3.1.1

CRACKING WINDOWS PASSWORDS

(CAIN)

Laboratory Overview

Objective

An understanding of what makes a “Strong” password, by using software programs created to crack password security using both dictionary and brute force attacks.

Information for Laboratory

A. Students will utilize Cain v2.65 to “crack” stored Windows passwords on the local machine

Student Preparation

The student will have completed requisite reading. The student will require paper for notes and should be prepared to discuss the exercises upon completion.

Estimated Completion Time

60 Minutes

Password Security

Most accounts on a computer systems usually have some method of restricting access to that account, usually in the form of a password. On most systems, the password is stored encrypted, so it can not be read easily. Typically the password is ran through some type of algorithm to generate a hash. The hash is usually more than just a scrambled version of the

original text that made up the password, it is usually a one-way hash. The one-way hash is a string of characters that cannot be reversed into its original text. Most systems do not "decrypt" the stored password during authentication, they store the one-way hash.

During the login process, you supply an account and password. The password is ran through an algorithm that generates a one-way hash. This hash is compared to the hash stored on the system. If they are the same, it is assumed the proper password was supplied. There are several different types of encryption methods used to store passwords.

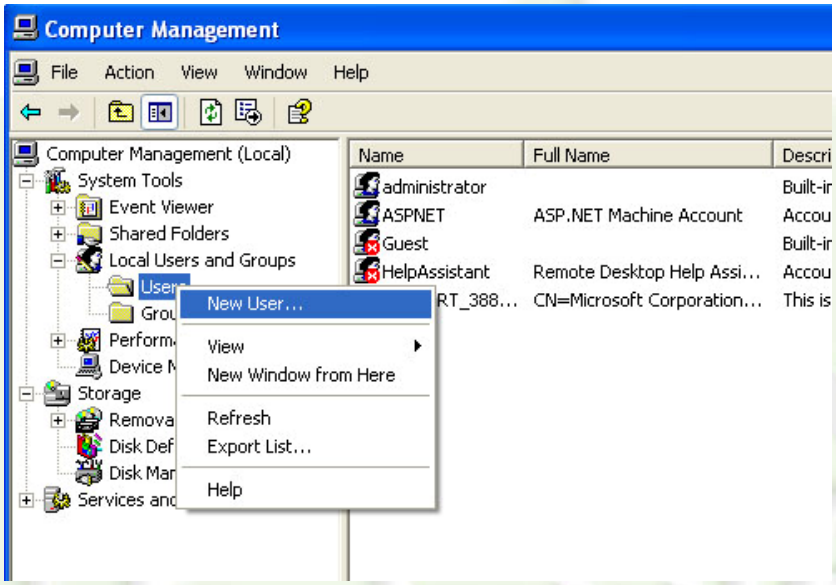
Microsoft Windows XP uses NTLM hashes when storing a users password. To crack a password requires getting a copy of the one-way hash stored on the computer, and then using the same NTLM algorithm to generate your own hash. Keep doing that until you get a match. When you get a match, whatever word you used to generate your hash is in fact the stored password. Since this can be rather time-consuming, programs have been written to automate the process. There are freeware password crackers available for Windows, Netware, and Unix.

Cain v2.65

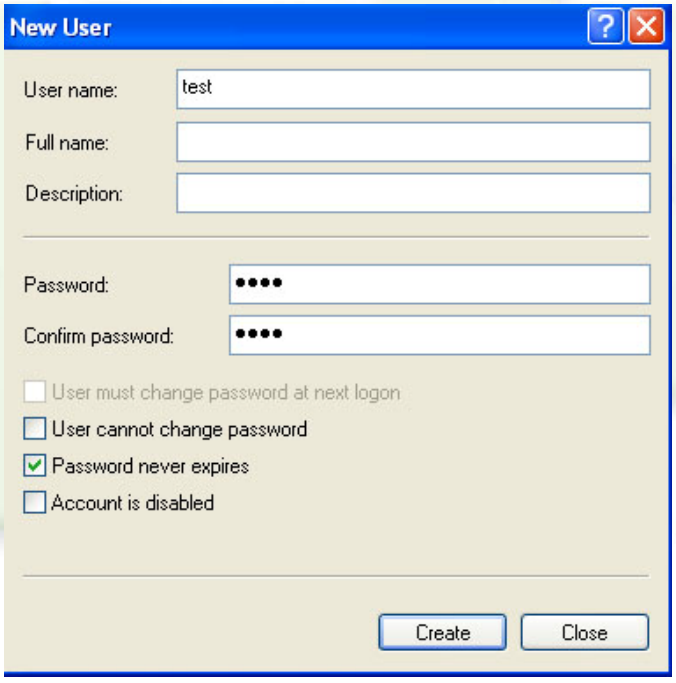
We will use Cain v2.65 to demonstrate how password cracking is done, and the concept of a strong password. Cain is a freeware program that can be downloaded via the Internet.

