ENPM809W - Introduction to Secure Coding

Lab - 5 – Attack Lab – Poor Authentication and Authorization

Author: Syed Mohammad Ibrahim
UID: iamibi
Email: iamibi@umd.edu

Phase 1: Burp Suite to intercept requests

a) Provide the URL of the WebGoat.NET application page where you are exercising the question. Do not include any query parameters or any other special characters in the answer.

   http://localhost:52251/WebGoatCoins/CustomerLogin.aspx

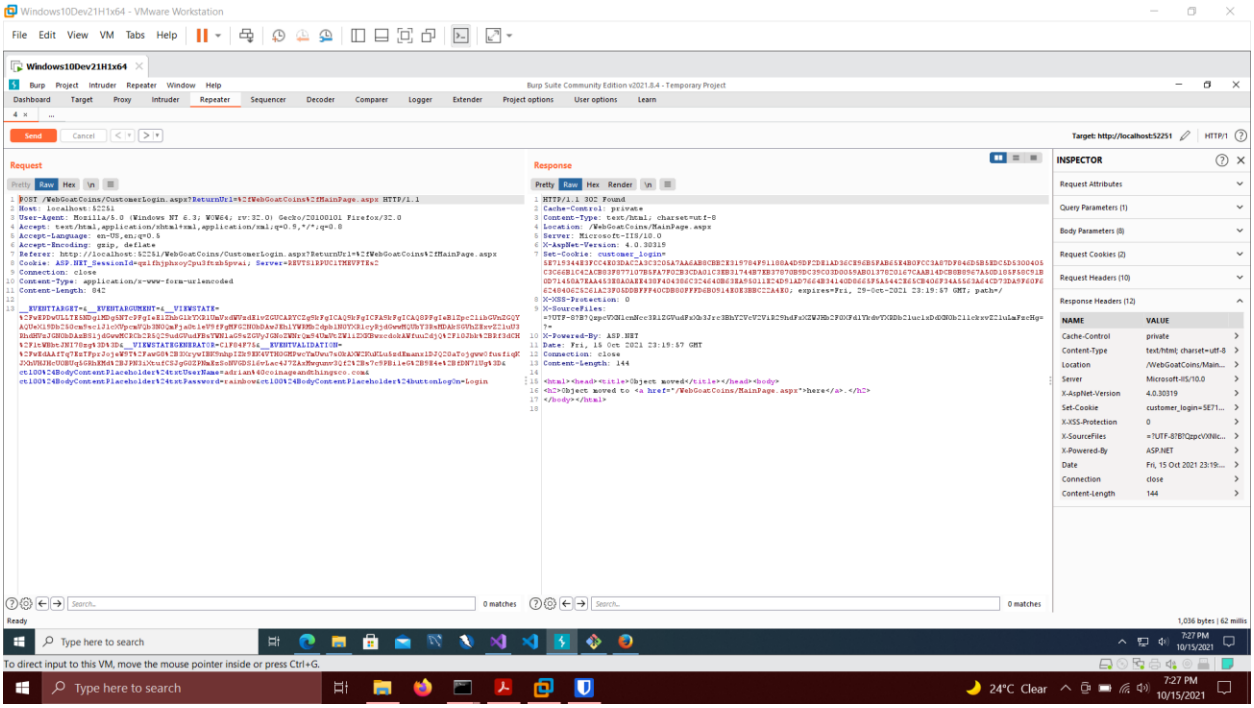
b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

   The input was intercepting the request using Burp Suite under proxy tab and then using the repeater option to resend the request. The request contained the original value of user that they entered, that is, email id and password as plain text.

c) Describe the output result.

   Using Burp Suite, I was able to view the request and its response from the server. There was no limit placed on the number of requests that can be made to the server from Burp Suite.

d) Provide a screenshot of the output.



e) Provide the CWE-ID, Filename, Line Number of the weakness that is at the heart of the vulnerability.

CWE ID(s):
- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-319: Cleartext Transmission of Sensitive Information

Filename: Configuration
Line Number: N/A

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

There are majorly two vulnerabilities involved:
1. The unlimited attempts allowed while authenticating a user.
2. Sending sensitive credentials as plaintext over the network.

Because of these vulnerabilities, an attacker can perform an unlimited number of login attempts for any/every email id of user that they find for the system.

