

SHA1 Password Exploitation in Python

A screenshot of a Mac desktop showing a terminal window. The terminal window has tabs for 'next.c' and 'SHA1.py'. The 'SHA1.py' tab contains the following Python code:

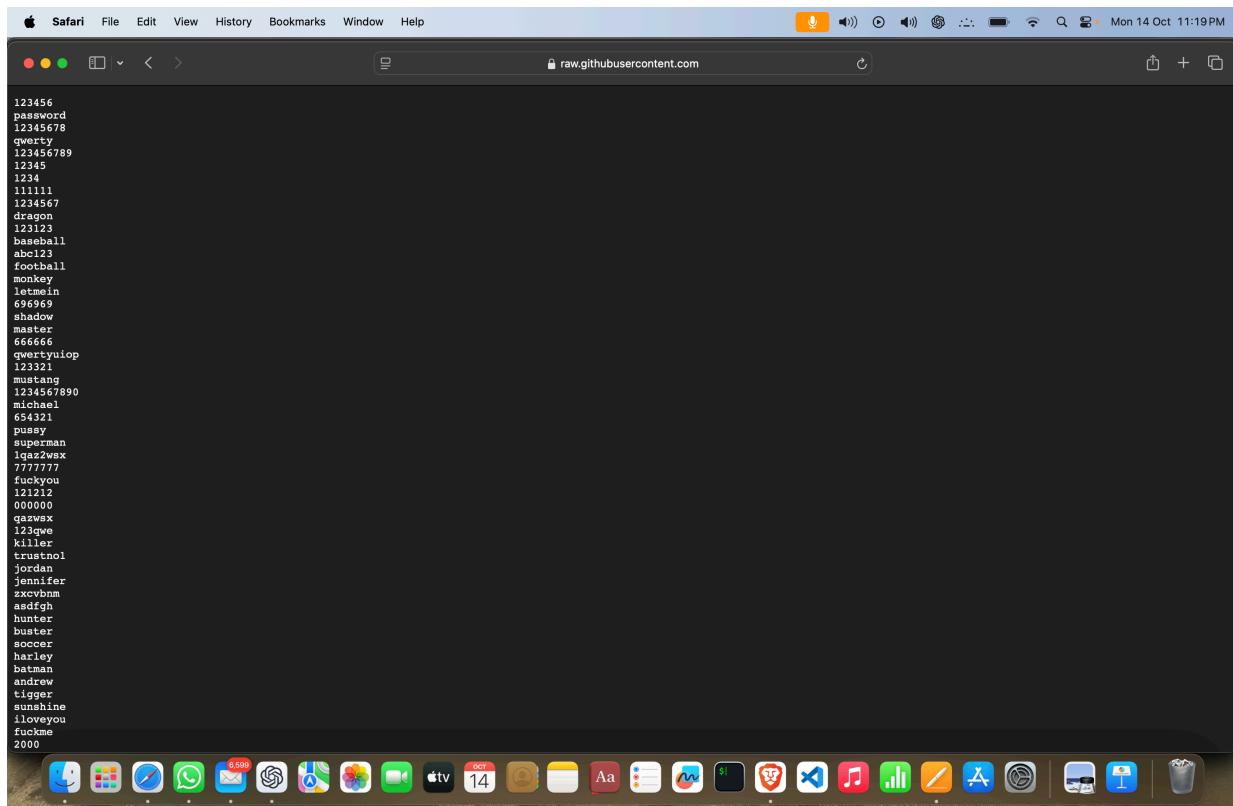
```
1 def main():
2     user_sha1 = input("enter the SHA1 to Crack")
3     cleaned_user_sha1 = user_sha1.strip().lower()
4
5 if __name__ == '__main__':
6     main()
```

The terminal output shows the command 'arkapravapanigrahi@Arkapravas-MacBook-Air next.c % 98]'.

