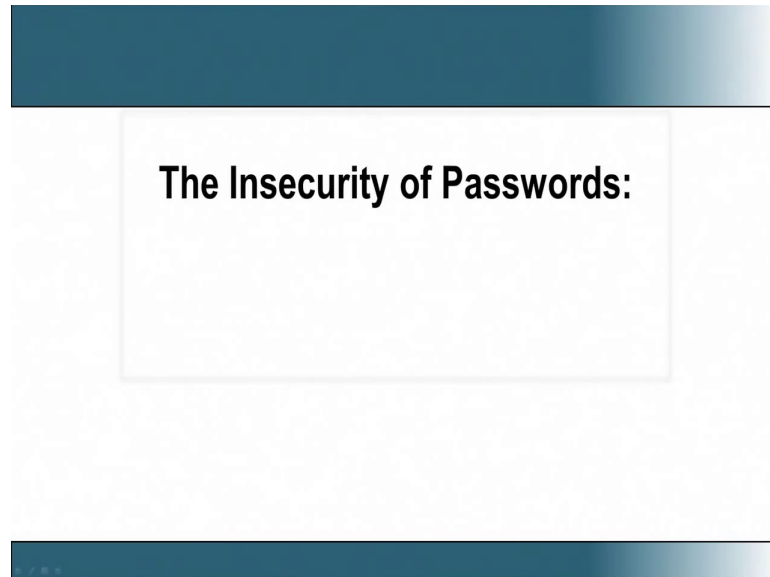


Introduction to Information Security
Prof. V. Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 09

Now, we discuss about the insecurity of passwords.

(Refer Slide Time: 00:10)



One of the simple thing that we notice, people always pick very weak passwords. What we mean by weak, a several things has listed in the slides.

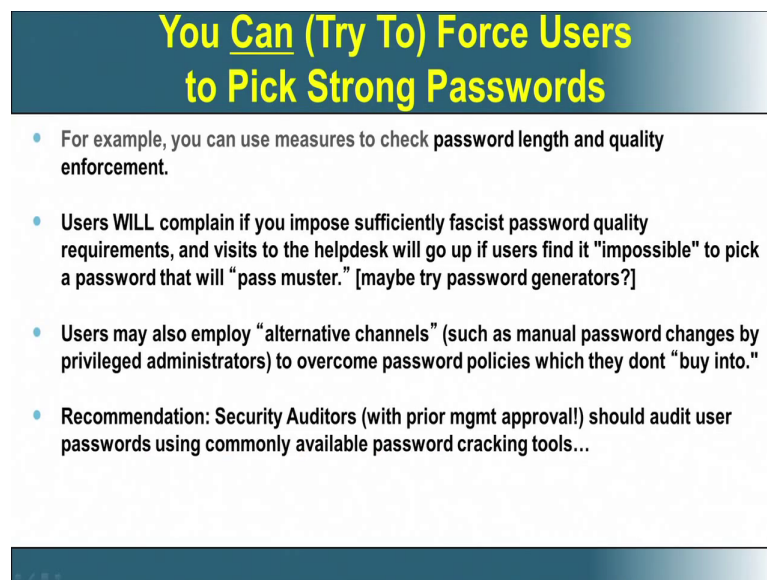
(Refer Slide Time: 00:32)

A presentation slide with a dark blue header and footer. The main content area is white. The header contains the title "1. People Will Pick Weak Passwords" in a bold, yellow, sans-serif font. Below the title, there are two bullet points in a blue font. The first bullet point lists several types of weak passwords: "password == username", "trivially short (6 character or less) passwords", "passwords that use only lower case letters", "passwords that are words in the dictionary", and "passwords that are a pattern on the keyboard (12345678, qwerty, etc.)". The second bullet point asks "Why are weak passwords a problem?" and explains that they can be successfully attacked with brute force attacks (password = a, b, c, d, ... z, aa, ab, ac, ad, ... az, ba, ... zzzzz), or via dictionary attacks (try all words found in the dictionary as potential passwords).

