Université Libre de Bruxelles
INFO-F405 - Computer Security
Project 1: Rainbow Tables

Teaching assistant: Naïm Qachri
nqachri@ulb.ac.be

2012-2013

1 The project

1.1 Introduction

The passwords that we use everyday are either stored locally on our computers, or stored remotely on servers. Usually, passwords are stored hashed in order to complicate the work of an attacker who would gain access to the file in which the passwords are stored. The hashed passwords are called fingerprints and have a fixed size. Despite this protection, it is now usual that attackers try to get access to passwords in clear. More precisely, the attacker will try to find passwords that when hashed produces known stolen fingerprints.

The most common techniques are the following :

– **The brute force attack** generates all the possible passwords and checks if their associated fingerprints matches stolen fingerprints.

– **The dictionary attack** precomputes all the fingerprints of a dictionary of passwords. Stolen fingerprints are then compared with all the fingerprints contained in the dictionary. Once a matching is found, the attacker obtains, thanks to the dictionary, the associated password in clear.

There is also another popular technique based on a particular table called **rainbow table**.

A rainbow table based attack offers a tradeoff between the needed storage resources and the computing time. The method is based on the hashing algorithm that is used to compute the fingerprint of the passwords and on a *reduction function* that maps (or reduces) a fingerprint to a potential password. For instance, the function takes a subset of the bits of the fingerprint and convert them into a string that will be the new password with the same length than the subset size. The only constraint for the reduction function is to be consistent and to return the same result for identical inputs. Being consistent means that if the reduction function returns passwords, then every possible output is at least mapped to one input.

Generating the rainbow table is simple : the attacker picks a password and applies n times, iteratively, the hashing and the reduction functions. After the n successive applications of bashings and reductions, the initial password and the last computed fingerprint are stored in the table. The process is repeated then with a new password. The new password can be chosen in a dictionary or on a random basis. We propose to choose it in a dictionary, where password are chosen in succession

(a dictionary that you can easily create yourself). The Fig. 1 shows a graphical representation of the algorithm.

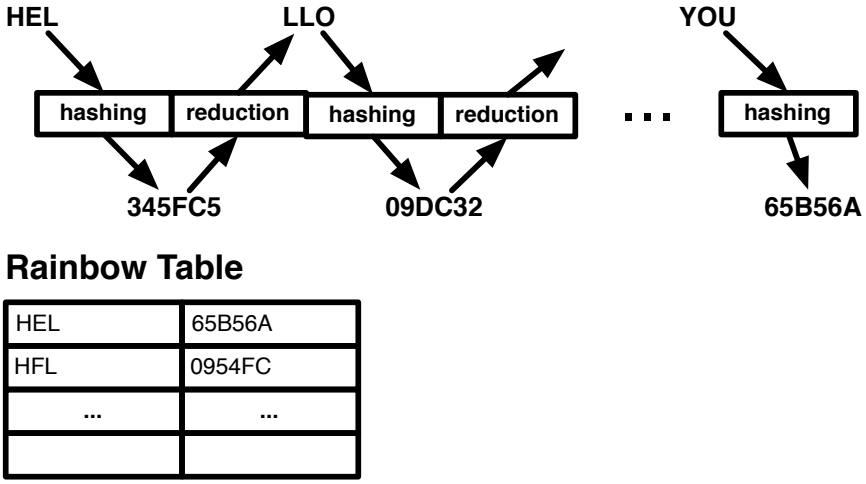

FIGURE 1 – sequence of reductions and hashing and the associated rainbow table

Using this table to retrieve a password consists of two steps (once the attacker has captured a fingerprint) :

1. if the stolen fingerprint matches one of the fingerprint stored in the table, successive hashings and reductions are computed on the related password in the rainbow table to find the corresponding password ;
2. else, the stolen fingerprint is reduced and hashed. If this new fingerprint does not match one of the fingerprint stored in the table, this step is repeated, else (if the fingerprint appears at last in the table) the algorithm jump to first step.