Step 1: Create test user

Right click on My Computer and click Manage. From Computer Management, double click on Local Users and Groups. Right click on the Users folder, and click New User as shown below.

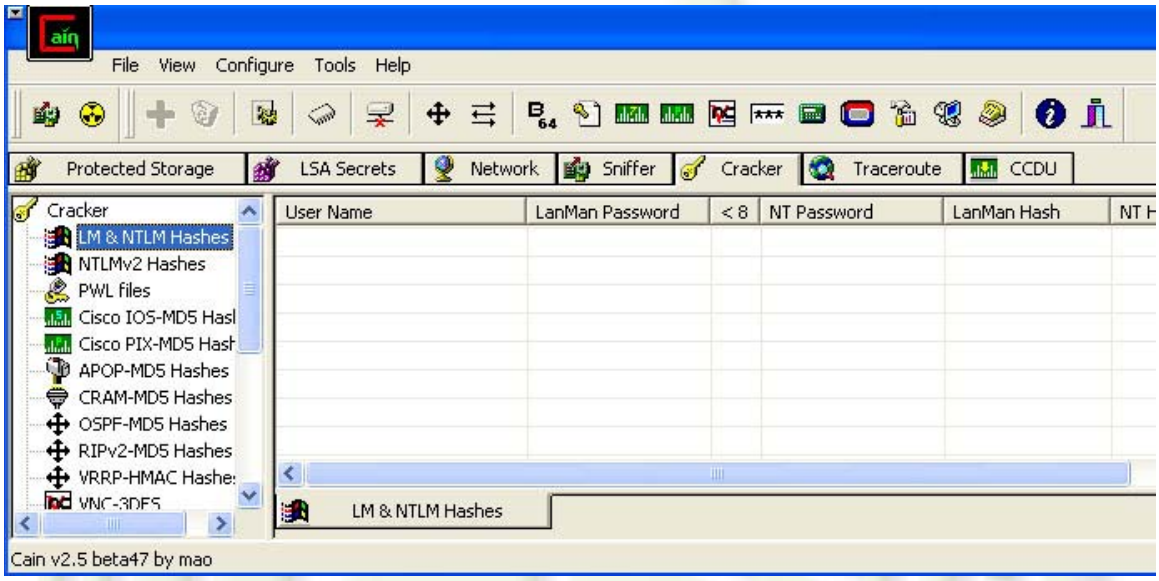


Create a user named test. Set the password to “test” (do not type the “quotes” only what inside). Unclick “User must change password at next login”, and click Create.

