Password Recovery Lab

CMSC 426/626 – Computer Security

Password-Based Authentication

The security of an operating system begins with establishing the identity of its users. By far the most common method for authenticating users is the use of passwords. When a users is enrolled in a system, they choose – or are given – a user name and password. To log in, the user provides both her user name and password, and the system determines whether the correct password was entered for the given user name.

In most cases, the system does *not* keep a list of the users' passwords, but rather stores a hash of each user's password (e.g. using SHA-256 or MD5). For example, suppose a particular system stores MD5 hashes of users passwords; then if my user name is `topha` and my password is `tophatime`, the user database would contain a line like:

```
topha:a715b1df45a28985727c8d40542630c1
```

Now, even if an attacker acquires the password file, he or she will not immediately learn my password or the passwords of the other users. Unfortunately, if users pick poor quality passwords, the attacker may still be able to recover many of the passwords.

The RockYou Compromise

RockYou is a developer of games for social media. In 2009, hackers acquired the password list for 32 million RockYou users – and the passwords were *not* hashed. The entire list was made public, providing a unique look at how users actually choose passwords. The ten most frequently used passwords were:

```
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
```

Not only does the RockYou compromise show us how weak many passwords are, but it also serves as a *dictionary* for password guessing attacks. That is, the 14 million unique passwords from RockYou are a pretty good set to try if we want to attack a system.

Basic Password Guessing Attack

We will use the RockYou list to implement our own off-line password guessing attack. By *off-line* we mean, an attack against compromised password hashes, as opposed to an *on-line* attack, in which we actually try to login to a system remotely. So, suppose we have one or more target hashes and we know what hash algorithm was used to produce them. Then we want to hash every password in the RockYou list and see if any of them match our targets.

To implement our attack, we first need to know how to compute hashes in Python. The following will compute the SHA-256 hash of the string `tophatime`:

```
>>> from Crypto.Hash import SHA256
>>> pwd = 'tophatime'
>>> sha2 = SHA256.new(pwd)
>>> print sha2.hexdigest()
```

Similarly, the MD5 hash can be computed as follows:

```
>>> from Crypto.Hash import MD5
>>> pwd = 'tophatime'
>>> md5 = MD5.new(pwd)
>>> print md5.hexdigest()
```

Verify that you can correctly compute the SHA-256 and MD5 hashes in Python by computing both hashes for the password `123456`, the most frequently used password in the RockYou list.

The SHA-256 hash of `123456` is:

```
8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
```

The MD5 hash of `123456 is:`

```
e10adc3949ba59abbe56e057f20f883e
```

Now, suppose you have have recovered a password file from a system that you know uses the MD5 hash algorithm and you see that the administrator's password hash is:

```
4f59a144a9c2d932f95a3c6160698c5f
```

You will try to recover the password by hashing the passwords in the RockYou list and seeing if any match the target hash. To do so, you must be able to read the file of passwords one line at at time, hash each line, and compare the result to the target hash. To get you started, here is some code that will open the file and read the passwords one at a time:

```
>>> infile = open('rockyou.100000.txt')
>>> for line in infile:
...       line = line.rstrip('\n') #remove newline
...       # Now compute the hash of line
...       # Compare the hash with the target
...       # Print the password if the hashes match
```

Note 1: You must use `rstrip()` to remove the newline character – if you skip this step, you will not find the correct password.
Note 2: in the sample code, we used the file `rockyou.100000.txt` which contains only the top 100,000 passwords from the RockYou list.

EXERCISE 1: Use the first 100,000 items in the RockYou list (contained in file `rockyou.100000.txt`) to find the administrator's password. You know that the system uses an MD5 hash.

A More Efficient Attack

Although our attack works, it is not terribly efficient, especially if we have multiple target hashes – every time we run the attack, the hashes of the passwords have to be re-computed. It would make more sense to compute the hashes once, save them, and re-use them.

The file `rockyou.100000.hash` contains the SHA-256 hashes and corresponding password for the first 100,000 passwords in the RockYou list. There are several ways we could use this list efficiently, but one in particular is to load the hashes and passwords into a Python dictionary. A *dictionary* is the same thing as a hash table or associative array in other languages: it is an array that is indexed by strings rather than position.

We can load the RockYou passwords and hashes into a dictionary as follows:

```
>>> infile = open('rockyou.100000.hash')
>>> pwtable = {}  # an empty dictionary
>>> for line in infile:
...         # remove \n and split into hash and passwor
...         fields = line.rstrip('\n').partition(' ')
...         # add the hash as a keyword with the password
...         # as its value
...         pwtable[fields[0]] = fields[2]
```

Now we can easily see if any given password hash is in our list:

```
>>> target =
'c09e3688481523bd55044838800156be38126d5a01a58ba04094cdce3002e023'
>>> target in pwtable
True
>>> print pwtable[target]
'4peace'
```

EXERCISE 2: You have gained access to the following password file:

```
root:c44a8bae970ecc11fb01d11be9886377c1ac9b8d2ff450d5ebefc5b78eaa95ef
sam:95c0d6e7a0e1ecf1dbd33ed89fedff8b1ffb3dbfb7c6288a6c47f0bec35e3124
topha:2415224ff13cef4a85213e06958fa0c5393e83fb631bf75c4fc38773e4fea1f5
sally:035440b51fb1a8a4f910567d9f7292f6fb55230f037cf0be7b8c16c649e416da
```

Recover as many of the passwords as you can. Which users' password could you not recover? Why couldn't you recover it?

