Identity, Authentication, and Access Control

License

Contents

General notes about the labs

Often the lab instructions are intentionally open ended, and you will have to figure some things out for yourselves. This module is designed to be challenging, as well as fun!

However, we aim to provide a well planned and fluent experience. If you notice any mistakes in the lab instructions or you feel some important information is missing, please feel free to add a comment to the document by highlighting the text and click the comment icon ( ), and we will try to address any issues. Note that your comments are public.

You should maintain a lab logbook / document, which should include your answers to the questions posed throughout the labs (in this colour).

Preparation

Start by loading the latest version of the LinuxZ template from the IMS system. If you have access to this lab sheet, you can read ahead while you wait for the image to load.

> To load the image: press F12 during startup (on the boot screen) to access the IMS system, then login to IMS using your university password. Load the template image: LinuxZ (load the latest version).

Once your LinuxZ image has loaded, log in using the username and password allocated to you by your tutor.

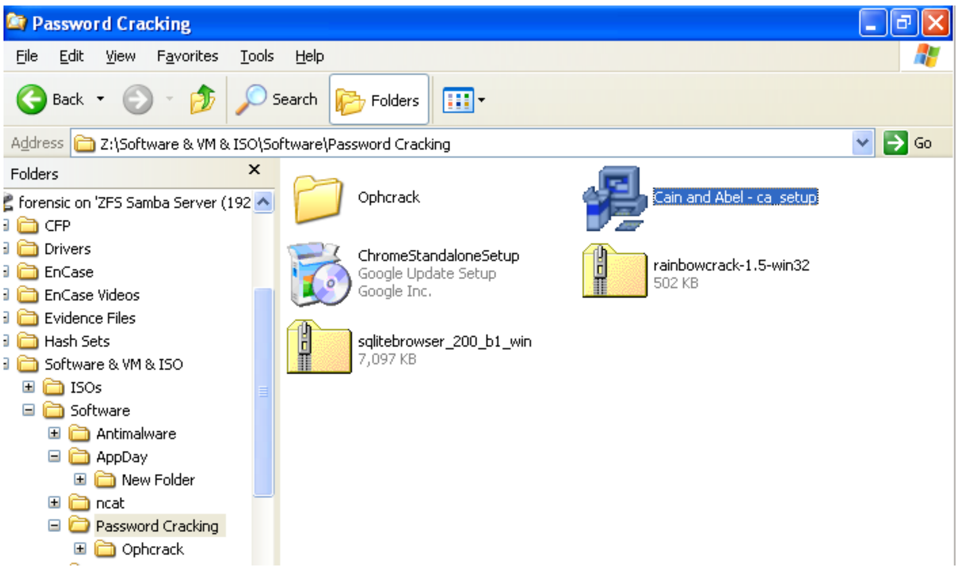
Using the VM download script (as described in the previous lab), download and **start** this VM:

- WinXP Pro - bridged with network share

This lab could be completed on almost any Windows system with Cain and Abel (you don't need to download the installer from the Internet if you are working in our labs). If you use a different version of Windows the steps will be slightly different, and will need some alteration. Older versions of Windows (XP and older) are best for illustrating the problems with LM hashes.

Start by installing Cain and Abel onto the Windows VM.

> As illustrated below, the installer can be found on the network drive in "Z:\Software & VM & ISO\Software\Password Cracking".



Installing Cain and Abel

> Copy the installer to the local disk (for example, My Documents) and then run it.

Introduction

This lab covers the foundations of authentication, identity, and access control. These labs are performed using Windows, although the concepts apply similarly to Unix systems.

In order for computer systems to treat users appropriately, the identity of each end user needs to be established and recorded. This process typically starts with a user *identifying* themselves. For example, by clicking on or typing their username when logging in. Next they prove their identity. This is known as *authentication*, when someone proves their identity, and subsequently their identity is associated with their session on the system. Later comes *authorisation*: what they are allowed to perform.

