DVAPI Vulnerability Assessment Report

# API2:2023 – Broken Authentication

## Objective

To identify and exploit a Broken Authentication vulnerability in DVAPI (API2:2023) by abusing insecure JWT handling. This includes extracting, cracking, and forging JWT tokens to gain unauthorized access to protected endpoints.

## Overview of DVAPI

DVAPI (Damn Vulnerable API) simulates real-world API flaws for learning and testing. It includes common vulnerabilities as categorized by OWASP, such as authentication bypasses due to weak token management.

## Vulnerability Description

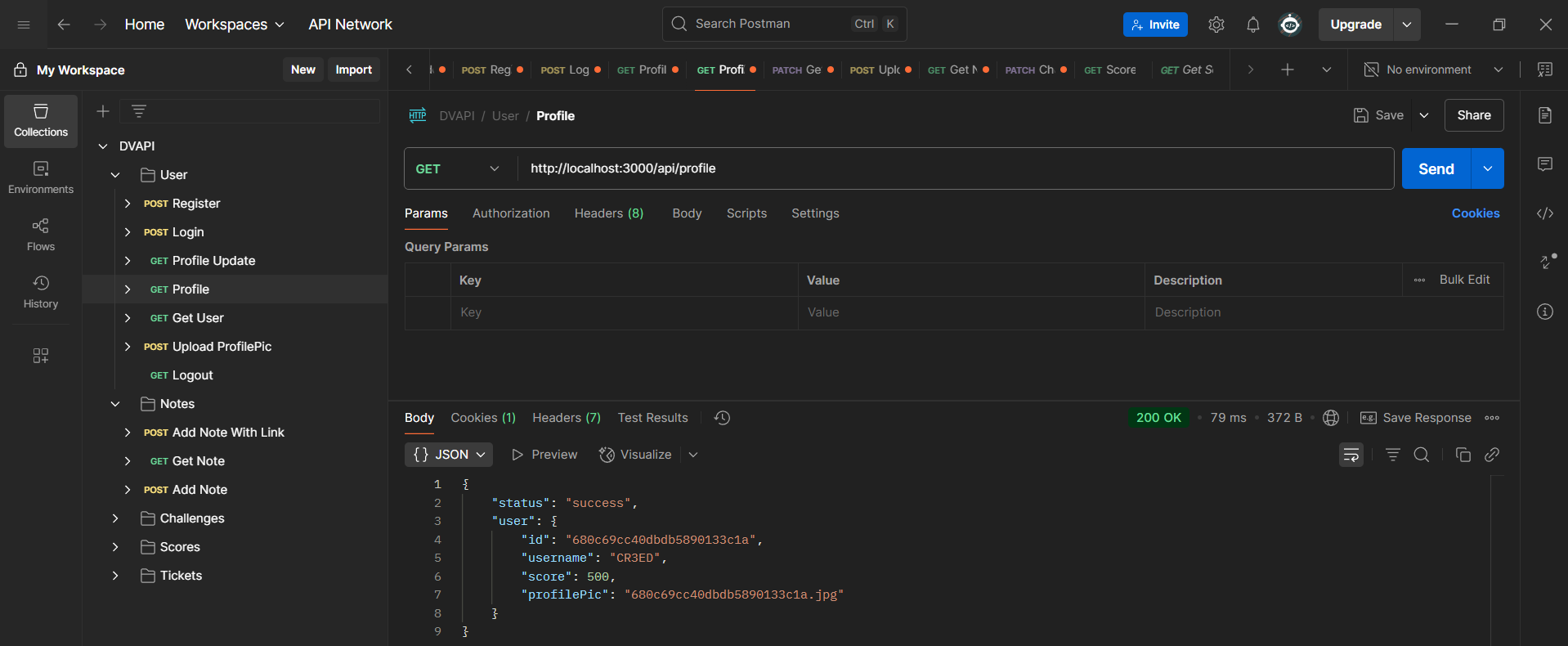
Broken authentication occurs when APIs fail to protect authentication mechanisms, such as session tokens or JWT secrets. In this case, the JWT secret used for token signing is easily guessable and not securely managed, allowing an attacker to forge a valid token.