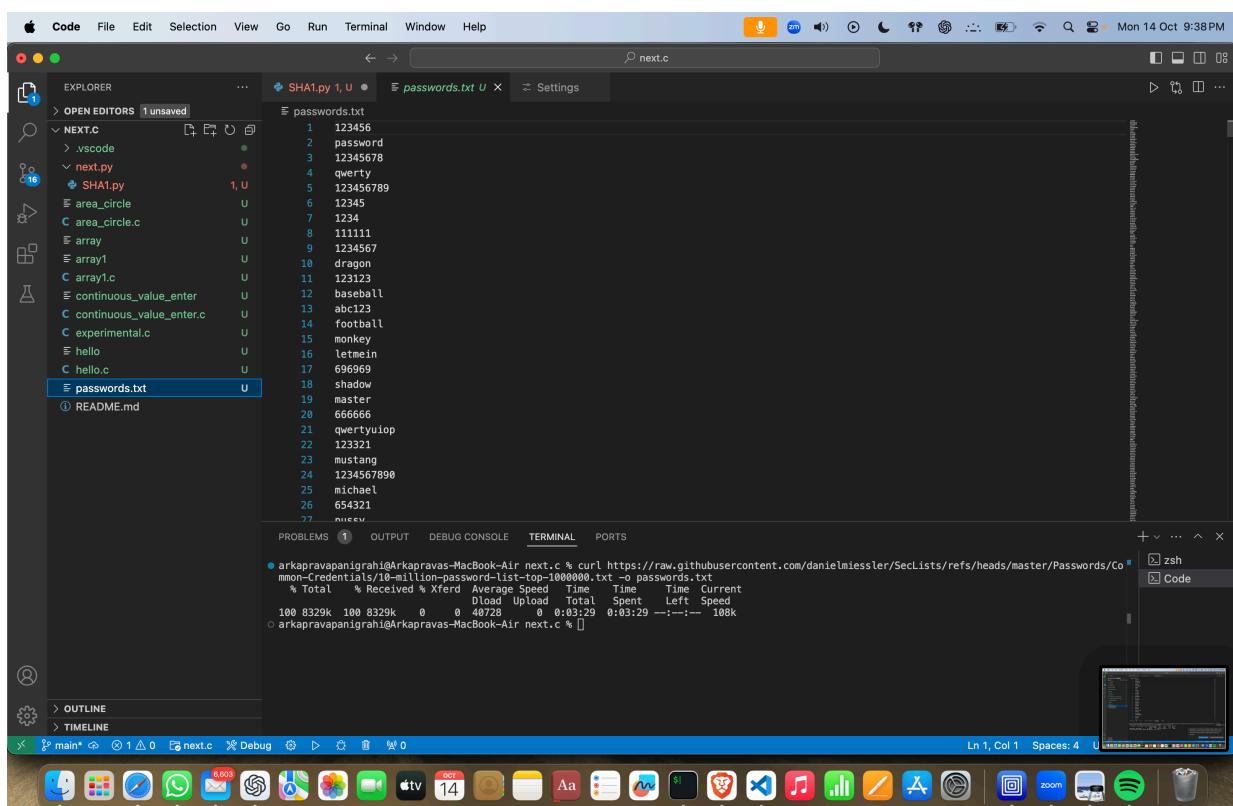
1. To exploit SHA-1 hash vulnerabilities in passwords using Python , first we have to write a code that is a SHA-1 hash cracker that takes a user-provided SHA-1 hash and compares it against the SHA-1 hashes of passwords in a file (passwords.txt). It reads each password from the file, hashes it, and checks if it matches the given hash. If a match is found, it prints the corresponding password; otherwise, it reports that the password could not be found.

A screenshot of a Mac desktop showing a GitHub search results page. The search query is 'SecLists / Passwords / Common-Credentials / 10-million-password-list-top-1000000.txt'. The results show a single file named '10-million-password-list-top-1000000.txt' by 'LethargicLeprechaun'. The file size is 8.13 MB and was last updated 74c24b5 · 4 years ago. A note says '(Sorry about that, but we can't show files that are this big right now.)'

