

# **KNOWN PASSWORD ATTACK**

**CSE-406**

**Section B**

**Group No-07**

Dibyendu Brinto Bose-1505092

Submitted to - Dr. Md. Shohrab Hossain



Department of Computer Science and Engineering  
Bangladesh University of Engineering and Technology  
(BUET)

DHAKA 1000

September 15, 2019

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Demonstration of my Tool</b>	<b>2</b>
2.1	INPUT: . . . . .	2
2.2	OUTPUT: . . . . .	3
2.3	Attack Webpage and Server setup . . . . .	3
2.4	Hashing Techniques . . . . .	3
2.5	Exploited Information . . . . .	4
2.6	List of Password . . . . .	5
<b>3</b>	<b>Reason for Work</b>	<b>5</b>
<b>4</b>	<b>Defence Mechanism</b>	<b>6</b>
<b>5</b>	<b>Future Work</b>	<b>8</b>
<b>6</b>	<b>Conclusion</b>	<b>8</b>

# 1 Introduction

**Known Password Attack** is a growing security attack for stealing vulnerable information about user and to exploit there privacy. In this Project I have implemented known password attack by knowing the **Hashing Technique** and **Hashing Value** of the password by matching the hash value with a list of **Most Common Passwords**. If the password is found then **My Attack Tool** will fetch the information by submit the form of that html page.

Details steps and implementation with snapshots of my Attack Tool is described below-

## 2 Demonstration of my Tool

I have Created a Attack Tool for Known password Attack which will take the **URL** of the webpage which attacker want to exploit and the **Hash Value** for the password and the **Hash Type**(SHA1,MD5) and will give output the **password** and the **information** after exploiting that URL if the password matches with a list of known common passwords. For the demonstration of this attack I have also made web pages using **HTML,PHP** which runs at **XAMPP** server.

### 2.1 INPUT:

- URL of the webpage
- Hash Value of password
- Hashing method

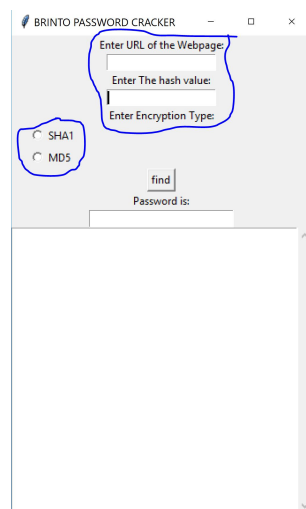


Figure 1: Inputs of Tool

## 2.2 OUTPUT:

- Password (if matches)
- Exploited Information

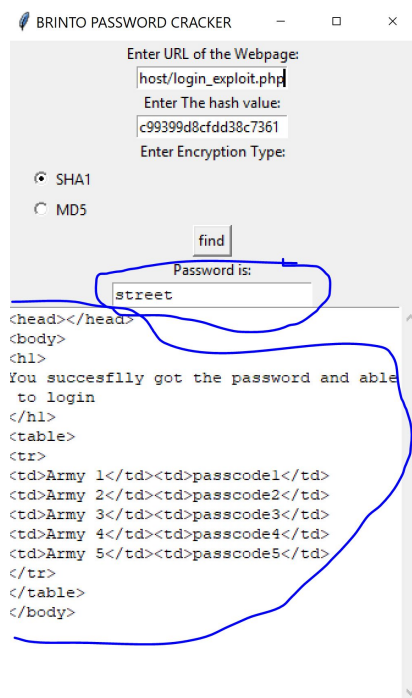


Figure 2: Outputs of Tool

## 2.3 Attack Webpage and Server setup

- As in real webpage it is tough and risky to run this attack so its good to run this attack in localhost and local webpages
- For demonstration purpose i have made *http://localhost.loginexploit.php* webpage by using **PHP and HTML**
- **MySQL** database is used in my demonstrative webpage
- this website is hosted in **XAMPP** local server

## 2.4 Hashing Techniques

- **SHA1** is the first technique that i have implemented here. it is a strong hashing for encryption of password. I have **implemented my own sha1 hash** in this project and i didn't use any build-in library. In Figure:5 the password crack of **sha1 encoding** is demonstrated. and also snapshot of the implementation is also included.

