On NT Password Security

By: Jos Visser

Version: 5th May 1997

This article was originally dated 4th april 1997, and has been reworked on 10th april 1997 after comments from interested parties and new insights. This version has been updated on May 5th (liberation day), in the 17th year of the reign of our Queen: Beatrice of Orange.

Table of contents

Introduction

On the 31st march 1997 a well known electronic news bulletin announced that a major flaw in Microsoft's "flag operating system" Windows/NT had been uncovered (see the original article). This article made strong suggestions that:

- The passwords of Windows/NT users could easily be cracked,
- This hack could be pulled of by "a reasonably skilled kid with a PC and a 28.8K modem,
- This hack could be used to automatically synchronize the passwords of users on Windows/NT and UNIX systems,
- An attack program could be sent by email to "steal" your Windows/NT passwords.

The various claims made in the original article left many Windows/NT users and administrators wondering about the security of their NT systems. We believe that this matter has been greatly exaggerated and that no breakthrough in cracking NT systems has actually been made. Some weeks later, more reports popped up about the ease with which NT password security could be subverted. In response to this, Microsoft has released a security HotFix which (amongst other things) adds stronger encryption of password information to Windows/NT (see ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postsp2/sec-fix/" and later in this article).

This article puts a technical perspective on the issue and investigates the true dangers and vulnerabilities of Windows/NT password security.

Sidestep: password security in general

It has been common knowledge for quite some time that userid/password authentication schemes are inheritantly weak. Users usually choose bad passwords (i.e., ones that can easily be guessed), write passwords down, mail passwords to co-workers or share passwords freely by telephone. For

this reason, high security environments do not rely on password authentication alone, but combine password security with smart cards or fingerprints.

Every system that uses a form of password authentication must store a representation of the password in order to check whether a logon attempt (local or through the network) is allowed. It should come as no surprise that passwords are of prime interest to hackers trying to break into a system. Secure operating systems must therefore ensure safe storage and transmission of passwords. I will first focus on the storage of passwords in the system password database. There are basically three ways of storing passwords:

- Storing the clear text passwords (a bad idea)
- Storing an encrypted version of the password (not a good idea either)
- Storing a unique obscured representation of the password (a fairly good idea)

It is obvious that storing the exact textual password in the system pasword database is an extremely bad idea. Modern operating systems (UNIX, Windows/NT) do not store passwords this way.

The least an operating system should do is encrypting stored passwords. In this way someone with access to the password database can not easily read the passwords stored there. However, encryption implies the possibility of decryption. A hacker who knows (or guesses) the encryption algorithm and the encryption keys can decrypt the stored passwords. Some operating systems that use encrypted storage of passwords allow the system administrator to decrypt any user's password. An example of such an operating system is MPE by Hewlett-Packard. It is nowadays considered unacceptable for an operating system to store encrypted passwords.

The third (and preferred) method of storing password is not to store the password at all! Instead, the operating system uses the password as a parameter in an irreversible mathematical function that calculates a *hash value*. This hash value is then stored in the system password database (which, strangely enough, does no longer contain passwords but hash values). In order to validate a logon, the system uses the password supplied by the user, performs the mathematical function and then compares the hash value with the value stored in the password database. The premise is that when the hash values are equal, the two paswords used to generate the hash values are equal as well: the user is allowed access. Both UNIX and Windows/NT use this third method of storing password information.

So if you look at a UNIX /etc/passwd file, the second field (the "password" field) does *not* contain passwords, but it contains the hash value that resulted from executing the UNIX hash function with the password as a parameter. Likewise, the Windows/NT security database (\WINNT\SYSTEM32\CONFIG\SAM) does not contain passwords, but hash values generated from the Windows/NT hash function with the user's password as an input parameter.

Cracking passwords

In operating system that store hash values instead of (encrypted) passwords it is **impossible** to decrypt user's passwords, even if you have full access to the system password database! For this reason, in UNIX the /etc/passwd database was usually readable by all users. This does not immediately lead to a breach of security because the hash values are not decryptable since they are generated with one-way (irreversible) functions.

However, systems storing hash values in publicly readable files are vulnerable to a so-called *brute force attack*. Because the hash functions are well documented, a hacker could try to generate all possible passwords, calculate their hash value and compare this hash value to the hash values stored in the system password database. Since the length of a password is bounded by a system limit, there is a finite number of possible passwords. It is theoretically possible to generate all possible passwords and hash them. Fortunately, the computing power needed in order to launch a full scale brute force attack is not generally available. In a system using eight character ASCII passwords there are approximately 7.2E16 possible passwords. Given a computer that can generate, hash and compare 1 million possible passwords per second it will take about 2258 years to search the entire password space. Since Windows/NT uses passwords of up to 14 characters a brute force attack with the same computer would take more than 600,000 billion years (or at least: very, very, long).