Phase 2: Brute-force

a) Provide the URL of the WebGoat.NET application page where you are exercising the question. Do not include any query parameters or any other special characters in the answer.

   http://localhost:52251/WebGoatCoins/CustomerLogin.aspx

b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

   The input to the Burp Suite was the request made for these three email ids juri@gold4allagescom.net, karttunen@coinsoffinland.net, mory@osakacoinageco.net The request contained the email id and initially a random password to generate a request. After generating the request and passing it to Repeater and then to Intruder, the Twitter Ban word list (399 words) was used to substitute multiple passwords for the above email ids. To identify whether the attack succeeded in any of the password attempt, a part of HTML tag was marked in the Burp Suite under Grep which said, “Incorrect Login!”.

c) Describe the output result.

   For each email id, after executing the intruder attack with words from the twitter ban list there was one password per email that returned empty html tag on “Incorrect Login!”, at which point I stopped the script and ran the password in the Customer Login portal. The password was correct, and I was able to login as the user.

d) Provide a screenshot of the output.

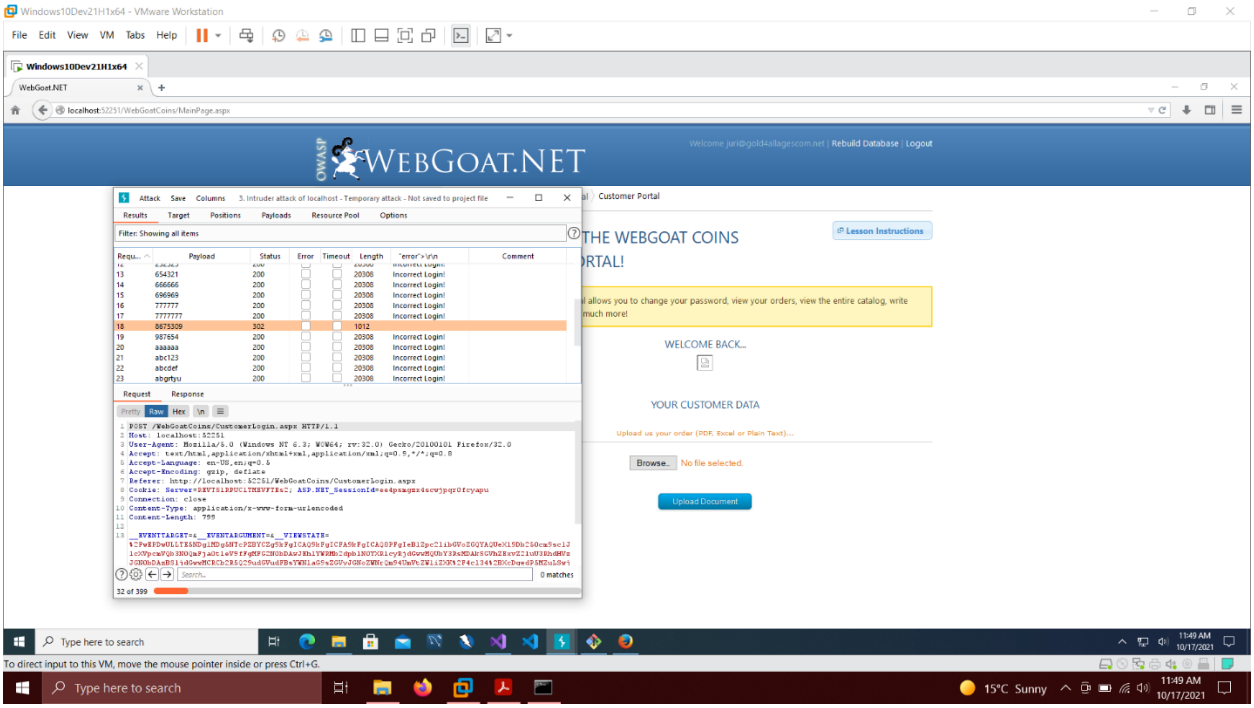


Figure – 1 – juri@gold4allagescom.net password cracked

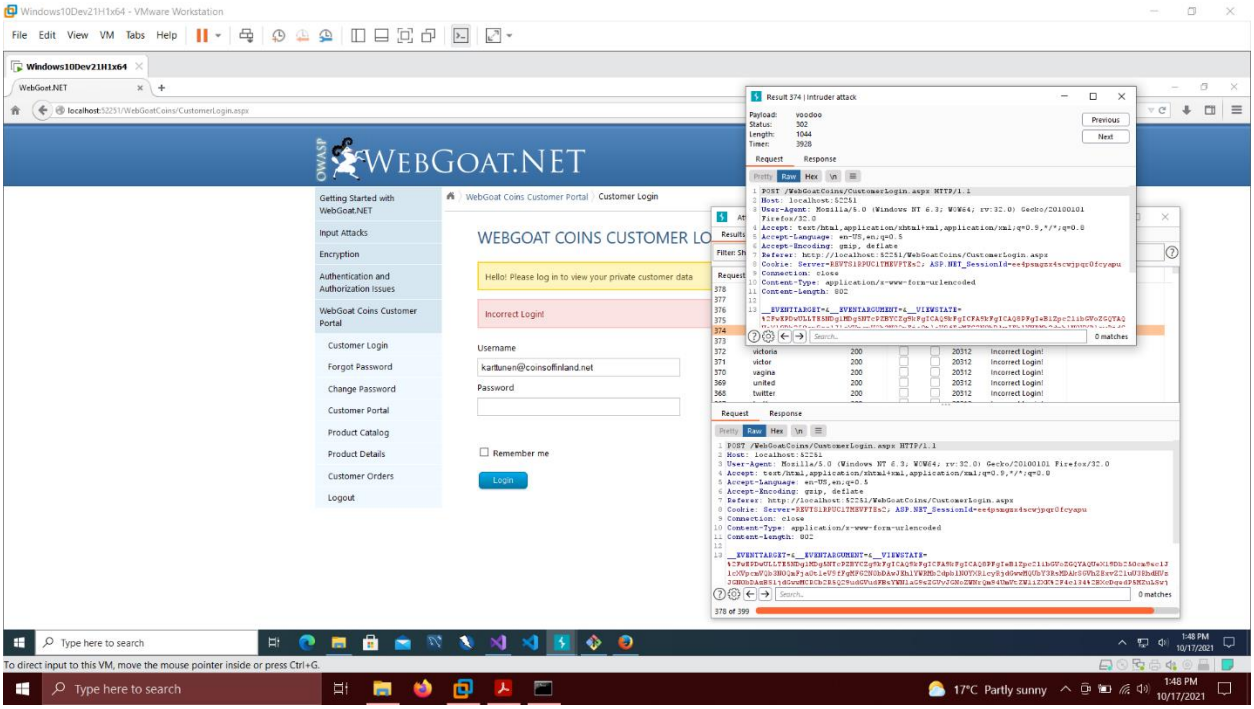


Figure – 2 - karttunen@coinsoffinland.net password cracked

