API Security Assessment Documentation

Account Takeover, Weak Hashing, IDOR and Data Exposure Analysis

Tester: Sreenath Thekkedan

LinkedIn

GitHub Repository

1. Executive Summary

This document outlines a hands-on API security assessment performed to identify weaknesses related to authentication, hashing, authorization controls, data exposure, and application behavior. During testing, several vulnerabilities were identified that demonstrated how an attacker could gain unauthorized access, impersonate other users, retrieve sensitive information, and exploit weak design decisions.

The findings show a combination of broken authentication, weak hashing, IDOR (Insecure Direct Object Reference), excessive data exposure, and improper error handling, which together create conditions for account takeover and privacy compromise.

2. Scope and Objectives

Objectives

- Understand API authentication behavior

- Inspect how tokens and credentials are handled

- Examine hashing mechanisms

- Attempt user switching without authorization

- Detect exposed or broken links

- Observe HTTP method handling and UI responses

- Relate results to OWASP API Security Top 10

In Scope

- Login and credential handling
- Returned response data
- Token behavior
- URL parameter exposure
- User identity switching
- Hash verification
- UI error display messaging
- Link scanning

Out of Scope

- Backend source code
- Infrastructure penetration
- Denial of service
- Social engineering

3. Methodology

Approach

- API request testing using Postman
- Credential and token observation
- MD5 hashing comparison to demonstrate reversibility
- Testing URL parameter tampering
- Verifying account switching behavior
- Broken link enumeration

- HTTP verb handling analysis

4. Detailed Findings

Finding 1: Weak Authentication

The API allowed credentials to be passed directly within the URL query string instead of securely in a POST body.

Risk: Exposure through browser history, proxies, logs

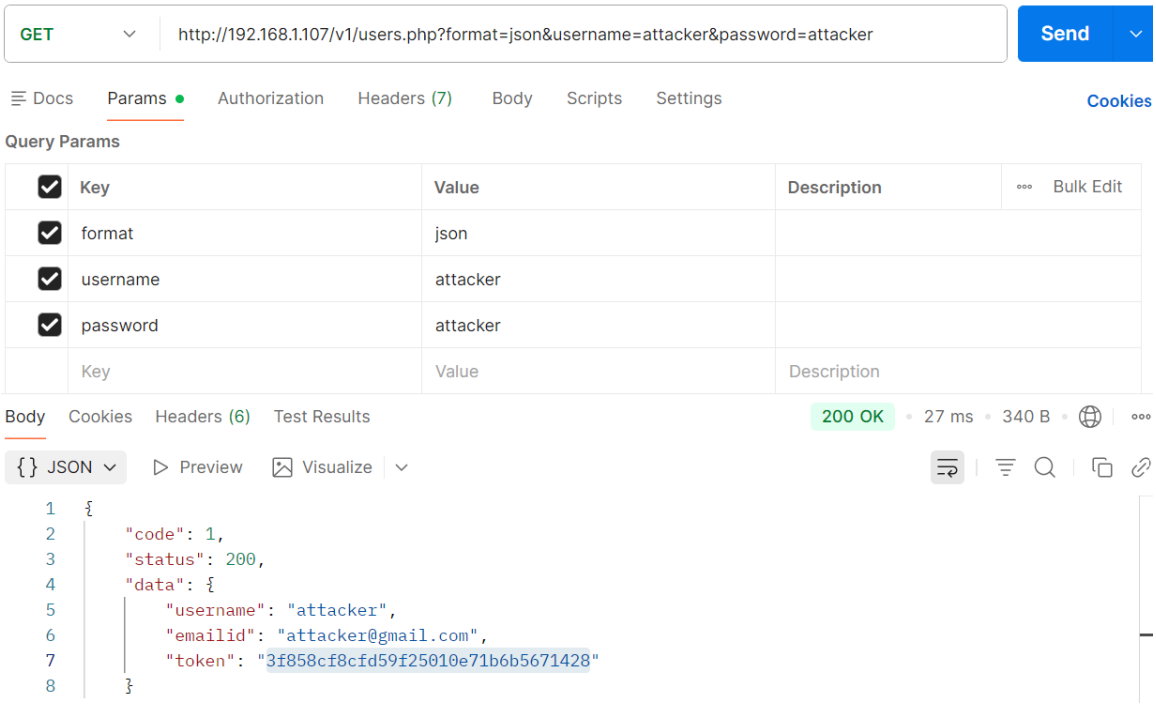
Impact: Credential harvesting and replay attacks