Identity

In Windows NT based systems (such as Windows 2000, XP, Vista, 7, and 8), a unique ID number is used to represent each identity on the system, known as a security identifier (SID). When making decisions about what actions allowed, the Windows security reference monitor (SRM) does not consider the username, such as “cliffe”, but rather the SID, such as “S-1-5-21-3621811015-3361144348-30301820-1013”. SIDs can also represent computers, or groups of users.

When you start a program, it runs as your SID, and any security sensitive action it takes is only allowed if that identity has permission to perform the action.

The environment variable 'username' contains a human-readable *username*. If you are not familiar with the term “environment variable”, then search for some information on the Internet.

From a command prompt, run the following command to display the current username:

```
echo My username is %username%
```

List the SIDs of all users on the current machine:

```
wmic useraccount get name,sid
```

Run this command and compare the output:

```
wmic useraccount where name='%username%' get sid
```

Note: On some Windows systems (although typically not Windows XP) the whoami command is also available, which also provides related information.

Question: Given the above, what command could you type to output the SID of a specific user named “Alice”?

The association between SID and other identity related information is visible via the registry editor. The Windows registry is a database that stores various configuration settings for the operating system, drivers, and application software. *Be careful, editing of registry values can result in serious consequences.*

Run:

```
    regedit
```

Browse to “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ ProfileList”

Select each SID and record the ProfileImagePath value.

Question: What do these represent?

There are certain well-known SIDs that always have a special meaning, or are always associated with specific types of users.

Question: Using the Internet, identify the meaning of these well-known SIDs:

S-1-1-0:

S-1-0-0:

S-1-5-21*domain*-501:

S-1-5-21*domain*-500:

Is the user that you are logged in as in the Windows XP system one of the above?

Typically normal users on the system have SIDs with a *relative identifier (RID)* (that is, the last section of a SID) of 1000+. The first normal user has a RID of 1000, the next 1001, and so on. What this means is that given a SID, regardless of the username, we can tell if it refers to a normal user, or an account defined by the operating system.

Question: Would changing the username of the Administrator account make it impractical for a local user to identify that the account is an admin account? Why not?

Question: If you deleted a Windows user account and then created a new account with the same username, it would not have access to all the same permissions of the original user. Explain why.

Using the appropriate feature in Control Panel, create a new user named “Joe”.

Set Joe's password to “Joe” (After creating the account, you can edit the user in order to add a password)

You can run a command as another user by using the 'runas' command.

Look at the available command line options, by running this command:

```
runas /?
```

Start notepad as Joe:

```
runas /user:Joe notepad
```

Goto the taskmanager (Ctrl+Shift+Esc), and view the processes tab, and confirm that notepad is running as the Joe user.

