Lab 9

Chetan Sahrudhai Kimidi - ckimidi

***Abstract*—Exploiting common authentication flaws and JWT Token related lab tasks.**

I. Introduction

In this lab, I will solve common authentication flaws in WebGoat 7.1, and JWT in WebGoat v2023.4.

II. Authentication Flaws - WebGoat 7.1

A. Password Strength

I solved various questions, related to how quickly a password can be cracked, depending on the complexity. This can be seen in Fig. 1.

My personal takeaway regarding why each password needs a different time is because, the cracking time is the brute-force time indirectly. So, when the password is only lower case letters, the ASCII range is small i.e. 97 to 122 only. But, if upper case letters are also used, then complexity increases. So, the crack time depends upon length of password times the range of possible password characters.

B. Forgot Password

I solved this using a brute-force approach, using BurpSuite Intruder. First, I intercepted the request sent for Forgot Password, for username admin, in my Burpsuite Proxy and I sent that request to Intruder, where I added a payload marker near the color field. I added a payload list of 10 common colors to parse through and check. After I started the attack, I could see the response length for 'green' was different from all other colors. When I checked its full response, I found out the favorite color of admin is green. This can be seen in Fig. 2 and Fig. 3.

C. Multiple Level Login 1

We have to use an expired TAN of the user and login successfully. Since TAN 1 is used, it cannot be re-used again. But, we can change which TAN is checked, for instance, by changing the HTML value from 2 to 1, in the Inspect section, as seen in Fig. 4. In this way, I completed this task, as seen in Fig. 5.

D. Multiple Level Login 2

Similar to the previous level, we need to change a HTML value using the Inspect feature. Here, I edited the username HTML value from Joe to Jane, and successfully logged in as Jane, as seen in Fig. 6 and Fig. 7.

III. Identity & Auth Failure - WebGoat v2023.4

A. Authentication bypass

To solve this, I opened up the Developer Tools section of the browser, and observed that there was a mechanism which was verifying whether the security questions are being validated or not, not the answers of the questions, but the questions themselves. This can be seen as verifyMethod with value of SEC_QUESTIONS, in Fig. 8. As suggested in the hints, I tried to directly remove the security questions, but this verify method was making sure that there needed to be some security questions definitely. So, I tried changing the name of the security questions and hit submit, then it actually verified without actually verifying the questions. What I did was, change the security questions from, secQuestion(0/1) to secQuestion(X/Y/A/B/...). This satisfied the verify method functionality, but also made sure that the security questions weren't actually being verified, due to the name change. This can be seen in Fig. 9.

B. JWT Tokens Steps 1-5

The first two lessons explained what JSON Web Tokens actually were and regarding the structure of a JWT, how it has 3 parts called header, payload/claims, and signature, and how it is base-64 encoded. In the third lesson, I solved a basic task of decoding a given token, using the JWT feature of WebWolf. I copied the token and pasted it into WebWolf JWT panel, and saw the username was 'user', as seen in Fig. 10. The fourth lesson explained about the authentication of a JWT token and how we can get a token. The fifth lesson was related to signatures and JWT signing. Here, I first selected Tom as the user and saw that I could vote but only admins with top privileges could reset the vote count. The goal was to attain admin privileges and reset the vote count. To achieve this, once I was logged in as Tom, I opened the Network panel in the Developer Tools section. Once open, I clicked on reset vote, and I observed there was a new POST request, with the name 'votings'. So, I understood that I can use ZAP and intercept this request, edit it with a modified token (for which, I can use WebWolf), and send it, to complete the task. I did the same, set the algorithm to null/none and the admin privilege to true in the WebWolf JWT editing panel, copied the modified token, pasted it in the ZAP resend window, and successfully completed the task. This can be seen in Fig. 11, Fig. 12 and Fig. 13.

C. JWT Tokens Steps 6-9

The sixth lesson was the detailed solution for the fifth lesson's task, and lessons 7,8,9 were regarding the code review and where possible weaknesses lie in the code, how they can

be solved, what happens if parts of the token are removed, such as signature or algorithm being set to none, and any alternatives, such as PASETO. I solved the quiz question in lesson 7, as seen in Fig. 14.

