Password Security and Exploits

- The most common method of implementing system and network security is password protection.

- Unfortunately password protection is quite often the weakest link in network and system defenses.

Types of Password Attacks

- Weak passwords are the most common exploits. If attackers can guess a user's password, they can gain access to a machine or network and have full access to any resources that user has access to.

- This can be extremely dangerous if the user has special access such as domain administrator or root privileges.

- The most common method of obtaining a password is by obtaining an encrypted password file and cracking it.

- Based on the system that the password file was acquired from, an attacker will first determine the encryption that was used.

- Then by using one of the methods listed below, an attacker can take a plain text password, encrypt it, and see if there is a match:

  - Dictionary attacks
  - Brute force attacks
  - Hybrid attacks

Dictionary Attack

- This technique exploits the fact that most users use common dictionary words (very convenient) as passwords

- A dictionary attack takes a file that contains most of the words that would be contained in a dictionary and uses those words to guess a users password.

- On most systems, a dictionary attack can be completed in a short period of time compared to trying every possible letter combination.

- A dictionary attack can be customized for different companies and users. For example, an attacker can add company jargon in the dictionary or use words from known user interests and hobbies.

- There are a large number of precompiled dictionaries available on the Internet, including foreign language dictionaries and dictionaries for certain types of companies.

- By carefully studying and understanding an environment, the chances of successfully cracking a password increase.

- It is very important to urge users not to use passwords that can be easily derived from their surroundings and interests.

Brute Force Attacks

- There is no such thing as an unbreakable password, the strength of the encryption scheme notwithstanding.

- All passwords are breakable; it is just a matter of how long it takes to break or crack it.

- Using a powerful enough computer, an attacker can try every possible combination of letters, numbers, and special characters, until eventually a password is cracked. Hence the term brute force attack.

- With a brute force attack, you start with the letter a and try aa, ab, ac, and so on; then you try aaa, aab, aac, and so on, until all possible combinations have been attempted.

- If an attacker knows that the minimum length for a password is six characters, the brute force attacks can start with aaaaaa and go from there.

- This technique is essentially a trade-off between the speed of the CPU and the time it takes to crack a password.

- As memory becomes cheaper and processors become faster, tasks that used to take a long time to can be accomplished in a very short period of time.

- Another serious threat to password security and encryption are **distributed attacks**. An attacker can make use of the large number of networked machines to achieve performances close to super computers to crack encryption techniques.

- Some very good examples of this can be found at http://www.distributed.net/.

Hybrid Attacks

- Dictionary attacks find only dictionary words but are quick, and brute force attacks find any password but take a long time.
- As administrators crack down on weak passwords and require users to have letters and numbers, most users simply add a couple of digits to the end of a password-for example, my password goes from milliways to milliways42.
- An attack that takes dictionary words but concatenates a couple of letters or numbers to the end-the **hybrid attack**.
- The hybrid attack takes your dictionary word and adds a couple of characters to the end.
- The following table shows the relationship between the different types of attacks.

	Dictionary attack	**Brute Force attack**	**Hybrid attack**
Speed of the attack	Fast	Slow	Medium
Amount of passwords cracked	Finds only dictionary words.	Finds every password.	Finds only passwords that have a dictionary word as the base.

Other Types of Password Attacks

- Attackers will usually take the path of least resistance, to acquire the information that they are after.
- Even though passwords can be made very secure, attackers can still compromise user passwords using methods such as:

  - Social engineering
  - Shoulder surfing
  - Dumpster diving

Microsoft Windows Passwords

- The password hashes for each account are stored in the security database in Windows. This is also referred to as the SAM or security account manager.
- The file is located in **\WINDOWS\system32\config\SAM**, and is usually world readable, however it is not accessible when the system is running because it is locked by the system kernel.
- During the OS installation, a copy of the password database is copied into the **WINDOWS\repair**. This copy is not very useful because no other accounts have been created yet so it only contains the default accounts.
- However, the Administrator is a default account. This is another reason to ensure that the administrator account has a strong password. If the administrator updates the repair disk, this information is also updated.
- When a user types a new plaintext password, Windows runs it through two hash algorithms, one for the regular Windows hash and one for the LANMAN hash.
- To calculate the regular hash, Microsoft converts the password to Unicode and then runs it through a MD4 hash algorithm to obtain a 16-byte value.
- To calculate the LAN Manager hash, Windows pads the password with O's until it has a length of 14 characters.
- It is then converted to uppercase and split into two 7character pieces. An 8-byte odd parity DES (data encryption standard) key is calculated from each half, and then the DES keys are encrypted and combined to get a 16-byte, one-way hash value.
- The goal with encryption is to make the time needed to perform a brute force attack on a password so long that it is impractical for someone to attempt to crack it.
- Encryption can also make the time it takes to perform a brute force attack so long that the value of the information expires before the attack is complete.
- Windows does not store user passwords as hashes, but as "check sum" of the passwords.
- The structure of the complex SAM-file uses the so called V-block, which has a size of 32 bytes and it includes hashes of the password for the local entering: NT Hash of 16-byte length, and hash used during the authentication of access to the common resources of other computers LanMan Hash, or simply LM Hash, of the same 16-byte length.

- The following is a summary of the algorithms are used in the formation of these hashes:

- **NT Hash formation**:

  - User password is to a Unicode-line.
  - Hash is generated based on this line using the MD4 algorithm.
  - The derived hash is encoded by the DES algorithm; RID (i.e. user identifier) is used as a key.
  - This step was necessary for achieving different hashes for users who have similar passwords.
  - Recall that all users have different RIDs (RID of the Administrator's built in account is 500, RID of the Guest's built in account is 501, all other users get RIDs equal 1000, 1001,1002, etc.).

- **LM Hash formation**:

  - User password is shifted to uppercase and padded by null bytes up to 14-byte length.
  - The field is divided into two halves of 7 bytes each, and each of them is encoded separately using DES; the output is 8-byte hash and total 16-byte hash.
  - The LM Hash is the encoded the same way as in the NT Hash formation algorithm.

- To improve the security of the password storing in Windows NT Service Pack 3 (and in all further NT systems up to Windows 2003), hashes are now encoded using an additional algorithm using the **Syskey** utility.

- The method Microsoft chose to implement passwords on Windows enables a perpetrator to crack passwords at a faster rate than on other systems, for example, Unix. This is due to two main flaws in the implementation.

- The first design flaw is in Microsoft's LAN Manager hashing scheme. Because NT/2000/XP/2003 is designed to be backwards compatible with earlier versions of Windows, it uses the LAN Manager hashing scheme, which breaks a password down into two 7-character words and does not have case sensitivity.

- This significantly weakens the strength of a password. LAN Manager was the predecessor to NT and Windows and was one of the first network operating systems.

- LAN Manager came out in the late 80's when machines were a lot slower and technology was just starting to be adapted. Therefore, for speed reasons, it

was decided to break the passwords up into two pieces because it was easier to process.

- Also in the 80's, 7-character passwords seemed highly secure and took a very long time to crack. Unfortunately this technology is still in use today when machines are far more powerful.

- Now with LAN Manager passwords, instead of trying to crack a password that is 12 characters long, a hacker would just have to crack one 7-character password and one 5-character password, which is much easier than cracking one 12-character password.

- The reason for this is because the longer a password is, the more possible combinations of characters a brute force attack has to try, which increases the time needed to crack a password. In any case, the longest password a hacker will ever have to crack in Windows is 7 characters long.

- Another problem with reducing the number of characters in a password is that most people use numbers or special characters at the end of a password, which means it is very likely that one of the two 7-character passwords contains only letters.

- A password containing only letters is much easier to crack than passwords with numbers and special symbols. For example, cracking the password marvin@8 would be fairly difficult and would take a long time to brute force because it has alpha, number, and special characters.