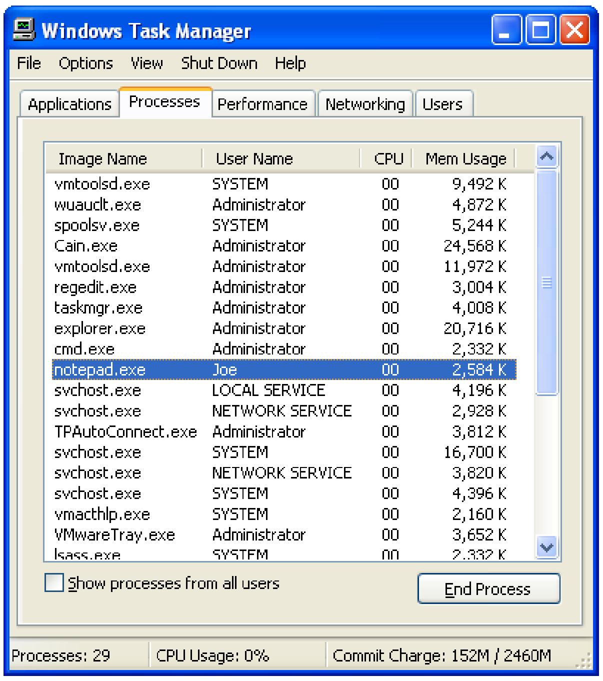


Image Name	User Name	CPU	Mem Usage
vmtoolsd.exe	SYSTEM	00	9,492 K
wuauclt.exe	Administrator	00	4,872 K
spoolsv.exe	SYSTEM	00	5,244 K
Cain.exe	Administrator	00	24,568 K
vmtoolsd.exe	Administrator	00	11,972 K
regedit.exe	Administrator	00	3,004 K
taskmgr.exe	Administrator	00	4,008 K
explorer.exe	Administrator	00	20,716 K
cmd.exe	Administrator	00	2,332 K
notepad.exe	Joe	00	2,584 K
svchost.exe	LOCAL SERVICE	00	4,196 K
svchost.exe	NETWORK SERVICE	00	2,928 K
TPAutoConnect.exe	Administrator	00	3,812 K
svchost.exe	SYSTEM	00	16,700 K
svchost.exe	NETWORK SERVICE	00	3,820 K
svchost.exe	SYSTEM	00	4,396 K
vmacthlp.exe	SYSTEM	00	2,160 K
VMwareTray.exe	Administrator	00	3,652 K
lsass.exe	SYSTEM	00	2,332 K

☐ Show processes from all users

Processes: 29 CPU Usage: 0% Commit Charge: 152M / 2460M

Notepad and users

Question: Does this mean that this notepad will act differently or have different permissions to when you just run “notepad”. Why?

Type some text and save the file to the desktop. You will notice that the file will not actually appear on *your* desktop.

Question: Where is the file?

Authentication

The above instructions included creating a password that is the same as the username. When this happens, this is known as a “**joe account**”. This is a particularly bad choice of password, because it can be guessed very easily, without any prior knowledge. This is an example of a **low entropy** password. Entropy, in this context, relates to the degree of randomness within the password. Using a long combination which includes non-alpha numeric characters can result in high entropy, a stronger password. A word from a dictionary, or a birth date are easy to guess. A password that is long and highly random will be harder to guess. However, when a password is extremely random it can be difficult to remember. Therefore, it can be helpful to use mnemonics to create passwords. For example: “I hate to eat *Fish* on a *Friday* the 13th” could result in the password: “Ihte*F*oa*F*t13”.

For local accounts, Windows stores the passwords in hash form in the Security Accounts Manager (SAM) database, which is a Windows registry hive. One-way hash functions enable us to store representations of passwords, without actually storing the password itself. An important feature of a one-way hash function is that the same input will always produce the same output, but it is impossible to determine the input used to generate a particular output, except by trial and error.

Question: Why is this preferable to simply storing the password clear-text in a file?

Password cracking

Create the following user accounts (as restricted users, rather than Administrators), and note how strong you think each password is:

- username: Alice, password: ljasdfpouklsdf213
- username: Bob, password: cat
- username: Clive, password: dog1