## Tools Used

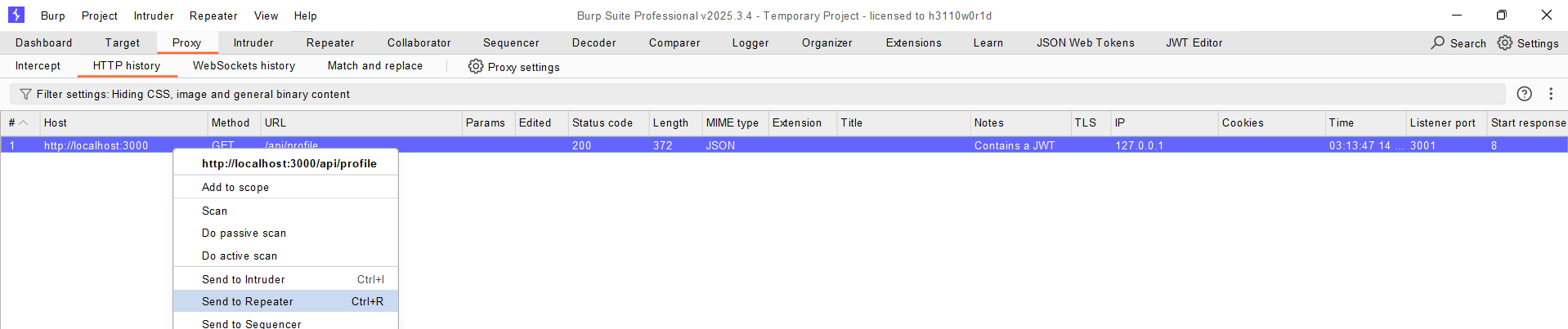
**• DVAPI:** Running at http://localhost:3000  
**• Postman:** Used to send API requests and inspect responses  
**• Burp Suite:** Used for intercepting JWT tokens and modifying them  
**• Hashcat:** Used to crack the JWT secret using a dictionary attack