- With the LAN Manager hash, a hacker would have to crack marvin, which is only alpha characters, so it is fairly easy to do, and then she would have to crack @8, which contains a number and special character.

- However, @8 would be very simple to crack based on the length. Thus, breaking up a password into two pieces makes it considerably easier to crack.

- A brute force attack takes considerably less time to crack two pieces compared to the time it takes to crack one piece.

- This is true because the two pieces can be cracked in parallel, so instead of trying every possible combination of 14 characters to crack the password, the hacker would only need to try every possible combination of 7 characters.

Salts

- Now let us look at the second reason why Windows passwords can be cracked in a shorter period of time.

- To make passwords harder to guess, they are often randomized. This way two users who have the same password have different hashes.

- When a password is encrypted, the system uses something called a **salt**, which is meant to make passwords a little harder to guess by randomizing the password.

- A salt is a random string that is combined with a password before it is encrypted. The second design flaw in NT is that it does not use a salt.

- Normally, when the user enters a new password, the system computes the hash and stores it. The problem with this is that if two people have the same password, the hash is the same.

- The way the system uses a salt is that for each user it calculates a random number-the salt. When the user enters a new password, the system first combines the password with the salt and then computes the hash.

- The system not only stores the hash, but also the salt with the user ID. Now, when a user authenticates to the system and he types his password, the system looks up the salt and combines it with the password, calculates the hash, and determines whether there is a match.

- This way, if two people have the same password, they will have different salts, and their passwords will be stored differently. This makes it a lot harder to brute force a password.

- Without a salt, an attacker can compute the hash of each word once and scan the entire list of user's passwords to see if there is a match. Because ten users with the same password will have the same hash, their passwords can be cracked with one attempt.

- With a salt, you have to compute the hash of each word for each user using their unique salt. Now, instead of computing the hash once and scanning the list, all the work has to be repeated for each user. In this way, using a salt makes it increasingly difficult, from a time perspective, to crack a series of passwords.

- For example, the following shows two users' passwords that are the same in a system where salts are not used:

  ford:.D532YrNl2G8c
  arthur: .D532YrNl2G8c

- As can be seen, because a salt was not used to randomize the password, the two encrypted passwords are exactly the same.

- A password cracker would only have to compute the password once and he would be able to crack both accounts at the same time.

- The following shows two users' passwords that are the same in a system where salts are used:

  ford:.D532YrNl2G8c
  arthur:WD.ADWz99Cjjc

- Although the passwords are the same, because the salts are different, the resulting encrypted passwords are different.

- A password cracker would have to compute the hash twice, once for each password and using a different salt each time.

- As pointed out earlier, this increases the time, especially if there are a lot of accounts on the system.
- Microsoft does not use a salt, so if two users have the same password, they are encrypted the same way.

- From a security perspective, the two things that Microsoft does to make cracking passwords even easier are:
  - Utilizing LAN Manager hashes, which break passwords into two 7-digit passwords.
  - Not using salt (or randomness), so two identical passwords are encrypted the same way.

Accessing the SAM-file and Importing the hashes

- As mentioned earlier, there's no way to directly read or edit this file because Windows will lock it once the system is up and running.
- If you try to read this file Windows messages about the file sharing violation, because this file is constantly opened for the system and only Windows may record data in this file.
- There are several methods of acquiring the **SAM** file data. One of them is PWDUMP program method (used in PWDUMP, LCP, LC5 programs, etc.).
- PWDUMP method works as follows: the program connects to the system procedure LSASS and using its rights (and methods) gets hashes from the SAM registry branch, i.e. from the SAM-file.
- A very effective way to recover the SAM file is to boot up an alternate operating system on the computer and access the file from a different partition.
- The easiest way to do this is to use a live CD like Knoppix and perform a boot from the CD-ROM with this disk in the drive.
- Once you are in Knoppix, mount the Windows partition and locate the SAM file in the Windows directory, and either copy it to a USB drive, floppy disk, CD, or upload it to another machine.
- Once you have the SAM file, it is a simple matter to feed into a password vcracker such as LCP, and select Import/Import from SAM file. This will load the hashes and you will be able to execute a variety of attacks on them.