The password will be the user name itself, it will be trivially shortlike 1234 or it can be full of low case letters or it uses words from the dictionary or patterns on some keyboard, these are all weak passwords in the sense that one can easily hack by what we call as brute force attacks. Brute force is we just keep trying different combinations and we try these combinations from different sources like we take dictionary and then take all the combinations and then we go and find out whether the user has used one of these combinations.

So, by just doing such type of brute force attack, a hacker can easily find out what the password is. The thing is that one can try to force user to pick strong passwords. For example, you can go and tell, it should have a special character, it should have capital, it should be not less than some length like 8, 8 characters. The moment you put this, then some people cannot even create passwords. So, they will be struggling to create a password and that means that your admin will have a problem. So, they will start calling help by saying none of the password works. So, if you start making more and more rules on how to create a password for a man, for a normal user, average user it becomes extremely difficult.

(Refer Slide Time: 02:20)



You Can (Try To) Force Users to Pick Strong Passwords

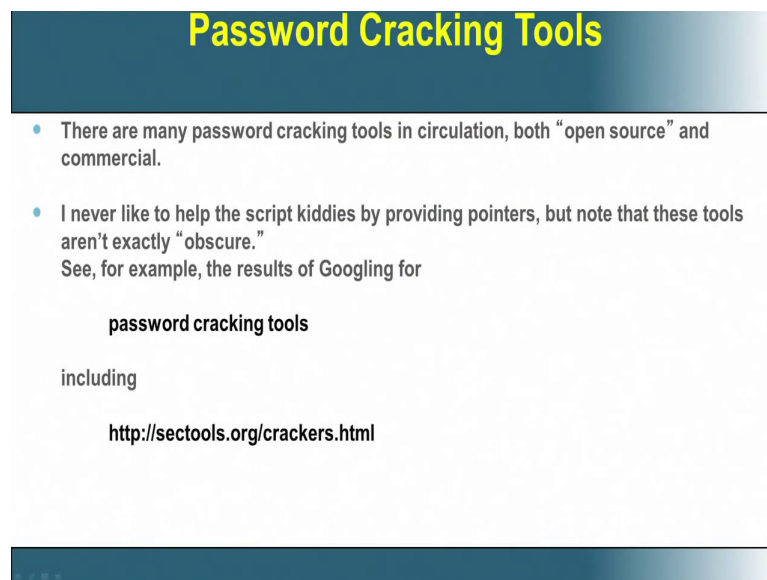
- For example, you can use measures to check password length and quality enforcement.
- Users WILL complain if you impose sufficiently fascist password quality requirements, and visits to the helpdesk will go up if users find it "impossible" to pick a password that will "pass muster." [maybe try password generators?]
- Users may also employ "alternative channels" (such as manual password changes by privileged administrators) to overcome password policies which they dont "buy into."
- Recommendation: Security Auditors (with prior mgmt approval!) should audit user passwords using commonly available password cracking tools...

The user can also employ some alternate channels which will create password given a set of rules, but then the same channel can be used by the hacker to hack those passwords. So, there is again a trade-off between a complicated password and the acceptability of that complicated rules from the users point of view. So, on the other front if you want to go and ensure that the users do have a strong password, the auditors who do an

information security audit with the permission of the management can go and with the prior approval, he can go and test some of these passwords and see if these passwords adhere to some strength.

They can use actually password cracking tools to find out, if some passwords are weak and then you can inform the users of the system saying these are the passwords that are weak and go and change them. So, these one way by which we can force users to pick strong passwords. But, still one has to remember there is a balance that we need to strike here between, how complex the passwords could be and how acceptable that complexity is for the user.