## Steps to Reproduce

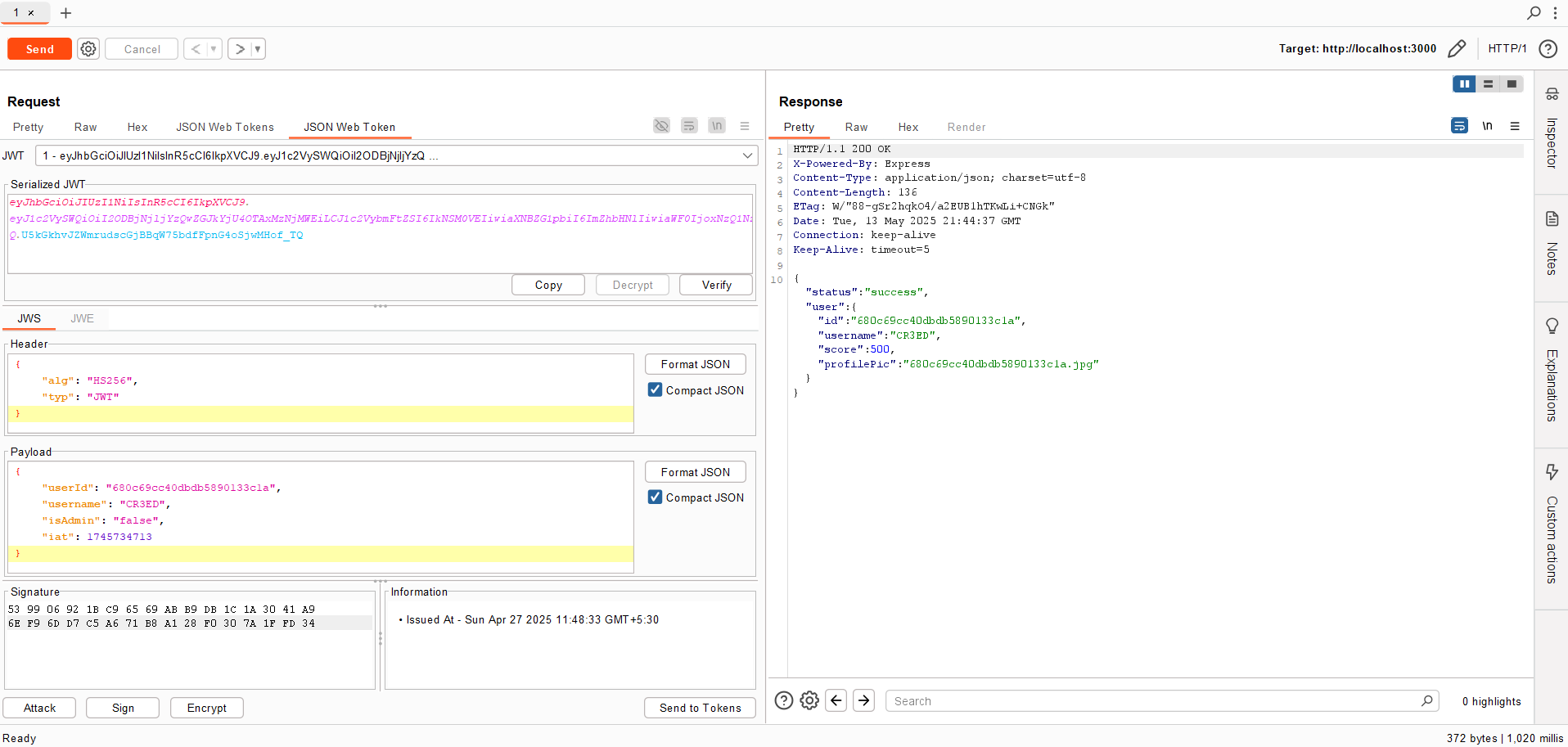
1. A request was made to `/api/profile` from Postman, retrieving a JWT token for a regular user.



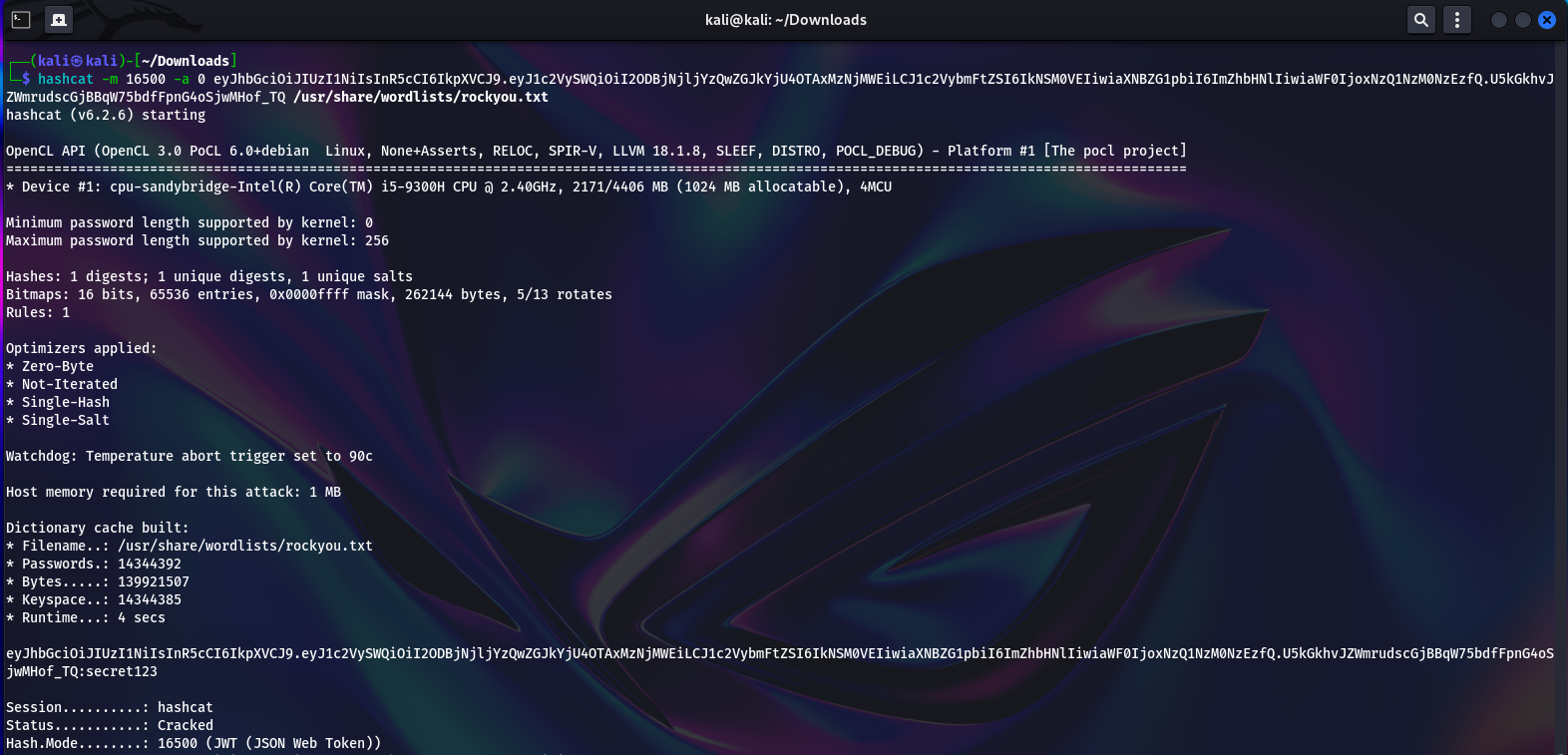
2. The request was captured via Burp Suite’s HTTP history and sent to the Repeater for modification.