IV. Password Reset Steps 1-5

The first lesson was basic information about the password reset functionality and its importance. In the second lesson, there was a basic task which made sure that we could read email with the WebWolf interface. I was successfully able to get the emails regarding password reset, after clicking on the Forgot Password, as seen in Fig. 15. The third lesson stated the mechanism behind the password reset, such as checking if the account exists or not. I had already solved the fourth lesson, using BurpSuite Intruder, in WebGoat 7.1 Authentication Flaws - Forgot Password, and I found out already that admins favorite color is green. So, I successfully completed that as seen in Fig. 16. The fifth lesson was basically viewing different kind of security questions, in case of, password rest. I viewed all the various questions there, and completed the task, as seen in Fig. 17.

V. JWT Tokens Step 10

The tenth lesson in JWT tokens was related to JWT cracking. The goal here was to find out the secret key, when given a token, and then return a new token, where the username has been modified to WebGoat. To find out the secret key first, I had to take the given token locally and process it by matching it (brute) with a big list of some 'n' number of top common words. To do this, I found out about a tool called hashcat. Hashcat is the world's fastest and most advanced password recovery utility, supporting five unique modes of attack for over 300 highly-optimized hashing techniques. And for the list of common words, I found a file on GitHub, as referenced in [3]. Now, for the brute force part, I installed hashcat on my VM, using sudo apt install hashcat, and used the command as seen in Fig. 18, to start the brute force guessing. After approximately 20 seconds and 146 word checks, hashcat found out the word (secret key) in the token was 'available' (see Candidate 1, in picture), as seen in Fig. 19. Then, I copied the given token into WebWolf and entered 'available' as secret key, and changed the username from 'Tom' to 'WebGoat'. I copied the modified token, and pasted it in the input box, and completed the task successfully, as seen in Fig. 20 and Fig. 21.

VI. Conclusion

In summary, this report concludes that I exploited various authentication flaws and solve advanced JWT tasks.

References

[1] PayPal 2FA
[2] JWT IO
[3] List of Words for Hashing
[4] Hashcat
[5] Hashcat usage for JWT Tokens

