Security Project Final Report

Student ID: **1605006** (Avijit Biswas)
Attack Tool: **Dictionary Attack and Known Password Attack**

Dictionary Attack and Known Password Attack

Introduction:

Dictionary attack and known password attack is one kind of brute force attack. These attacks are performed in offline and online methods. In our project, we demonstrate online attacks on a http server.

Steps of Attack:

We are going to build a http server with nodejs. There are basically signup and login pages. We have stored the name and hashed password of users in a database. Then we have used our dictionary and known password file to crack the password of any user.
There are three basic fundamental steps of our attack.These steps are explained below:

1. **Collecting dictionary and known password list:**
   We have collected some common english words from this link https://github.com/dwyl/english-words/blob/master/words.txt
   and it is our dictionary.txt file to succeed in the attack. Moreover, we have collected some common and popular passwords from this link https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt
   and it is our known_password.txt file to succeed in the attack.

2. **Building a website:**
   Generally, dictionary and known password attacks are performed on an online live site. But for security purposes, we have built a website hosted locally. We have used a local machine as host where we have used mySQL database to store our users' cardinals(**name and**

password). The attacker's program communicated with this server to perform a dictionary attack and known password attack.
In our database, there is only one table/schema whose name is demouser. There are 3 attributes in this schema which are id,name,password. Here, “password ” field is the hash value of the original password. We have designed basic signup and login pages to simulate our attack.

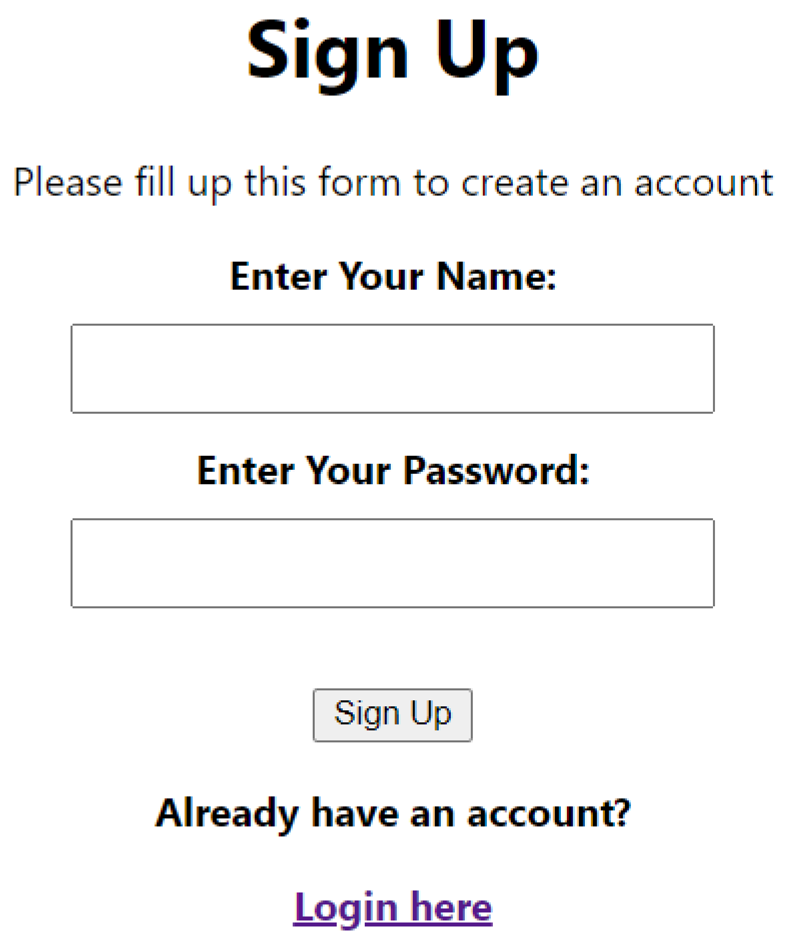


Fig: SignUp web page

We have created a user by using this signup web page. After successful signup, username and password will be stored in the database.