- As mentioned earlier, modern Windows systems use additional hashes encryption using the **Syskey** utility.
- Until recently hashes from the SAM-file copied from these systems couldn't be decoded due to the lack of information on this algorithm.
- Now there are several utilities, including LC5, LCP, and others that can extract hashes from SAM-files encoded by this algorithm.
- However, decoding the hashes also requires the **SYSTEM** file, located in the same folder as the SAM file.
- This file contains some registry keys that are necessary to decode the SYSKEY algorithm hashes.
- This file is also locked by the system only but, it can be copied in the same way that was used to acquire the SAM file.

Windows Password-Cracking Programs

- There are several programs that can be used to crack passwords in a Windows environment:

  LC5 (@stake)
  LCP (www.lcpsoft.com)
  Ophcrack

- **LC5** is a powerful and feature-rich of the programs listed, but it is a very expensive commercial product.

- **LCP** by contrast is free and contains most of the features available in LC5. This application however has not been updated in some time and it will not run on 64-bit OS's.

- **Ophcrack** is a free Windows password cracker based on rainbow tables. It is a very efficient implementation of rainbow tables. It comes with a Graphical User Interface and runs on multiple platforms.

Ophcrack

- The Ophcrack Windows password cracker is currently the best and fastest, free Windows password recovery tool available.

- It cracks Windows LM hashes using rainbow tables. The program includes the ability to import the hashes from a variety of formats, including dumping directly from the SAM files of Windows.

- There is also a LiveCD version which automates the retrieval, decryption, and cracking of passwords from a Windows system.

- Rainbow tables for LM hashes of alphanumeric passwords are provided for free by the developers. These tables can crack 99.9% of alphanumeric passwords of up to 14 characters in usually a few seconds, and at most a few minutes.

- Larger rainbow tables (for LM hashes of passwords with all printable characters, including symbols and space) are available for purchase from Objectif Securité.

- **Features:**
  - » Runs on Windows, Linux/Unix, Mac OS X, ...
  - » Cracks LM and NTLM hashes.
  - » Free tables available for Windows XP and Vista.
  - » Brute-force module for simple passwords.
  - » Audit mode and CSV export.
  - » Real-time graphs to analyze the passwords.
  - » LiveCD available to simplify the cracking.
  - » Loads hashes from encrypted SAM recovered from a Windows partition, Vista included.
  - » Free and open source software (GPL).

The LCP Utility

- LCP is a free Windows password cracking utility that is very similar in functionality to LC5.

- The following is a summary of features provided with LCP:

- **Importing User Account information:**
  - import from local computer
  - import from remote computer
  - import from SAM file
  - import from .LC file
  - import from .LCS file
  - import from PwDump file
  - import from Sniff file

- **Password recovery mechanisms:**
  - dictionary attack
  - hybrid of dictionary and brute force attacks
  - brute force attack
  - Brute force session distribution:
  - sessions distribution
  - sessions combining

- **Hash computing:**
  - LM and NT hash computing by password
  - LM and NT response computing by password and server challenge.

- **Other General features:**

- LCP provides a separate tool for extracting Security Identifiers and user account information.

- A security identifier (SID) is used by the operating system to uniquely identify a well-known user account or group account, irrespective of the localized language used by the system:

  - SID & User names extracting tool (SID) for Windows NT/2000/XP/2003.
  - SID for a given account name
  - Extracting an account name for single SID or account names for an SID range

Protecting Against Password Crackers

- There is no practical way to completely prevent password cracking. As long as you have a network, and you are connected to the Internet, attackers or insiders will be able to find some way to extract or capture password hashes and crack them.

- The following are some ways to increase the security of Windows passwords:

  - Disable LAN Manager authentication
  - Enforce the use of strong passwords
  - Have a strong password policy
  - Implement SYSKEY security enhancement
  - Use one-time passwords
  - Use Biometric authentication
  - Audit access to key files

UNIX Password Crackers

- UNIX passwords do not have the same vulnerabilities that Windows passwords have, but UNIX has its own set of issues to deal with.