(Refer Slide Time: 03:49)



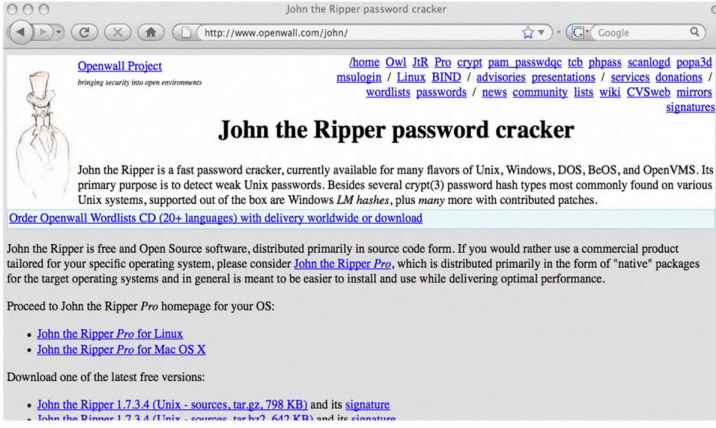
Password Cracking Tools

- There are many password cracking tools in circulation, both "open source" and commercial.
- I never like to help the script kiddies by providing pointers, but note that these tools aren't exactly "obscure."
See, for example, the results of Googling for
password cracking tools
including
<http://sectools.org/crackers.html>

There are lot of password cracking tools, though this is not the intention for us to teach how to crack a password here. But, nevertheless it is very important for someone in this area to understand that there are open source password cracking tools. So, we give some examples you can go to the website that this shown in the slide, it is sectools dot org.

(Refer Slide Time: 04:10)

A Sample Tool From the Sectools.org Listing



John the Ripper password cracker

John the Ripper is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, BeOS, and OpenVMS. Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix systems, supported out of the box are Windows *LM hashes*, plus *many* more with contributed patches.

John the Ripper is free and Open Source software, distributed primarily in source code form. If you would rather use a commercial product tailored for your specific operating system, please consider [John the Ripper Pro](#), which is distributed primarily in the form of "native" packages for the target operating systems and in general is meant to be easier to install and use while delivering optimal performance.

Proceed to John the Ripper *Pro* homepage for your OS:

- [John the Ripper Pro for Linux](#)
- [John the Ripper Pro for Mac OS X](#)

Download one of the latest free versions:

- [John the Ripper 1.7.3.4 \(Unix - sources, tar.gz, 798 KB\)](#) and its [signature](#)
- [John the Ripper 1.7.4.4 \(Linux - sources, tar.gz, 747 KB\)](#) and its [signature](#)

Now, we have a very famous password cracking tool, the ripper password cracker and if you carefully looking to the first few lines, it is work on different operating systems like flavors of Unix, Windows, dos, BeOS, open VMS, etcetera. So, it is sort of a very versatile tool, the John the Ripper password cracker.

(Refer Slide Time: 04:37)

One Password Cracking Approach Particularly Worth Highlighting: Rainbow Table Methods

- Most computational algorithms, including password cracking algorithms, involve tradeoffs between time and space, e.g., I can use more CPU, or more RAM or disk storage, depending on how (algorithmically) I decide to attack a problem. Rainbow tables use pre-computed stored hash chains (e.g., more storage) to dramatically reduce the CPU/clock time required to crack password hashes.
- For more information on Rainbow Tables, see:
<http://project-rainbowcrack.com/> and
<http://project-rainbowcrack.com/tutorial.htm> and
<http://www.ethicalhacker.net/content/view/94/24/>

77

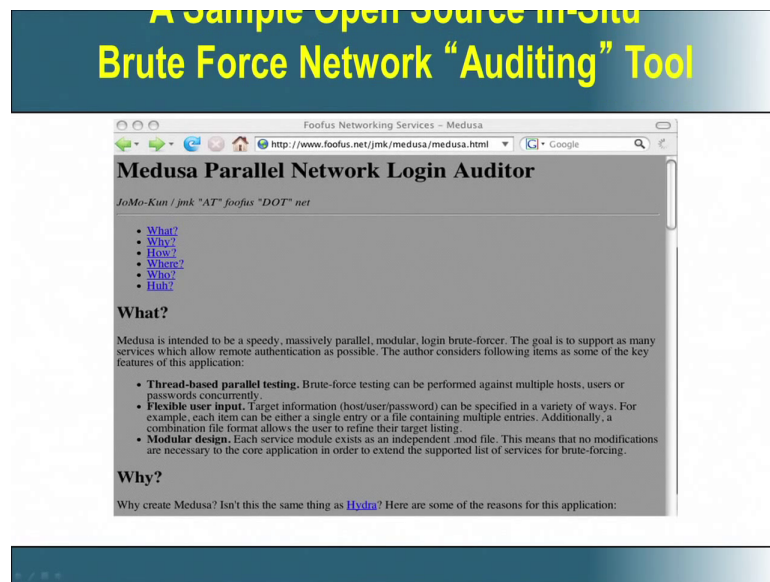
Another one interesting approach for password cracking is called the rainbow table method. We will talk about rainbow table method in the level two course, but one of the things that we need to understand here and we also encourage the users to go to the different websites that we have listed here and understand more about rainbow tables. Why it is called a rainbow, because it does variety of cracks and variety of operating