3. **Attacker's code by using socket programming in cpp:**

    We have used a tcp socket for establishing the connection. The code of creating a tcp socket is given below:

    socket_descriptor = socket(AF_INET, SOCK_STREAM, 0);

    Then we have configured the server by giving port no 3001 and IP 127.0.0.1(localhost).

    server.sin_port = htons(3001);

    server.sin_addr.s_addr = inet_addr("127.0.0.1");

    After the tcp connection is established, we send a http request packet to the server by POST method. For this, we have made a structure called HTTP_Packet.

    ```
    struct HTTP_Packet{
    public:
       string method;
       string targetURL;
       string version;
       string content_type;
       string content_length;
       string request_body;
       };
    ```

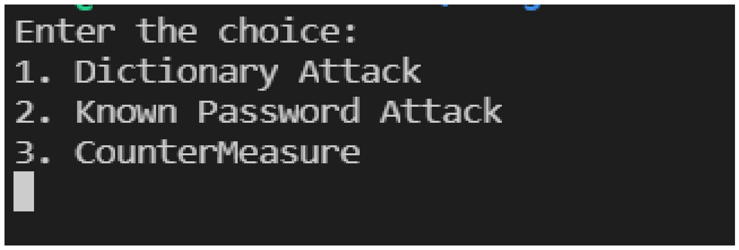
    Here, we have given the username and password as request_body and the length of this request_body is given as content_length. After sending data, the server responds back with some data and we read this data with the help of the socket. If we get status code 201, we break the loop of the reading file, print the success msg and close the tcp socket.

Reasoning of success of the attack:

The success of the dictionary attack depends on if we can guess the correct password.If the user is not using any word from the dictionary or mix-up the words, our attack can be unsuccessful.
Our attack tool can tell us for this website,if any user uses a password from the dictionary,we can detect this. But it also depends on knowing the username of the user.For this simulation purpose, we create a user and choose a password from the dictionary and create another user and choose a password from the known password file. Our attack tool can successfully find out that the password was taken from the dictionary and known password file. So,in this way our attack is successful.

Observed Output:

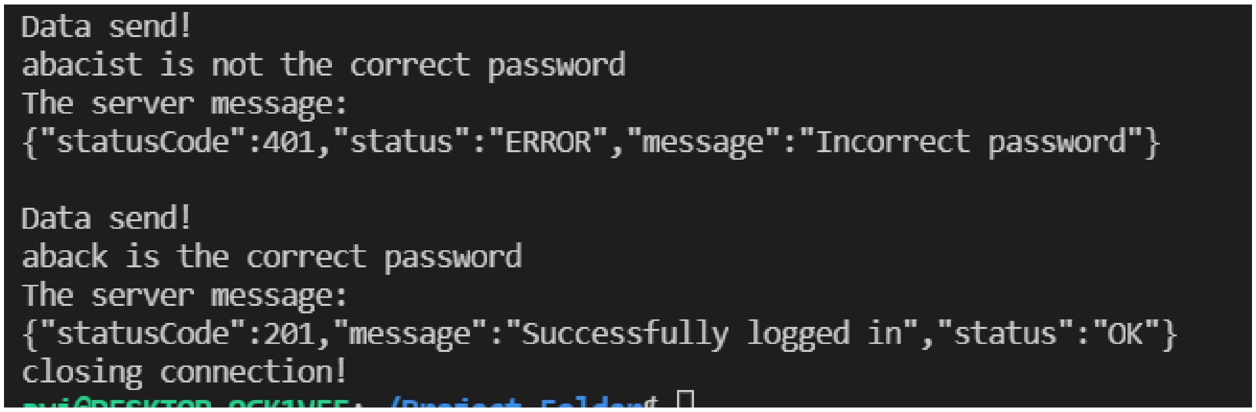
For demonstration purposes, we assume that we know the username and suppose that password is given from a dictionary and known password file. In a dictionary attack, the username is **"avijit"** and in the known password attack, the username is **"navid"**. For a dictionary attack, the password we have chosen is **"aback"** and for known password attack, the password we have chosen is **"ferrari"**.
At first whenever we run the attacker's code, the following terminal is seen:



```
Enter the choice:
1. Dictionary Attack
2. Known Password Attack
3. CounterMeasure
```

Fig: Choice of Attack and CounterMeasure

If we enter the choice 1, then a dictionary attack will be occured. Each word from the dictionary file is given as the password of the request body and the post request is sent to the server. In respect of each post request, the server responds back with a status code along with a message. If the password is wrong, the status code is 401 and if the password is correct, the status code is 201.