Unfortunately, systems usually limit the characters that can legally occur in passwords and, above all, users choose bad passwords. Analysis has shown that the amount of information present in a typical password is far less then the available number of bits would suggest. If you assume that users typically choose password shorter then eight characters, and use only letters and digits in their passwords, the number of passwords you have to search shrinks tremendously. A brute force attack, although still requiring a lot of computer power, becomes easier under those assumptions. Several claims have been made that a small number of Pentium Pro powered computers could deliver the computing power to hash, store and compare the universe of 'wimpy passwords'.

To worsen things, users normally choose pronouncable passwords that are easy to remember. It makes therefore much more sense to try the password 'april' before you try '44YThg6e'. Although both are perfectly valid passwords, users are much more prone to choosing a well known word or name as their password. This fact has lead to the *dictionary attack*. Instead of mindlessly generating all possible passwords, smart hacking systems use a database of common words and a few rules of thumb to generate all plausible passwords. The databases of these programs commonly include standard dictionaries, names of persons, types of cars and so forth. Common tricks such as reversing the password and substituting 0's (zeroes) for o's (ooo's) are part of the program logic as well. These smart password crackers are remarkably good at guessing peoples passwords. The best known password cracker for UNIX is the "Crack" program which is quite sophisticated. To run Crack you only need a well equiped modern PC (Pentium 133Mhz, 32 MB RAM, 1GB hard disk), so running these tools is basically within everyones reach. The latest version of Crack (5.0a) has been ported to Windows/NT as well! (see http://www.nmrc.org/files/nt).

So, although the passwords in a UNIX or Windows/NT password file are not decryptable using a straightforward algorithm, it is not impossible to crack one or more passwords using a brute force or dictionary attack. This fact has lead UNIX vendors to implement *shadow password files*. In UNIX systems equiped with this feature the password information (hash value) is removed from the publicly readable password file and placed in a separate (shadow) password file (typically /etc/shadow) which is accessible by the system administrator only (and by tools running with administrative capabilities). In Windows/NT, user and password information is stored in the SAM part of the registry. The SAM subkeys are protected with Access Control Lists that deny everyone (even the system administrator) access to the information stored there. So ordinary users do not have access to the stored password information. The system administrator can off course modify the ACL's to allow him/her access! To protect the password hash values even more, standard Windows/NT obfuscates the hash value a bit by DES-encrypting it with a user dependent (but easily determinable) number (for experts: the RID).

Rememember: dictionary attacks are only feasible because users choose bad (easily guessible) passwords!

Hashing functions

An important factor in password security is the quality of the hashing function. It should typically be a one-way function that generates a unique hash value for a given password. Ideally, two passwords that are almost the same (e.g. differ only in one position) should generate a completely different hash value (the proof of this is left to the reader as an exercise).

UNIX

UNIX uses a DES-like algorithm to calculate the hash value. The password is used as the DES key (eight 7-bit characters make a 56 bit DES key) to encrypt a block of binary zeroes. The result of this encryption is the hash value. Note: the password is not encrypted, it is the key used to perform the encryption! A strong feature of UNIX is that is introduces two random characters in the alogrithm (the *salt*). This ensures that two equal passwords result in two different hash values. From viewing the UNIX password file you can not deduce whether two persons have the same password. Even if they do, the salt introduced in the algorithm ensures that the hash values will be different.

LanManager

LanManager uses a 14-byte password (112 bits) which is split in two 56-bit entities. These two 56-bit entities are used to DES-encrypt a fixed (known) 8-byte magic number The result of these two DES-

encryption is concatenated to form a 16-byte hash value.

Windows/NT

Windows/NT uses the Internet standard MD4 hashing algorithm to generate a 16-byte hash of the Unicode encoded password.

Comparison of UNIX and Windows/NT hash functions

A strong feature of the UNIX hashing algorithm is that it uses a random element in order to diversify the hash function results. The effect of this is that two users with the same password will have different hash values stored in their user definition. This makes it more difficult for a hacker to perform password attacks because there is no straightforward way to hash a generated password. The password cracking program has to introduce the salt as well, lengthening the time it takes to perform the attack.

Because Windows/NT does not introduce a random element, attack programs can simply hash a would-be password and compare the hash value to all password fields in the NT password database. This comparision will yield all users using that same password.

Performing a Windows/NT password attack

As I've shown in this document, Windows/NT is (just like UNIX) vulnerable to a dictionary attack. In order to perform a dictionary attack a hacker must have:

- A list of Windows/NT users and their hashed passwords
- A password cracking program

pwdump

Windows/NT stores users and their associated information in the SAM (Security Accounts Manager) part of the registry. As far as passwords are concerned NT stores both the LanManager (DES) and the Windows/NT (MD4) hash values. This amounts to two 16-byte hash values. These hashes have been obfuscated a bit by DES-encrypting them. (Un)Fortunately, solid research (presumably by Jeremy Allison) has uncovered the keys with which these hash values have been encrypted. It is relatively easy to write an NT program that reads the SAM database, decrypts the hash values, and writes them out into a file.