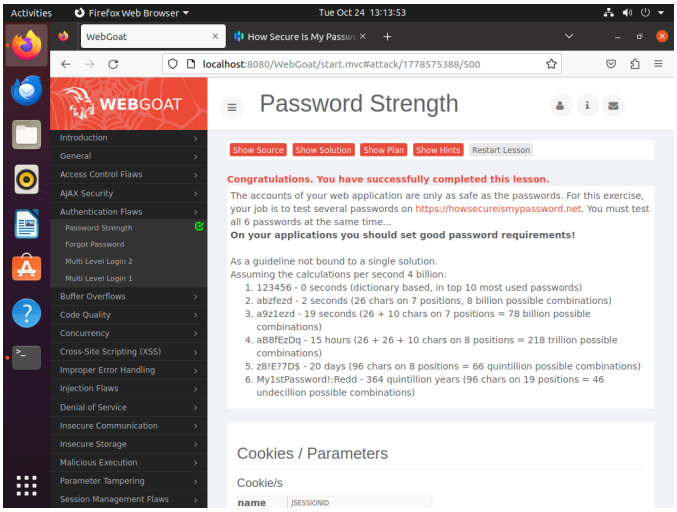

Fig. 1. Password Strength

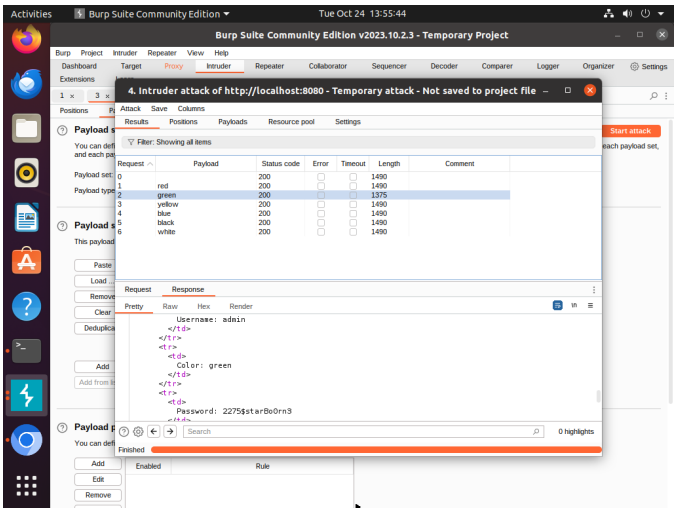

Fig. 2. Intruder Attack

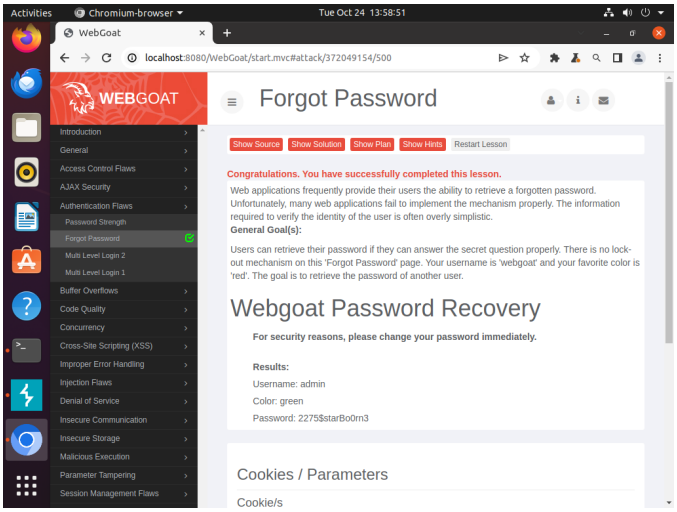

Fig. 3. Color green!