OWASP Mapping: **API2 - Broken Authentication**

Recommendation:

- Use POST payload instead of URL parameters

- Enforce secure session establishment



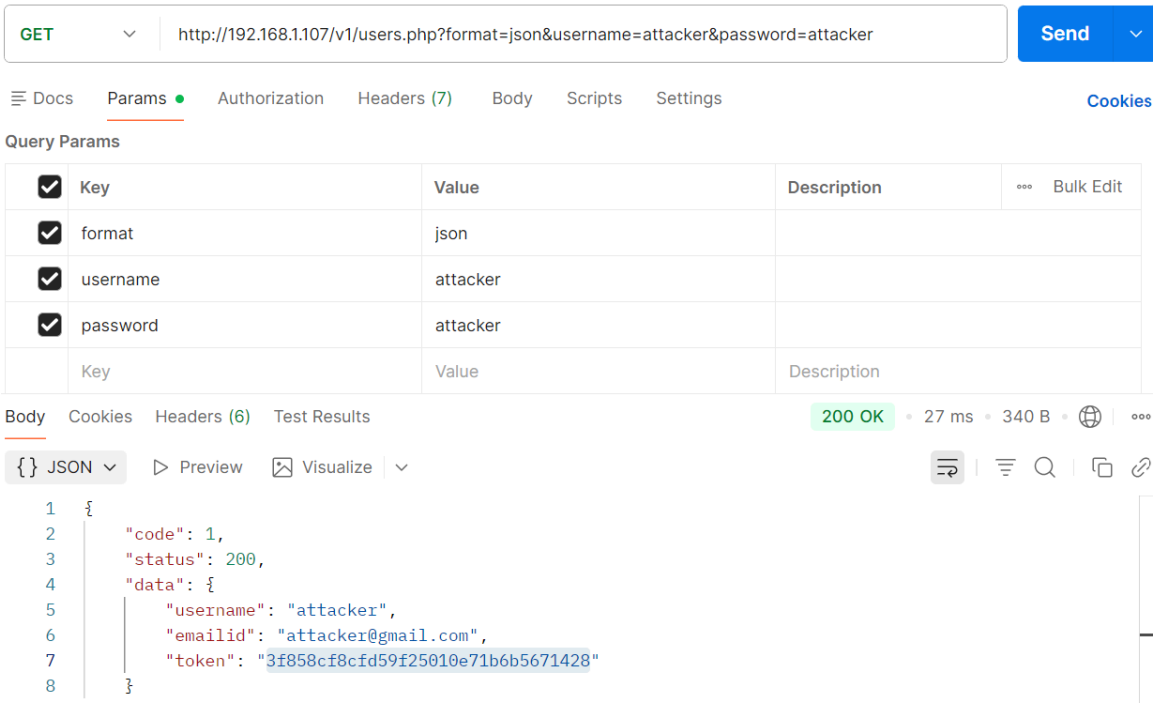
Finding 2: Sensitive Data and Token Exposure

The API response revealed user token, email, and identifiers in plaintext.

Risk: User impersonation

Impact: Privacy breach and compliance concerns

OWASP Mapping: **API3 - Excessive Data Exposure**



Recommendation:

- Remove sensitive fields from responses

- Use minimal data return principle

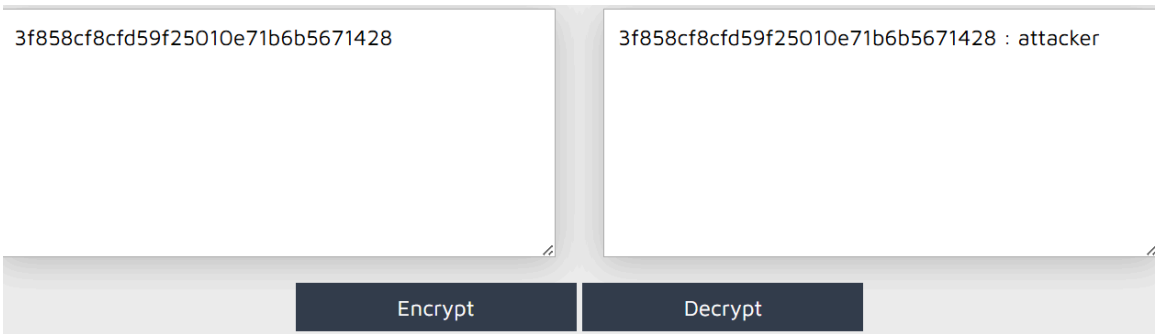
Finding 3: Weak Hashing (MD5)

The system used MD5 hashing, which was shown to match a reversible hash.