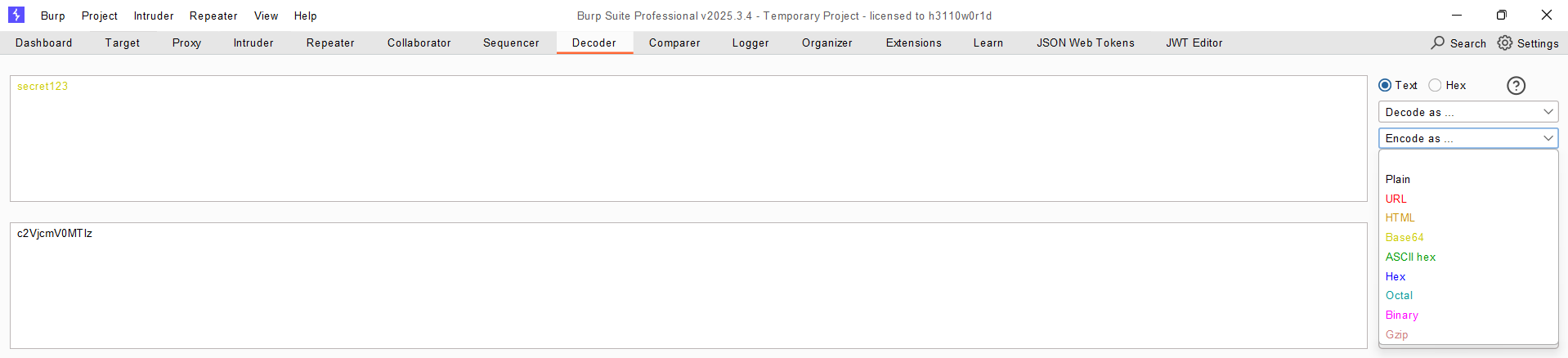
3. The JWT was copied from the Repeater's JSON Web Token tab for offline analysis.