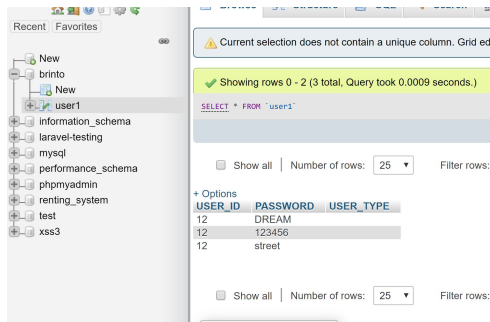


Figure 3: Database for demonstration webpage

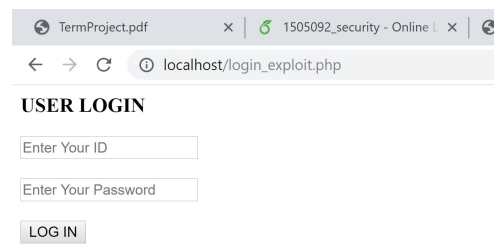


Figure 4: Demonstration webpage

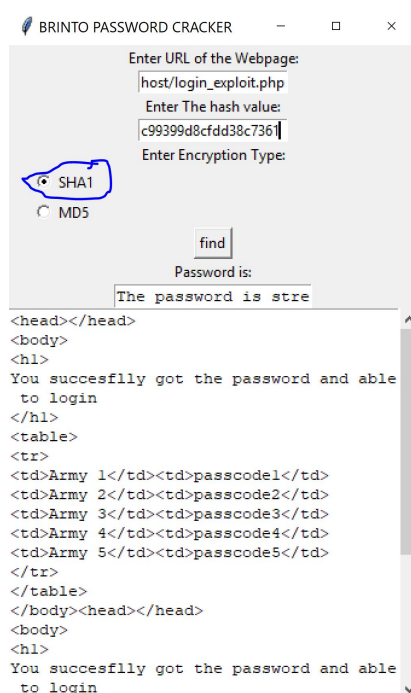


Figure 5: Exploit of Sha1

- **MD5** hashing is another strong technique for password encryption. I have also **implemented my own md5 hash** in this project and i didn't use any build-in library. In Figure:6 the password crack of **MD5 encoding** is demonstrated. Snapshot of the md5 hash class is given here-

## 2.5 Exploited Information

- After successful guessing of the password my tool also **fetch the exploited information** from the webpage.
- simple **HTML form Submission** is used here. thus the Attacker successfully get the information by the Known password attack.
- the code fragment of this part is demonstrated in figure:7

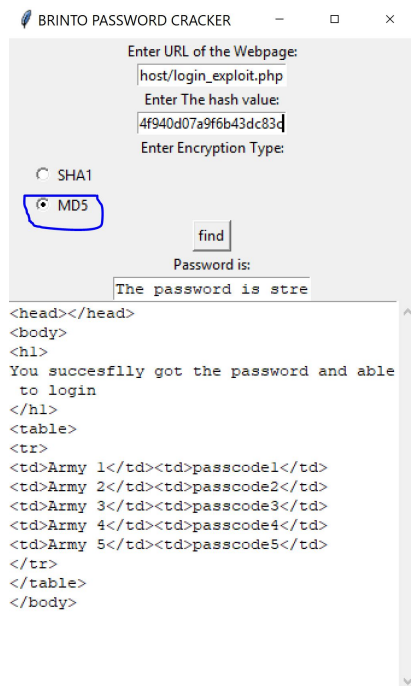


Figure 6: Exploit of MD5 encoding

```

if require == shalhash:
    br = mechanize.Browser()
    br.set_handle_robots(False)
    br.open(url)
    br.select_form(name="x")
    br["id"] = "12"
    br["password"] = str(guess)
    res = br.submit()
    content = res.read()
    with open("mechanize_results.html", "wb") as f:
        f.write(content)
    T.insert(END, content)
    T1.insert(END, str(guess))
    nff+=1

```

Figure 7: Code fragmant for fetching information

## 2.6 List of Password

List of password is fteched from a webpage <https://raw.githubusercontent.com/danielmiessler/SecLists/master/Credentials/10-million-password-list-top-10000.txt> where all the **common** and **Vulnerable** passwords ar given

## 3 Reason for Work

By making and using the above tool known pass word attack **successfully work** in this tool.there are certain of measures and steps for the successful working of this attack-

