Rainbow Hacking for SHA-256 of Numbers

Author: Mohsen Moghaddas

Course: Beginner Python Programming

Instructor: Amir Emad(Jadi) Mirmirani

Abstract

This paper discusses the implementation of a Python program designed to crack SHA-256 hashed passwords for numeric inputs. The project leverages a rainbow table approach, precomputing and storing a range of numeric values along with their corresponding SHA-256 hash values. The primary objective is to demonstrate a practical application of Python programming skills in the realm of password security and cryptography.

Introduction

In the field of cybersecurity, password hashing is a fundamental practice to safeguard user credentials. One widely-used hashing algorithm is SHA-256, which is renowned for its cryptographic strength. However, in this project, we explore a scenario where an attacker attempts to reverse SHA-256 hashes of numeric passwords. Our project's primary aim is to create a Python program that can efficiently crack SHA-256 hashed numeric passwords.

Methodology

Our approach involves the use of a rainbow table, a precomputed set of values and corresponding hash results. This table allows us to perform a quick lookup and retrieve the original password for a given SHA-256 hash. In Python, we use the hashlib library for the SHA-256 hash computation and storage of the hashed values.

1. **Generating Rainbow Table**: We generate a rainbow table for numeric values in the range of 1000 to 9999. For each numeric value, we calculate its SHA-256 hash and store the pair in a dictionary, where the hash value serves as the key and the numeric value as the value.

```
for i in range(1000, 10000):
    sha256_dic[hashlib.sha256(str(i).encode()).hexdigest()]
= i
```

2. **Cracking SHA-256 Hashes**: To crack SHA-256 hashes provided in an input file, we read the hashes and use the rainbow table for lookups. For each hash found in the table, we retrieve the corresponding numeric value and store it in a list along with the username associated with the hash.

```
for row in reader:
    list_new.append((row[0], sha256_dic[row[1]]))
```

3. **Outputting Results**: Finally, we write the cracked password-user pairs to an output file.

```
with open(output_file_name, 'a') as f:
    for x, y in list_new:
        f.write("%s,%s\n" % (x, y))
```

Results

Our program successfully cracked the SHA-256 hashed passwords and provided the original numeric values for the given hashes. The project demonstrates the efficiency of rainbow tables in password recovery for simple numeric passwords. However, it's important to note that this method is not applicable to more complex, random, or alphanumeric passwords, where brute-force attacks or dictionary attacks would be less efficient.

Conclusion

In this Python programming project, we developed a rainbow table-based approach to crack SHA-256 hashed numeric passwords. The successful implementation showcases the importance of strong password policies and the limitations of this specific attack method. While this project serves an educational purpose, it highlights the critical need for robust hashing algorithms and password security practices in real-world applications.

References

1. Python Documentation: hashlib - Secure Hash and Message Digest Algorithms. (n.d.). Retrieved from https://docs.python.org/3/library/hashlib.html
2. Ferguson, N., Schneier, B., & Kohno, T. (2010). "Cryptography Engineering: Design Principles and Practical Applications." Wiley.
3. Matyas, S. M. (1984). "Hash Functions for Data Integrity." In Advances in Cryptology—CRYPTO ’84 (pp. 335-347). Springer.
4. Rainbow Tables. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Rainbow_table