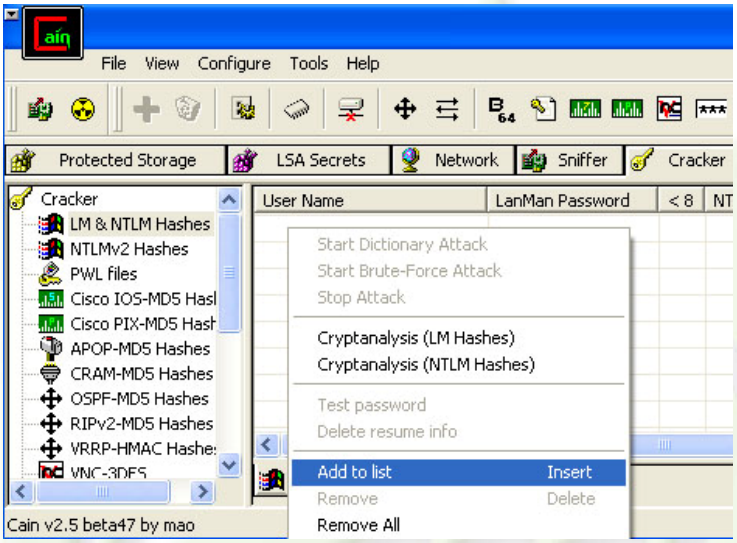


Step 2: Open Cain

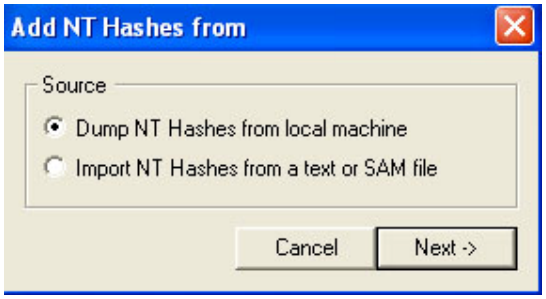
When Cain is open, click on the Cracker tab, and LM & NTLM Hashes as show below.



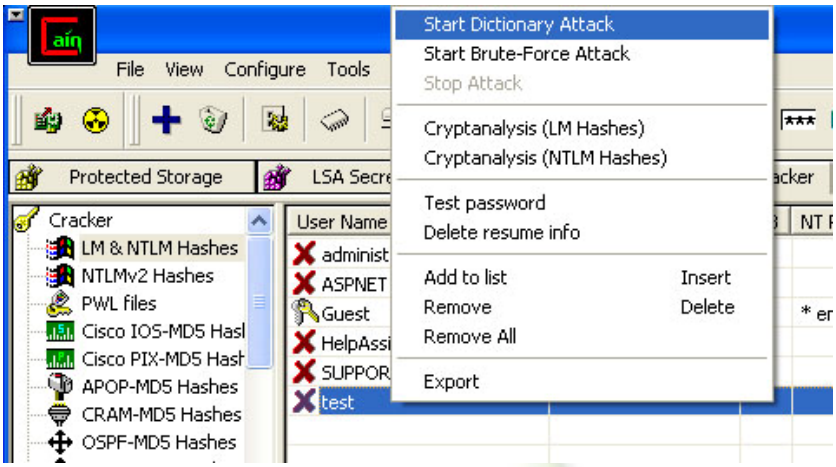
In the center of the workspace, right click and click Add to list.



Click Next to Dump NT Hashes from local machine



The user names will all appear on the cracking list. Right click on the user test, and click Start Dictionary Attack.

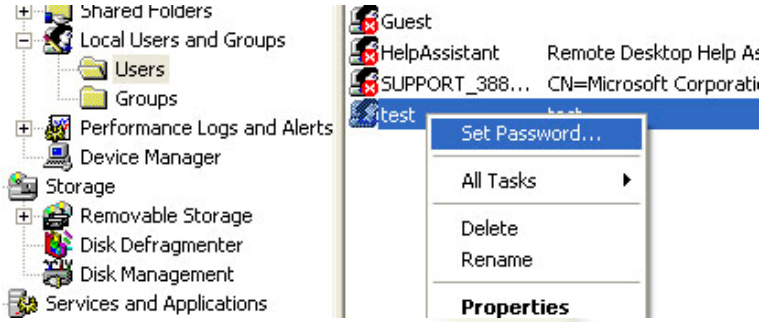


Notice below, the password “test” was found. You can also view the actual NTLM Hash.

You should have noticed that it only took a split second to find the password. A dictionary based attack uses a word list, containing every word in the dictionary, names, and some other misc. items. Since it can usually run through the whole list in less than a minute on a local machine, using a word from the dictionary is not a good choice for a password. Close Cain.

Step 3: Changing the test user password

From Local Users and Groups, click on users. Find the user test, and right click on it. Click Set Password…



Change the password to “aaaa”

Step 4: Crack the password

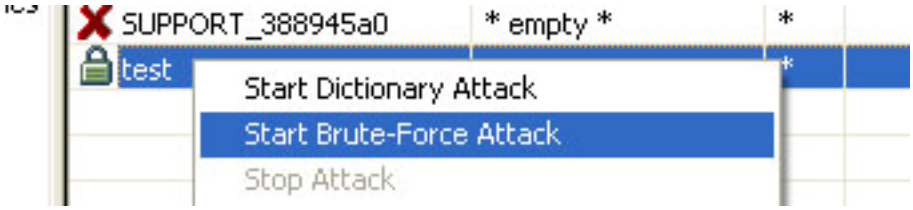
Select all the accounts listed in Cain, and press delete on the keyboard to clear the list. Since we changed the password on the user test, the hash has changed, and Cain needs to be updated with the new hash.