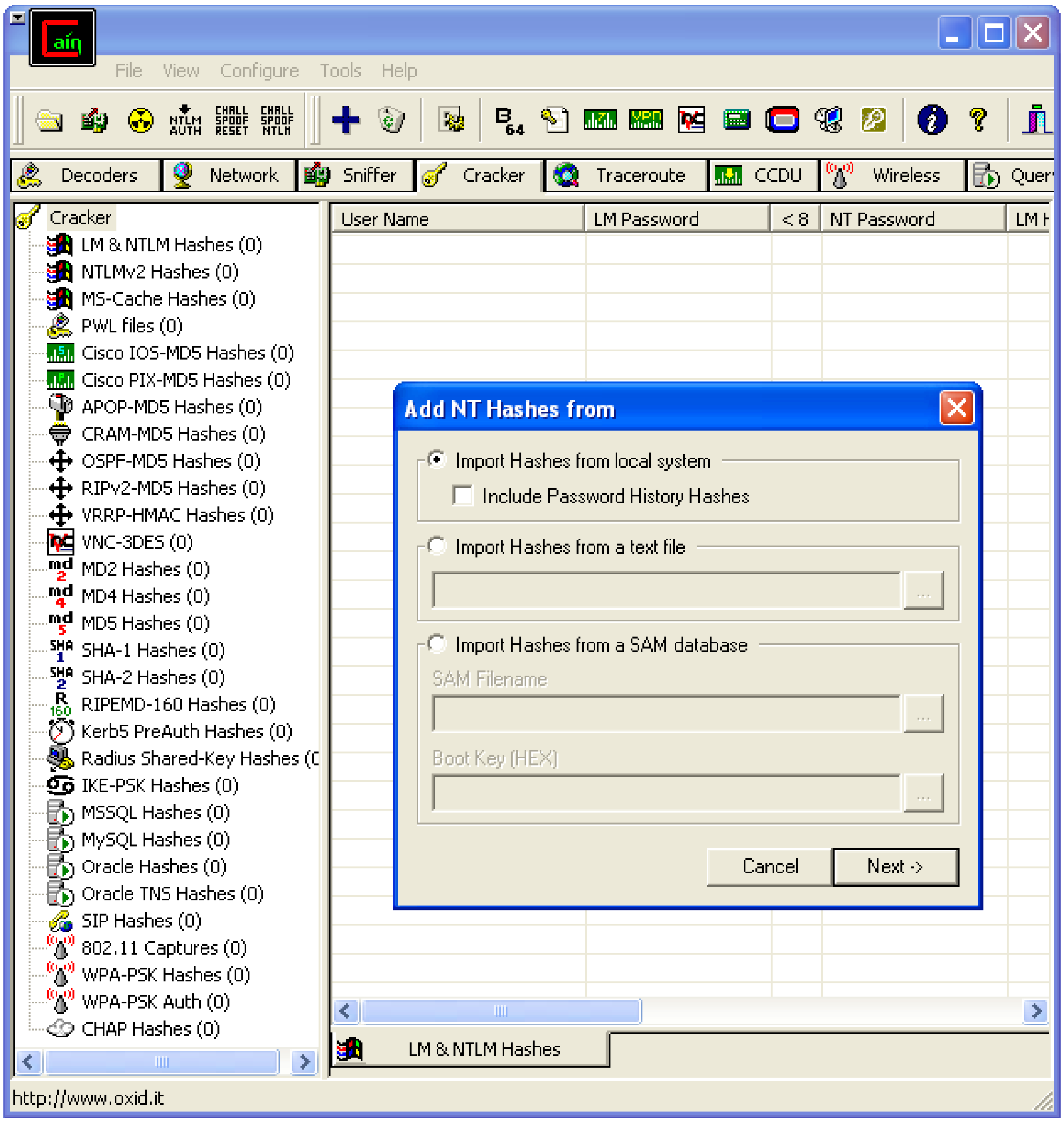
- username: Dee, password: 01012012
- username: Eve, password: Ihte*F*oa*F*t13
- username: Fred, password: s3cr3t

The SAM database is locked while Windows is running, and depending on the version of Windows, the database may be also encrypted using Syskey (although the decryption key is typically stored in the registry). This can make extracting the hashed passwords more difficult, but not impossible. A number of different tools can be used to extract the SAM.

Once extracted, a password cracking program can be used to guess the passwords, by attempting likely character combinations. *Dictionary attacks* use word lists to guess passwords based on words, while *brute-force attacks* attempt every possible combination of characters, and can take much longer.

Start Cain and Abel (aka Cain), an open source security tool; one of *many* tools that can crack SAM hashes. If you followed the instructions to install Cain, an icon will be on the desktop.

Click the “Cracker” tab, right click the empty list, and click “Add to list”. Leave “Import hashes for local system” selected, and press “Next”.