Salted Hashes

The reason we can perform the basic password guessing attack so efficiently is that a given password always hashes to the same value. Thus I can build a dictionary once and use it over and over to recover different passwords. *Salting* is a technique that makes it more costly for an attacker to guess passwords. Rather than hashing just a user's password, we hash the password along with a small amount of random information, called the *salt*. The salt is different for each user, so even if several users have the same passwords, their hashes will be different. Also, an attacker needs to know a users's salt before they can attempt to guess their password.

Suppose now you have acquired a salted password file. The salt values are given in the file and the hash value is just the MD5 hash of the salt concatenated with the password:

```
root:6954222:d98f1c1410daefd5a7efa893ae7bf237
sam:1258244:e8c101595f2701d59626b68d9368b592
topha:11030922:40f1b3e282ca6f3a4f53a856d742760b
sally:13064284:75b363a560580edc76b0451c2b3a6e8b
```

Now to try recover one password I have to run through the entire password list:

```
>>> salt = '6954222'
>>> target = 'd98f1c1410daefd5a7efa893ae7bf237'
>>> infile = open('rockyou.100000.txt', 'r')
>>> for line in infile:
...       line = line.rstrip('\n')
...       md5 = MD5.new(salt+line)
...       if md5.hexdigest() == target:
...            print 'password is: ' + line + '\n'
...            break
password is: timberlake
```

EXERCISE 3: Recover the remaining three passwords from their salted hashes.

Lab Assignment

Your assignment for this lab is to recover as many passwords as possible from a given set of hashes *using only code that you have written*. Using any password cracking package — commercial or open source — will be considered a violation of course Academic Integrity policies.

The target hashes are provided, in Python format, in the file `hashes.py`. Within the file, they are divided into three groups:

1. Unsalted hashes of passwords based on the RockYou dictionary
2. Salted hashes of passwords based on the RockYou dictionary
3. Hashes of five-character random passwords (*only needed by Grad Students*)

All of the hashes have been computed using the SHA-256 implementation in the PyCrypto Python library. For the salted hash data, the salt and the hash are separated by a semicolon, with the salt given first. To compute the salted hashes, the salt and password were combined by concatenating the salt and the password (e.g. `sha.update(salt + pw)`).

For both the salted and unsalted hash groups, some of the passwords are straight from the first 100,000 entries of the RockYou dictionary, some are words from RockYou with a digit appended or prepended, and some are concatenations of two RockYou words. *In all cases, only words from the first 100,000 entries of the RockYou dictionary have been used*.

Lab Submission and Grading

Your lab must be submitted on Blackboard and should include the following:

1. A *brief* write-up
2. The password recovery code that you wrote (Python, Java, C, or C++)

The write-up must describe your approach to the password recovery problem and include all recovered passwords. The lab can be completed entirely in Python, and Python is preferred, but you may submit code in Java, C, or C++. In particular, you may choose to use a compiled language for efficiency reasons.

Grading of the lab will depend on both the number of passwords recovered *and* a review of your code. There is a minimum number of passwords you must recover to earn an A, B, or C:

Numer of Passwords Recovered	Highest Possible Grade	Rationale
5	C	Among the group one and two hashes, there are five for which the passwords is a RockYou word. You must recover these five passwords to receive a C.
7	B	Among the group one and two hashes there are six for which the password is either a RockYou word with a digit appended or prepended or the concatenation of two RockYou words. You must recover at least **two** of these more difficult passwords to receive a B.
9	A	Among the group one and two hashes there are six for which the password is either a RockYou word with a digit appended or prepended or the concatenation of two RockYou words. You must recover at least **four** of these more difficult passwords to receive an A.
10 - 11	Extra Credit	Extra credit will be granted for recovering additional passwords.

GRADING CRITERIA FOR UNDERGRADUATE STUDENTS (CMSC 426)

Important: These are *maximum* grades. Grades may be reduced if the code submitted does not support the claimed password recoveries.

Additional Assignment for CMSC 626 (Graduate Students)

Lucky Grad Students — you get to learn some more about TMTOs! Undergraduates may also complete this task for Extra Credit. You will finish a TMTO attack to recover some of the five-character random passwords (group three).

You are provided with the following components of a partially-completed TMTO attack:

1. The program `build_table.py` that is used to build the TMTO tables
2. A zip file `catalog.zip` that contains 1024 already-built TMTO tables

You must write the program that carries out the actual TMTO attack (the every-time work) and apply it to recover as many passwords as possible from the hashes in group three (five-character random passwords). Grading for graduate students will depend on the number of five-character random passwords recovered in addition to the number of group one and two passwords recovered:

Number of Group 1 and 2 Passwords Recovered	Number of Group 3 Passwords Recovered	Highest Possible Grade	Rationale
5	0	C	Recovery of the simple dictionary passwords from groups one and two is the minimal acceptable accomplishment for graduate students.
7	10	B	Graduate students must perform at the undergraduate B level *plus* have moderate success against the group three hashes in order to earn a B.
9	20	A	Graduate student must perform at the undergraduate A level *plus* have a high level of success against the group three hashes in order to earn an A.
10 - 11	25 - 50	Extra Credit	Extra credit will be granted for recovering additional passwords.

GRADING CRITERIA FOR GRADUATE STUDENTS (CMSC 626)

Important: These are *maximum* grades. Grades may be reduced if the code submitted does not support the claimed password recoveries.

Graduate students must include the group three passwords they recovered in their lab write-up and turn-in their TMTO code along with the write-up and any other code written to recover password from group one, two, or three.