Lab 3: MD5 and Rainbow Tables

50.020 Security

Hand-out: February 14
Hand-in: February 21, 9pm

1 Objective

- Hash password using MD5
- Crack MD5 hashes using brute-force and rainbow tables
- Strengthen MD5 hash using salt and crack again the salted hashes

2 Hashing password using MD5

- To warm up, compute a couple of MD5 hashes of strings of your choice
  - Observe the length of the output, and whether it depends on length of input
- To generate the MD5 hash using the shell, try `echo -n "foobar" | md5sum` to compute hash of `foobar`
- To generate the MD5 hash using python, use `import hashlib` module and its `hexdigest()` function.

3 Brute-Force and dictionary attack

- For this exercise, use the fifteen hash values from the `hash5.txt` (newline separated)
- Create a `md5fun.py` Python 3 script to find the corresponding inputs that create the challenge hash values. Try the following two approaches:
  - Brute-Force attack: Find the corresponding input by computing the hash values for each possible combination. You need only to consider passwords with 5 lowercase and or numeric characters."
  - Dictionary attack: To help reduce the search space we provide a dictionary with newline separated common words in `words5.txt`. For this approach, notice that a resulting hash may be generated from the original word by its permutation and including numbers to it, e.g. hello -> lh78o"
  - For each method, take note of the computation time of your algorithm to reverse all fifteen hashes. Consider the `timeit` python module: `https://docs.python.org/3/library/timeit.html`

4 Creating Rainbow Tables

- Install the program rainbrowcrack-1.7-linux64.zip (`http://project-rainbowcrack.com/rainbowcrack-1.7-linux64.zip`).

  ```
  $ unzip rainbowcrack-1.7.1-linux64.zip
  ```

- Use **rtgen** (`http://project-rainbowcrack.com/generate.htm`) to generate rainbow tables with the characteristics shown below.
  - Five characters input
  - Only lower case letters and numeric characters.
  - Chain length is 3800.
  - Chain number is 600000.
  - Part index is 0.
  - Table index is 0.
- Use **rtsort** (`http://project-rainbowcrack.com/generate.htm`) to sort the rainbow table to make searchable by **rcrack**.
- Use **rcrack** (`http://project-rainbowcrack.com/crack.htm`) to crack the list of fifteen passwords from `hash5.txt`.

5 Salting

- Extend your Section 3 Python 3 script to append one random lowercase character as salt value to all the elements of the list of passwords you recovered in the previous part of this exercise. Rehash the password using MD5, and store the newly hashed passwords and their salt values into a new file called `salted6.txt` (remember to store the new password as well, maybe in a `pass6.txt` file). The functional definition of our salt strategy is the following:

  ```
  saltedhash(password) = hash(password||salt)
  ```

- Generate a new rainbow table using **rtgen** (with new parameters) to break the hash values. As before, sort the table using **rtsort**.
- Compare the timing of the new table generation and lookup vs the previous values
- Try to break as many salted hashes as possible.
- Prepare a 1-page writeup explaining the differences between salted and non salted rcrack strategies and compare the timings.

6 Hash breaking competition

- We provide a list of hashes in `hashes.txt`
  - They are of different difficulty - not all are equally hard
  - There are no easy rules about length or characters allowed any more!
- Implement an optimized script and try to reverse as many of those hashes as possible. You can also use other tools as you want.
- Write a one page explanation on the approach you use to crack those passwords.
- Submit the answers as a csv file containing two columns. The first column is the md5 hash of the password you break, and the second column is the plain text password.

7 Hand-in

- Submit your `md5fun` script which breaks the supplied `hash5.txt`, generate the salted hashes and the relevant files. Put your username and mention the timings in your header.
- Prepare the writeup as explained in Section 5.
- For Section 6, submit the found hard hashes together with writeup and code in a zip file.