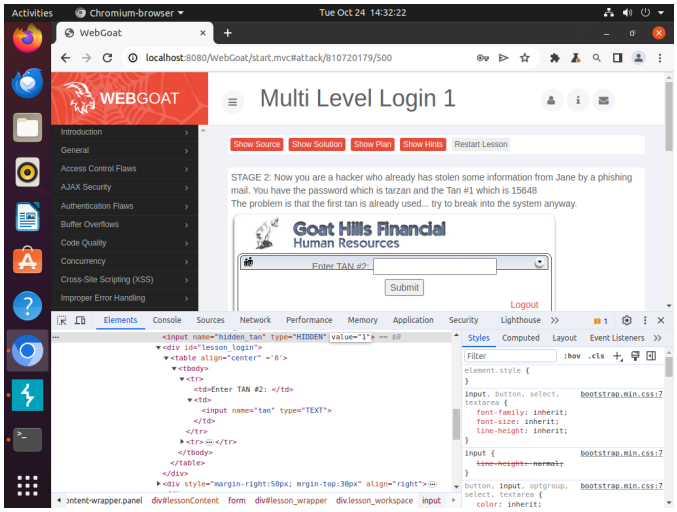

Fig. 4. Changing value

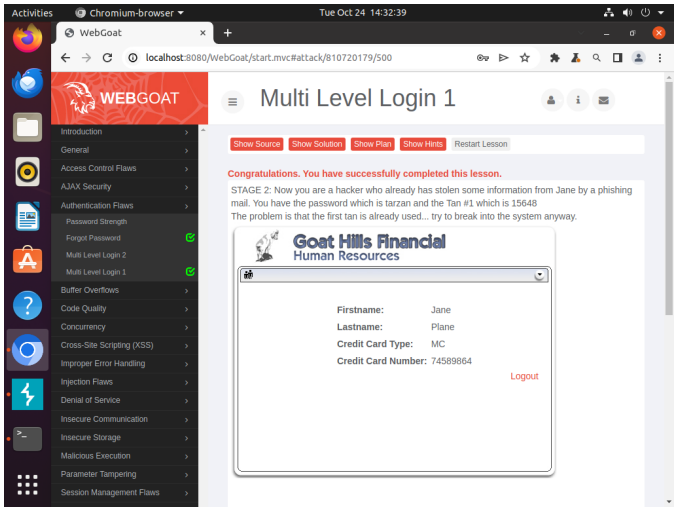

Fig. 5. Task done

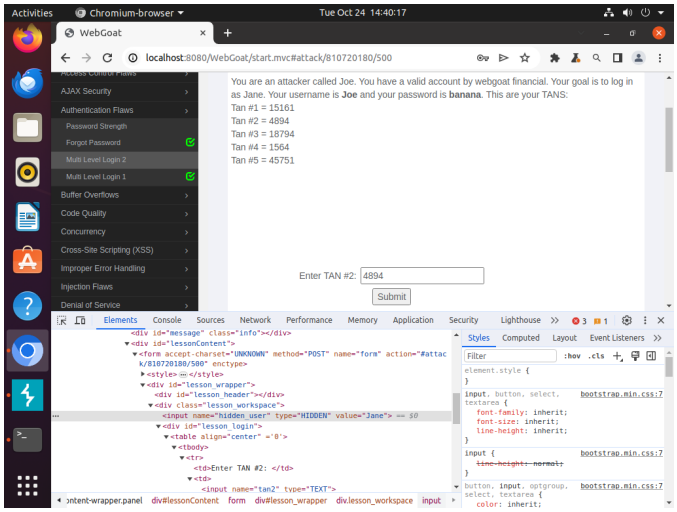

Fig. 6. Joe to Jane

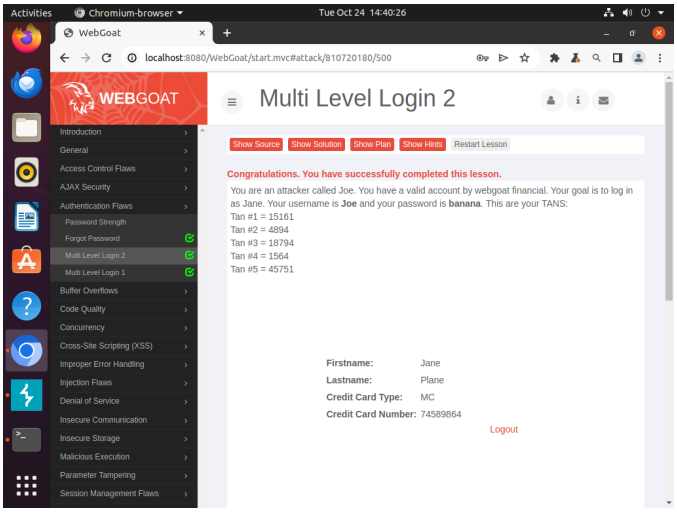

Fig. 7. Task done

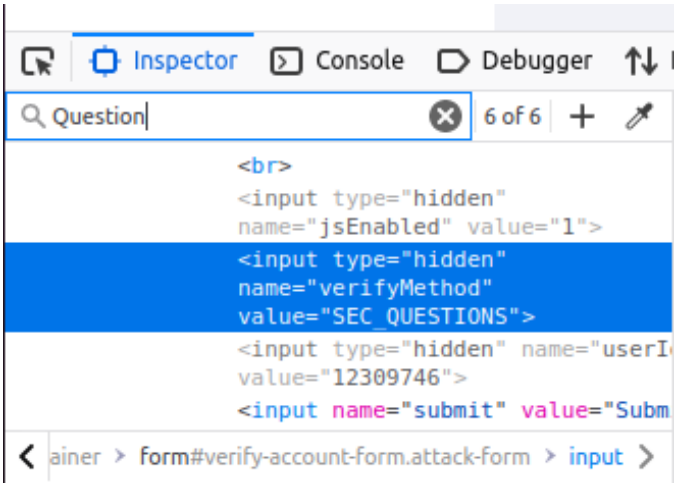

Fig. 8. security questions???

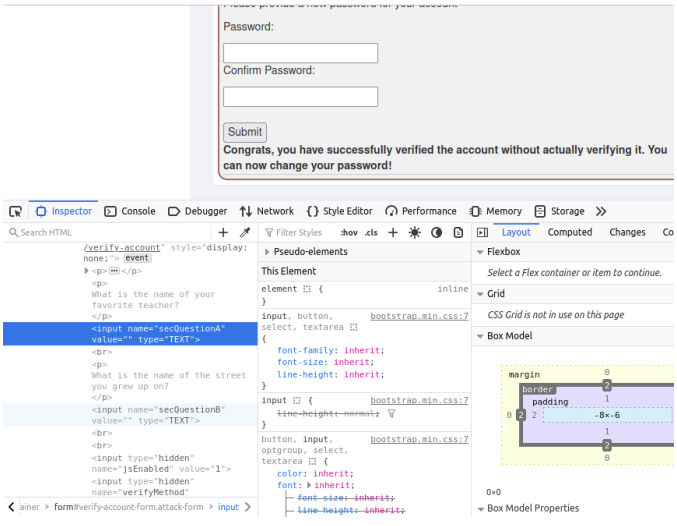

Fig. 9. Authentication BYPASSED!

