**Password Security and Cracking on Windows**

**Introduction**

If a server is to authenticate a user's login, the server must know the user's password so that it can compare the password that has been entered against the stored password. Passwords are the most commonly used mechanisms to authenticate users on to a system and/or protect sensitive information. Bad passwords can be guessed or cracked.

***Objectives:***

* Examine what makes a good password
* Discuss how passwords are stored
* Discuss different hashing algorithms
* Investigate techniques used to guess or crack passwords.

**Passwords**

What constitutes a good or bad password? Bad passwords lie on one of two ends of the spectrum. They are either easily guessed or cracked (i.e. your spouses name, your children’s name…) or they are difficult to remember (these you either forget or you write down where someone else can find them). A good password on the other hand is hard to crack but also easy to remember

Any computer that's running Windows 2000, XP, or 2003 stores users’ passwords within the Security Accounts Manager (SAM). Technically, the SAM doesn't contain the passwords themselves. The password is hashed, and then the hash is encrypted and stored within the SAM. When you first read about the way that Windows stores passwords, it sounds fairly secure. There are a couple of problems with the storage method. For starters, the encryption key is stored on the server right along with the hashes that the key encrypts. This hashing algorithm is fairly well known. There are several utilities available off the shelf that could allow anyone who has physical access to the machine to reset a password without knowing any sort of login credentials. There are also a few "password recovery utilities" that are allegedly able to retrieve passwords directly from the SAM database.

Windows 2003 Server encrypts passwords using one of two hashing mechanisms: LM-hashes or NT-hashes. Hashing is a numeric algorithm designed to create a unique representation of a password that the average user cannot understand. LM Hash is considered much weaker than NT Hash. In this tutorial, we will begin with a high level discussion of password encryption. We will examine passwords on a Windows 2003 server that we have setup for you. You will log into the system as an administrator, create a user account and then use various password-cracking tools to explore the security LM hashes versus NT hashes.

Before we begin, you will need to log in to your preconfigured snapshot (Password Cracking) Virtual Machine and create new users accounts. The snapshot to be used for this exercise is entitled ***Password Cracking Exercise***.

**Windows**

Windows can store passwords in two ways: A LM Hash (considered weak) or a NT Hash (considered pretty strong if users can come up with a good strong password). For local users windows stores the passwords in the SAM database locally on the machine*. (FYI: When your password is stored in cache the credentials are actually hashed again adding another layer of security)*

***What is a hash?***

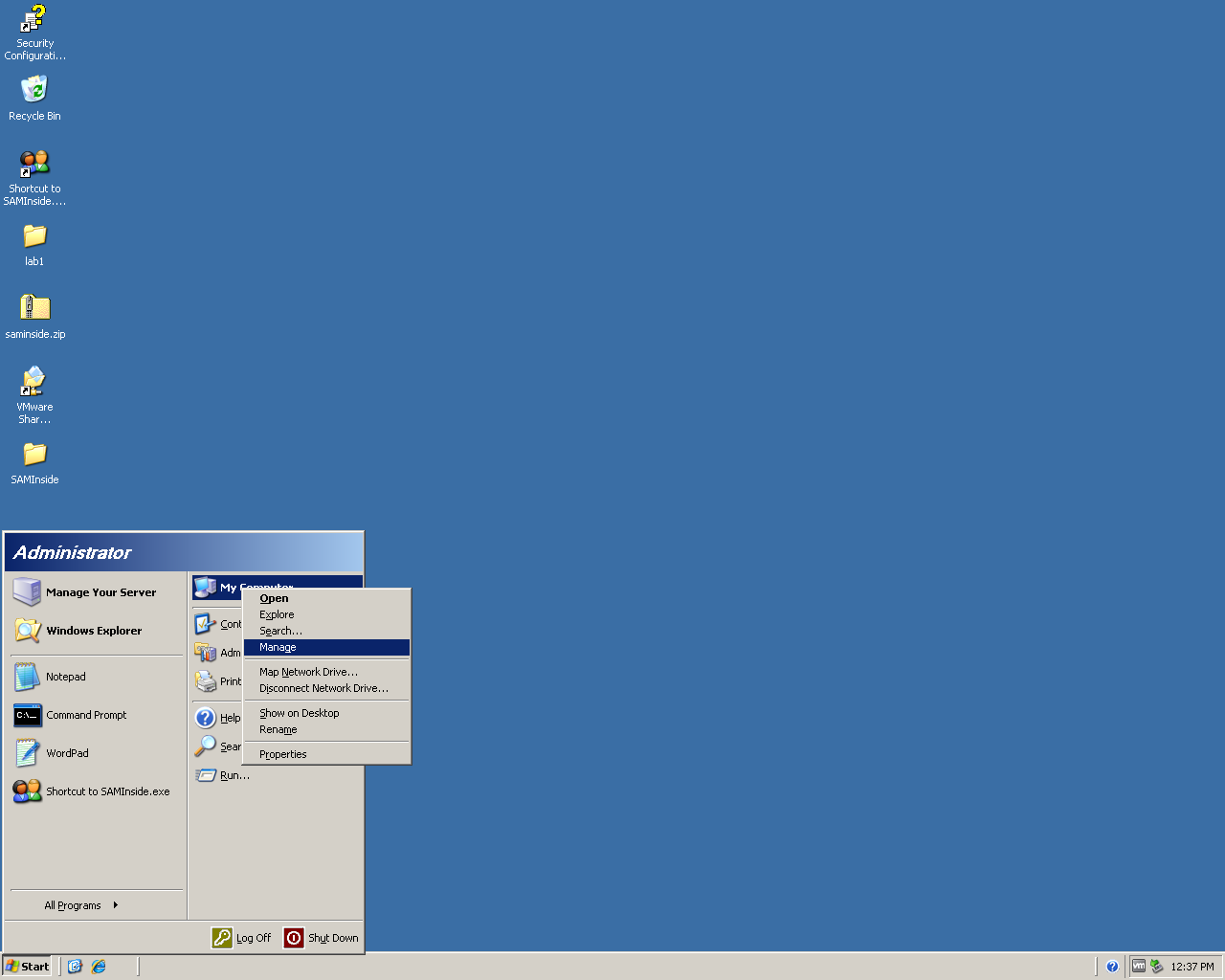
LM stands for “LAN Manager” and is an older (pre Windows NT) technology. For a LM Hash -- among other things DES (Data Encryption System) is used to create a hash. LM hashes aren't really hashes. In LM your password is converted to upper case and broken into two 7-character chunks. These chunks are used as the key to encrypt a well-known constant with DES. It worked OK years ago, but it's cryptographically weak. The characteristics of the LM hash algorithm are listed below [1]:

1. The user's password is restricted to a maximum of fourteen characters (assuming one byte per character)
2. The user’s password is converted to uppercase
3. The user's password is encoded in the System OEM Code Page.
4. The password is null-padded to 14 bytes.
5. The “fixed-length” password is split into two seven-byte halves.
6. These values are used to create two [DES](http://en.wikipedia.org/wiki/Data_Encryption_Standard) keys, one from each 7-byte half.
7. Each of the two keys is used to DES-encrypt the constant [ASCII](http://en.wikipedia.org/wiki/ASCII) string “KGS!@#$%”,resulting in two 8-byte cipher-text values.
8. These two cipher-text values are concatenated to form a 16-byte value, which is the LM hash.

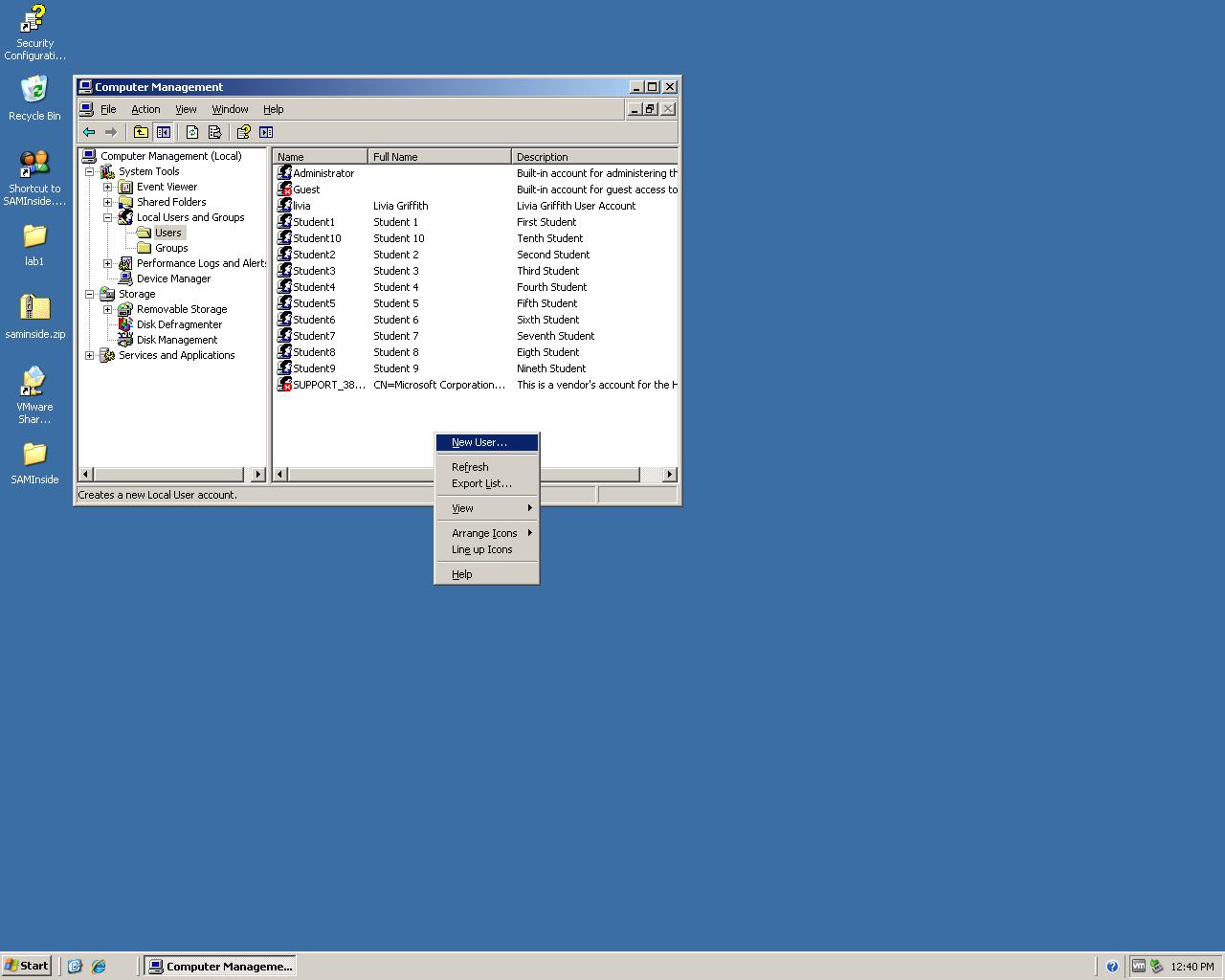
Several weaknesses can be noted to this “hashing” technique. First the passwords are limited to only 14 characters. Given that there are 95 printable ASCII characters there are 9514 possible password combinations. Secondly, since passwords longer than 7 characters are divided into two seven character halves each half can be attacked separately. This reduces the options to 957, which is even easier to crack. A third weakness is that LM hashes do not use a *cryptographic salt* in the algorithm. The term *cryptographic salt* refers to a random value that is applied to the algorithm to make it unique. By applying a *salt* helps to further disguise a password. Assume that two people use the same password. Without applying a random *salt* value the stored encrypted hash value would be the same. Applying a *salt* adds an additional layer of security.