Add NT Hashes from

[✓ Import Hashes from local system]

[ ] Include Password History Hashes

Import Hashes from a text file

Import Hashes from a SAM database

SAM Filename

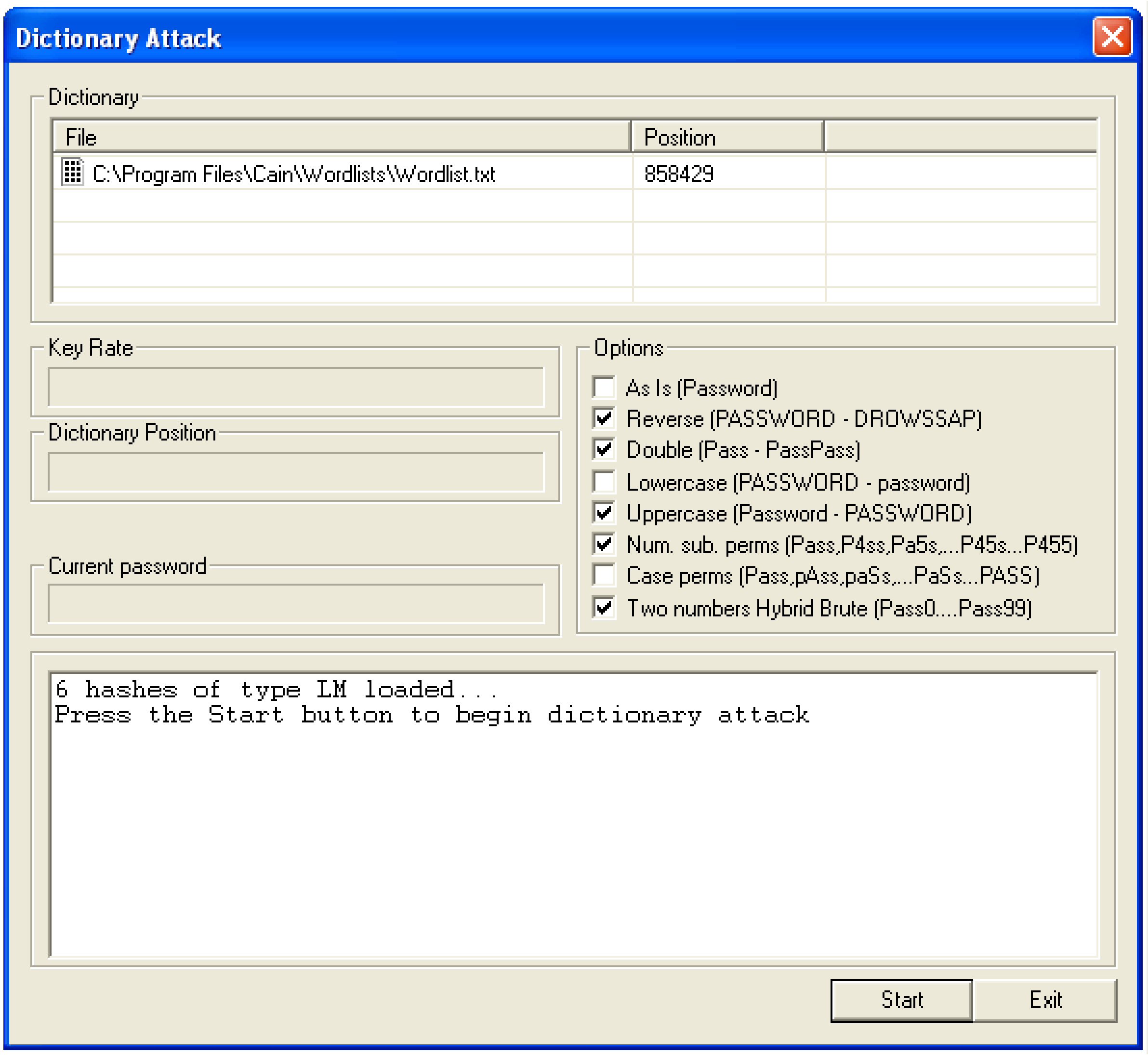
Boot Key (HEX)

Cancel   Next ->

Importing hashes from a local SAM database

Note that you can now see the LM and NT *hashes* for each password. This is how the passwords are stored on disk on our Windows host.