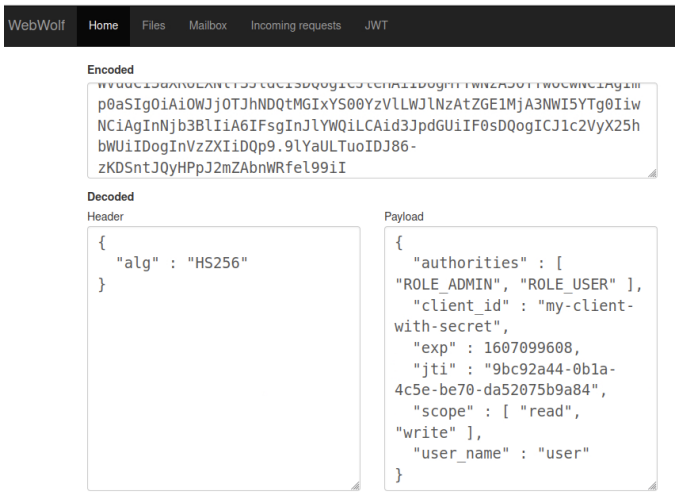

Fig. 10. WebWolf JWT

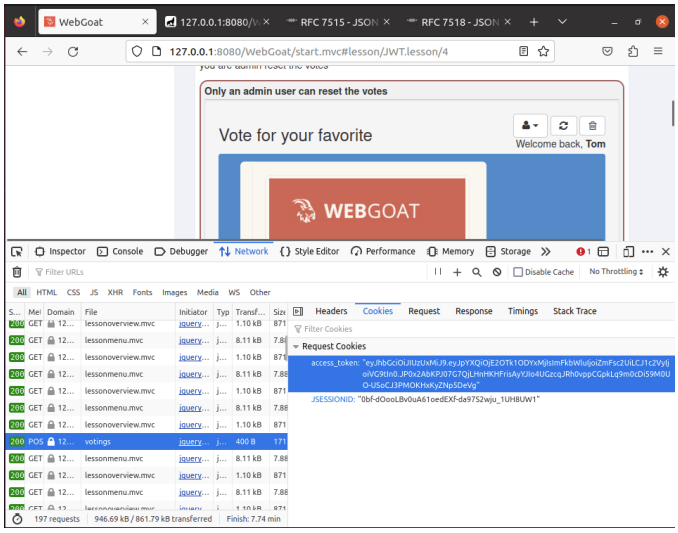

Fig. 11. POST Request

Fig. 12. Modified JWT

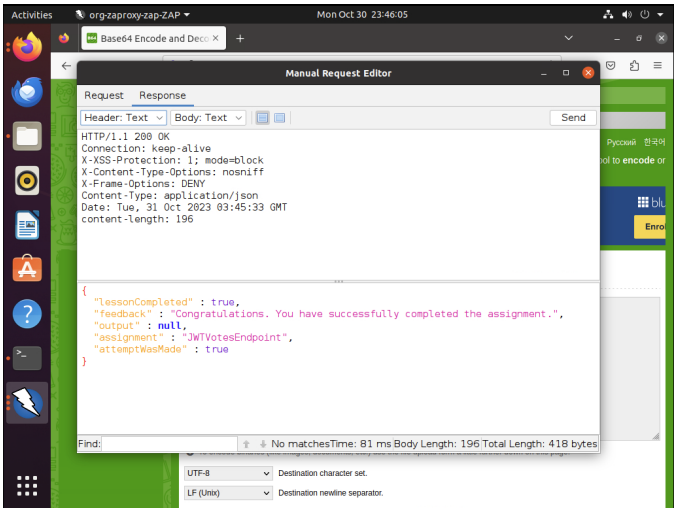

Fig. 13. Task completed!