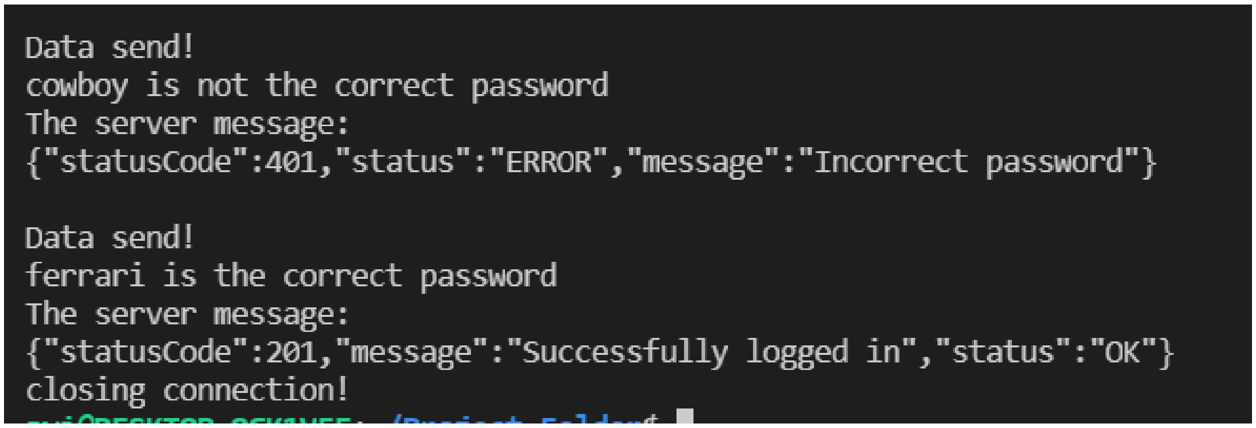


```
Data send!
abacist is not the correct password
The server message:
{"statusCode":401,"status":"ERROR","message":"Incorrect password"}

Data send!
aback is the correct password
The server message:
{"statusCode":201,"message":"Successfully logged in","status":"OK"}
closing connection!
```

Fig: Dictionary Attack

So whenever we get the right password “aback” from the dictionary, server responds back with **201** status code and by seeing this status code, we can say that our dictionary attack is successful.
On the contrary, if we enter the choice 2, a known password attack will be occured.



```
Data send!
cowboy is not the correct password
The server message:
{"statusCode":401,"status":"ERROR","message":"Incorrect password"}

Data send!
ferrari is the correct password
The server message:
{"statusCode":201,"message":"Successfully logged in","status":"OK"}
closing connection!
```

Fig: Known Password Attack

In this case, when we get the correct password “ferrari”, the server responds back with the status code **201** along with the message implying that the attacker successfully logged in the website.

Countermeasure Design:

We have designed two countermeasures and implemented one of them. First countermeasure is that we can lock the user after multiple failed login attempts such as after 5 or 10 login attempts. If the attacker continuously tries to login with all passwords from the dictionary, he will be locked after 5 or 10 failed login attempts and in this way, the attack will be unsuccessful.
Another fruitful countermeasure is to use **reCAPTCHA** in login form.