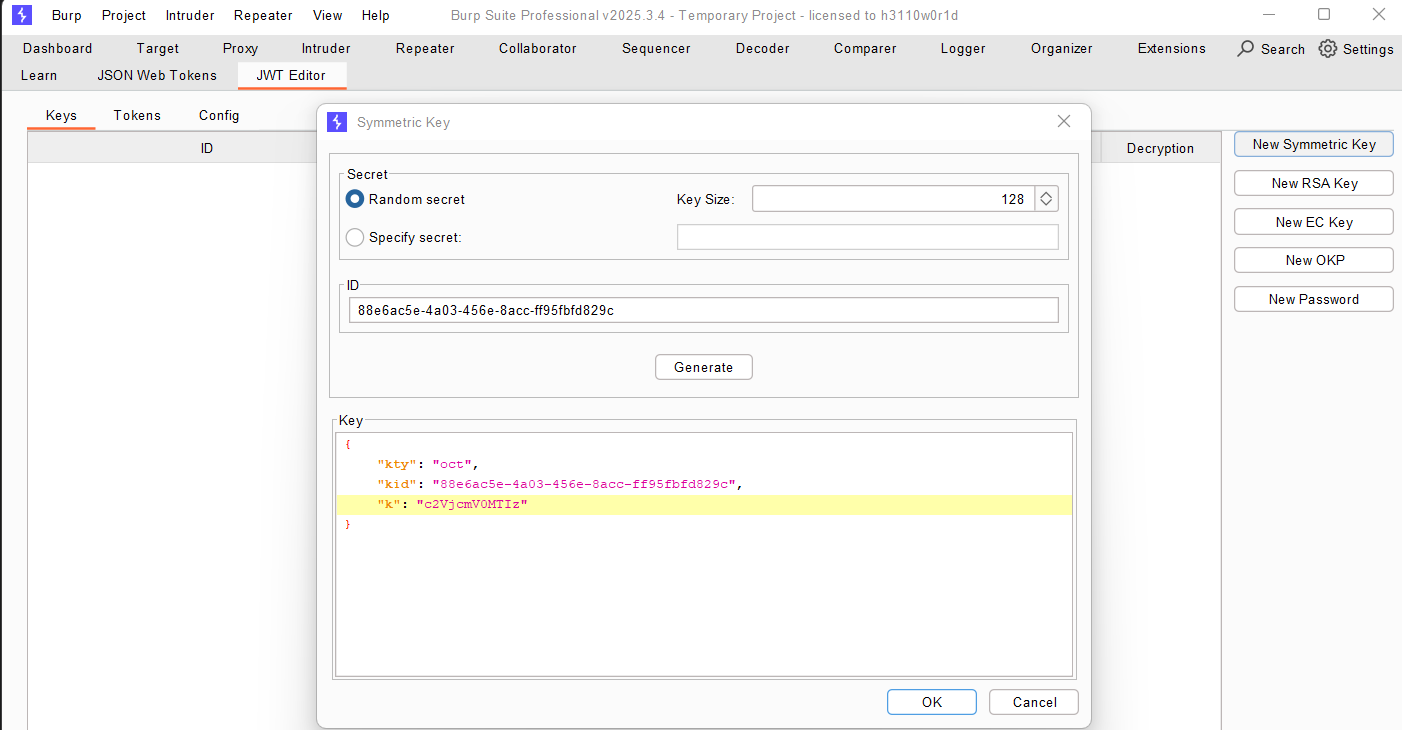
4. The JWT was then cracked using Hashcat with the `rockyou.txt` wordlist, successfully revealing the secret.