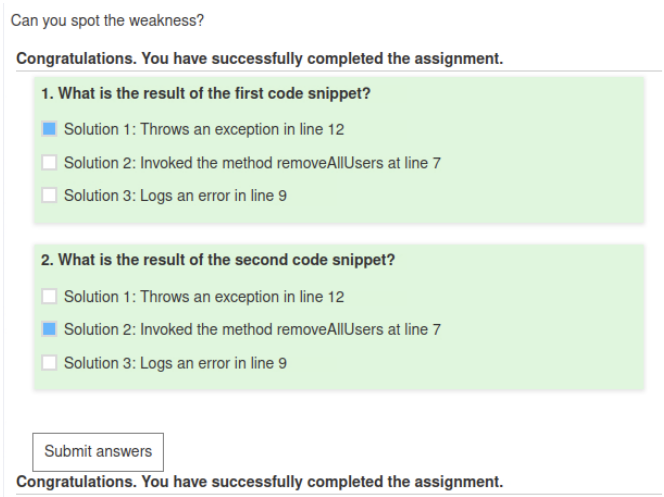

Fig. 14. Quiz done

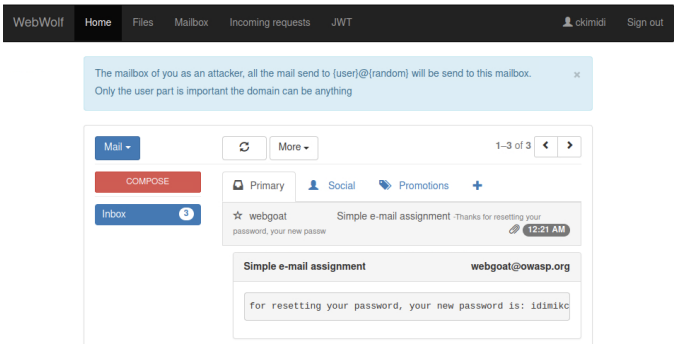

Fig. 15. Password Reset Lesson 2

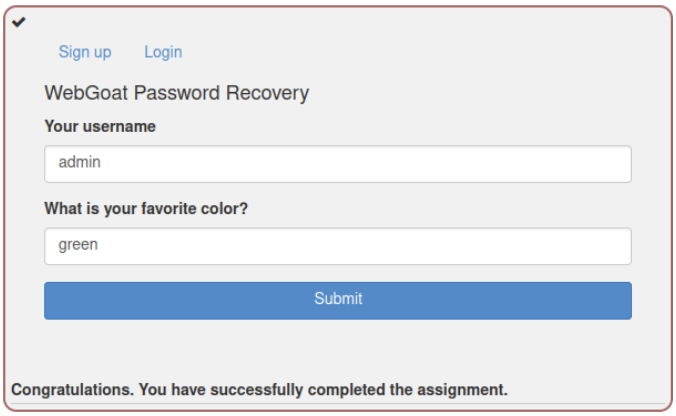

Fig. 16. Password Reset Lesson 4

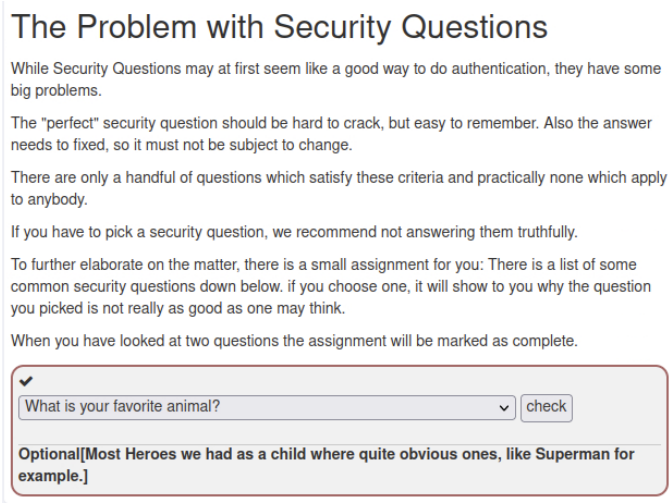

Fig. 17. Password Reset Lesson 5

