Assignment 2: Cracking Hashes and Exploring Diffie-Hellman Key Exchange Vulnerabilities

January 19, 2025

Part 1: Cracking Hashes with and without Salt

Objective

The objective of this part of the assignment is to help you understand the vulnerabilities of hashing algorithms by performing attacks on salted and unsalted hashes. You will write Python programs to crack MD5, SHA1, and SHA256 hashes, both with and without salts, and analyze the effectiveness and difficulty of these attacks.

Motivational Story

Imagine you are part of a cybersecurity team in a prominent organization. Recently, the organization's database was hacked, and the attackers extracted password hashes. These hashes are now available in six files (unsalted and salted). Your mission is to analyze the security of these hashes by using a publicly available password list containing at least 5 million passwords. Your task is to uncover the actual passwords used in the organization by cracking both unsalted and salted hashes. Through this exercise, you will determine which type of hash is easier to crack, explain why, and provide insights into the security measures that can be employed to protect sensitive information.

Files Provided

- md5hashes.txt
- sha1hashes.txt
- sha256hashes.txt
- md5saltedhashes.txt
- sha1saltedhashes.txt
- sha256saltedhashes.txt

Task 1: Cracking Unsalted Hashes

1. Write a Python script (`attack1.py`) that:
   - Loads the target hashes (MD5, SHA1, and SHA256) from the provided files.
   - Uses an online list of commonly used passwords containing at least 5 million entries as a dictionary. For each password, generate a hash using MD5, SHA1, and SHA256, and check if this hash matches any of the hashes in the provided files. Output the cracked passwords.
2. Analyze the time taken for cracking the unsalted hashes. Use `time` to measure the time taken for each algorithm.

Task 2: Cracking Salted Hashes

1. Write a Python script (`attack2.py`) that:
   - Loads the salted hashes (MD5, SHA1, and SHA256) from the provided files.
   - Extracts the salt and hash from each salted entry (format: `salt:hash`).
   - Hashes each password from the dictionary, combines it with the salt, and compares it to the stored salted hash.
   - Outputs the cracked passwords.
2. Analyze the time taken for cracking the salted hashes.
3. Explain how salted hashes increase complexity in terms of security.

Report Task

Write a report demonstrating your understanding of the following concepts:

1. The challenges of cracking salted hashes and how salts increase security.
2. The reasons why MD5 and SHA1 are considered insecure, including their vulnerabilities and the advantages of using more secure algorithms like SHA-256 or SHA-3.
3. The limitations of relying solely on hash cracking, considering factors like rate-limiting, cryptographic protections, and hash iteration complexity.

Ensure your report is well-structured and clearly explains each concept.

Deliverables

1. Python scripts (`attack1.py` and `attack2.py`).
2. Report PDF

Part 2: Exploring Diffie-Hellman Key Exchange Vulnerabilities

Objective

The objective of this part of the assignment is to explore the vulnerabilities in the Diffie-Hellman key exchange mechanism by simulating a secure communication scenario. You will interact with a server (oracle) that implements a vulnerable version of Diffie-Hellman and attempt to uncover the private key used by the server.

Background

Diffie-Hellman key exchange is a method used to securely exchange cryptographic keys over a public channel. Despite its widespread use, certain configurations, such as the use of small prime numbers, make it susceptible to attacks like brute-forcing the private key. Additionally, the protocol is vulnerable to man-in-the-middle (MITM) attacks if not properly authenticated.

In this part of the assignment, you will act as a client and interact with a server (oracle) that uses a vulnerable configuration of Diffie-Hellman. Your task is to exploit the vulnerabilities to deduce the server's private key and simulate a man-in-the-middle attack.

Assignment Tasks

Part 1: Understanding the Server(oracle)

1. The server generates Diffie-Hellman parameters (prime P and generator G) based on your unique entry number.
2. The server sends Public Key (P, G) to client (you) based on your entry number.
3. We have provided IP address and port number to use in client script to connect with server.

Part 2: Client Implementation

1. Write a client program that connects to the server.
2. Send your entry number to the server to receive the corresponding P and G.
3. Implement the Diffie-Hellman key exchange to compute the shared secret.
4. Analyze the server's response and attempt to deduce the server's private key.

Part 3: Exploiting the Vulnerability

1. **Brute-Force Attack:** Utilize the small prime number to attempt a brute-force attack on the server's private key.
2. **Man-in-the-Middle (MITM) Attack:** Simulate a man-in-the-middle attack by intercepting and modifying the key exchange messages between the client and the server.
3. You have to write a client code and a server code, the server will act as the MiM, between the Oracle that we provide and your client.
4. Document your approach and findings for both attacks.

Deliverables

1. **Client Code and MiM code:** Your implementation of the client program and the malicious server acting as MiM.
2. **Report:** A detailed report documenting your methodology, analysis, and the private key you uncovered. Include code snippets and any challenges faced during the process.
3. **Results:** Submit the deduced private key, details of the man-in-the-middle attack, and any insights into the vulnerabilities.

Instructions for Submission

1. Upload your client code and report in a ZIP file to Gradescope.
2. Ensure your entry number is included in the report and used in your client program.

Deadline

All submissions must be made by 27th January 2025. Late submissions will incur a penalty as per the course policy.