- The following are some tools that are used to crack UNIX passwords:

o Crack
o John the Ripper
o XIT
o Slurpie (for distributed environments)

- Password information is contained a file **/etc/passwd**, which stores all of the user IDs and encrypted passwords in the same file.

- This file was a text file and contained the user ID, encrypted password, home directory, and default shell. The following are some sample entries from a passwd file:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
.
.
.
aman:$1$VDFVP$AbAyFFY13e63NmIyls49U1:500:500::/home/aman:/bin/bash
foo:$1$E30q1c$9mxf1jr2/MybYvMx/i.:501:501::/home/foo:/bin/bash
bar:$1$BuDrVRwl$Ww1P4A4PELmtGaDV7.:502:502::/home/bar:/bin/bash
```

- The general format for the passwd file is as follows:

Username:encrypted passwd:UID:GID:full-name:home directory:shell

- The file has world readability, so any user with an account on the system has read access to the file, which enables the attacker to copy it and crack it.
- To create a solution to the readability problem, UNIX, on its latest versions, splits the passwd file into two files.
- The passwd file still exists, however it contains everything except the encrypted passwords.
- A second file, shadow, was created, which contains the encrypted password. The shadow file is only accessible to root.
- In the above password file, users **root**, **bin** and **daemon** are using shadow passwords.

- The following is a sample shadow file:

```
root:$1$rpÜpat$YsvXMYSIeA3dzYqjNTNh0:11747:0:99999:7:::
bin:*:11747:0:99999:7:::
daemon:*:11747:0:99999:7:::
   .
   .
   .
```

- Note that any entry in the shadow file that has an asterisk (*) identifies accounts that should never log in, and the system should deny any attempt of someone trying to log in with this user account.
- By using a shadow file, the effectiveness of password cracking is reduced, due to the attacker's inability to have root access to crack a file.
- The easiest way to tell if a system is using shadow files it to type **more /etc/passwd.** After the user ID, see if there is an x or random characters.
- If there is an x, the system is using shadow files; if it is random text, it is not.

UNIX Password-Cracking Programs

- **Crack**, is probably one of the most familiar password crackers for the UNIX environment.

- Recall the three basic methods of cracking programs: **dictionary**, **brute force**, and **hybrid** attacks.

- **Dictionary attacks** are usually performed first because they are the fastest, but they are not guaranteed to find every password. Also, it is important to remember that a dictionary attack is only as good as the dictionary that is used.

- Therefore, it is important to spend a little time to build a solid dictionary that contains most of the common words that users of a company might use for their password.

- A **brute force** attack takes the longest amount of time to perform, but it is guaranteed to find every single password. It might take 100 years, but it will eventually crack every password.

- Because most passwords are a dictionary word with a couple of numbers or special characters appended to the end, a **hybrid attack** is also used.

- A hybrid attack takes more time than a dictionary attack and less time than a brute force attack, but it usually finds several additional passwords that a dictionary attack could not crack.

Crack

- **Crack** is a password-cracking program that is designed to quickly locate weaknesses in UNIX passwords.

- Crack uses standard guessing techniques to determine the passwords. It checks to see if the passwords are any of the following:

- The same as the user ID
- The word *password*
- A string of numbers
- A string of letters

- Crack works by encrypting a list of likely passwords and seeing if the results match any of the user's encrypted passwords, which must be supplied prior to running the program.

- Crack, written by Alec Muffet, is available at CEPIAS (which use to be Coast) and is run out of Purdue University at http: //www.cerias.purdue.edu/.

- It can be downloaded from:

ftp://ftp.cerias.purdue.edu/pub/tools/unix/pwdutils/crack/

- CEPIAS has an extensive number of other useful tools available to download. Muffet's web site is http://www.crypticide.com/users/alecm/security/c50-

faq.html and contains some information on Crack and other programs he has written.

- Crack uses a dictionary and connotations of words to break passwords that have been encrypted with Crypt (which is the algorithm UNIX uses to encrypt passwords).

- Crack computes the encrypted password of the guess to see if there is a match. Crack's primary function and original purpose is to crack passwords on a UNIX machine.

- Several versions have been ported to other operating systems, but they do not work as well as the original program, which was designed for UNIX.

- There are several other useful tools available with this program. **Reporter** is used to view the results of running Crack, and **shadmrg** is used to combine passwd and shadow files.

Configuring and installing Crack

- The following is a high-level breakdown of the necessary steps that need to be performed to configure Crack:

1 . Download the Crack file.

2. Unzip the file using gzip: gunzip crack5.0.tar.Z

3. Untar the file: tar -xvf crack5.0.tar

4. Read manual.txt.

5. Edit the script file as per the installation notes.

Compile the program: **Crack -makeonly**
Crack -makedict

- After you successfully download Crack, first unzip the file crack5.0.tar.z by typing the following:

  gunzip crack5.0.tar.Z

- This process creates a file called crack5.0.tar. Now you need to untar the file by typing the following:

tar -xvf crack5.0.tar

- After the tar files are extracted, a directory called c5Oa is generated with subdirectories holding configuration files, documents, scripts, source codes, and so on.

- Some of the key files are **Makefile**, **Peporter**, and **Crack**. A manual.txt file is also created that is the same as the readme file.

- The Crack script file must be edited to and reconfigure the values of CRACK-PATH, C5FLAGS, CC, CFLAGS, and LIBS to suit the operating system.

- Finally, issue the commands Crack -makeonly and then Crack -makedict. After these two steps, a binary executable Crack file is created if no errors are generated during the compilation process.

- To check the password file on the local system run

./Crack /etc/passwd

- The Reporter script is used to check the results of the Crack program to see which passwords have been cracked:

  ./Reporter

- This script outputs the results of which passwords were cracked. This can also be piped to a file. Guesses are listed chronologically, as Crack continues to run over the course of days or weeks.

- Crack has several options that can be used. The following are the most common used ones:

debug

  - Allows us to see what the Crack script is doing.

recover

  - Used when restarting an abnormally terminated session.

fgnd

  - Runs the password cracker in the foreground while stdin, stdout, and stderr are sent to the display so that users can better monitor what is occurring.

fmt

  - Allows the user to specify the input file format that should be used.

n

- Allows the user to jump to a specific spot in the rule base and start password cracking from a specific rule number "n."

keep

- Prevents deletion of the temporary file used to store the password cracker's input. This is helpful for determining what the user did or troubleshooting problems.

mail

- Emails a warning message to anyone whose password is cracked.

network

- Runs the password cracker in network mode.

nice