2. We search for a password file in the internet that contains many passwords for testing



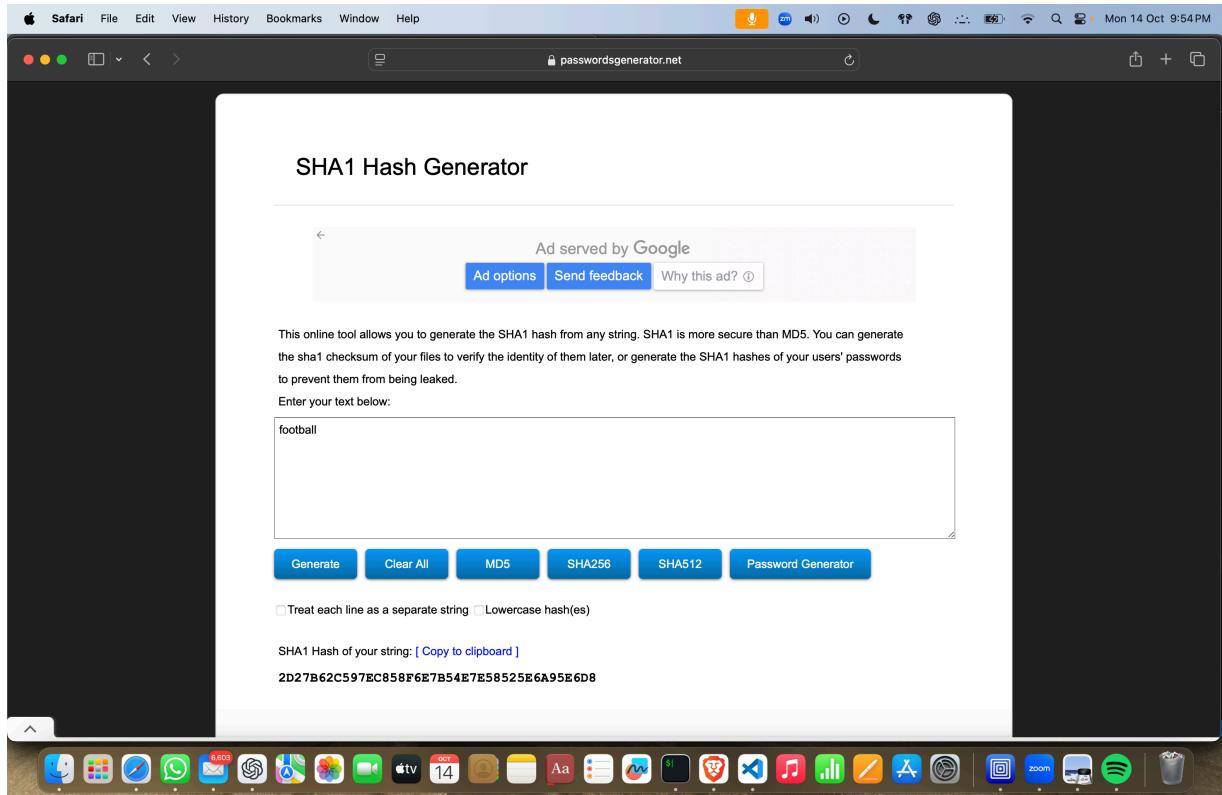
3. We can select any one password from the password list , here we select password football for the testing purpose.



4. Then we download the password list by using curl command in the terminal of VS code

The command used - curl <https://raw.githubusercontent.com/danielmiessler/SecLists/refs/heads/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt> -o passwords.txt