systems like Linux, Windows and also a variety of protocols like ssh..

The interesting thing about rainbow is that it is when we do a normal brute force approach, it takes a lot of time and space, space in sense of memory. Now, this rainbow table method actually uses some precomputed stored hash strings. In a hash, a password is not stored just as a password, but it is actually converted into or obfuscated into a very complex string using a hash function. So, this actually uses some precomputed, stored hash chains to quickly go and crack a password.

So, this is the very interesting technique you will discuss about rainbow table methods in the level 2 course. But, I still suggest that you can go to the websites and understand, what is rainbow table method for password cracking, it is a very interesting one.

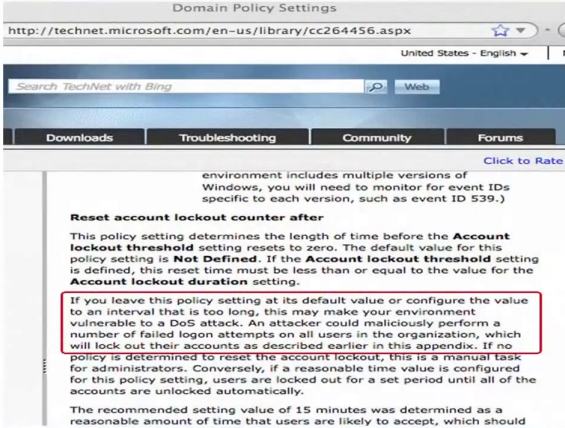
(Refer Slide Time: 06:21)



So, there are many other open source tool which can do a brute force cracking of the password, this is one brute force network auditing tool which can do this.

(Refer Slide Time: 06:43)

Can "Brute Forcers" Accidentally (or Intentionally) DDoS Your Users?



The screenshot shows a web browser window with the URL <http://technet.microsoft.com/en-us/library/cc264456.aspx>. The page title is "Domain Policy Settings". Below the navigation bar, there is a search box and a "Web" button. The main content area is titled "Reset account lockout counter after". The text explains that this policy setting determines the length of time before the Account lockout threshold setting resets to zero. The default value for this policy setting is **Not Defined**. If the Account lockout threshold setting is defined, this reset time must be less than or equal to the value for the Account lockout duration setting. A red box highlights a warning: "If you leave this policy setting at its default value or configure the value to an interval that is too long, this may make your environment vulnerable to a DoS attack. An attacker could maliciously perform a number of failed logon attempts on all users in the organization, which will lock out their accounts as described earlier in this appendix. If no policy is determined to reset the account lockout, this is a manual task for administrators. Conversely, if a reasonable time value is configured for this policy setting, users are locked out for a set period until all of the accounts are unlocked automatically." Below this, it states: "The recommended setting value of 15 minutes was determined as a reasonable amount of time that users are likely to accept, which should".

Now, another consequence of hacking the password as such password hacking is actually a disclosure of the private information. So, it is against the confidentiality property and that is why we want to stop hacking password. But, the hacking of the password, the policies that are used to prevent hacking of a password can affect the availability of a system.

So, for example, we could have a policy saying that for every three attempts of login, if you use a wrong password for three wrong logins, immediately your system will be disabled for say 30 minutes for you to access again. So, if someone actually wants to stop you from using the system, if the hacker wants to stop you from using the system, he just has to use your login id and login 3 times with the wrong password, so the next 30 minutes you cannot use the system. And if you does every 25th minute again and again or every 30 minutes again and again, you can permanently you will not be in a position to use the system.