Select all the accounts, and right click, “Dictionary attack”, on “LM Hashes”. Add the dictionary (right click and select the dictionary) from “C:\Program Files\Cain\Wordlists\Wordlist.txt”. Enable (check) the “As is” option (leaving the other options checked).



File	Position
C:\Program Files\Cain\Wordlists\Wordlist.txt	858429

Options

- [ ] As Is (Password)
- [✓] Reverse (PASSWORD - DROWSSAP)
- [✓] Double (Pass - PassPass)
- [ ] Lowercase (PASSWORD - password)
- [✓] Uppercase (Password - PASSWORD)
- [✓] Num. sub. perms (Pass,P4ss,Pa5s,...P45s...P455)
- [ ] Case perms (Pass,pAss,paSs,...PaSs...PASS)
- [✓] Two numbers Hybrid Brute (Pass0....Pass99)

```
6 hashes of type LM loaded...
Press the Start button to begin dictionary attack
```

Starting a dictionary attack

Click start.

After the dictionary attack is complete, click “Exit”, and view the LM Password field.

Question: Which of the passwords were cracked??

Attempt a brute-force attack. Leave this running as long as possible.

This *may take some time* (you can stop once you think you have found enough), so I suggest you **continue with the other tasks while the password cracking is running**, and come back to it to see which of the passwords are cracked and how long it took.

Windows XP and earlier typically store “LM hashes”. Note that due to weaknesses in the way LM hashes are calculated it is possible to crack these in a relatively short amount of time. They are particularly weak because they are case insensitive, the password is limited to 14 characters length and are divided into two 7 character strings before being hashed separately. All of these factors lead to the passwords being easy to crack. Modern PCs can crack LM password in a matter of hours.

Techniques such as pre-computed hash tables (such as "rainbow tables") can make cracking even faster. https://crackstation.net/, uses techniques such as these.

Copy one or more LM hashes and paste them into CrackStation, to see if that can crack any faster.

Tip: you can right-click, Export to a file, then open the file to copy the hashes.

Note that other hash functions are stronger and take longer to crack.

Newer versions of Windows can be configured to not store LM hashes.

Question: Given the inherent security weaknesses, why would anyone have enabled LM hashes on Windows XP when a more secure alternative existed?

Question: Search the Internet for information on how to disable LM hashes on Windows XP:

Question: Which of the passwords were cracked? Does this match the password strength you noted earlier?

Windows NT includes a Local Security Policy dialog, which can be used to configure security settings. Open Local Security Policy, from the Control Panel, Administrative Tools (Hint: may be found under "Performance and Maintenance/Administrative Tools" when in WinXP category view).

Under "Security Settings", "Account Policy", "Password Policy", set a minimum password length to what you believe is a safer value. Remember, the longer a password is, the longer it would take to crack it.

Question: Justify your password length rule:

Look at the other password configuration options available.

Question: List options that would be appropriate for a small business computer containing point-of-sale and accounting software, and describe the security benefits of those options:

Enable the password rules you believe would be appropriate.

Test your rules by attempting to change the passwords of existing users, in ways that should be prohibited.

Note that in larger organisations that use Windows, *domain accounts* can be managed over a network by a *domain controller*, a server that manages accounts. Domain accounts are centrally managed, and can be configured so that users only need one password to login to any computer that is connected to the domain, rather than having accounts individually managed on each computer.

Access Control

Access control defines which entities are authorised to access specific resources.