**Exercises Preparation**

We will start by adding three new local users to the system. Take your mouse and click on the start button. Right click on ***My Computer*** and select the ***Manage*** option.



Expand the ***Local Users and Groups*** section and choose the ***Users*** folder in the left hand column. In the right hand section Right click and choose ***New User***



Add three new users with the following criteria:

Username: *testuser1*

Full Name: *Test User 1*

Descripton: *Your Name Student Account*

Password: *Any password you would like to use*

*Be sure to* ***uncheck*** *the box next to “User must change password at next logon” and that all other checkboxes are* ***unchecked.***

Username: *testuser2*

Full Name: *Test User 2*

Descripton: *Second Test User*

Password: *Any password you would like to use*

*Be sure to* ***uncheck*** *the box next to “User must change password at next logon” and that all other checkboxes are* ***unchecked.***

Username: *testuser3*

Full Name: *Test User 3*

Descripton: *Third Test User*

Password: *Any password you would like to use*

*Be sure to* ***uncheck*** *the box next to “User must change password at next logon” and that all other checkboxes are* ***unchecked.***

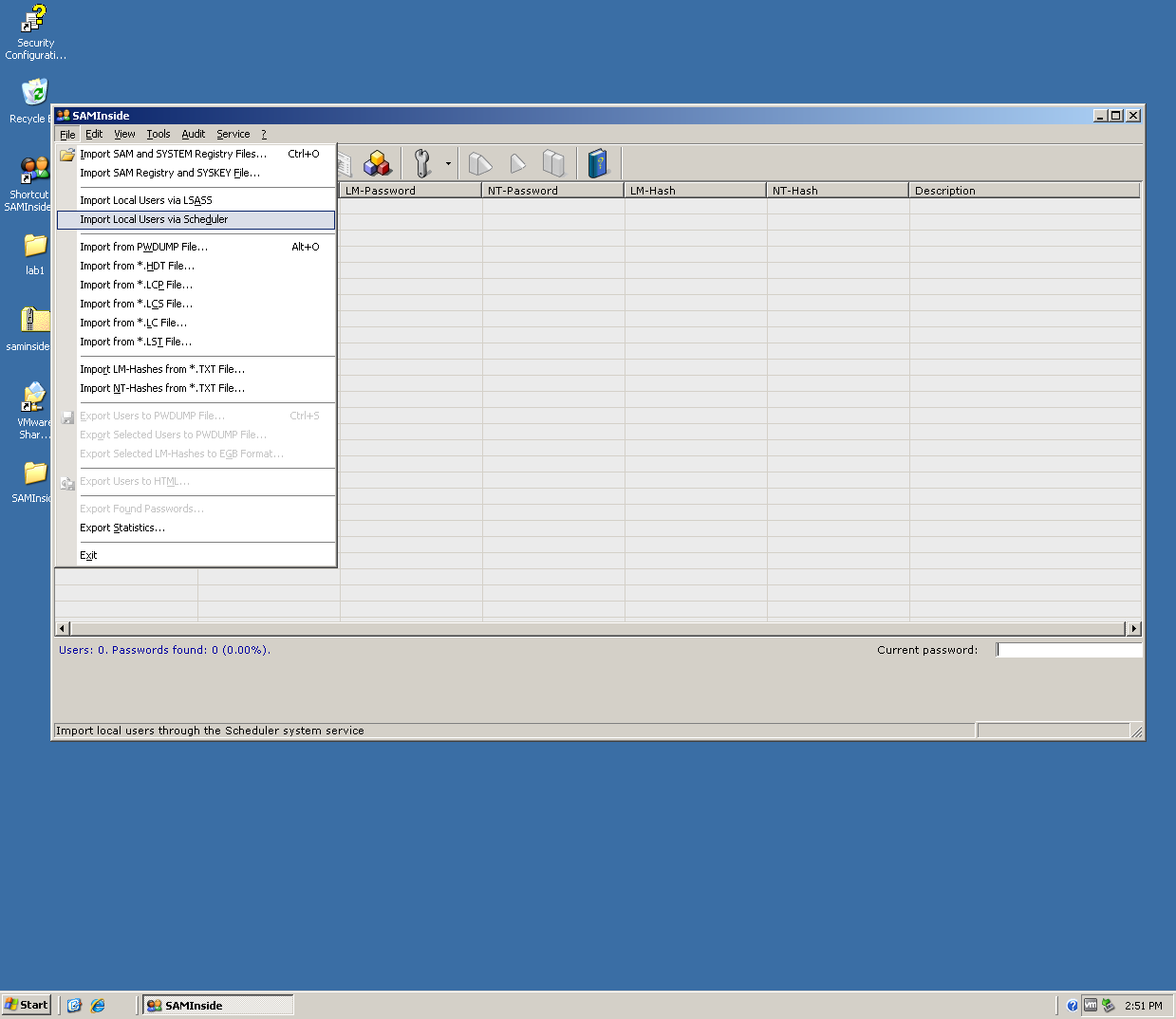
Close the *Computer Management* Window

**Exercise 1 – Cracking LM & NT hashed Passwords Using SAMinside**

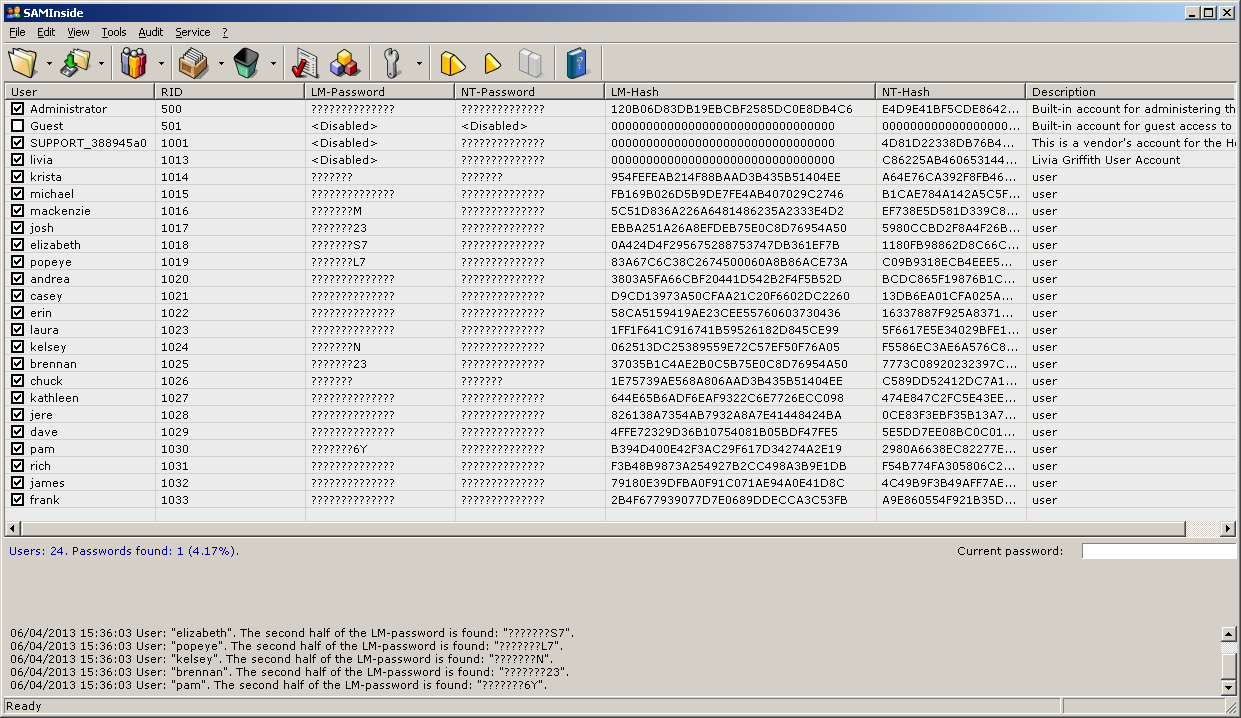
**Summary:**

* Start up SAMinside password cracking software
* Load the local user information from your current system
* Attempt to crack the passwords for the accounts located on the machine using a LM Hash brute force attack.
* Attempt to crack the passwords for the accounts located on the machine using a LM Hash dictionary attack.

Now we will test passwords security. Double click on the desktop icon named ***SAMinside*** to start the *SAMinside* password-cracking program. (If you receive a security-warning click on the *Run* button.) You should now see a window that looks similar to a spreadsheet with headings that include User, RID, LM-Password, NT-Password, LM-Hash, NT-Hash, Description. This program is called SAMinside and can be used to, among other things, to crack passwords. Let’s play with this a bit. First we will need to import the current user list into SAMinside. To do this click on the *File* menu and select ***Import Local Users via Scheduler.*** This may take a moment and will import all the local users that you saw in the *Computer Management* Application Window.



A list similar to the one below should appear.



Password cracking procedures are located under the *Audit* menu located at the top menu bar. When we click on this option we see that boxes are checked in front of LM-Hash Attack and Brute-force Attack. This means that the program is setup to do a brute force attack on the LM-hash password algorithm. As we mentioned earlier passwords can be stored in either LM-hash or NT-hash formats. If you look in the body of the Window you will be able to tell what type of hash was used for each user’s password. If a password is 14 characters or less Windows 2003 by default will store passwords in less secure LM-Hash format.

Let’s run a test attack and see what happens. We will start with the default LM-Hash/Brute Force attack. Check that these two boxes are checked and then right click on ***Start Attack***. Let’s let it run for several minutes and see what happens. Check the contents occasionally to see if you notice any changes in the body of the Window. As characters become discovered they will be displayed under the LM-Password column. Also notice the changes in the *Current Password* window. You may also note that in the lower left hand section of the Window statistics about the number of *Passwords Processed* and the *Time Remaining* are displayed.

There are two different types of attacks that can be used: Brute Force and Dictionary Attacks. Another name for a *brute force* attack is an *exhaustive key search*/attack. In this attack the system performs an exhaustive and systematic check of all possible key combinations until a valid combination is found. The larger the key the more time consuming this can become. A *dictionary attack* takes a dictionary (or list) of commonly used passwords and applies them to the password until a discovery is made. Running and attack takes time so this will run for a while.

While we are waiting let’s discuss the display above. Notice that some of the LM-Passwords have letters already filled in. This is password discovery already at work. This also confirms for us that LM-hashes are just not that sophisticated.