There are more stringent policies, especially in internet banking where if there are 3 to 4 logins with wrong passwords or wrong id, then the entire internet banking itself is disabled and you need to go sometimes in person to enable it back. Though it is a very important security measure to safe that your account from the bank perspective, but then it can be used by a hacker to deny you the service.

So, policies that are put to ensure confidentiality, in this case password can turn out to be detrimental, for some other property in this case we are seeing the availability being the

violated. So, this is the coupling between confidentiality, integrity and availability. If I go and increase the confidentiality, increase assurance for confidentiality I may go and decrease the assurance for availability and this is a very interesting example in the direction.

(Refer Slide Time: 09:31)

More In-Situ Brute Forcing Points To Consider

- Are you limiting the number of password attempts you allow? For example, after three failed logins do you stop listening to additional login attempts from a source IP, but **only for a period of time?** [see preceding slide!]
- Do you log (and analyze!) all authentication failures?
- If you use the same authentication service for multiple network applications, do you track/limit/log login failures from ALL of them? Or are you only protecting one service (such as sshd) while allowing someone to flood "more obscure" services (such as POP3 or IMAP) with query traffic w/o those attacks being detected and handled?
- Moving services off their default ports and/or using port knocking can

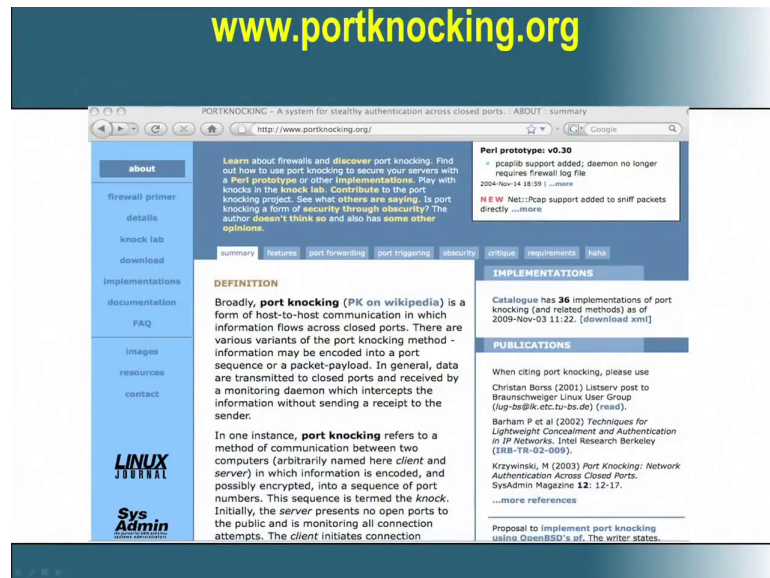
So, another interesting thing is the more on the, what we call as port knocking. The port knocking is something like there are different ports. For example, if you access a web server say HTTP, it uses port 80, FTP uses port 21, the SSH uses port 22. So, port knocking is to keep, there are some ports which will not be exposed, but then one way of a denial of service is to do this port knocking, where you keep repeatedly requesting for a port and hence making it unavailable or very scarcely available for a genuine user.

So, another thing is that suppose let us now again come back to confidentiality. If I am using intuitive brute forcing of saying that if you have a wrong password, if you enter wrong password for some time, you go and cancel the login or say for example, three file logins, use stop listening to additional login attempts only for a period of time. So, do you log and analyze, all these authentication failures is very important. Because, if it coming from a genuine user who has forgotten his password or is it is coming from an attacker, who wants to denial the service to the user. So, every 30 minutes if you find the user for a consistent period of time giving wrong passwords, then there should be something else that needs to be done to safe guard the interest of the genuine user.

And other important thing is that if the same password can be used for multiple network

applications, it is not just I am trying to login for one application. But, I may be doing with the same password I can do multiple applications. Are we going to stop all the applications or are we going to prevent for a single application, so these are something needs to be addressed, when we look at passwords.

(Refer Slide Time: 11:59)



So, this particular slide talks more about port knocking and how port knocking can be used for a denial of service type of attack.