- Runs the password cracker at a reduced priority for other jobs to take priority over the CPU.

“John the Ripper” Password Cracker

- **John the Ripper** is a fast password cracker, currently available for many flavors of Unix (11 are officially supported, not counting different architectures), DOS, Win32, BeOS, and OpenVMS.
- John can crack the following password ciphers: standard and double-length DES-based, BSDI's extended DES-based, FreeBSD's (and not only) MD5-based, and OpenBSD's Blowfish-based. It can also crack AFS passwords and Win NT LanMan hashes given a command line option.
- John the Ripper can be downloaded from: http://www.openwall.com/john/
- Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos AFS and Windows NT/2000/XP/2003 LM hashes, plus several more with contributed patches.
- This utility can also take a collection of wordlists (available on CD) for 20+ human languages, lists of common passwords, and files with the common passwords and unique words for all the languages combined.

Password cracking modes

- **Wordlist Mode***:*

- This is the simplest mode John supports. John checks passwords against a wordlist file and optionally tries permutations of those words.

- **Single Crack Mode:**
  - In this mode, John gets account information on each user and uses pieces of it as passwords to try.
  - For example, suppose the user account "**abdulla**" is owned by **Aman Abdulla**. John would try **Aman**, **Abdulla**, **ABduLlA**, **abDullA**, and other permutations of information associated with **abdulla** to crack the password.

- **Incremental Mode:**
  - Also known as a brute force attack. Given a character set, John will try every combination of those characters up to 8 characters long.

- **External Mode:**
  - John's user can write pseudo-C functions that John uses to generate the words it tries. See the documentation for the details.

- The following is a summary of steps for building and installing John:

  1. **tar zxvf john-1.6.tar.gz**
  2. **cd john-1.6/src**
  3. **make linux-x86-any-elf**
  4. **cd ../run (change to directory containing *john* executable)**

- Note that if the system uses shadow passwords (very likely on any modern UNIX system), you will need access to the ***/etc/shadow*** file on the target system.

- The file ***/etc/shadow*** contains the password hash needed to crack the password and is by default only readable and writable by the root user.

- To merge the **shadow** file into the **passwd** file, John provides a utility called: ***unshadow***.

- To create a password file for John to work on, first merge the passwd and shadow files into the run directory under your John installation as follows:

  unshadow /etc/passwd /etc/shadow > passwd.1

- Now run John on the merged password file. For simple passwords:

nice -n 20 john/run/john -single passwd.1

- And for dictionary attempts:

nice -n 20 john/run/john -w:john/run/wordlist -rules passwd.1

- Quicker dictionary cracks can be performed by disabling rules and only using a wordlist. Do this by removing the ***-rules*** switch:

nice -n 20 john/run/john -w:john/run/wordlist passwd

- The most powerful mode available is incremental (brute force) mode. This will use the default incremental mode parameters, which are defined in **john.ini**'s [Incremental:All] section.

- In the configuration file supplied with John these parameters are to use the full 95 character set, and to try all possible password lengths, from 0 to 8. (By zero password length I mean a hashed empty string, this sometimes happens.)

- This mode will run forever or until it cracks the passwords; read doc/MODES for an explanation.

- The following will invoke John in incremental mode:

  john -i passwd.1

- It is important to note that this mode attempts ALL possible combinations. That means we have a total number of combinations defined by:

- Thus for a 95 character set and an 8 letter password, we will have a total of **6634204312890625** combinations (i.e., **6.6342P (peta) strings to go through**!).

- Don't forget to set **MaxLen** in **john.ini** if your passwords exceed its default of 8 (and strong passwords should).

- To examine the status of John, press ENTER while it is running. It will display the current string pattern it is attempting to crack.

- Cracked passwords will be displayed on STDOUT (if running interactively) as well as stored in a file named **john.pot**.

Protecting Against Password Crackers

- There is no practical way to completely prevent password cracking. As long as you have a network, and you are connected to the Internet, attackers or insiders will be able to find some way to extract or capture password hashes and crack them.

- The following are some ways to increase the security of passwords:

  - Use Shadow files
  - Use Passwd+ to enforce strong passwords
  - Use one-time passwords
  - Use Biometric authentication
  - Audit access to key files
  - Inventory all active accounts

Shadow Files

- Shadow files make it difficult for users to gain access to the encrypted passwords.

- If your UNIX system is not using shadow files, you should either upgrade the operating system or use a program that will convert your **passwd** file to a **shadow** file.

- With shadow files an attacker needs root access to extract the encrypted passwords.

- Shadow files do not eliminate the threat, they just reduce the threat by increasing the chances that only legitimate users can access the passwords and run Crack.

- Crack can still be used on a system with shadow files; it simply requires an extra step and extra access for the user.

- To run Crack on such a system, an attacker must have root access to read the shadow file or some way to acquire a copy of the file.

- The copy of the shadow file must be merged with the **passwd** file before running Crack.
- To merge the files, Crack comes with a ***shadmrg.sv*** script that enables the user to combine the two files.
- The **shadmrg** script does not use arguments and must be edited and configured for the system on which it runs.
- The John password cracker also provides a script, called ***unshadow*** that will merge the shadow file into the passwd file.