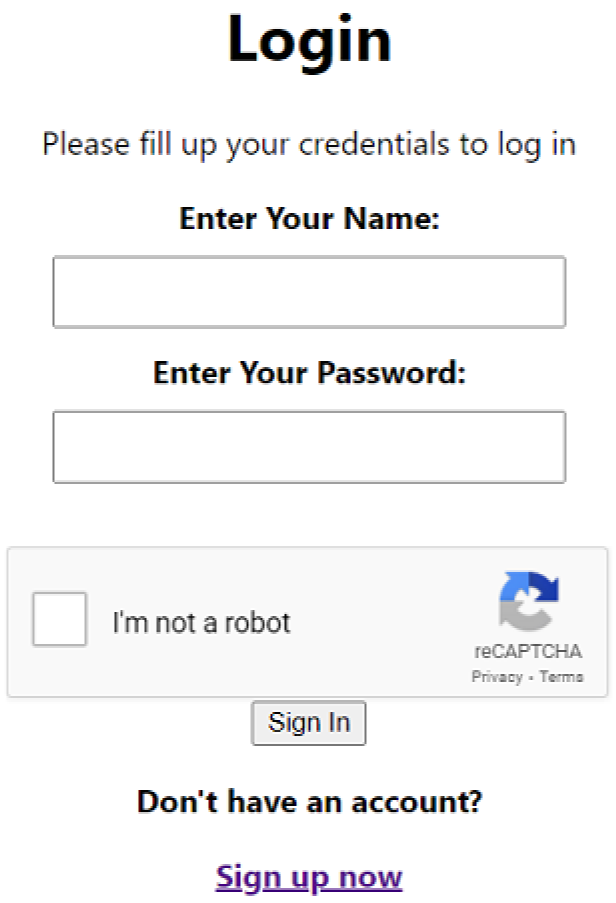
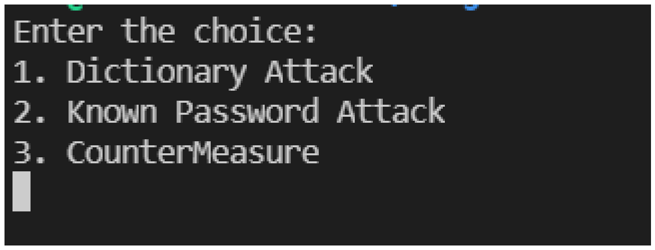


Fig: new login web page with reCAPTCHA

If the user has to fill up a reCAPTCHA during login, it must be difficult for the attacker to crack the password by dictionary and known password attack.Because,in this way, the attacker has to find the captcha solution to make a login attempt. But it is very hard. In this way, we can prevent the dictionary attack and known password attack.

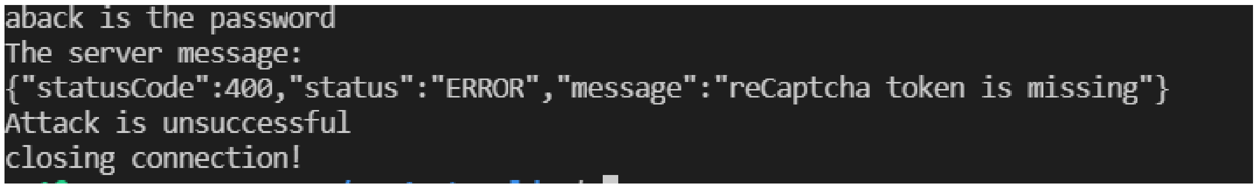
<u>Simulation of Defence Mechanism:</u>

For the demonstration purpose, we try to login the website with the correct password **"aback"** along with username **"avijit"**. After running attacker's code, we see the terminal which is given below:



```
Enter the choice:
1. Dictionary Attack
2. Known Password Attack
3. CounterMeasure
```

In the above terminal, if we enter the choice 3, the defense mechanism will be shown saying that the attack will be unsuccessful despite knowing the correct password. Due to using reCAPTCHA in the new login form, the attacker does not know the **token**(client site key and server secret key) and so can not login the website with the correct password.



```
aback is the password
The server message:
{"statusCode":400,"status":"ERROR","message":"reCaptcha token is missing"}
Attack is unsuccessful
closing connection!
```

Fig: CounterMeasure of Attack

In the above picture, we can see that the server responds back with **400** status code even though we have given the correct password **“aback”** because the reCAPTCHA token is missing. In this way, we can prevent this attack.