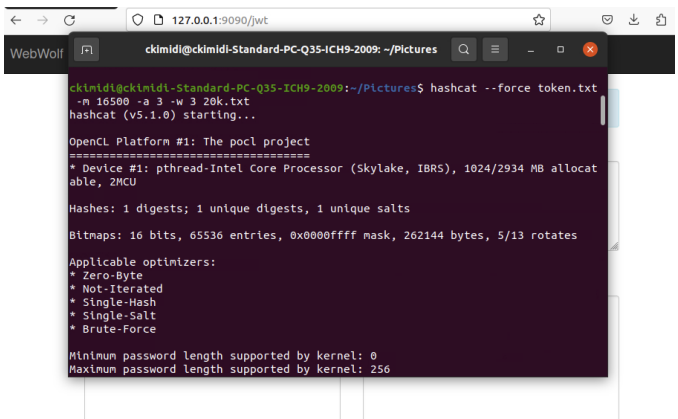

Fig. 18. Hashcat command

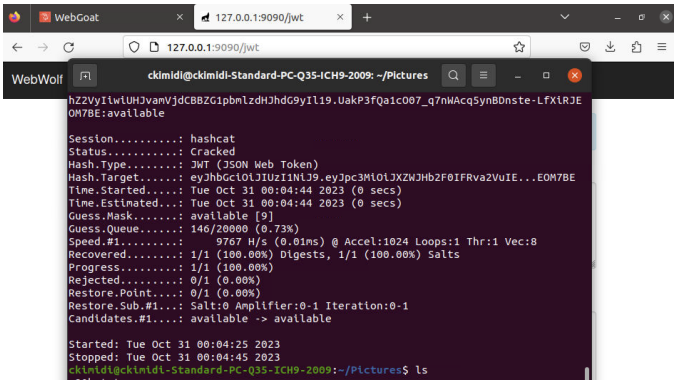

Fig. 19. Secret key found!

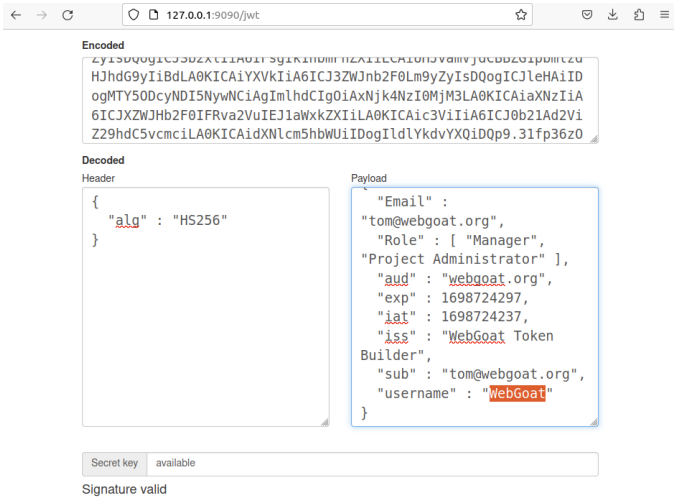

Fig. 20. Modifed username to WebGoat

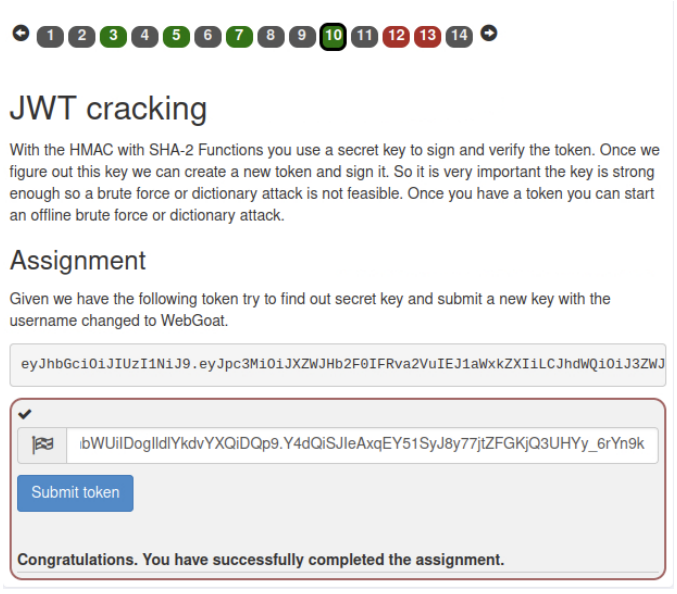

Fig. 21. Tasks completed!