***Windows passwords do not use salt values when creating a hash:***

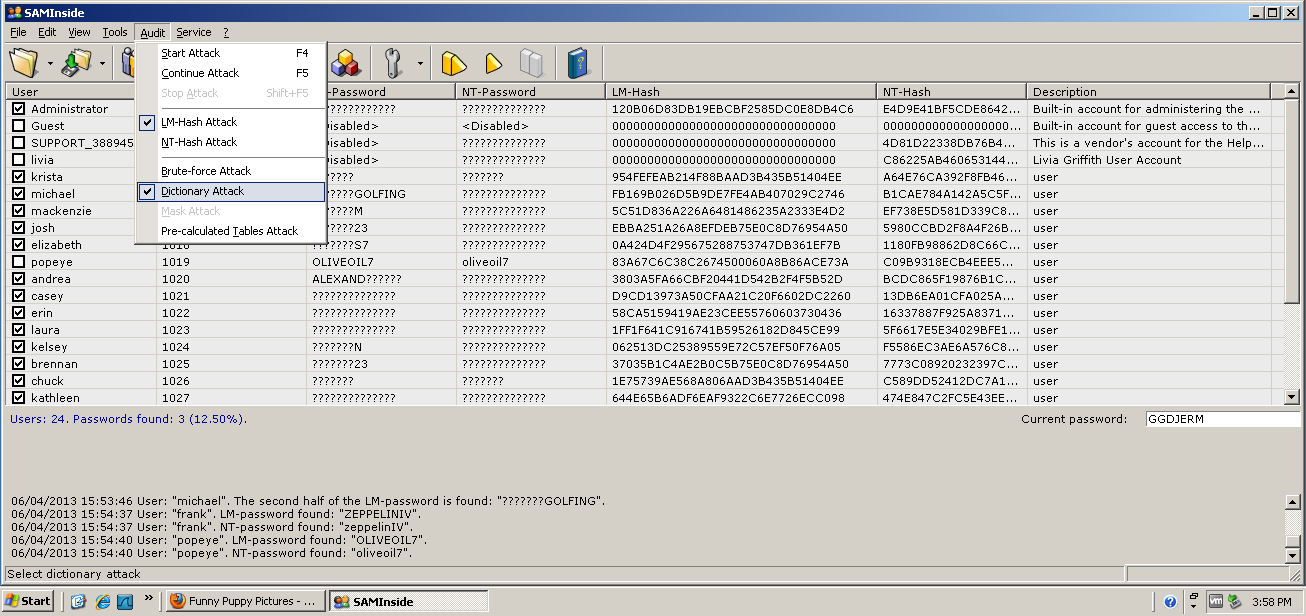
Notice also that it appears no two hashes are the same. This means that no two users have the same password. Using a randomized salt value is imperative to securing passwords.

Windows does not use salting when storing the passwords hashes, thus all Windows   
computers will create the same hash for same password. You can actually buy   
on-line pre-computed hashes (but to store all possible combinations for LM   
Hashes it would take 166 Terabytes of space and for all possible NT Hashes   
generated from password shorter then 15 characters it would take about   
140,959.235.198 Exabyte’s).

Generally speaking a NT Hash can only be cracked if the user used an easy to guess password (dictionary attack) while a LM Hash can be cracked in few days even if users came up with really complex passwords (but majority of passwords stored as LM Hash were cracked in a day or less).

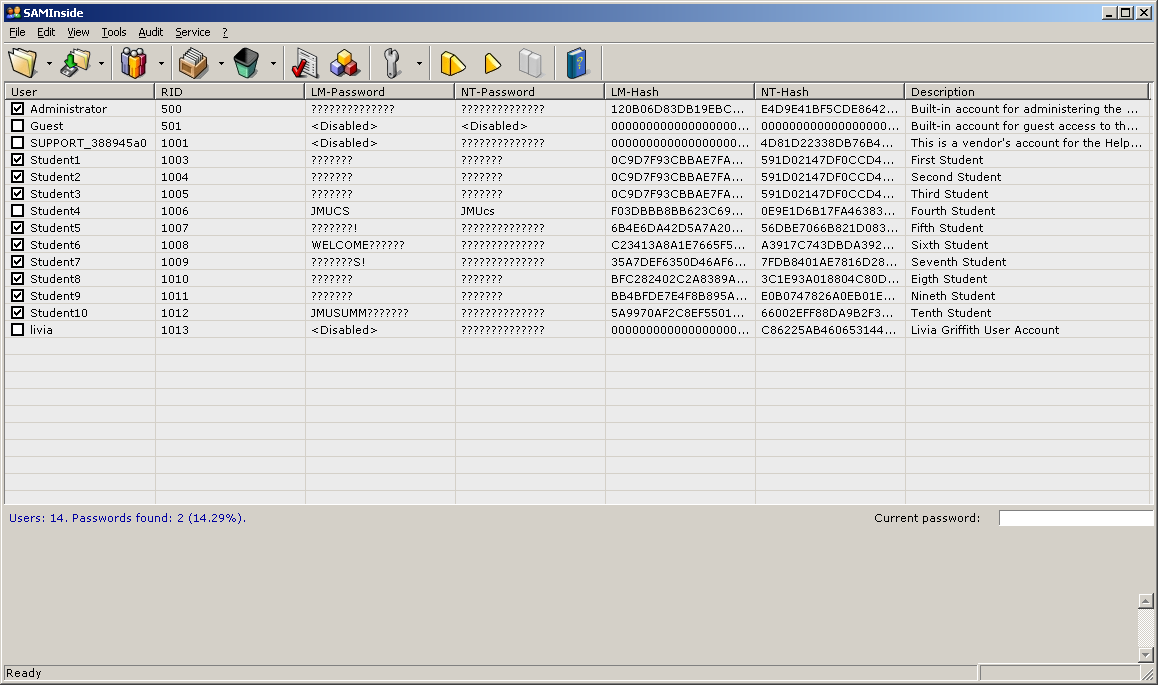
NT stands for “New Technology” and first came into existence with the Windows NT operating system. For NT Hash - MD4 is used. NT hashes are real hashes. The hash is a single application of MD4 on the (case-preserved) password; the result is stored in Unicode.

Now you should stop the attack and observe the results. Have some clues appeared as to what the passwords are? Now try the same exercise but change the attack to be run on the NT-Hash. Wait several minutes before stopping and see what happens. To stop the attack click on *Audit🡪 Stop Attack.* Look in the body of the window under the Password columns and see if you notice any changes. Now let’s try a dictionary attack. Choose the ***Audit*** tab. Leave the *LM-Hash Attack* selected but now check the box next to *Dictionary Attack* and the click ***Start Attack***



Notice how much faster this ran. This is partly due to the very small dictionaries we had loaded prior to the start of this course. *Dictionary attacks*, like *brute force* attacks can run for days. Look at the results. Were any other passwords discovered?

Below is another example. In this example we are viewing the results after both and LM Hash attack and an NT Hash attack have been performed.



In this example the password for Student4 has been discovered. It was a simple word so discovery was made for both the LM and NT Hashes. Notice that the true “case sensitive” password is revealed in the NT Hash. As we noted earlier this is because LM Hashes convert all letters to uppercase. Another worthy observation can be noted for Student6, Student7 and Student10. Parts of the LM Hashed passwords were discovered for these three users. This is not true when trying to crack the NT Hash. This is proof that the NT Hashing algorithm is more secure than the LM Hashing algorithm.

SAMinside is a popular tool for cracking passwords. A free demo version is available for download on the Internet but the full version requires a minor payment.

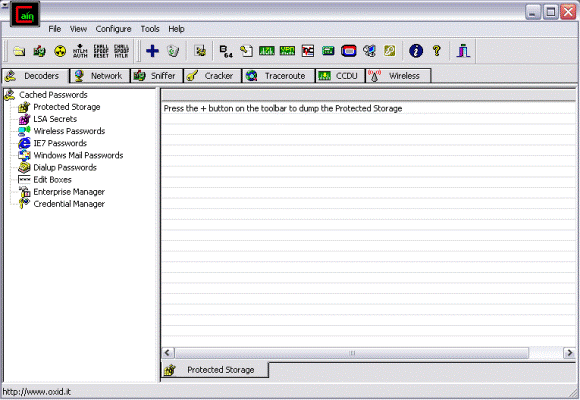
**Exercise 2 – Cracking LM & NT hashed Passwords Using Cain and Abel**

**Summary:**

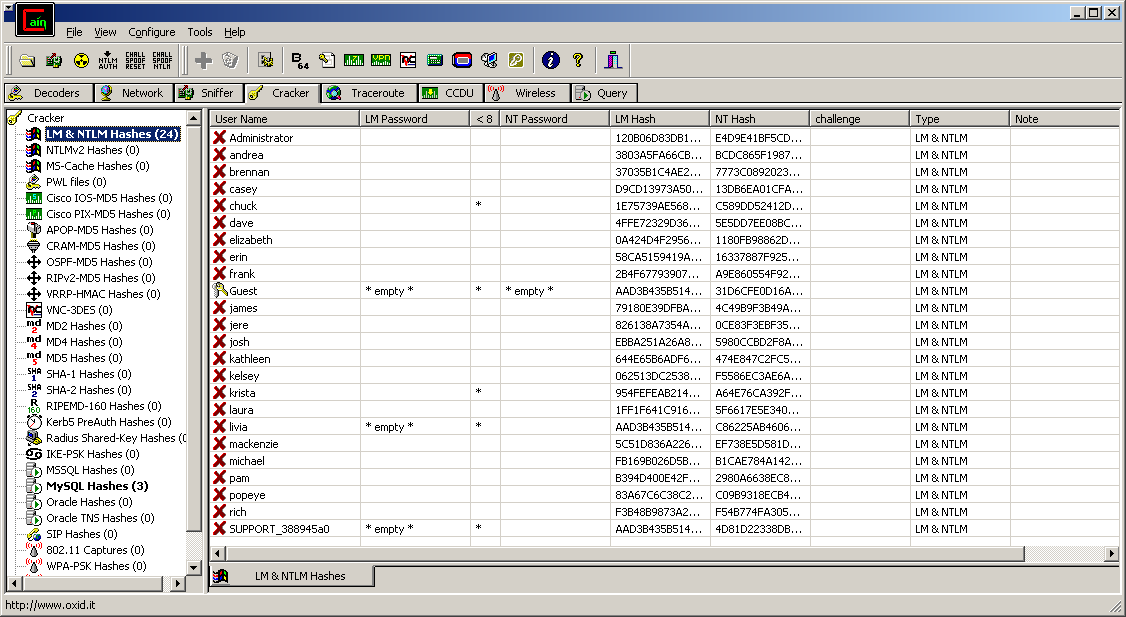
* Log into Windows 2003 Server
* Add a user with a password of less than 14 characters
* Start up Cain and Abel password cracking software
* Attempt to crack the passwords for the accounts located on the machine using a LM Hash brute force attack.
* Attempt to crack the passwords using other approaches

Another well-known and powerful cracking tool is Cain and Able. Cain & Abel is a password recovery tool for Microsoft Operating Systems. It allows easy recovery of various kind of passwords by sniffing the network, cracking encrypted passwords using Dictionary, Brute-Force and Cryptanalysis attacks and much more. Let’s take a look at this tool and see how cracking works.