Every file in an NTFS filesystem has an access control list (ACL), which lists all of the subjects (typically users) that are authorised to have access. Each entry in a NTFS ACL is known as an access control entry (ACE). An ACE defines the permissions granted (or denied) to a specific SID.

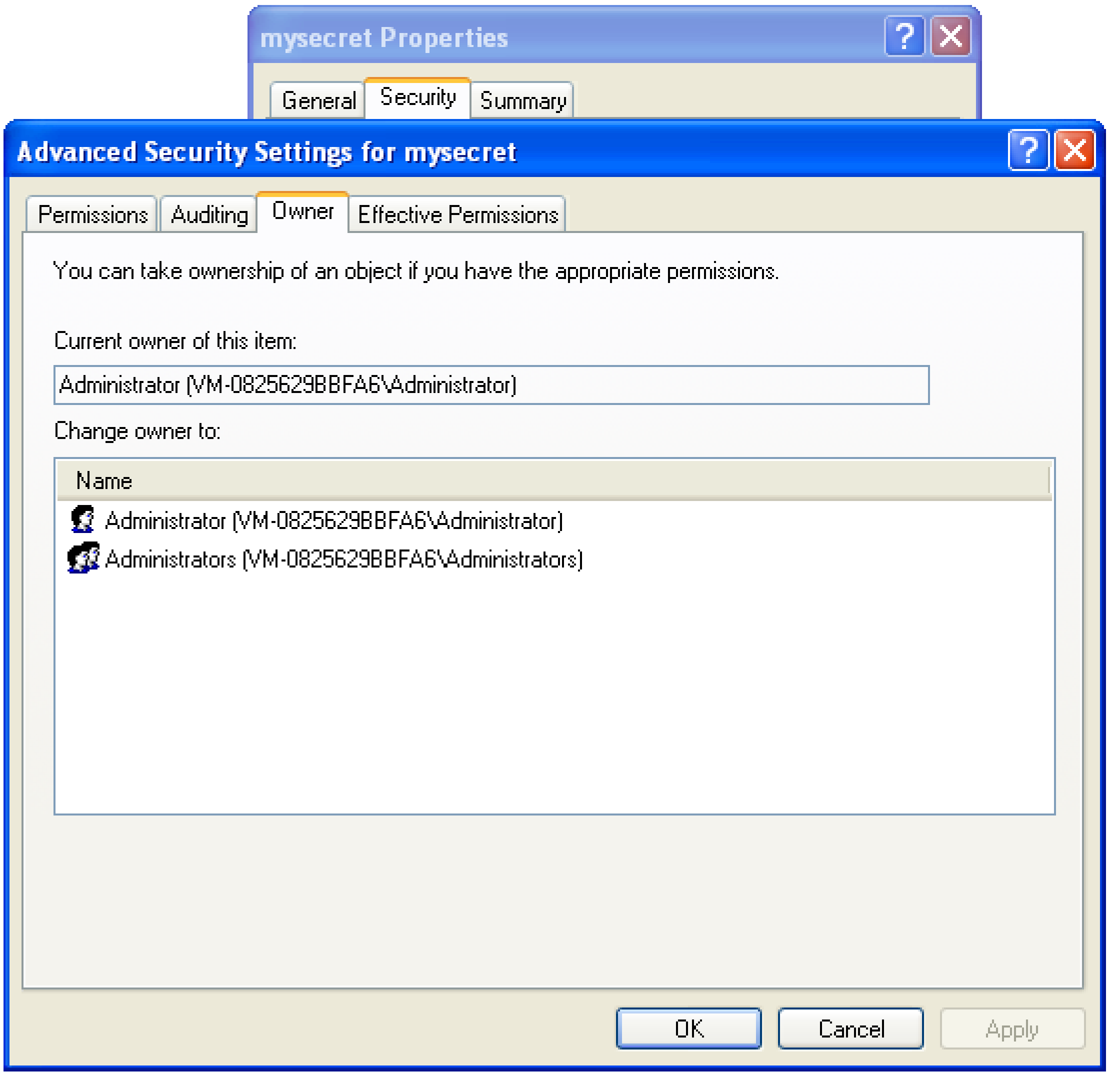
Using Notepad (invoked from the Start menu), create a new text file in the root "C:" directory, called "mysecret.txt", and containing some text. For example, "This is a secret, I will only share with some".

