University of Miami

# Password Cracking

ECE 576

Nigel John
10-30-2019

## Objectives:

1. To understand how hash tables are used to lookup passwords
2. To understand the creation and use of Rainbow tables
3. Realize the limitations of Rainbow Tables
4. Use available tools for cracking UNIX passwords

## Task 0: Install python (v3), python IDE, python cryptography

For this lab, you are going to need to have python installed on your laptop. If you already have this then you can skip this step.

1. Download the latest python (v3) from www.python.org for your operating system
    a. Your SEEDs systems have python 2 but not python 3
    b. Your Kali has both, but you may need to update
2. Follow the installation instructions for your system
    a. See: https://wiki.python.org/moin/BeginnersGuide
3. For programming information:
    a. https://wiki.python.org/moin/HowToEditPythonCode
4. Install the python cryptography module
    a. Open a Terminal window (or Command shell)
    b. Type: pip install cryptography
        i. check https://cryptography.io for installation issues
5. Download and install an IDE:
    a. IDLE: python ships with a standard IDE, IDLE, written in pure python
    b. PyCharm: www.jetbrains.com/pycharm
    c. Eclipse with PyDev: www.eclipse.org (my personal preference)
    d. Xcode on the Mac also supports python
6. Familiarize yourself with the platform
7. The python documentation is at:
    a. Main documents: https://docs.python.org/3/
    b. Tutorial: https://docs.python.org/3/tutorial/index.html

## Task 1: Creating and Using a Hash Dictionary

For this task you are to create and use a Hash Dictionary to break passwords. In this case the passwords are the set of all 5 character passwords, **all uppercase, exactly 5 chars, no more, no less**.

### Task 1.1: Create the Dictionary

Estimate the size of the password space.

Assuming that you have to store the hash of the password and the password on each line of a text file (the dictionary), estimate the size of the file.

Now:

1. Generate all combinations of passwords and for each combination
    a. Create a SHA256 hash of the value
    b. Store the hash and the value to a file
2. Save the file, this is your dictionary

### Task 1.2: Using the Dictionary

You are going to use the dictionary to lookup a random password and compare against a brute-force check of every password. During this you will time the operations so that you can project the time it will take to lookup the hash in the dictionary (linear search) vs its size vs the time it takes to do a brute force search.

1. Create a random 5-character password
2. Find the hash of the password
3. Time how long to do each of:
    a. Search the dictionary to find the password
    b. Generate and hash all combinations in order until you find the password
4. Repeat a few times to get and average time form each

## Task 2: Create and use a Rainbow Table

Rainbow Tables are used to reduce the size of the dictionary to a more manageable size at the expense of more computation during the search for a password.

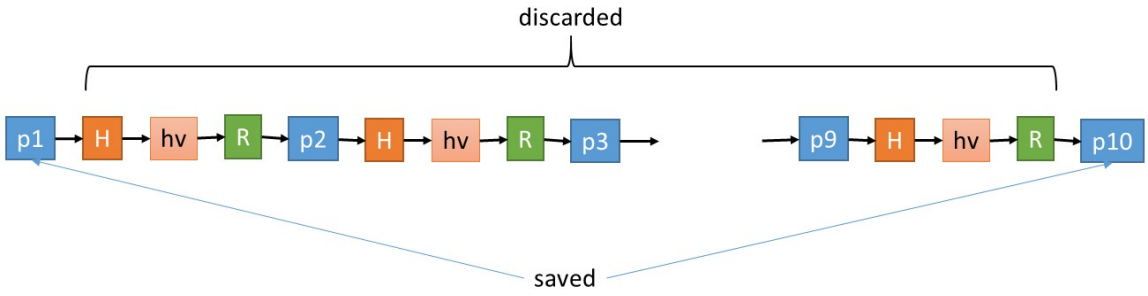
The Rainbow Table consists of a series of pairs of passwords derived from a chain of calculations consisting of repeated application of hash function and a reduction function.

The hash function represents the same hash used to create passwords, while the reduction function represent a mapping of a has value to a valid password (NOTE: the reduction is NOT a reverse of the hash). The start of a chain is shown below:

The length of these chains vary based on the size reduction needed, but the longer the chain the more computation will be needed to find the password.

### Task 2.1: Create a Rainbow Table

For this lab, you will used a chain length of 10. So that there are 8 hidden passwords contained in each chain and only the first and last kept in the file. This looks as follows:




The number of chains to create is also a problem, too few chains and they may not contain the password, too many increases the size of the resultant data file. Too many chains also creates another problem, they may merge to create duplicates sub-chains which makes finding the password more computationally intensive.

In this lab you will start with chains of length 10 and using 10% of the passwords as starting points.

The reduction function is also important. Repeating the same reduction function can create more merged chains, while using different reductions creates less merges, but increases computations.

In this lab the reduction function will be a very simple function: Simply convert the hash value to a base-64 representation and then take the first five characters as the generated password.

1. Generate all combinations of passwords and for 10% of the combinations
    a. Calculate the chain starting at the chosen password
    b. Record the final password
    c. Save the start and end of the chain to a file
2. Save the file, this is your rainbow table

### Task 2.2: Test the Rainbow Table

To test the rainbow table:

1. Create a random 5-character password
2. Find the hash of the password
3. Search the table:
    a. Apply the reduction on the hash
    b. Search the table to see if any chains end with the resultant value
    c. If not:
        i. Hash the new value and then return to step 3.a
        ii. Repeat until found or you have reached the length of the chain, in which case the password in not in the table
    d. If found:

        i. The password you are looking for is the one that produced the hash at the point you started, i.e. the one before the hash
        ii. To find this you must look at the first password in the chain and recalculate the chain from the start to just before the hash you were looking for.
        iii. The password before this is the one you are looking for.
4. If you do not find the password, then the table is too small, you will have to recalculate the table with more starting passwords.

## Task 3: Breaking Unix Passwords

Unix passwords make using dictionaries and rainbow tables impossible by including a salt with the original password. The salt is a randomly generate value that randomizes the hash of the same password. This means that the attacker will have to create dictionaries or rainbow tables for each possible salt, which greatly increases the space requirements.

To break these passwords, you need to use a brute force approach.

There are many tools to do this, the one your will use is called 'John the Ripper', and has a front-end GUI called 'johnny'

1. Load up you Kali unix
2. Copy the /etc/passwd and /etc/shadow file to your home directory
    a. Unix seperates the general user information and the salted/hashed password into separate files, you need to rejoin these
3. Run 'unshadow <passwd_file> <shadow_file> > <combined_file>'
    a. This wlll recombine the files
4. Start johnny
    a. Load the newly combined file into the system
    b. There are many options to create passwords, single use, wordlists, rules generated etc.
    c. Try a simple wordlist first
    d. If this does not work, try with all options.
    e. Note johnny can take a very long time to find some passwords as it tries very exhaustive searches for them
        i. You can test the effectiveness of this search by first creating a sample user with a simple password (such as 'password'). This will allow you to check the times to find simple passwords.
        ii. You can either add more sample users with increasingly complex passwords or just change the password of the initial sample user.
5. Record how long it takes to find the passwords.