Once cleared, repeat Step 2, dump the NT hashes from the local machine to Cain. Find the user test, and run the dictionary attack on it.

Notice that test is locked, the dictionary based attack could not produce a matching hash. “aaaa” is not on the wordlist.

Step 5: Cracking the password Brute-Force

Right click the user test, and click Start Brute-Force Attack



You should notice that the brute-force attack found a match very fast. "aaaa" would not be a strong password. Brute-Force attacks simply try all possible passwords until it gets the password. It will use every letter and number combination possible. From a cracker perspective, this is usually very time consuming. However, given enough time and CPU power, the password eventually gets cracked, all depending on the length of the password. A 4 letter password may only take a few minutes, but longer passwords could take hours, weeks, or even years to crack.

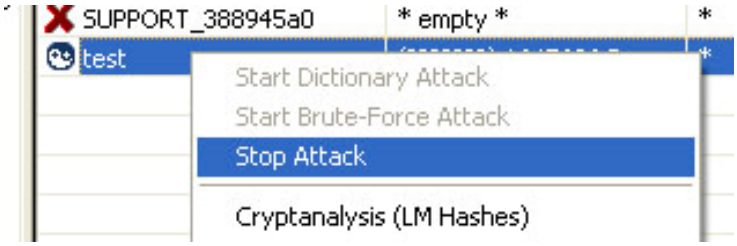
Step 6: Trying another password

Change the test users password to "!!aa"

Clear Cain, and then update with the new user info.

Start a Brute-Force attack on the test user.

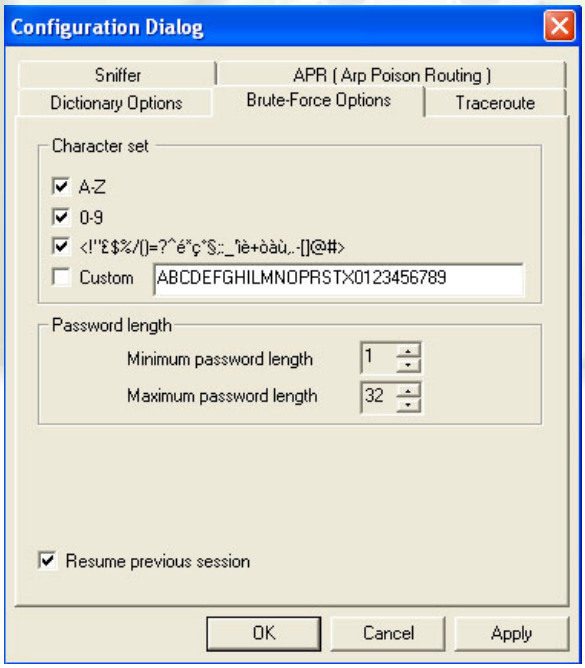
You will notice that after running for several minutes, Cain has not yet cracked the password. Right click on the test users and click Stop.



Step 7: Modifying the Brute-Force options

From Cain, on the top tool bar, click Configure, and click on the Brute-Force Options tab.

Check the symbols box and the click OK



Step 8: Cracking Brute-Force

With the options configured to also use symbols, Start a Brute-Force Attack on the test user.

This will help the brute-force attack on the password “!!aa”, as the ! is a symbol. Without this option, even a brute-force attack would never crack a password with a symbol in it.

Within a few minutes, Cain should have been able to crack the password. If its has taken more then a few minutes, check the brute-force options, clear the user list, reset the password, and try it again.

Step 9: Analysis

Try some of your own passwords, see if they can be broken from a dictionary based attack, or a Brute-Force attack.

A word to the wise: Brute-force attacks take time. The longer the password the harder it is to generate the matching Hash. Using symbols with number/letter combinations not found in the dictionary make for strong passwords.

Which of the following do you think are “Strong” passwords, and why?

Car	!#%asddsa!#%	password
Bobby	aa!!	!!g23h**h23g!!
Fido	!!aaaa!!	tqbf1978
MaNaGemEnT	marybethmikejoe	$$m0n3y$$

Summary Discussion

A classroom discussion should follow the lab. Review the lab questions and your analyses as a group. Share your experiences and knowledge with the class.

Appendix:

This lab was developed using Cain version 2.65, which can be obtained from:

http://www.oxid.it

Use the projects link. An online manual is also available.

The OS environment for this lab was Windows XP Professional, Version 2002, Service Pack 2 (8/04).