Browse to that directory in Windows Explorer/My Computer.

Go to Tools (menu), Folder options, View. And make sure that "Use simple file sharing" is **disabled**.

Have a look at the "security" settings for the file (right click the file and click properties).

Click advanced and look at who the owner of the file is.



You can take ownership of an object if you have the appropriate permissions.

Current owner of this item:

Administrator (VM-0825629BBFA6\Administrator)

Change owner to:

Administrator (VM-0825629BBFA6\Administrator)

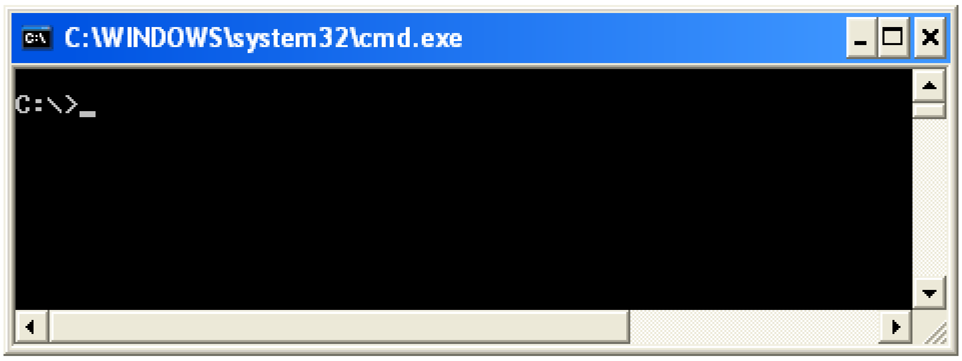
Administrators (VM-0825629BBFA6\Administrators)

File ownership

Now lets experiment with who can open the file…

Open a command prompt, and move to the “C:\” directory:

> Hint: use “c:” to change to C, and “cd ..” to move up directories, ask your tutor if you need help.



```
C:\>_
```

C:\ command prompt

Now run the following commands to see which users will be allowed to access the file:

```
notepad mysecret.txt
```

(This runs as your current user, it should print the contents)

```
runas /user:Alice "notepad c:\mysecret.txt"
```

This runs as Alice, and will probably display the contents and allow edits, since by default often all local users are granted access

Now, back in the ACL settings, add a row that grants Alice read permission, but denies write.

Hint: “Add”, type “Alice”, then check the boxes that apply.

Re-run:

```
runas /user:Alice "notepad c:\mysecret.txt"
```

Now Alice can read but cannot change the contents. If she opens the file in notepad, she can't save any changes.

```
runas /user:Alice "notepad c:\mysecret.txt"
```

Deny Eve permission to read to the file.

Confirm that Eve cannot read the file.

ACLs provide a powerful security tool to specify which users are allowed to access your files. However, note that NTFS ACLs can be quite complicated, since permissions can be inherited from directories.

Question: Would ACLs still be of any use if two people share the same user account login? That is, the two people both use the same computer without having separate accounts.

Earlier versions of Windows that were not based on Windows NT, such as Windows 95, 98, and Me, did not restrict access based on identity. These systems had user profiles that did not provide any security, they were only used to provide users with customised desktop settings. However, Unix has enforced identity and access controls since the 1970s.

Conclusion

You are now familiar with the very important concept of identity and authentication. You have also experimented with offline password cracking, and noted that some one-way hash functions, such as LM, are particularly weak. You have also experimented with access controls in Windows, which is a form of access control lists, to restrict what the separate users on a computer can access.