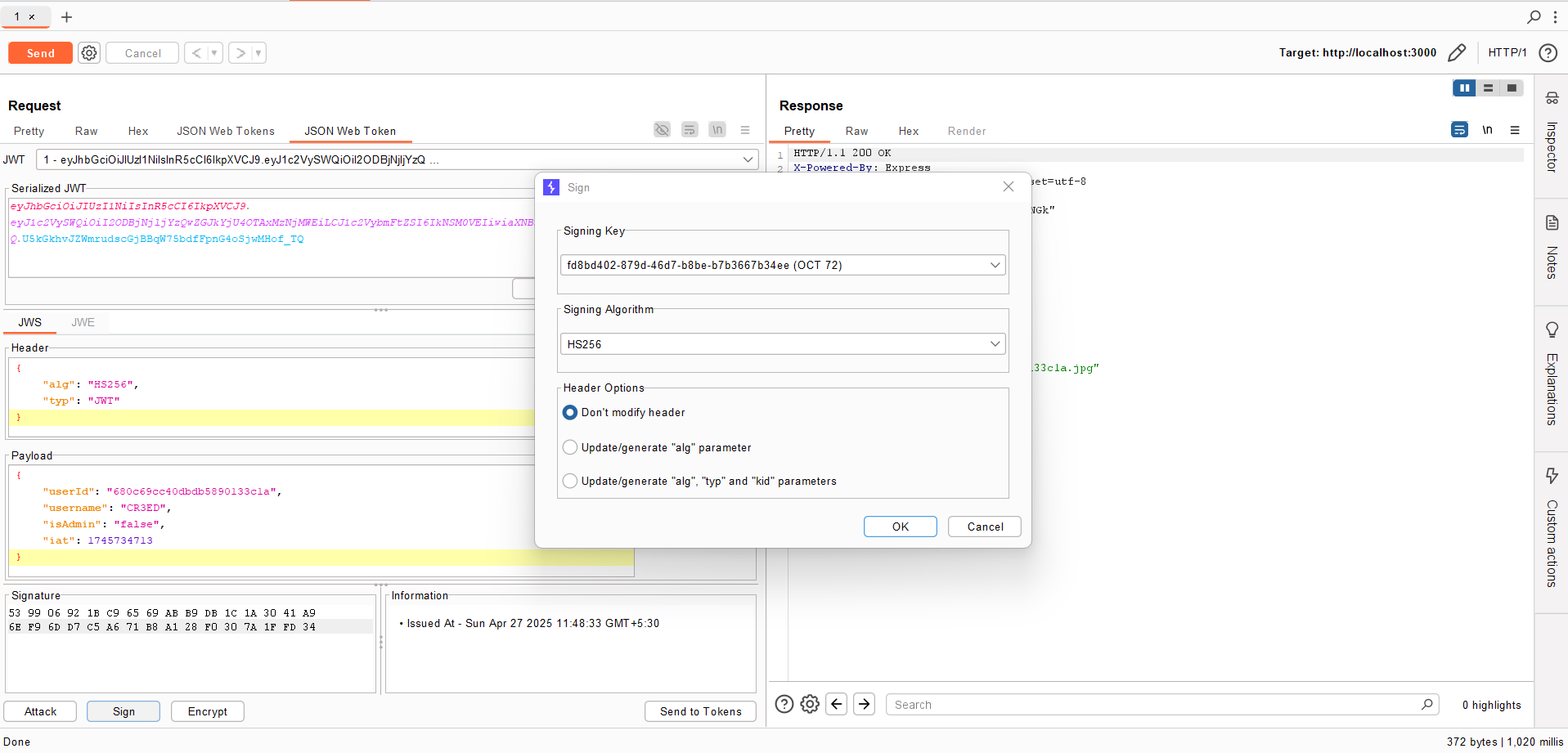
5. The cracked secret was encoded in Base64 in Burp Suite's Decoder.