(Refer Slide Time: 12:11)

Shadow Password Files

- Of course, offline password cracking tools require password hashes as input, and these days most operating systems store password hashes as shadow files, accessible only by privileged users.
- Shadow files are still of great interest to crackers, being among the top 10 targets seen by the Honeynet Project (see www.honeynet.org/book/export/html/14)
- Hmm... what might a cracker do? Well, lets assume he/she manages to obtain access to your backup server, or to your unencrypted backup media, including a copy of /etc/shadow ... Suddenly, (s)hes got plenty of fodder to feed to his/her favorite password cracking tool(s).
- Now you know the reason why Security Gurus harp on the importance of encrypting and/or controlling access to your backups!

So, let us come and look at the shadow password files, now very interestingly some of the password cracking tools actually required password hashes as input, they do not need

actually password as input. So, we can look at the honey net project site which we have stated there, you will realize that the input to the password cracking tool is not the password, but it is going to be the hash of that password which is actually stored in a shadow password file.

So, that is why many of the people involved in security, says that you should not have a file system that is unencrypted. The shadow password file is actually stored in a directory in the file system and if you are not encrypting the file system, then the shadow password file will be available for some one to steal and that will become a very, very interesting input or very important input for lot of password cracking tools.

So, one of the recommended procedures for implementing a secure system is to also go and encrypt the file system. What you mean by encrypting the file system? All the files that are stored in the disk are in some encrypted form and it gets decrypted when you load the file. While if I am going to encrypt all the files, then the shadow password file is also encrypted and so a hacker, who steal these file will not be in a position to find out what are all the hashes.

So, this is the double level protection that is needed, so that we have more security in our systems. The password encryption algorithm, so you enter a password that password is encrypted and stored in the shadow password file and the password is signed, it is converted to some other form in a very obscure form and stored in the shadow of a password file.

(Refer Slide Time: 14:37)

Passwords Encryption Algorithms

- You should be sure that you're using a strong password encryption **algorithm** (assuming your operating system gives you that choice).
- Some systems may still be storing passwords encrypted with the traditional "crypt" based on 56-bit DES encryption. That's pretty bad. You shouldnt use it.
- There are alternative password encryption algorithms available on at least some systems, such as Redhat Linux, including MD5 (identified by the leading \$1\$ in the printable form of the password file), and Blowfish (tagged with \$2\$ or \$2a\$), however I would recommend SHA256 (\$5\$) or SHA512 (\$6\$), which may be the default on (some) distros. See people.redhat.com/drepper/SHA-crypt.txt and kbase.redhat.com/faq/docs/DOC-15806 and httpd.apache.org/docs/2.2/misc/password_encryptions.html

So, people use different encryption algorithms and if an encryption algorithm is not strong, then one can go and find the password by doing some reverse Engineering. So, one of the recommended encryption algorithm as of today is the SHA 256 or SHA 512 and you can see the website that is listed there to understand more about these encryption algorithms and when you have an installation, you can actually go and check the password file to find out which encryption algorithm is used specifically in the Redhat Linux distribution.

So, using of a strong password encryption algorithm is also very, very important for ensuring protection of password. Now, comes another interesting thing from the windows prospective.

(Refer Slide Time: 15:52)

LAN Manager Password Hashes on Windows

- On many versions of Windows, password hashes are stored in multiple formats: LM hash (for "compatibility"), plus other formats. LM hash format hashes are vulnerable to fast brute force attacks. For example, LMCrack says, "The design goal of LMCrack was to walk a large key space based on a dictionary style attack rather than on a comprehensive brute force attack and to complete the task in under 5 minutes. The result is a program that utilizes a database of pre-computed hashes, which can search an effective key space of 3 trillion passwords in less than 60 seconds with an average success rate of 50+%."
- See "How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases," support.microsoft.com/default.aspx?scid=KB;EN-US;q299656& (or tinyurl.com/no-lan-man-hashes)

The windows add something call a LAN manager, which was very close to ldap (15:55) where you store the password. The password hashes were actually stored on active directories in early times and then there was a project call LM crack which was specifically target that to retrieve the passwords from the LAN manager. It is very interesting to read that the objective or the goal of LM crack, the design goal of LM crack was to walk large key space based on a dictionary style attack rather than on a comprehensive brute force attack and to complete the task in under 5 minutes.