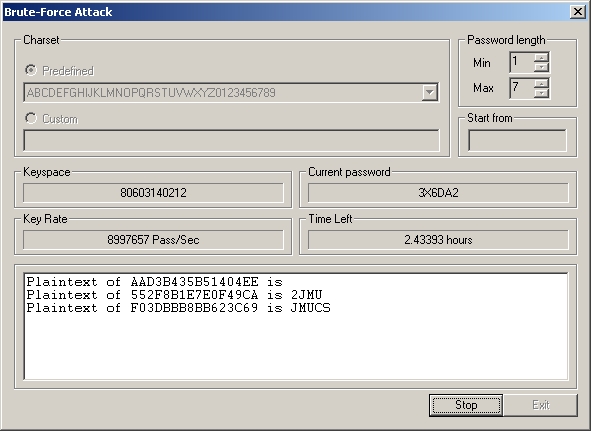
Open the "Cain and Abel"  xpcrack00



Choose the “*Cracker*” xpcrack tab in the third row. Notice the list of subcategories displayed in the left column. Cain and Able is an in depth tool with the ability to crack multiple types of password Hashes. Since we are working with Windows 2003 account passwords we want to select the “*LM & NTLM Hashes*” and then press the plus xpcrack0 key. Note that the “Import Hashes from local system” line is marked and press *next.* The list of users on your system should now be displayed and the list should look similar to that one below:



This list should look very similar to the columns you saw in SAMinside. One new additional column of relevance is also displayed. The *“<8*” column indicates which passwords contain 8 or less characters. These are potentially less secure and easier to crack. *Right Click* in the center of the window and choose *Select All.* Now all rows should be highlighted.Right click in the center of these rows and choose *Brute Force 🡪 LM Hashes* and choose *start*. This will perform a brute force attack using the Predefined dictionary of ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789. Remember that LM-Hashes are converted to all uppercase letters by default. Other letter, number and symbol combinations could have been selected to perform a more complicated cracking attack but we will use this one for simplicity sake. You should now see a screen similar to that below:

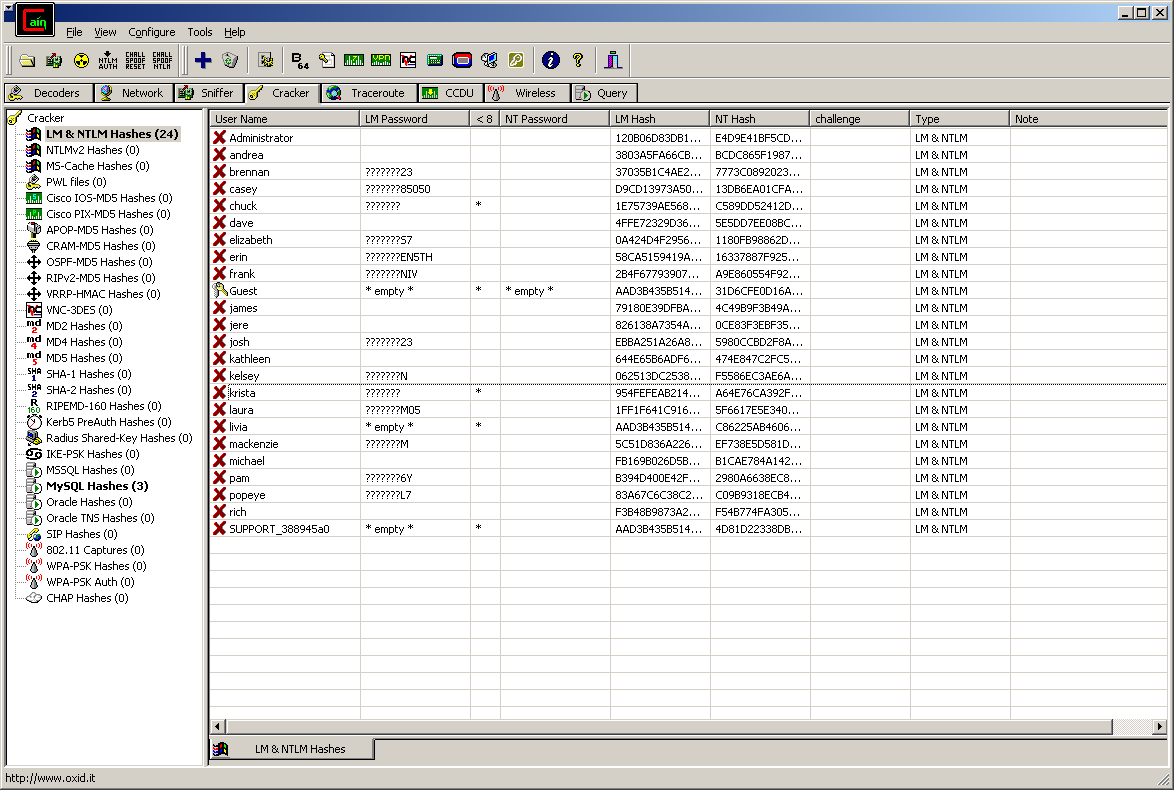


Let’s discuss some of the information displayed. The *Keyspace* section shows the current possible keys being used. The *Current password* is the password being tested. Two time fields appear. The *Key Rate* give you statistics of the number of Passwords per Seconds are being tested and the *Time Left* is an estimate of remaining time to complete all password tests with the given character list. The lower section displays results as they are discovered. Here we see two noteworthy findings:

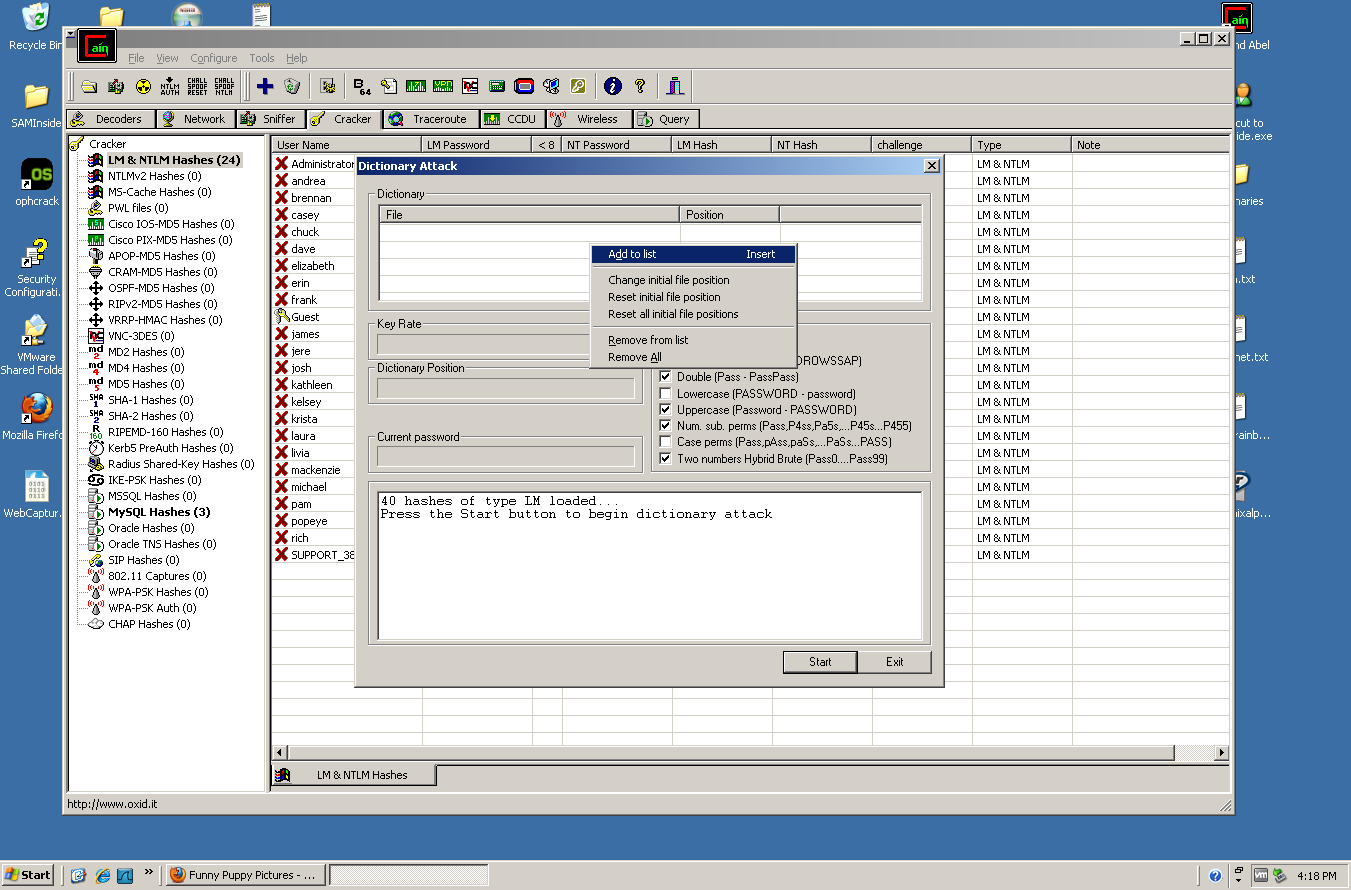
***Plaintext of 552F8B1E7E0F49CA is 2JMU***

***Plaintext of F03DBBB8BB623C69 is JMUCS***

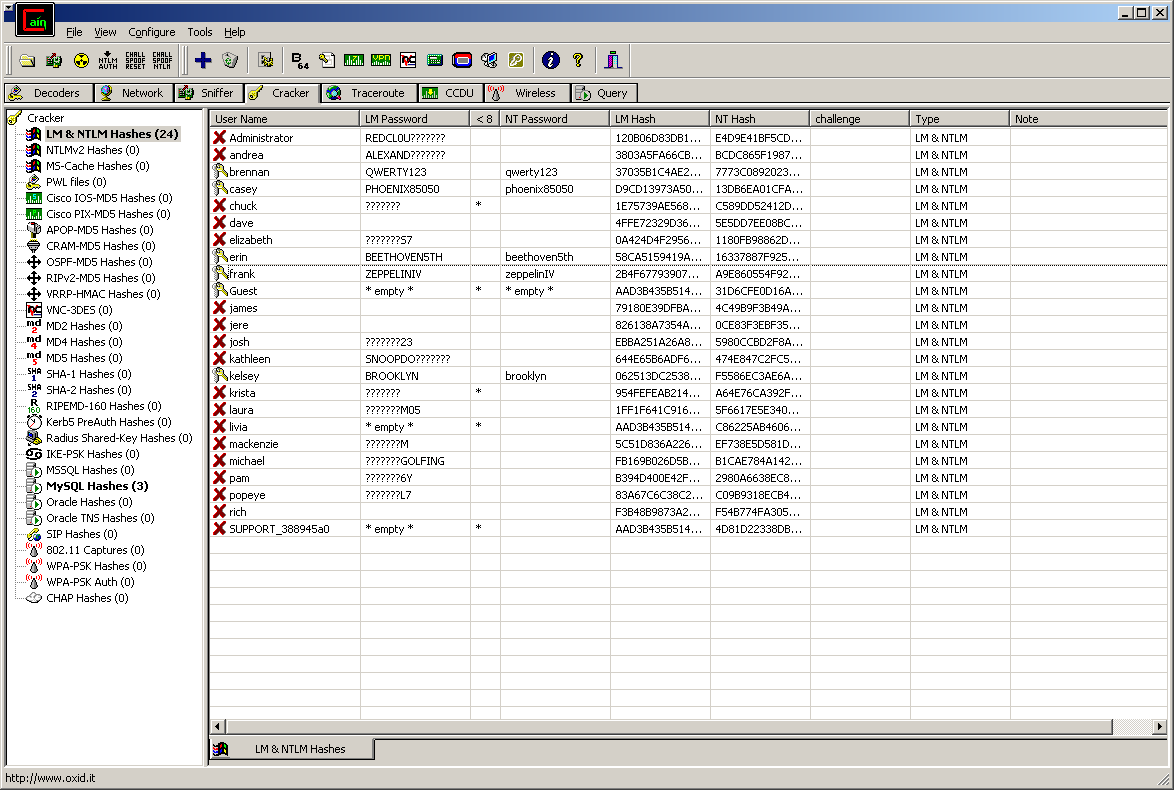
These two lines display LM hash discoveries as they pertain to the user passwords. Let your attack run for a few minutes and see what discoveries you find. When you are ready stop the attack and exit the Brute-Force Attack window. Now take a look at your main section. Are there changes in the passwords listed? Below is an example that should be similar to what you are seeing:



Now let’s run a test using a dictionary attack and compare the results. To do this you should first right click and select all. Then right click and select dictionary attack > LM Hashes. The following window should appear



*Right click* in the ***Dictionary*** section and choose Add to list. Go to your desktop and add **cain.txt**. Perform the same procedure again to add **honeynet.txt** the click *Start*. Notice the additional passwords that were discovered.