- both **SHA1** and **MD5** hashing are implemented correctly and checked with the online value for the known passwords and all the values of online and **my implementation** matches correctly.Figure:8 is demonstrating the results.

```

>string to be hashed: 'hello'

54-ELEMENT TABLE 'T' (fromsine function):

T[0]=3614090360 T[1]=3905402710 T[2]=606105819 T[3]=3250441966 T[4]=4118548399 T[5]=1200080426 T[6]=2821735955 T[7]=4249261313
T[8]=1770035416 T[9]=2336552879 T[10]=4294925233 T[11]=2304563134 T[12]=1804603682 T[13]=4254626195 T[14]=2792965006 T[15]=1236535329
T[16]=4129170786 T[17]=3225465664 T[18]=643717713 T[19]=3921069994 T[20]=3593408605 T[21]=38016083 T[22]=3634488961 T[23]=3889429448
T[24]=568446438 T[25]=3275163606 T[26]=4107603335 T[27]=1163531501 T[28]=2850285829 T[29]=4243563512 T[30]=1735328473 T[31]=2368359562
T[32]=4294588738 T[33]=2272392833 T[34]=1839030562 T[35]=4259657740 T[36]=2763975236 T[37]=1272893353 T[38]=4139469664 T[39]=3200236656
T[40]=681279174 T[41]=3936420074 T[42]=3572445317 T[43]=76029189 T[44]=3654602809 T[45]=3873151461 T[46]=530742520 T[47]=3299628645
T[48]=4096336452 T[49]=1126891415 T[50]=2878612391 T[51]=4237533241 T[52]=1700485571 T[53]=2399980690 T[54]=4293915773 T[55]=2240044497
T[56]=1873313359 T[57]=4264355552 T[58]=2734768916 T[59]=1309151649 T[60]=4149444226 T[61]=3174756917 T[62]=718787259 T[63]=3951481745

OPERATION TABLE 's':

s[0]=7 s[1]=12 s[2]=17 s[3]=22 s[4]=7 s[5]=12 s[6]=17 s[7]=22
s[8]=7 s[9]=12 s[10]=17 s[11]=22 s[12]=7 s[13]=12 s[14]=17 s[15]=22
s[16]=5 s[17]=9 s[18]=14 s[19]=20 s[20]=5 s[21]=9 s[22]=14 s[23]=20
s[24]=5 s[25]=9 s[26]=14 s[27]=20 s[28]=5 s[29]=9 s[30]=14 s[31]=20
s[32]=4 s[33]=11 s[34]=16 s[35]=23 s[36]=4 s[37]=11 s[38]=16 s[39]=23
s[40]=4 s[41]=11 s[42]=16 s[43]=23 s[44]=4 s[45]=11 s[46]=16 s[47]=23
s[48]=6 s[49]=10 s[50]=15 s[51]=21 s[52]=6 s[53]=10 s[54]=15 s[55]=21
s[56]=6 s[57]=10 s[58]=15 s[59]=21 s[60]=6 s[61]=10 s[62]=15 s[63]=21

*DBUFFER:

```

Figure 8: Matching of My implementation with original

- as i have also implemented **webpage** and used **local server(XAMPP)** so the overall html page is fetched easily. otherwise it would be much risky and troublesome for real websites.It is demonstrated in Figure:9



Figure 9: Own webpage

- if **CSRF** token cold be found in the html page of the webpage then this attack would be tough to exploit because then the id of the **form** cannot be extracted
- As HTML code is extracted correctly and **FORM-ID** is known using **inspect element** so form is submitted successfully with the cracked password

## 4 Defence Mechanism

- I have also implemented a **Defence Mechanism** for this attack.
- If **Random String** is concated with the password and stored in database then it will be much more harder to crack the hashing
- so now **different hashing value** for same **password** will be available for the attacker. and so if he has the correct hashing after that he wont be able to crack the password and wont get the valuable information
- This mechanism for defense password attack is called **password salt**

- There is also some more defence mechanism for this attack like **Random Delay**, **Blocking method**
- By the defence mechanism random hash for same password will generate to attacker. That's why password cracking will be much more tough
- That's why for a password "street" there is 2 hash value for 2 different time which is the main theme of the defense mechanism.

```
"C:\Users\brinto_dibyendu\PycharmProjects\helloworld\venv\Scripts\python.exe" "C:\Users\brinto_dibyendu\PycharmProjects\helloworld\hello.py"
street : b'71cfa8a7b88347cc8d05771d49cebb885e5d32b7b863e5efd057b8895b1c3a9d54feda5af7b20edb0630f78b6a88a03672c023aaa6a3a0c9cc8dfc44c89fe865bbd4ed5a0a49b1
street : b'e48b5f96b43927bf1a5ee075ef3fd1b4fad4d7df60f8a5a75b4d2787032b3f21ac8c697f97be0296f546ec7b93cd65314b2202149675c2bbcc34caa765c48f3a419a0947d73e7
```

Figure 10: 2 value for same password

```
function RandomPassword() {
    $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
    $pass = array(); //remember to declare $pass as an array
    $alphaLength = strlen($alphabet) - 1; //put the length -1 in cache
    for ($i = 0; $i < 5; $i++) {
        $n = rand(0, $alphaLength);
        $pass[] = $alphabet[$n];
    }
    return implode($pass); //turn the array into a string
}
```

Figure 11: Code Fragment for Defence

Options		
USER_ID	PASSWORD	USER_TYPE
12	DREAM	
12	123456	
12	street	
15	hello	
18	7	
20	brintoPJHEd	

Figure 12: Random string concated with password



## 5 Future Work

- In future more encryption techniques and more information about passwords will be used .
- If possible for real websites and central server ,this attack will be demonstrated
- for defence mechanism, blocking method like if someone try to access multiple times then that URL will be blocked. this things could be added
- If someone try to use more password then random delay will be added for slow down the process

## 6 Conclusion

This tool successfully able to crack password which is demonstrated at the targeted website and **Defence mechanism** is also demonstrated here. In future more hashing technique will be included and more defence mechanism will be included. Overall my tool successfully work for cracking and making defence for password.

Overall Summary of the attack tool and defence mechanism is-

- This attack tool will exploit the information after knowing the hashing information from the website of given URL if the password matches
- If the password doesn't match then it will store that
- Defence mechanism is selected **password salt** as it will add random string to the password to prevent password attack.