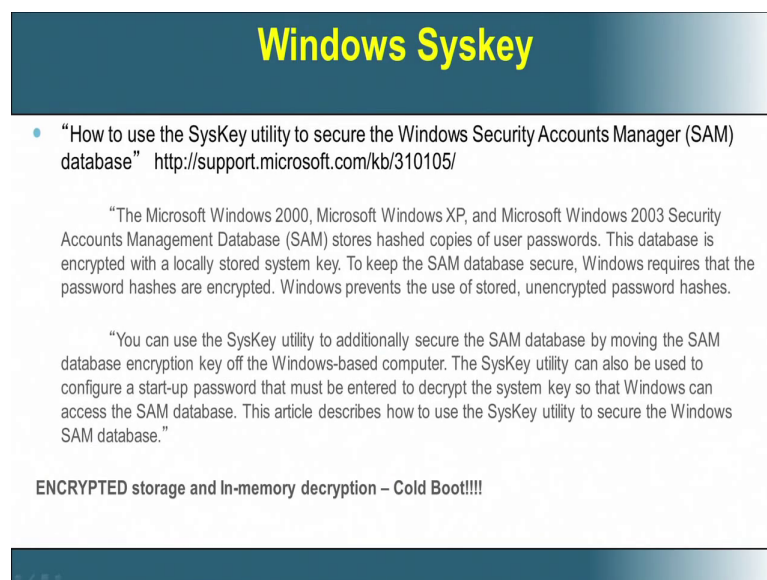
The result was a program that utilizes a data base of precomputed hashes which can search an effective key space of 3 trillion passwords in less than 60 seconds with an average success rate of 50 percent or more. So, how to prevent windows from storing a

LAN manager hash of a password in active directory and local security access module databases., You can go and look at the support page even by Microsoft.

So, this was one very interesting reported attack on password which was on the LAN manager hashes that were stored by windows. Windows also have a SysKey for encrypting the SAMdatabase and decrypt in the memory on a need baisi. The SysKey is actually a system generated key and the SysKey need not be actually part of your file system, it can be stored at some other place where integrity and confidentiality can be more ensured.

Now, what SysKey does? The SysKey is basically used to encrypt your file system. So, when somebody accesses your disk, they can't understand any information, because everything is encrypted.

(Refer Slide Time: 18:10)



Windows Syskey

- “How to use the SysKey utility to secure the Windows Security Accounts Manager (SAM) database” <http://support.microsoft.com/kb/310105/>

“The Microsoft Windows 2000, Microsoft Windows XP, and Microsoft Windows 2003 Security Accounts Management Database (SAM) stores hashed copies of user passwords. This database is encrypted with a locally stored system key. To keep the SAM database secure, Windows requires that the password hashes are encrypted. Windows prevents the use of stored, unencrypted password hashes.

“You can use the SysKey utility to additionally secure the SAM database by moving the SAM database encryption key off the Windows-based computer. The SysKey utility can also be used to configure a start-up password that must be entered to decrypt the system key so that Windows can access the SAM database. This article describes how to use the SysKey utility to secure the Windows SAM database.”

ENCRYPTED storage and In-memory decryption – Cold Boot!!!!

But, when the files are moved to the memory, the same SysKey can be used to decrypt it and so the decryption happens in the memory. Now, this can be certainly a way by which you can get more security for the passwords, because the passwords are encrypted and they are only decrypted when it comes into the memory. The hashes are also decrypted only when it comes into the memory. But, then recent technologies like what we call as a cold boot, where in we can take the processor out and study or the memory out and we can actually get glimpses of the memory, glimpses of what is stored inside the memory.

This type of cold boot can basically go and read the decrypted data that is already inside the memory. So, that is why many of the secure systems today need what we call as

tamper detect which will prevent an attacker to physically open the enclosure and go and study the contents of the memory using such techniques like the cold boot.