Risk: Hash cracking within seconds

Impact: Password compromise

OWASP Mapping: **API8 - Security Misconfiguration**



Recommendation:

- Replace MD5 with bcrypt, scrypt, or Argon2

Finding 4: IDOR Allowing Account Takeover

I was able to switch between users without proper authorisation logic.

Risk: Full account compromise

Impact: Access to private user accounts

OWASP Mapping: **API1 - Broken Object Level Authorization**

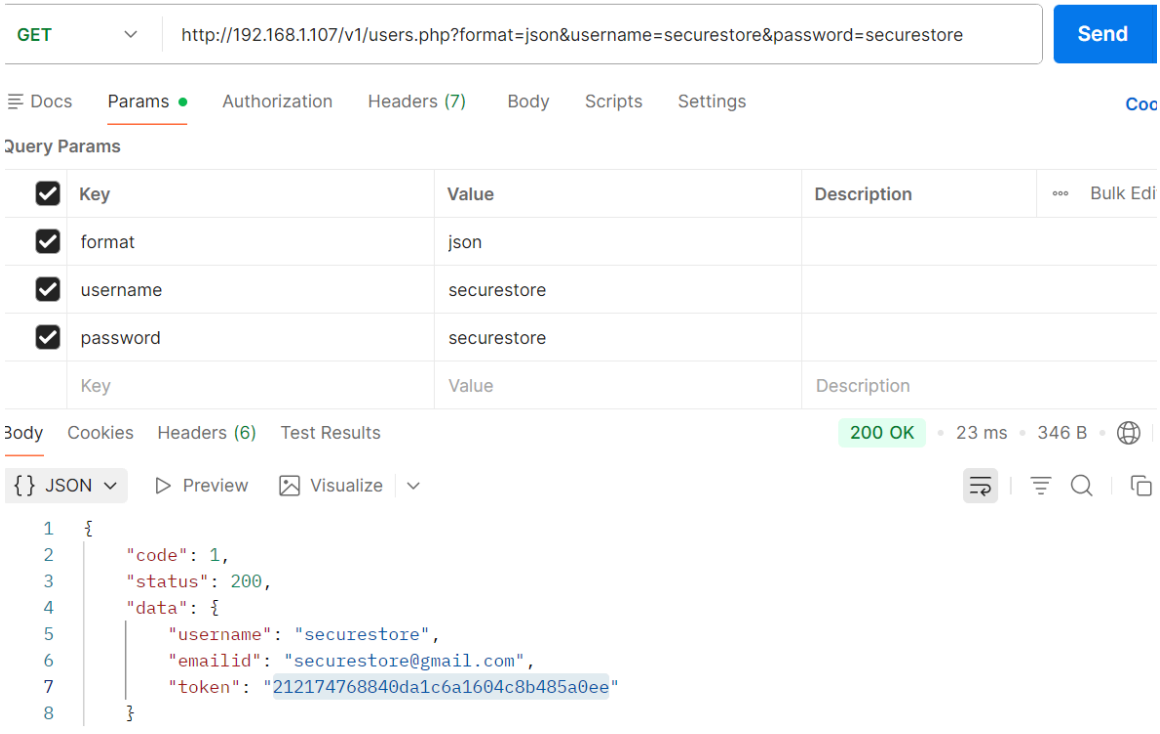
MD5 Hash Generator

Use this generator to create an MD5 hash of a string:

securestore

Generate →

Your String	securestore
MD5 Hash	212174768840da1c6a1604c8b485a0ee Copy
SHA1 Hash	9b5bb83c415e8dd23143eb44089700a5296ca6c7 Copy



Was able to move from one user(attacker) to another user(securestore)

Recommendation:

- Enforce user ownership verification on every request

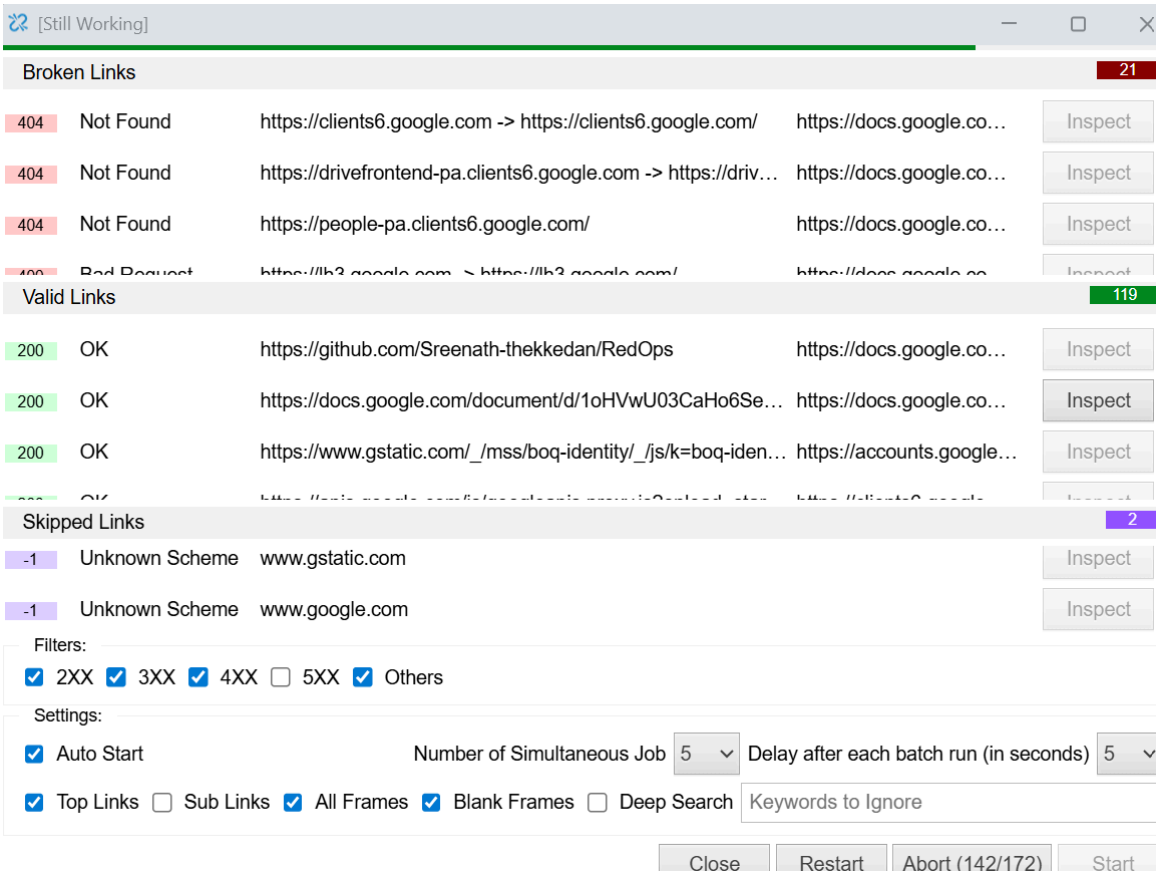
Finding 5: Broken Link Exposure(none was found on this website but tested it with another website and added for learning purpose)

A link scanning plugin revealed unused and outdated external references.

Risk: Reveals internal assets and legacy services

Impact: Attack surface expansion

OWASP Mapping: **API9 - Improper Asset Management**



Recommendation:

- Remove deprecated links

Finding 6: Improper Error Handling(none was found on this website but tested it with another website and added for learning purpose)