Here we use -o command to rename the file as password.txt



5. We go to the hash generator to generate a SHA1 hash for the selected password- football for verification purpose

```

import hashlib
def convert_text_to_sha1(text):
    digest = hashlib.sha1(
        text.encode()
    ).hexdigest()

    return digest

def main():
    user_sha1 = input("Enter the SHA1 to Crack: ")
    cleaned_user_sha1 = user_sha1.strip().lower()

    with open("./passwords.txt") as f:
        for line in f:
            password = line.strip()
            converted_sha1 = convert_text_to_sha1(password)

            if cleaned_user_sha1 == converted_sha1:
                print(f"password has been found: {password}")
                return

    print("could not find the password")

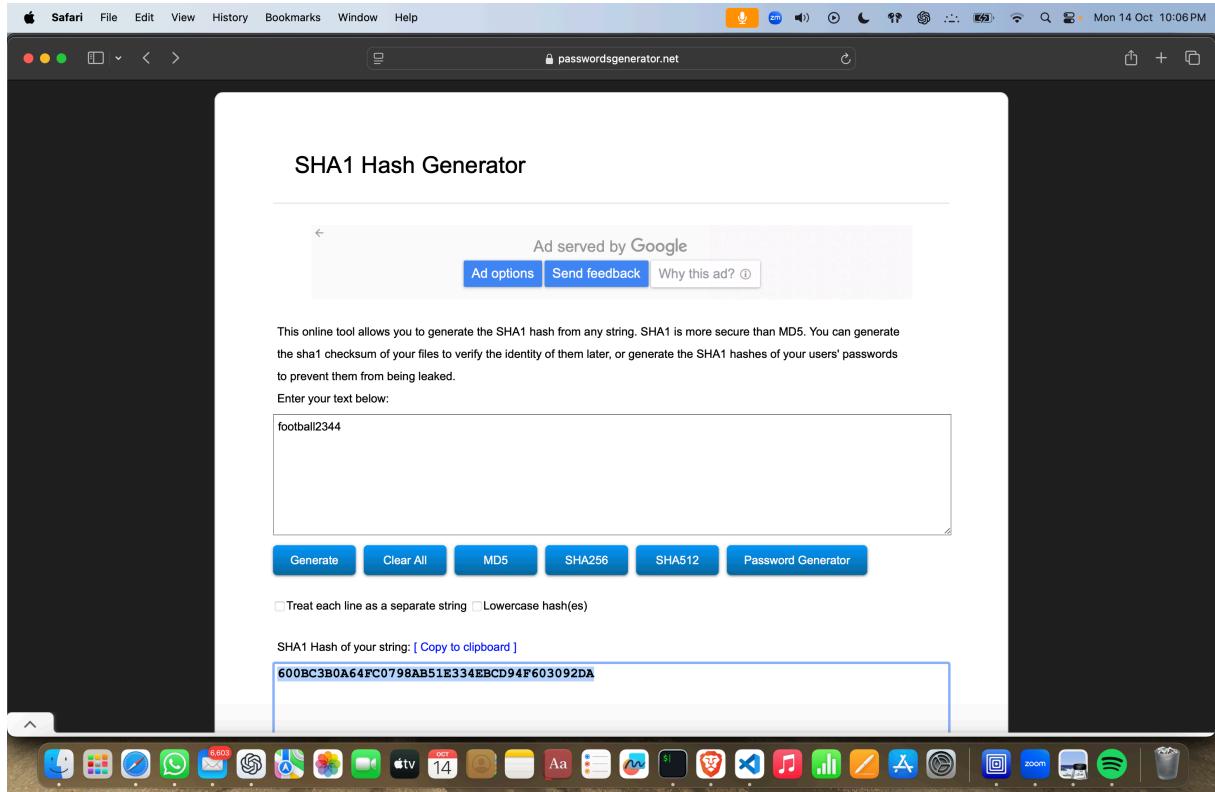
if __name__ == '__main__':
    main()

```

6. Next we run the code and we input the SHA1 hash of football that we got from the hash generator website

The hash is verified with all the passwords present in the password list and finally the hash matches with password- football

This signifies that our code is working properly



7. Now we are going to reverse verify it , we are going to generate SHA1 hash of a password that is not present in the password list

Hare it is football2344 , this password is not present in password list

```

next.py > SHA1.py > main
1 import hashlib
2 def convert_text_to_sha1(text):
3     digest = hashlib.sha1(
4         text.encode()
5     ).hexdigest()
6
7     return digest
8
9 def main():
10     user_sha1 = input("Enter the SHA1 to Crack: ")
11     cleaned_user_sha1 = user_sha1.strip().lower()
12
13     with open("./passwords.txt") as f:
14         for line in f:
15             password = line.strip()
16             converted_sha1 = convert_text_to_sha1(password)
17
18             if cleaned_user_sha1 == converted_sha1:
19                 print(f"password has been found: {password}")
20                 return
21
22
23
24
25
26     if __name__ == '__main__':
27         main()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- arkapravapanigrahi@Arkapravas-MacBook-Air next.c % python3 -u "/Users/arkapravapanigrahi/Documents/GitHub/next.c/next.py/SHA1.py"
 enter the SHA1 to Crack: 2D27B62C597EC858F6EB54E7E58525E6A95E608
 password has been found: football
- arkapravapanigrahi@Arkapravas-MacBook-Air next.c % python3 -u "/Users/arkapravapanigrahi/Documents/GitHub/next.c/next.py/SHA1.py"
 enter the SHA1 to Crack: 600BC3B0A64FC0798AB51E334EBCD94F603092DA
 could not find the password

8. We are going to run the python code again and we are going to enter the SHA1 hash of the password football2234. The code gives a output "could not find the password" , as the particular password football2344 is not present in the password list

This proves that our code for SHA1 password exploration in python is working properly.