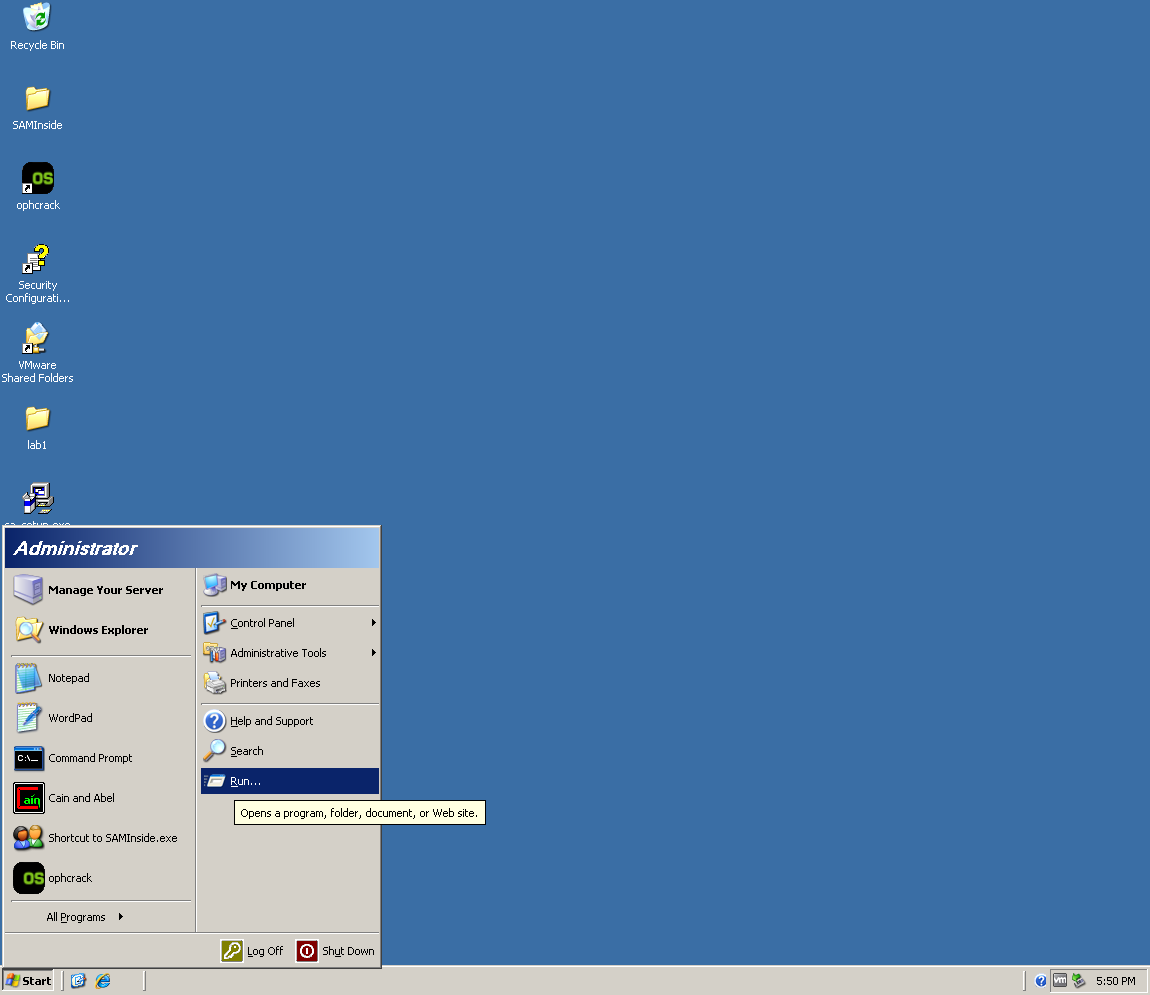
**Exercise 3 – Disabling LM Hashes in the Windows Registry**

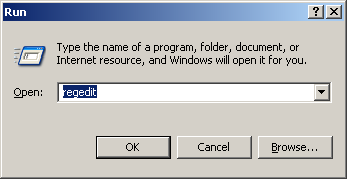
As stated earlier Windows generates and stores user account passwords by using two different algorithms. When you set or change the password for a user account to a password that contains fewer than 15 characters, Windows generates both a LAN Manager hash (LM hash) and a Windows NT hash (NT hash) of the password. These hashes are stored in the local Security Accounts Manager (SAM) database or in Active Directory. We have shown that the LM hash is relatively weak compared to the NT hash. In this exercise we will show how you can prevent Windows from storing an LM hash of your password.

**Important:**  This section contains steps on how to modify the registry. However, serious problems might occur if you modify the registry incorrectly. ***Therefore, make sure that you follow these steps carefully!*** In the event of an error you will need to drop back to your original Snapshot. On a system outside of the lab you will want to investigate using the *Backup Utility* or the *NTbackup.exe* command-line tool to create a backup of your registry.

To disable the LM Hash you need to add the DWORD value “*NoLMHash*”. This can be accomplished through the use of the Registry Editor detailed below:

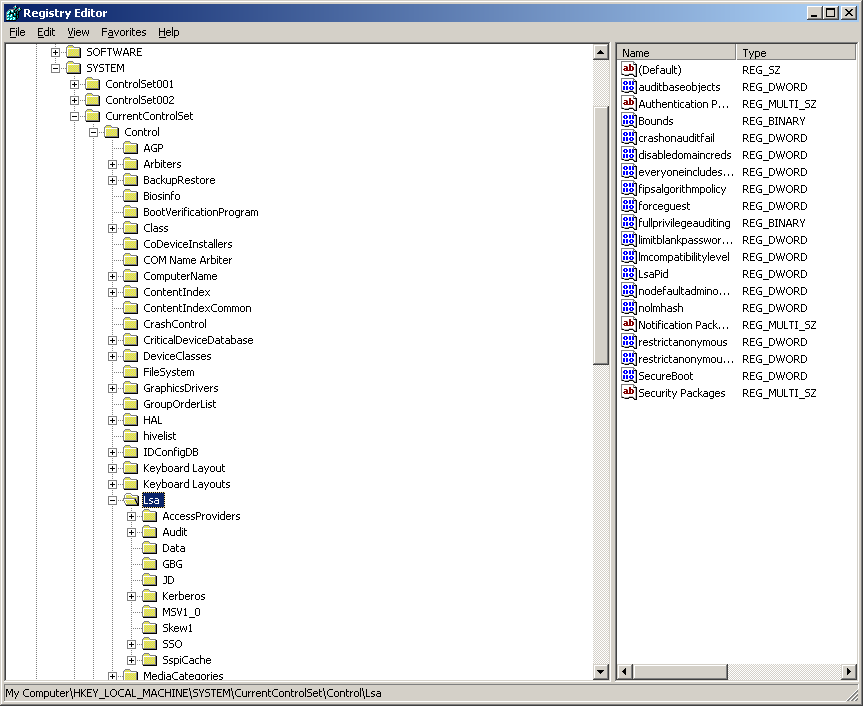
1. Click **Start**, click **Run**, type ***regedit***, and then click **OK**.



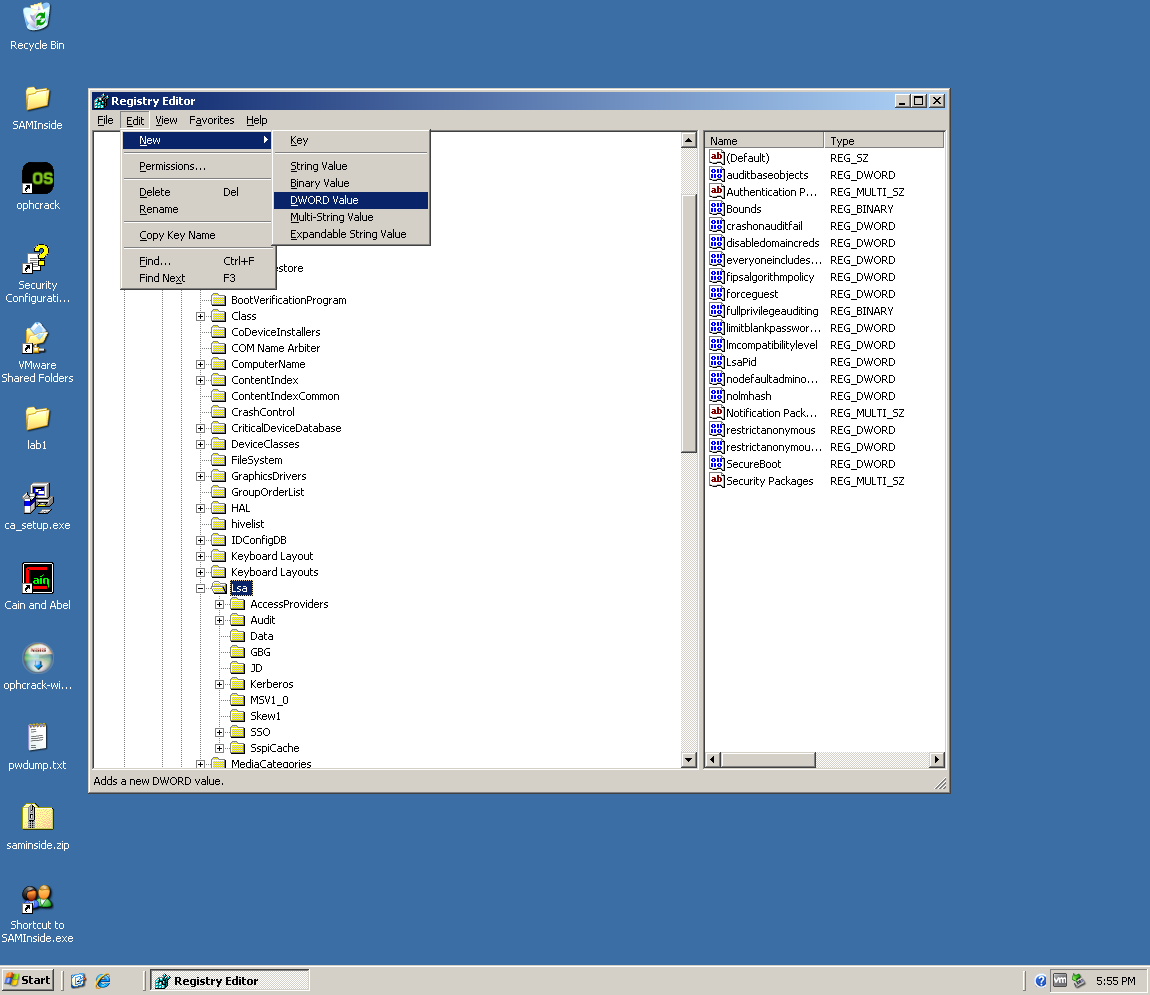


1. Locate and then click the following key in the registry:

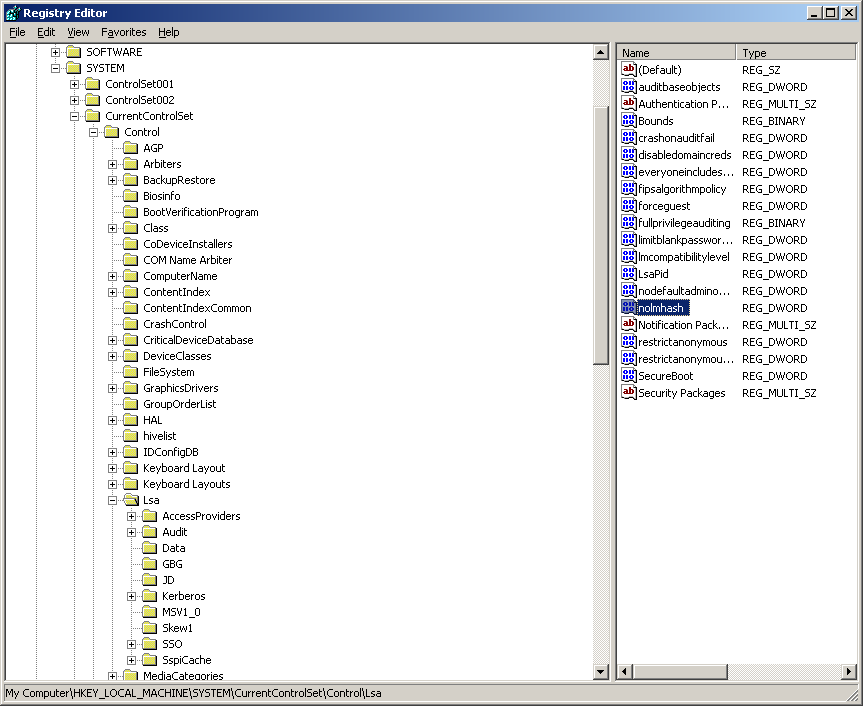
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa



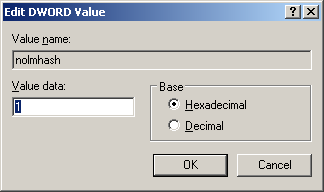
1. On the **Edit** menu, point to **New**, and then click **DWORD Value**.