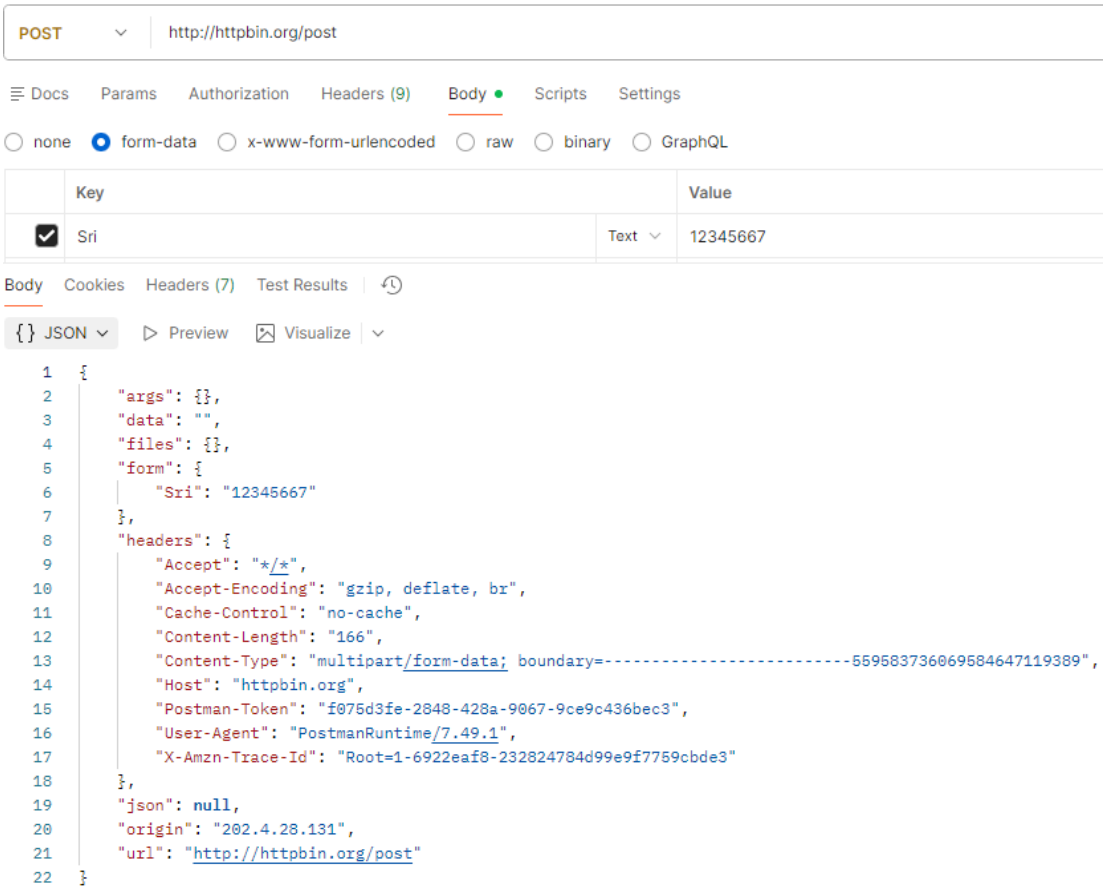
UI displayed “Method Not Allowed” instead of access restriction messaging.

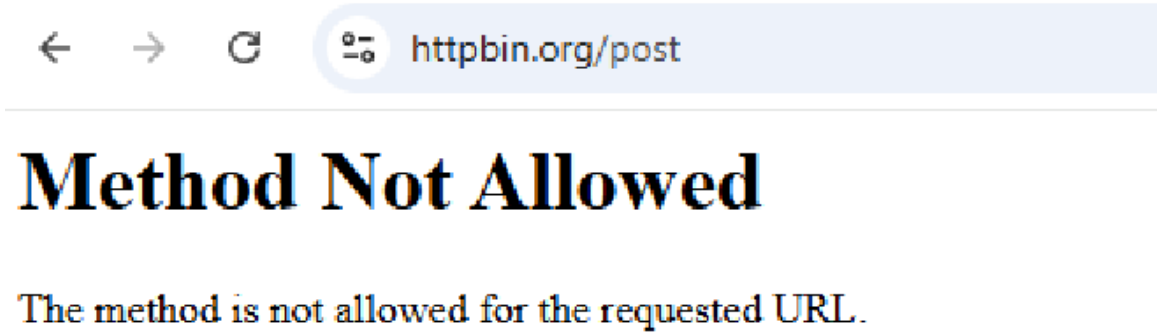
Risk: Reveals backend behavior patterns

Impact: Helps attackers determine allowed HTTP methods.

OWASP Mapping: **API7 - Improper Security Handling**

Checked with https://httpbin.org/ with random key and value parameters, and found it is authorising any user to access the data.





Recommendation:

- Standardize error messages

- Log details only server-side

5. Overall Severity Rating

Critical: 1

High: 3

Medium: 2

Low: 0

6. Business Impact

If exploited, these weaknesses could lead to:

- Unauthorized account access

- Exposure of user identity and credentials

- Legal and privacy implications

- Reputational damage

- Trust erosion

7. Recommendations Summary

Highest priority:

- Replace MD5 hashing

- Remove token and email exposure

- Enforce authorization on all user data access

Medium priority:

- Avoid using credentials in URLs

- Improve error messaging behavior

Lower priority:

- Remove broken and unused links

8. Tools Used

- Postman

- MD5 hash generator

- Browser broken link checker

- httpbin.org

9. Lessons Learned

-Small security gaps combine into major exploitation paths

- APIs require stricter validation than web interfaces

- Hashing choice is a critical security decision

- Authorization must always be verified server-side

- Error messages reveal more than developers realize

10. Conclusion

The assessment successfully demonstrated practical security weaknesses that can lead to account takeover, data exposure, and unauthorized access. Addressing the identified vulnerabilities will significantly strengthen the application’s security posture and resilience.