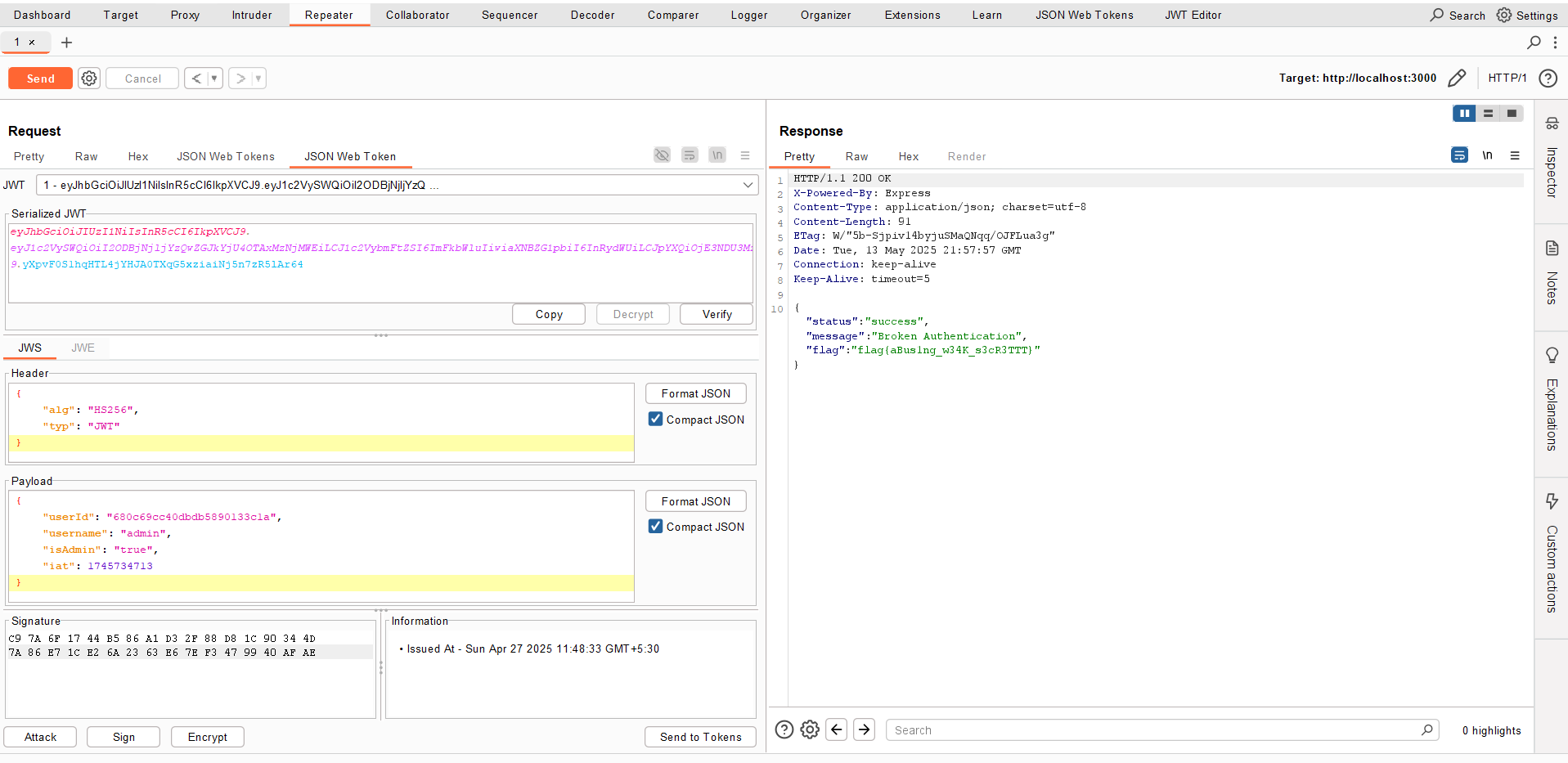
6. In the JWT Editor a new symmetric key was created and the base64 encoded password was placed in the k value of the key



7. Back to the repeater the new symmetric key was selected as the signing key



8. Finally the request was sent revealing the Flag



Flag: "flag(aBusing\_w34K\_s3cR3TTT}"

## Impact

An attacker can impersonate users or escalate privileges by forging valid JWTs. This can lead to unauthorized data access, account compromise, or total system control if admin tokens are forged.

## Mitigation Strategy

• Use strong, unpredictable secrets for signing JWTs.  
• Store secrets securely and rotate them periodically.  
• Implement token expiration and enforce revocation.  
• Avoid exposing internal JWT signing logic or keys.  
• Use secure algorithms (e.g., RS256) with key validation.