1. Type ***nolmhash***, and then press **ENTER**. **Note**: it may already exist. If so, move to step 5.
2. On the **Edit** menu in the right hand column, *Right click* on the ***nolmhash*** value and click **Modify**.



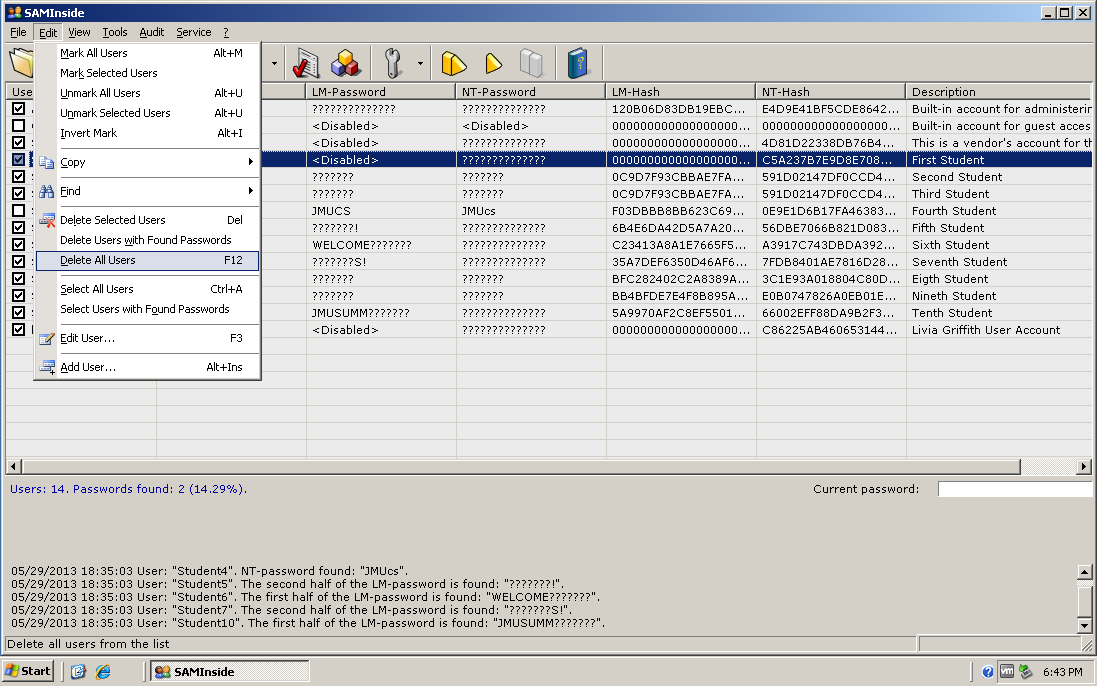
1. Type 1, and then **click OK**.



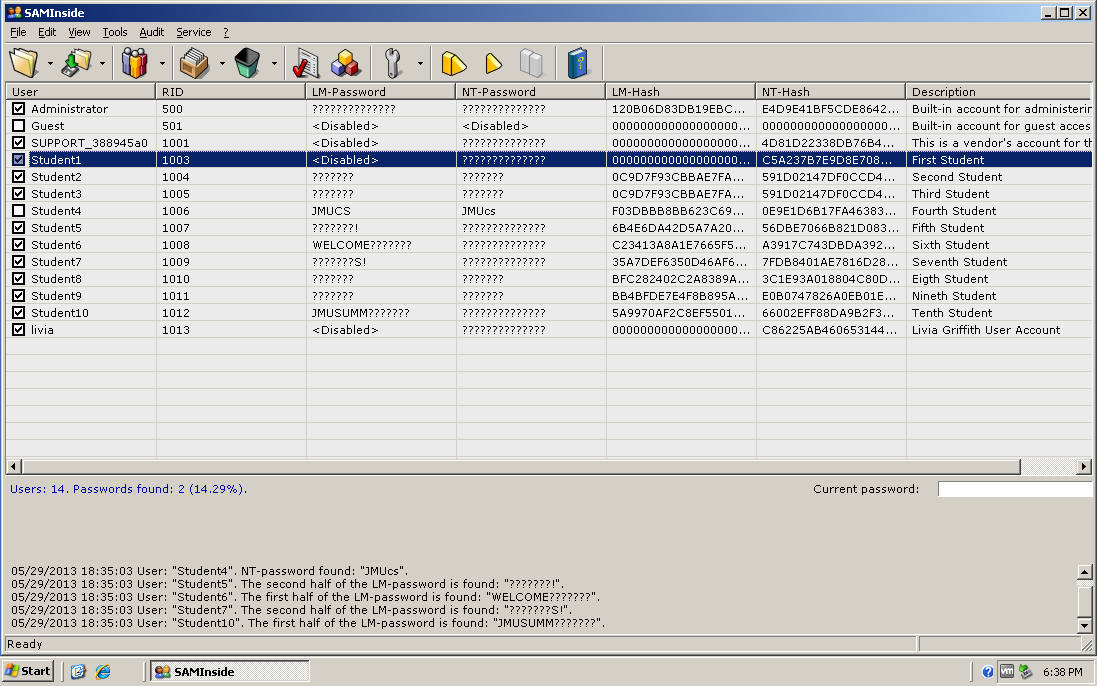
1. Restart your computer and log back in
2. Go to ***Start***, *Right Click* on **My Computer** and select ***Manage***
3. Go to ***Local Users and Groups -> Users****,* in the right hand column ***right click*** on *testuser1* and click ***Set Password…*** Change the password for *testuser1*.



1. When you receive a warning window click ***Proceed***
2. Enter a new password and click ***OK***
3. Reopen **SAMinside** by double clicking on the desktop icon.
4. *Click* the ***Edit*** tab and select ***Delete All Users***. This will delete the old users that were loaded in Exercise 1.



1. Reload the users and their hashes by clicking on ***File*** and ***Import Local Users via Scheduler***



Take note of the user whose password you changed (in the above example the user is *Student1*, on your system it should be *testuser1.* You can see that the LM-Password is now disabled and the LM-Hash contains all zeros. Also, note that all other LM-Hashes have not been zeroed out. This means that these accounts are still vulnerable to an LM-Hash attack. Close SAMinside.

**Exercise 4 – Cracking MySQL database passwords using a simple lookup table.**

MySQL is a popular free database available for download on the Internet. Versions of MySQL exist for both Windows and Linux operating systems that also help in its popularity. MySQL lists user accounts in the user table of the MySQL database. Each MySQL account can be assigned a password in which the computed hash value is stored. The original hashing method in older versions of MySQL produced a 16-byte string. In MySQL 4.1.1, the hashing method was modified to produce a longer 41-byte hash value. The current version of MySQL uses a double SHA1 hashing algorithm that is considered sophisticated

**Cracking techniques**:

Up to this point we have been using brute force and dictionary attacks to crack encrypted/hashed passwords. In this exercise we will introduce another technique: a cryptographic analysis attack using simple **rainbow tables**.

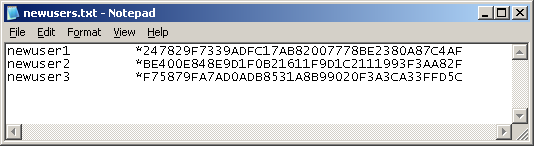
**Dictionary Attacks:**

Unlike Brute Force attacks, which are based on generating different character combinations for password guessing, dictionary attacks are attacks based on a given wordlist. Several creative wordlists of different lengths and sizes can be found free on the Internet. Complete dictionaries are only one kind of wordlist. Other lists exist containing commonly used passwords. These words are not found in the dictionary but have proven to be popular when people are creating passwords.

**Lookup Tables:**

Lookup tables are the simplest form of what is known as a rainbow table. A lookup table contains pre-computed hash values. In a sense it can be compared to creating the brute force hashes ahead of time and storing them in a table. The users can then use a simple tool, like findstr.exe, to search the table for the given hash value. If the hash value is found then the user will be able to see the clear text password. We will take a look at how this works. For this example we have generated a simple lookup table for 4 character passwords that are all alphanumeric. All passwords of length 4 or less made up of letters and/or digits are in the rainbow table (624=15 million entries). There is conceivably no real life application for such a simple table but it is useful for learning how lookup tables work. The longer the password and/or the more characters the larger the table. This can take a lot of extra storage and a long time to generate. Rainbow table **chaining** is a technique that helps in shrinking down the size of the tables. We touch upon that a bit in Exercise 5.

Let’s test out our simple rainbow table. Open the file newusers.txt by double clicking it on your desktop.



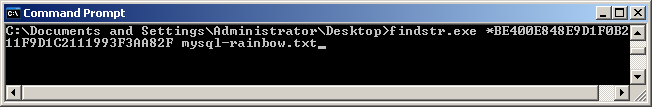
Now go to ***Start🡪 Command Prompt*** and open up a command prompt and then cd to your Desktop:

***C:\Documents and Settings\Administrator>cd Desktop***

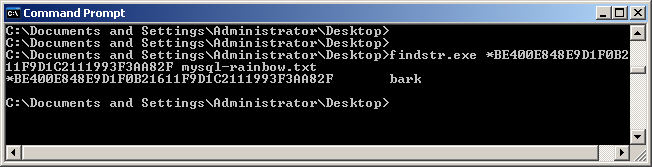
Findstr.exe is a program that will search a file for a given string. The syntax for use is:

C:\>**findstr.exe** <*some\_string\_of\_characters> <* ***filename.txt>***

You will want to *copy* and *paste* the first hash value from the newusers.txt file to the command line as an argument to *findstr.exe*. Type the following at the command line, copying and pasting the hash value from the newusers.txt file to search for the first hash value:



The *findstr.exe* command should return the hash string followed by the word “bark” which is the password.



**Exercise 5 – More Cracking MySQL database passwords while comparing multiple cracking techniques.**

.

**What Are Rainbow Tables?**

A **rainbow table** is a pre-computed table for reversing cryptograhic-hashing functions used for cracking password hashes. These tables are usually used in recovering the plaintext password, up to a certain length consisting of a limited set of characters. A rainbow table is a lookup table offering a time-memory tradeoff. A common application of rainbow tables is to make attacks against hashed passwords feasible.

## So how does it work?

A rainbow table is a compact representation of related plaintext password sequences (or chains). Each chain starts with an initial password, which is passed through a hash function. The resulting hash is then fed into a reduction function, which produces a different plaintext password. The process is then repeated for a fixed number of iterations. The initial and final passwords of the chain comprise a rainbow table entry.