Unfortunately, due to collisions, this method creates many duplicates in the password chains, and therefore within the storage. This is why, instead of using n times the same reduction function, the algorithm uses n different reduction functions (four different reduction functions are required for this project). If you consider each reduction function as a color, the name of *rainbow* table becomes logical.

When the algorithm uses n reduction algorithms, the attack becomes :

1. if the stolen fingerprint matches one of the fingerprint stored in the table, successive hashings and reductions are computed on the related password in the rainbow table to find the corresponding password ;
2. else :
   - a) initialize i to 1 ;
   - b) computes from the ith reduction to the nth and compares the new obtained fingerprint with the fingerprints stored in the table. If there is a match goes to step 1, otherwise increment i and store the fingerprint in a queue. If $i = n$ then goes to step 2c, else goes to 2b ;
   - c) pick the next fingerprint available in the queue and goes to step 2a.

There are many possible optimizations for the step 2. We ask you to implement the attack in the most efficient way.

1.2 The hashing algorithm

In the framework of this project, the hash function will be based on the `DES` encryption algorithm. `DES` will be used in the `ECB` chaining mode and the password to be hashed will be used as the encryption key. The message that will be encrypted with `DES` will be a publicly known 64 bits constant. The resulting 64 bits of the encrypted password must then be truncated to the 24 least significant bits. Since the key used by a `DES-ECB` algorithm has a size of 56 bits (without counting the 8 parity bits), in this project the password will be the 12 least significant bits of the key, the 44 remaining bits will be set to zero.

1.3 The reduction functions

The only constraint for the reduction function is to be consistent and to return the same result for identical inputs (but collisions are possible). It means that you have many choices. For instance, we could choose a reduction function that maps a fingerprint to a password as it follows : if the fingerprint has a size of 24 bits and a password has a length of 12 bits, the 12 most significant bits of the fingerprint will be the next password.

`Example of reduction function application :` *0x616BC3* $\rightarrow$ **0x616**

2 Realization of the project and deliverables

The project must be done by groups of four students.

You are asked to implement a program that cracks passwords from fingerprints generated by the presented hashing algorithms and to provide a report. You can use `Java` and `C++` for your project. You must use the following cryptographic libraries for the implementation of `DES` : `Crypto++` for `C++` language and the `Java Crypto Library` for `Java` language. You can use any common data structures, such as vectors, lists or matrices, to implement and optimize your rainbow table. We ask you to use four different reduction functions for your table. The choice for the functions is let for the students.

Some aspects of the project is simplified for reasons of usability and testing. The program must print at the end of the execution a list of the different possible keys that matches the cracked fingerprint.

Some parts of the system have voluntarily not been completely described in this document in order to let you investigate how to implement them efficiently.

Your report has to mention details about your choices of implementation and to justify your choices for the reduction functions, the pros and cons of these choices. It must not exceed 5 pages.

Do not forget to mention your names and your group number in the report. You have also to provide all the files needed to install and configure your program(a makefile must be included).

Your complete report has to be submitted to the secretariat of the computer science department (2N8.104) on paper format.

You have also to send an electronic version of your project to *infof405@lit.ulb.ac.be*. The object of the email must be "[INFO-F405] Project 1 Group x", where x is your group number. Your report, the implemented project and the different files must be compressed within a zip file that will be attached to this email. The name of your zip file must be "Secu12013+group number" (Secu120133.zip for the group 2, for example).

Oral defences will be organised during the week 7 (or 8, depending on the calendar that you read). The place and the date will be communicated later on the virtual university.

Project delivery instructions

To respect scrupulously !

1. Your project must include **your name** and **your group number**.
2. Your project must be **typewritten**. Handwritten project will **not be corrected**.
3. Your implementation must be **commented**.
4. Your must respect the constraints of programming languages and libraries.
5. You must respect the following terms :
   - Deadline : **the 28th october 2013 before 4pm**
   - Where : **"Student" secretariat of the computer science department, local 2N8.104**

   The secretariat closes at 4pm. **After 4pm**, the project will be considered **overdue**, and the project will be scored 0. This deadline and the sanction are also valid for the asked email.