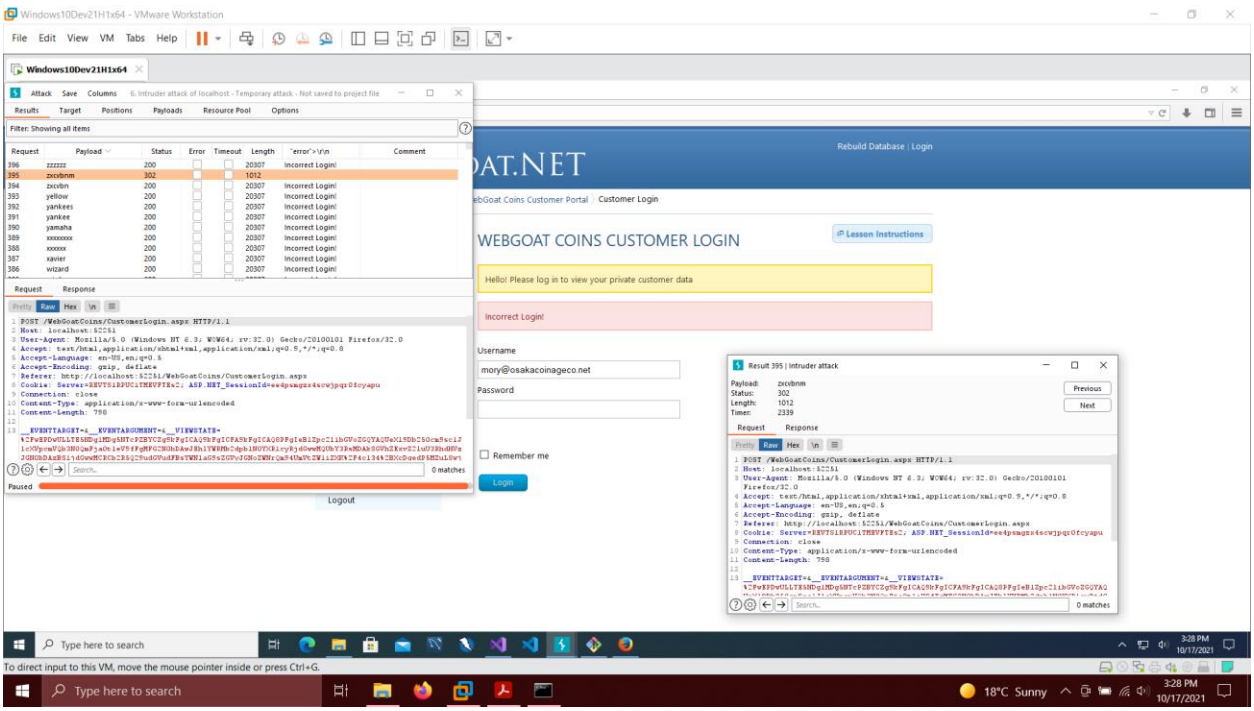


Figure – 3 - mory@osakacoinageco.net password cracked

e) Provide the CWE-ID, Filename, Line Number of the weakness that is at the heart of the vulnerability.

CWE-ID(s):

- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-521: Weak Password Requirements

Filename: Configuration
Line Number: N/A

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

A combination of unrestricted number of login attempts, cleartext transmission of credentials over the network and weak password requirements can make the attacker perform a rainbow table attack on any user's account if the attacker gets hold of their account email id. An attacker with a huge resource power would be able to retrieve passwords successfully after brute-forcing the passwords over the network.

Phase 3: Forgot Password

a) Provide the URL of the WebGoat.NET application page where you are exercising the question. Do not include any query parameters or any other special characters in the answer.

   http://localhost:52251/Content/ForgotPassword.aspx

b) Describe and provide the Input given to the application. Provide the input as is with no decorations. If the input is a URL, provide the URL encoded string. If the input is provided as text to multiple fields, list the fields and the input provided for each. If the input is non-printable characters, please provide the bytes provided to the input in Hexadecimal format. For example: 9ABCD01234.

   Approach 1: Using Burp Suite Brute-Force
   The initial input contained an email address (taken from the questions) which then required the user to provide an answer to a security question. For each email, a request was made through the Burp Suite using the interceptor and repeater and using the intruder to carry out a dictionary attack from the twitter ban list on the security question. The request that was captured in the Burp Suite contained URL escaped plaintext email and plaintext answer to the security question. To check whether the Burp Suite succeeded, I used an HTML tag ( ) to identify whether the answer was correct or not.

   Approach 2: Using Code and Burp Suite
   Going over the code, it is visible that the answer to the security question was stored in the key called "encr_sec_qu_ans", encoded two times. Intercepting a request in Burp suite and using a base64 decoder two times gave away the result.

c) Describe the output result.

   Approach 1:
   For every email id, the response for the security question was found in the Twitter Ban list. The HTML tag would not appear for the correct answer request. After putting the answer to the security question, a success message was displayed with the password decoded. There was no limit put on the number of attempts that can be made to the server.

   Approach 2:
   Using a Base64 decoder twice on the value of "encr_sec_qu_ans" intercepted by Burp Suite gave away the answer.

d) Provide a screenshot of the output.

   Approach 1:

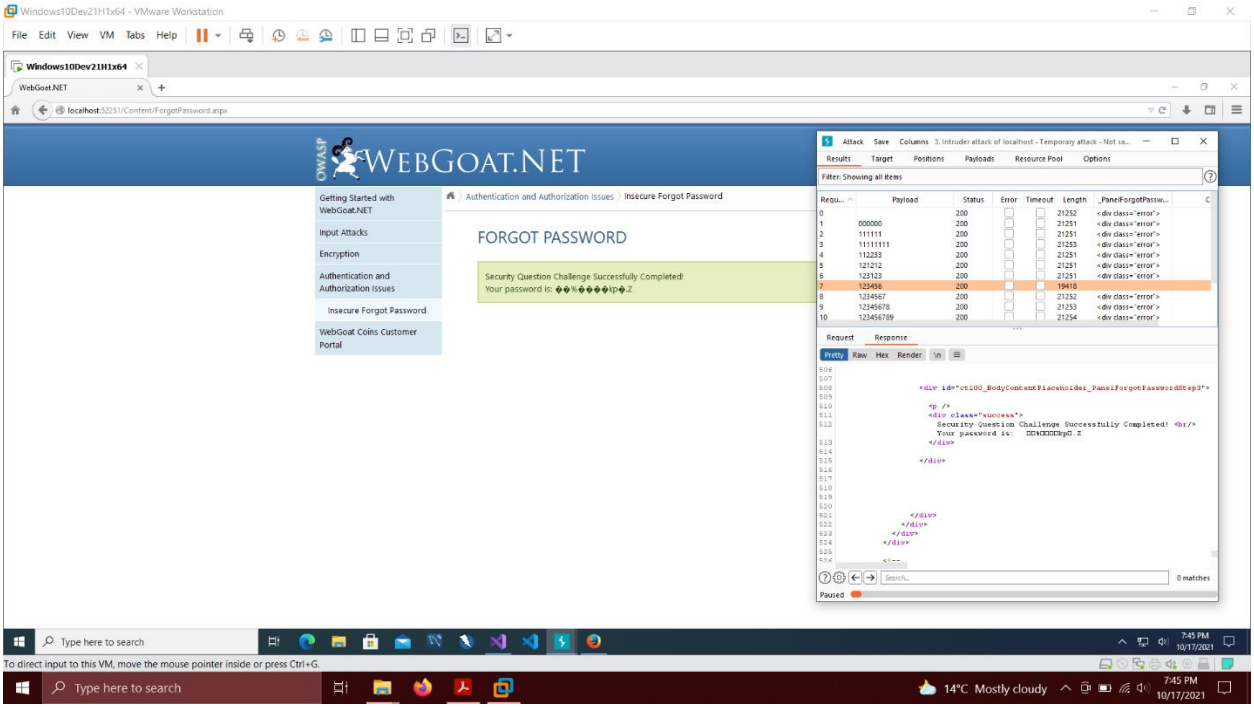


Figure – 1 - julie@golddepotinc.com security answer recovered

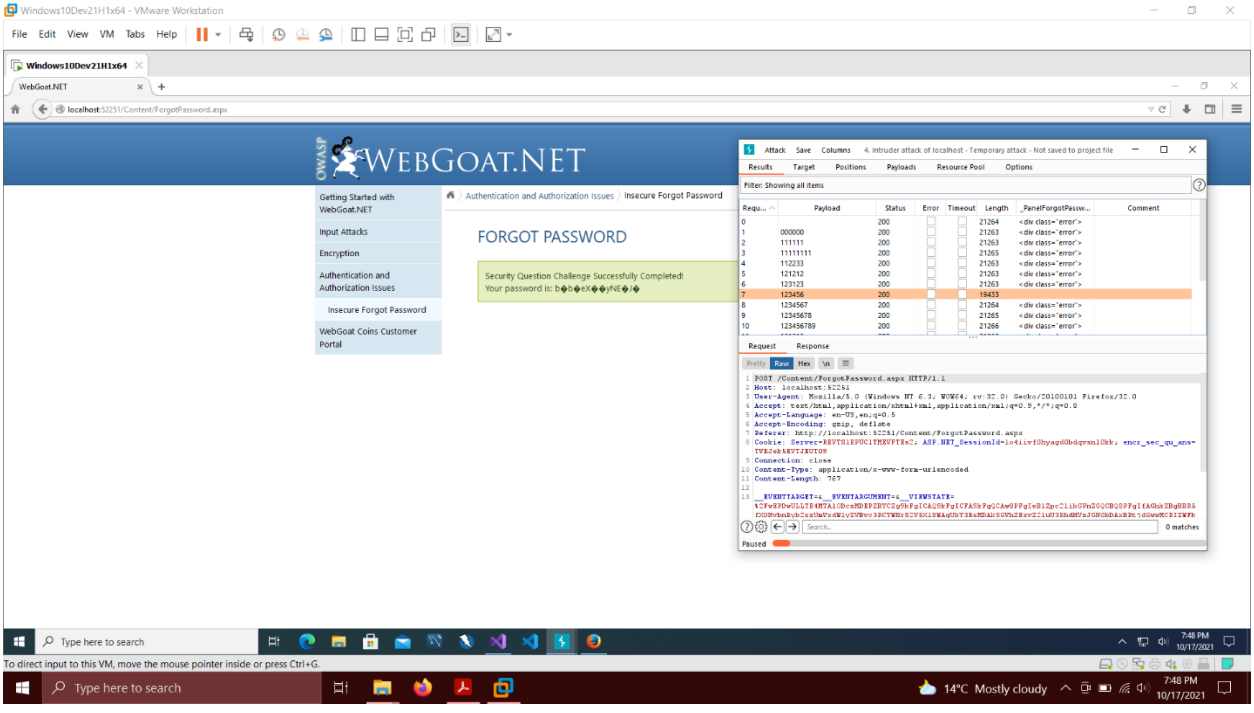


Figure – 2 - calaghan@australiangoldnetwork.net security answer recovered

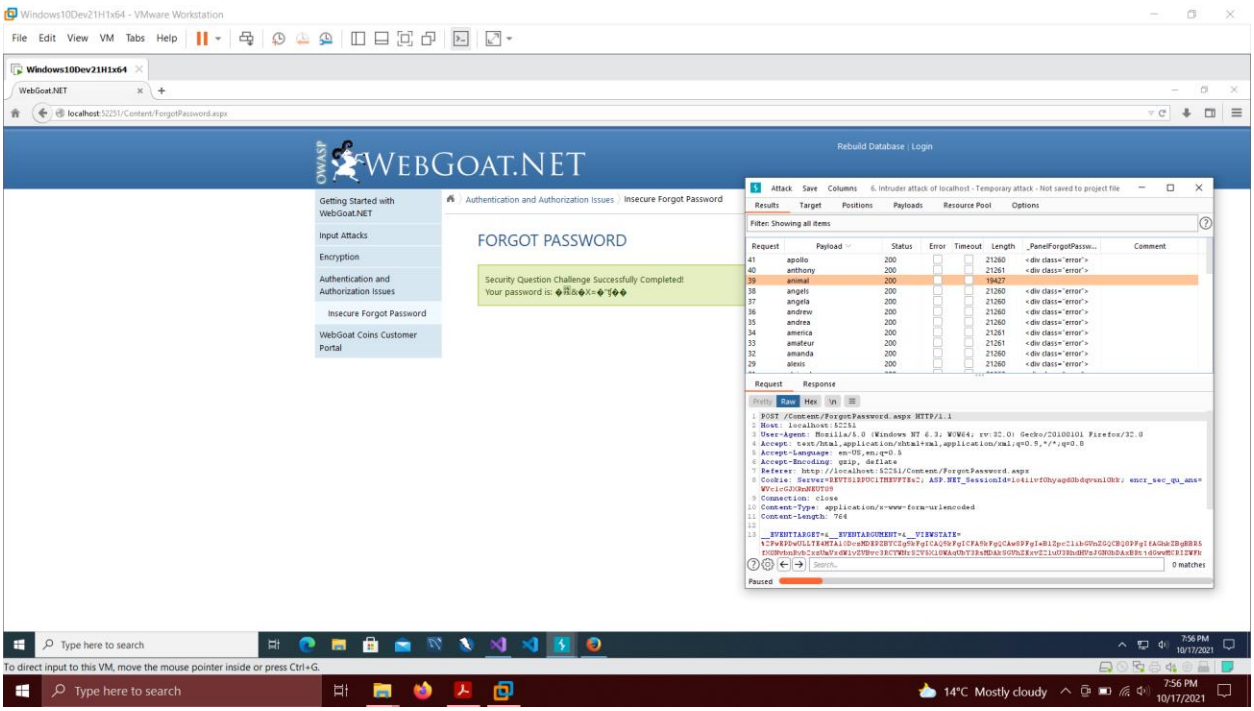


Figure – 3 - semenov@kremlincollectables.com security answer recovered

Approach 2:

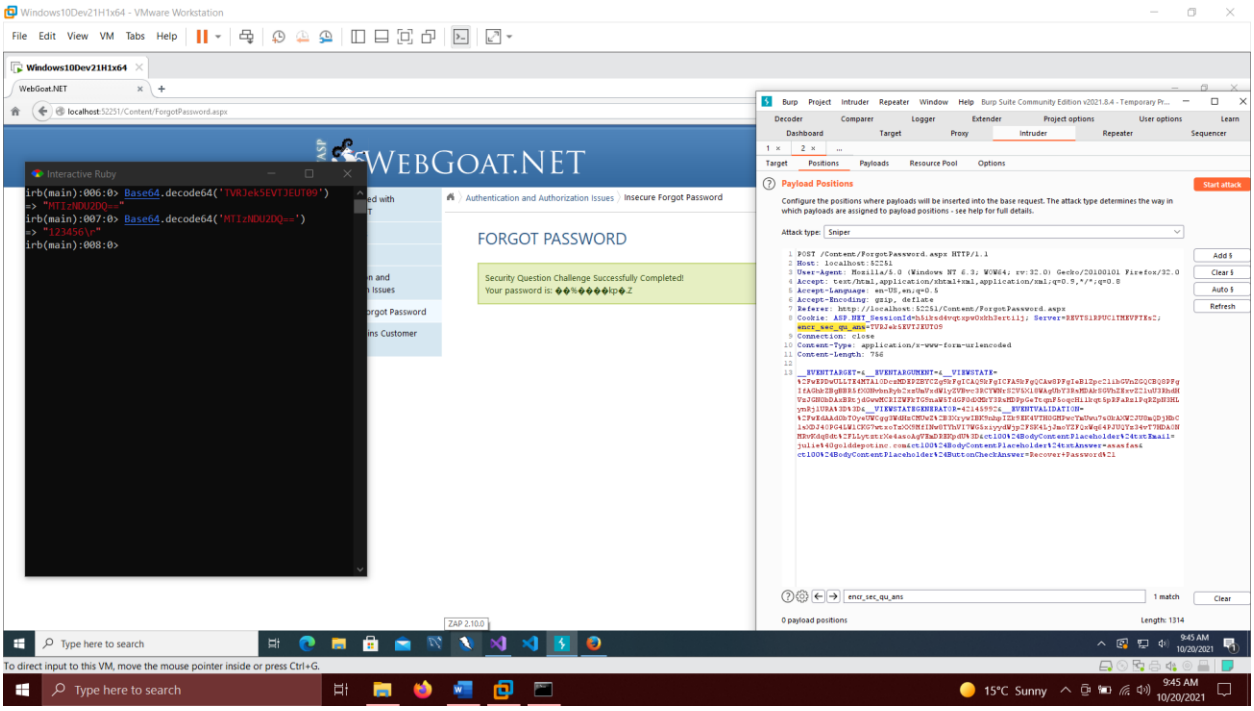


Figure – 4 - julie@golddepotinc.com security answer recovered

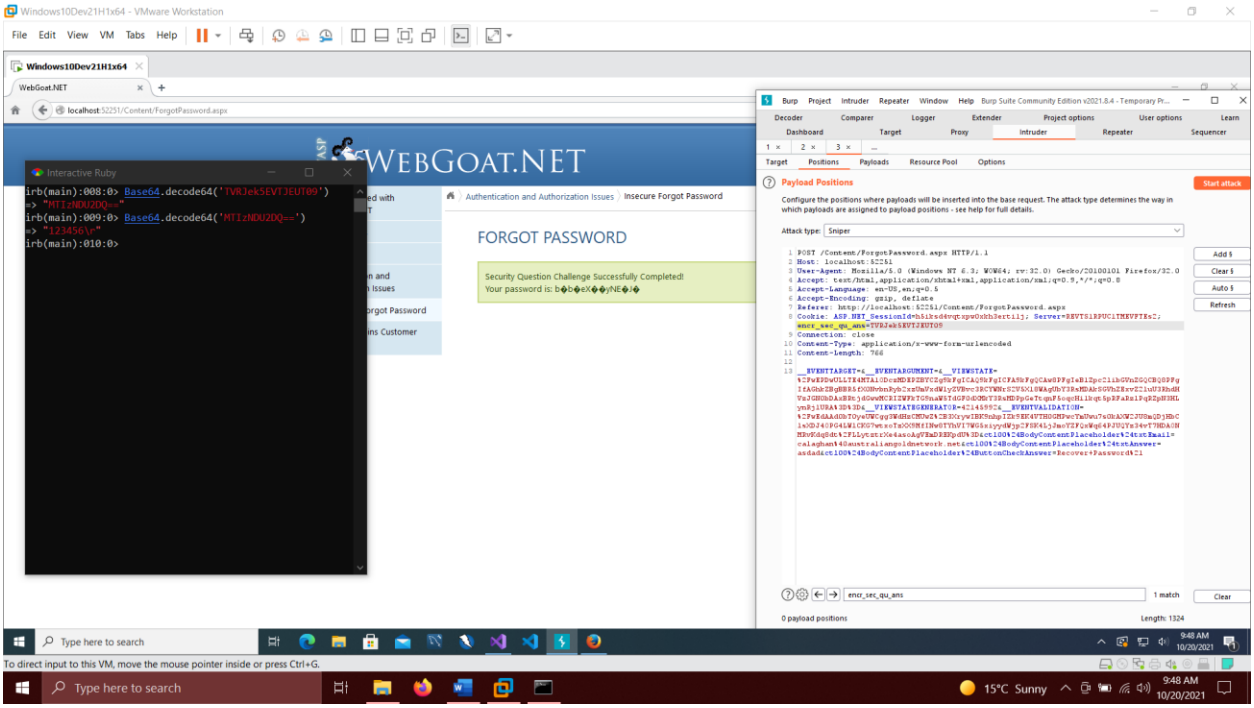


Figure – 5 - calaghan@australiangoldnetwork.net security answer recovered

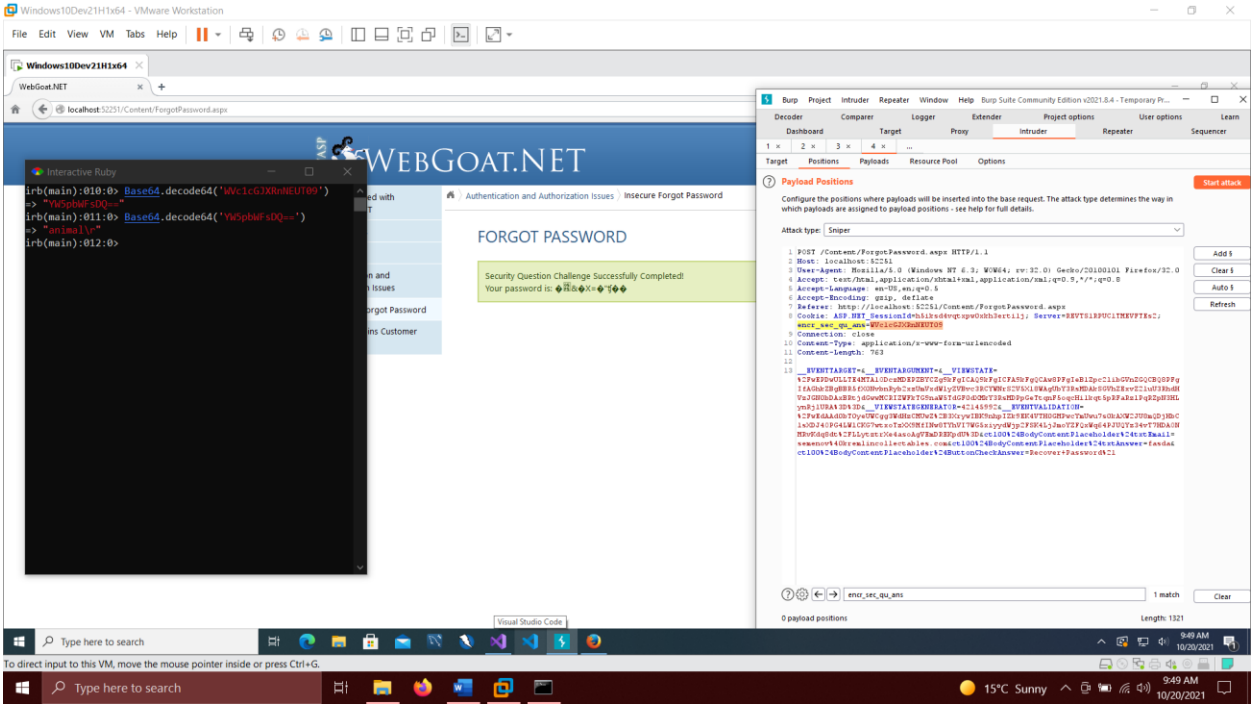


Figure – 6 - semenov@kremlincollectables.com security answer recovered

e) Provide the CWE-ID, Filename, Line Number of the weakness that is at the heart of the vulnerability.

CWE-ID(s):

Approach 1:
- CWE-307: Improper Restriction of Excessive Authentication Attempts
- CWE-319: Cleartext Transmission of Sensitive Information

Filename: Configuration
Line Number: N/A

Approach 2:
- CWE-640: Weak Password Recovery Mechanism for Forgotten Password
- CWE-261: Weak Encoding for Password

Filename: ForgotPassword.aspx.cs
Line Number: 47

f) Describe the vulnerability and what role the weakness plays in allowing the input (attack vector) to compromise the application.

Approach 1:
The unrestricted attempts on figuring out answers to security questions and cleartext transmission of credentials over the network can provide an attacker the required answer to the security question. An attacker can perform a rainbow table attack or gather information by observing the activity of victim over internet and guess the answer to security question.

Approach 2:
The password recovery mechanism is sending the correct answer to security question over the network using a weak encoding method which can easily be deciphered. The request should never contain the correct answer or at least be secured with a strong cipher/hash function.