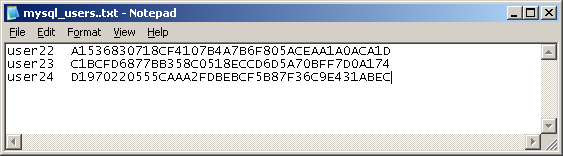
Increasing the length of the chain decreases the size of the table. It also increases the time required to iterate over each chain, and this is the time-memory trade-off of the rainbow table. In a simple case of one-item chains, the lookup is very fast, but the table is very big. Once chains get longer, the lookup slows down, but the table size goes down.

## How can I protect my password(s)?

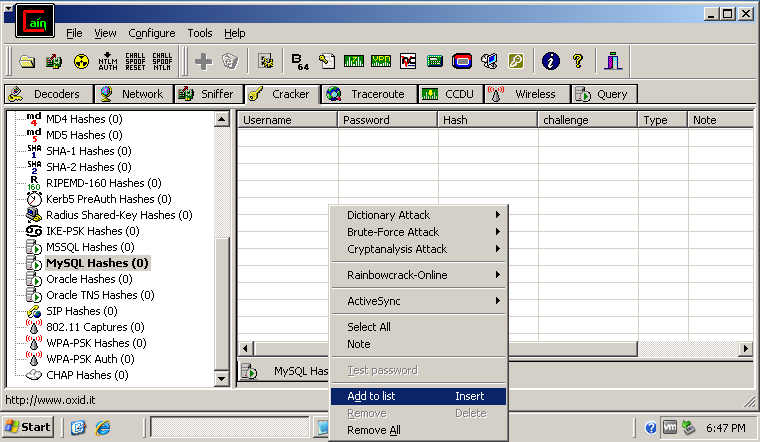
Rainbow tables tend to have little or no success when extrapolating outside the range of symbols or password length computed into the table. So, choosing a password that is longer or contains symbols not accounted for inside a Rainbow table can be very effective. Because of the sizable investment in computing processing, Rainbow tables beyond 9 places in length and a character set containing symbols are not yet common. As development around Rainbow tables is continuing, the length of passwords that are endangered is growing.

**Comparing cracking techniques on simple unsophisticated MySQL user passwords:**

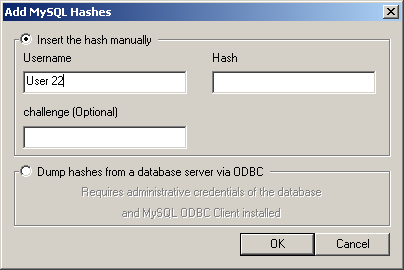
To start this exercise start up ***Cain and Abel*** by clicking the icon located on your desktop. Loading mysql hashes into Cain and Abel requires either a direct connection to a database or manual entry. In this exercise we will manually enter 3 users by copying and pasting their hash values. Notice on you desktop a text icon named ***mysql\_users.txt***. *Double click* on the icon, which should open up the file in ***Notepad***.



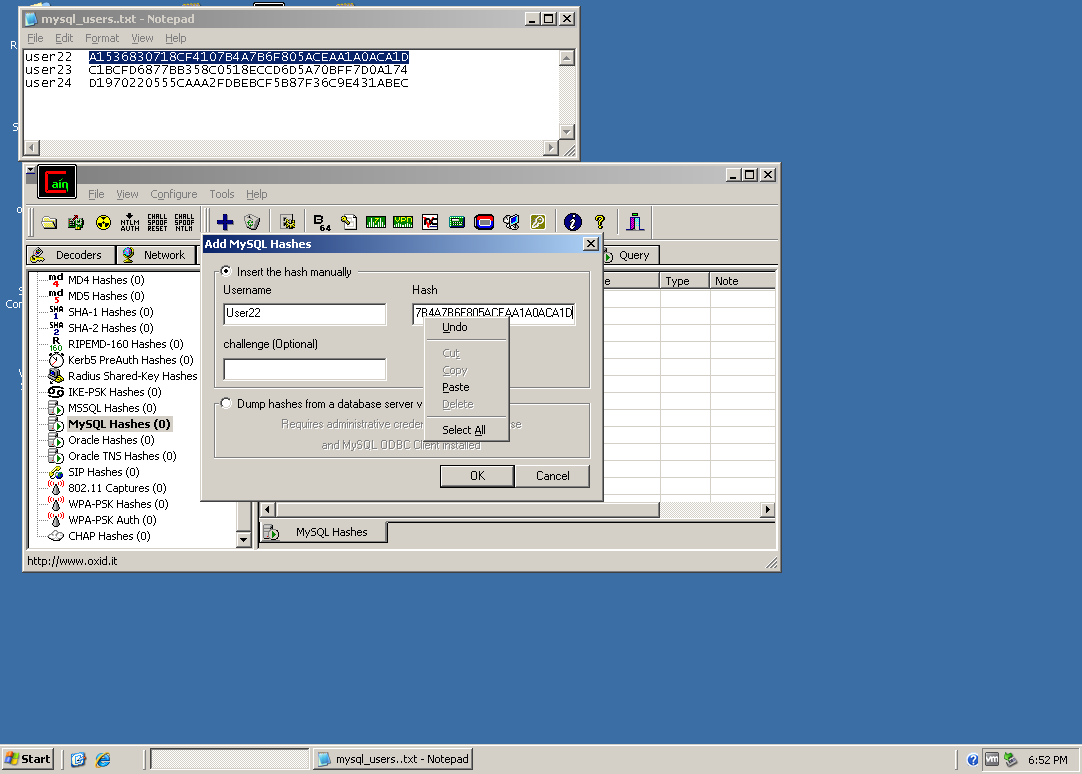
Now go to your Cain and Abel Window. Choose the ***Cracker*** xpcrack tab in the third row. In the Left-hand column click on ***MySQL Hashes***. Now *right-click* in the large open area to the right that appears lick a spreadsheet and choose ***Add to list***.



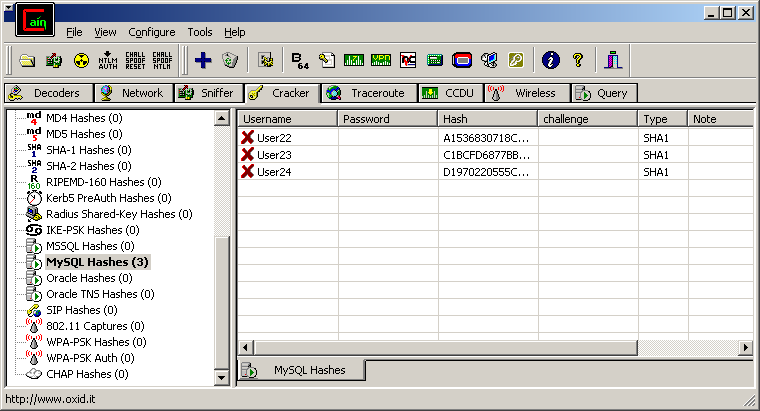
The following window should appear. Add User22 in the Username field:



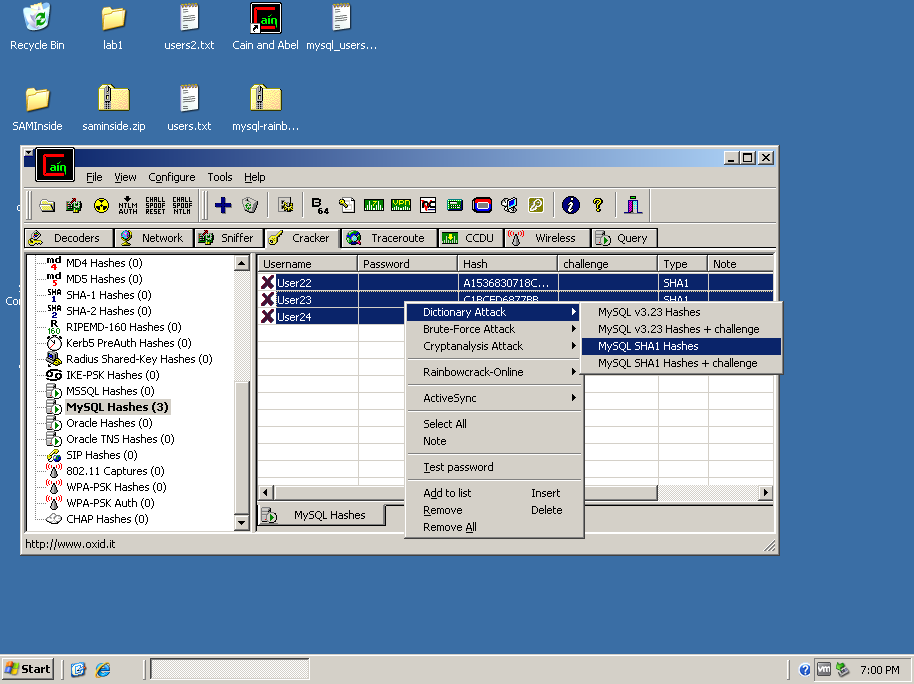
You will now need to highlight the hash value in the Notepad User list and copy/paste it to the Hash section of the above window.

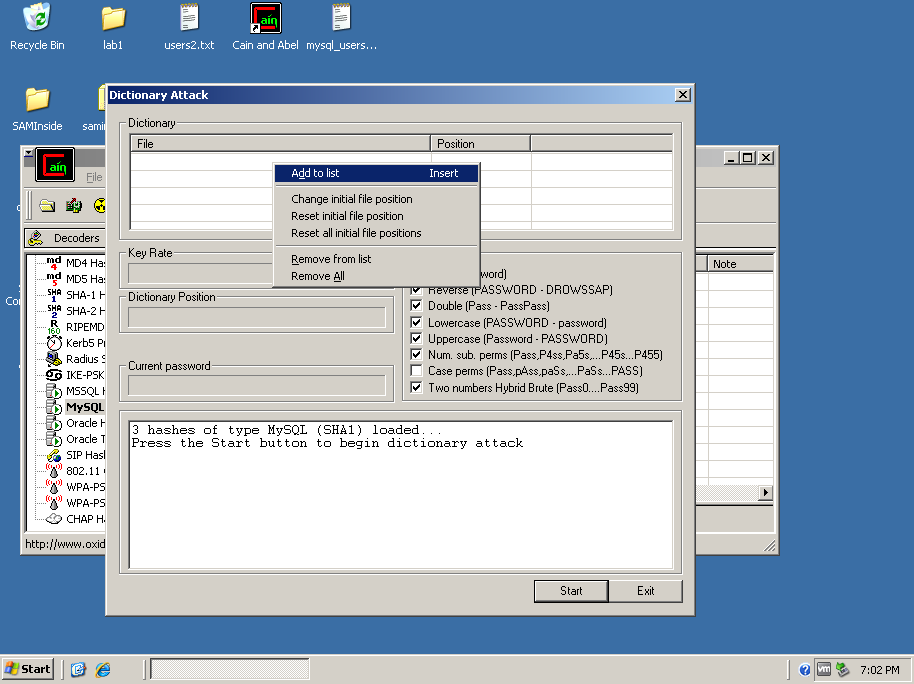
****

Now click ***OK*** and notice that this line has been added to the main body of **Cain and Abel**. Also note that the Type has been filled in as SHA1. As we mentioned earlier MySQL uses SHA1 hashing. Now repeat the same procedure to add User23 and User24. Be sure to cut and paste their hashed passwords. At the end your table should look similar to the one below:

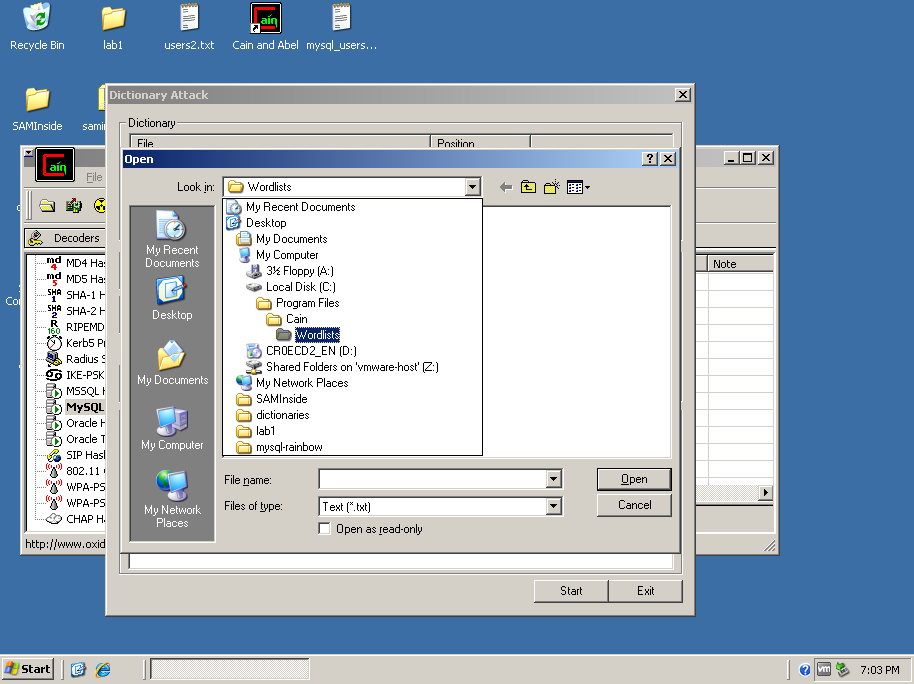


Now we will run the three different cracking techniques and see what happens. We will begin using a dictionary attack. Right-click in the body of Cain and Abel and choose Select All. All three User lines should now be highlighted. Right click in the center of the highlighted area and choose *Dictionary Attack 🡪 MySQL SHA1 Hashes*. The Dictionary Attack should appear.

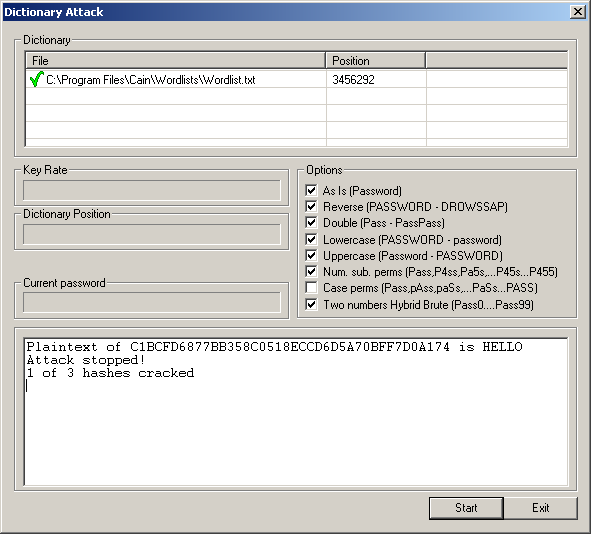


****

In the Dictionary section, remove all previously populated lists (right click > select Remove from lid). Next, click Add to list and navigate to the *Wordlists* folder located in *C:\Program Files\Cain* and open ***Wordlists.txt***:



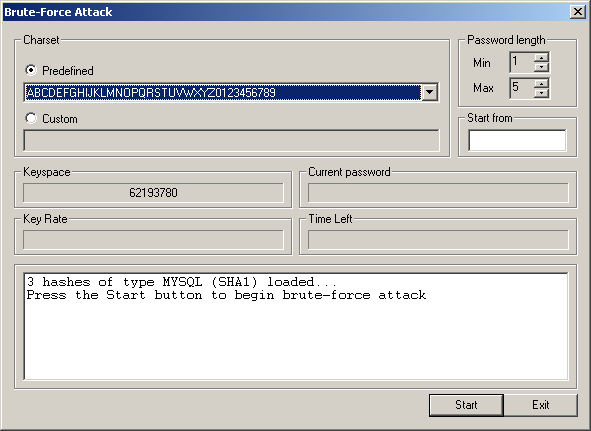
This wordlist.txt file should now appear in the Dictionary section. Choose Start to begin the dictionary attack. This will take a minute or two to run. Watch to see if any passwords are discovered. At the end you should notice one password has been cracked as can be seen below:



Click Exit and notice that this password has now been filled in next to User23. Now let’s run a Brute Force attack and see what happens. While the Users are all still highlighted right click and choose *Brute Force 🡪 MySQL SHA1 Hashes.* When the *Brute-Force* Window appears go to the Password Length section and change the limit from 1 to 16 to 1 to 5. For the sake of saving time we will make this change since we know that our passwords are only five characters in length.

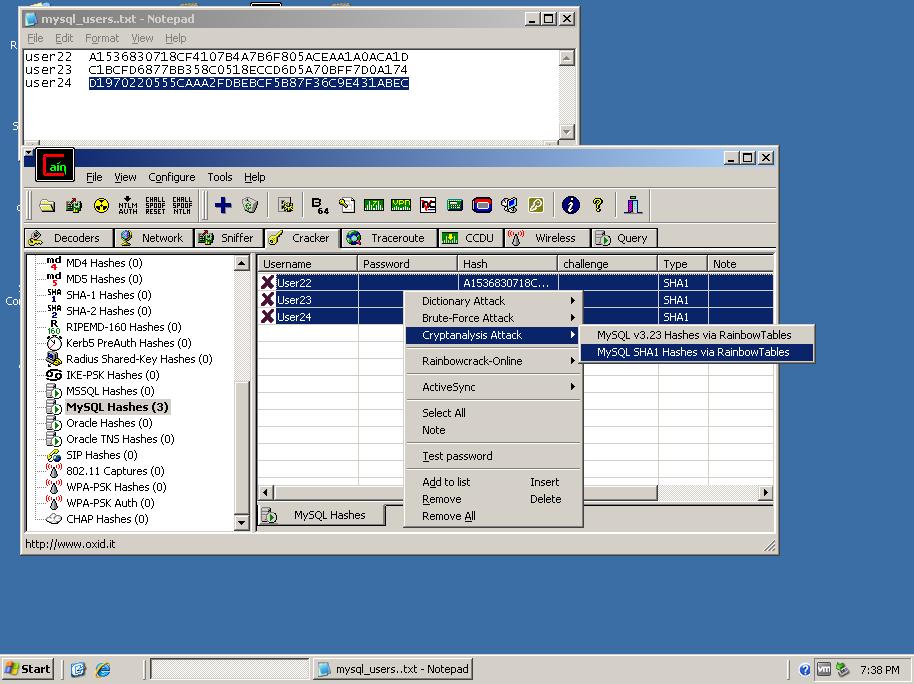
In a real attack five character would most likely not be sufficient. Now we will choose our predefined character set. Make sure the button box next to Predefined is selected and click on the arrow to list the various character sets. For this example, again in the interest of time, we will choose the Capital alpha Numeric list (you will need to scroll up to find this and it is the 2nd choice from the top):

**ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789**

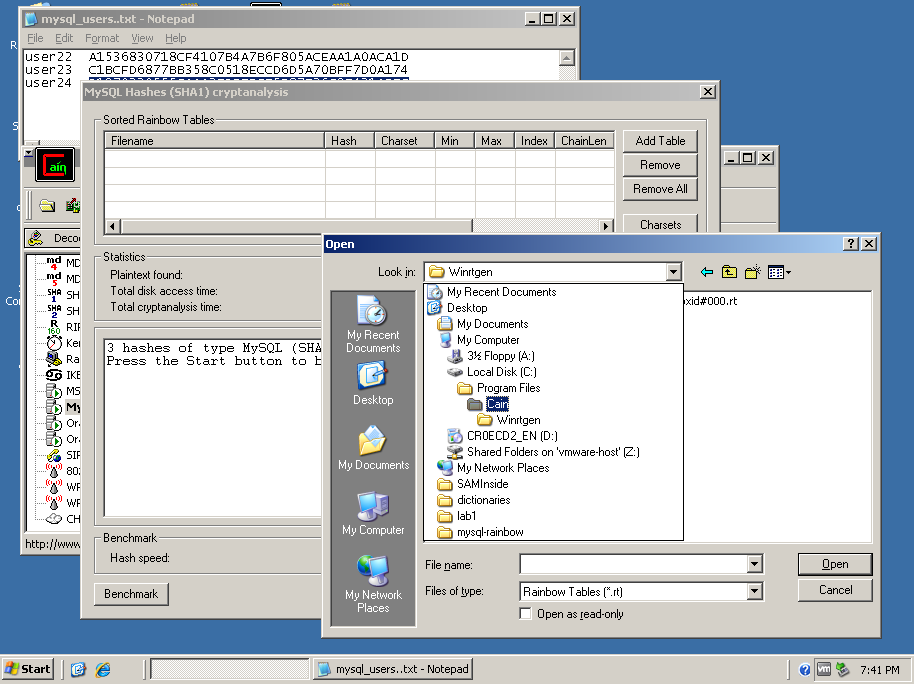
****

Click ***Start*** to begin the attack. Note that User22 is immediately displayed since we had already cracked this password. This time we discovered our other two passwords as well. Now all three have been cracked. Exit the Brute-Force Attack and notice now that all three passwords have been filled in.

Let’s compare this outcome with running against a simple Rainbow Tree. Since all passwords have been found we will need to delete these entries and re-enter them. Right click in the highlighted area and choose Remove all. Select Yes to Delete all entries. Now click on the plus xpcrack0 key and add the three users, copying and pasting their hashed passwords as you had done before. Once you have completed adding the users, right-click to select all and then right click and select Cryptanalysis Attack 🡪 MySQL SHA1 via Rainbow Tables:



In the MySQL Hashes (SHA1) cryptanalysis window click Add Tables. Then navigate to the C:\Program Files\Cain\Winrtgen folder:



Choose the file named: ***mysqlsha1\_alpha-numeric#1-5\_0\_2400\*400000\_oxid#00.rt***

Now click start and notice how much faster the passwords are cracked. Although this is a simple example with unsophisticated passwords this gives you a feel for how dangerous Rainbow Table attacks can be.

**Summary:**

These exercises gave you a sample of a few of the sophisticated programs used for password cracking that are available on the internet. The only defense to password cracking is a good offense. This consists of:

* Creating good passwords that are
  + Lengthy in size
  + Contain upper, lowercase letter as well as numbers and symbols
  + Are hashed using more sophisticated methods
  + Have a salt applied to them to avoid duplicate hashed values

**References:**

Account Passwords and Policies  
[http://www.microsoft.com/technet/prodtechnol/windowsser...](http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/security/bpactlck.mspx" \t "_blank)  
  
How to prevent Windows from storing a LAN manager hash of your password in   
Active Directory and local SAM databases  
[http://support.microsoft.com/kb/299656/en-us/](http://support.microsoft.com/kb/299656/en-us/" \t "_blank)

Hack Passwords with Cain and Abel

### <http://www.101hacker.com/2010/11/hack-passwords-with-cain-and-abel.html>

# How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases

### <http://support.microsoft.com/kb/299656>

# Password Hashing in MySQL

### <https://dev.mysql.com/doc/refman/5.5/en/password-hashing.html>

### 