Now, there is no need for you to run to your C compilers and write such a program. It already exists: pwdump.exe (see http://www.nmnrc.org/files/nt). This program performs exactly the tasks described above: it walks through your SAM database and writes out a file with userid's and the hash values associated with the user's password. The current version of pwdump.exe uses standard NT registry functions to walk through the SAM user database. In order to do so it first has to change the access control list of those registry keys to allow reading them. This needs Windows/NT administrative privileges. The effect of this is that you have to be a Windows/NT system administrator in order to run the current version of pwdump.exe.

So in order for a hacker to mount a password attack he/she first has to gain administrative access to your Windows/NT computer.

This severely limits the general usability of this attack because once a hacker has gained administrator access to Windows/NT, there are loads of other mischief that can be wrought without going to a lenghty dictionary attack.

Apart from gaining administrative access him/her self, a hacker can try and trick a system administrator in running a pwdump like program. Modern World Wide Web and email technology like ActiveX and MIME contain features through which programs are transmitted through the Internet and executed on the workstation of the user. By packaging pwdump in an ActiveX control and inviting a system administrator to view a web page a hacker can obtain the much needed dump of the NT password database.

It is therefore of the utmost importance that system administrators never run any program from an untrusted source!

System administrators should utilize the built-in security features of ActiveX and their email software in order to prevent malicious programs sent to them through the Internet of running on their machine. Unfortunately, many system administrators do not take the necessary care to prevent viruses, worms and trojan horses from running on their machine.

It should be stated here that Windows/NT is in this respect not less secure then UNIX. Any system administrator in UNIX can obtain a copy of the shadow password file (which contains the password hash values). What does make UNIX more secure is that the hashing function is stronger (uses salt).

Other NT password sources

There might be other places where copies of NT password information can be obtained. First of all the SAM database is world readable by default (try: cacls \WINNT\SYSTEM32\CONFIG\SAM). This file is normally in use by system components so you can not read it. However, copies of this file might exist (SAM.SAV) which are readable. I am not fully sure that the actual password hashes are stored in these files, but if they are, it is possible to retrieve them. The hidden implication here is that if the SAM files are readable, it might be possible to extract the password hash values with a normal user account! To my knowledge, all current methods of obtaining the password dump from within NT require administrative access.

It has been pointed out to me that the Windows/NT installation process leaves a copy of the password database in the \WINNT\REPAIR directory. Now, this SAM only contains the builtin Administrator and Guest accounts, but these might be enough to gain access in order to get the rest of the password database. The bottom line: protect your repair diskettes!

NTCrack

After having obtained a dump of the NT SAM database a hacker can mount an offline dictionary attack. This involves generating passwords (usually from a word list) and running the LanManager (DES) or MD4 (Windows/NT) hashing functions on them. There are currently two tools that can be used to crack a pwdumped NT database: "NTCrack" and "Crack 5.0a for NT".

"NTCrack" is a fairly straightforward tool that hashes a list of words and compares the hash values with an NT password database. It can be obtained at http://www.secnet.com. Because the Windows/NT hashing functions do not use a salt (like UNIX) the hashing/matching process proceeds at tremendous speed. Rumours are that it takes NTCrack about 5 minutes to match a 860,000 word word list against a SAM database dump containing 1,000 users!

"Crack 5.0a for NT" is an adapted version of the sophisticated UNIX Crack program. It comes with powerful algorithms and extensive multi-lingual word lists. See http://www.nmrc.org/files/nt for Crack 5.0a for NT.

Sniffing passwords from the network

Another way of obtaining passwords or information about passwords (hash values) is wiretapping the network (local or Internet) as client-server connections are being made. Tapping the local area network is extremely easy once you have physical access to the network cable. Microsoft has made tapping the network much easier with its "Network Monitor" tool, which features unlimited network tapping, protocol decoders and a nice GUI. Tapping the Internet is a bit harder because you need access to a central location such as a provider backbone. However, most provider employees are capable of tapping their own network and several reports of hackers tapping ISP backbones have surfaced over the last few years.

Older versions of the SMB protocol send the users password over the network without any encryption! Modern system (such as Windows/NT) support this for backward compatibility, but do not use this themselves.

The standard SMB protocol now used by LanManager and Windows/NT does not send password

information directly across the network. Instead it uses a challenge response mechanism. When a user logs on to a client computer he/she enters the password. The client computer calculates the password hash value and remembers it for future use. Whenever a client connects to an SMB server, the server generates an 8-byte random value which it sends to the client. The client uses the remembered 16-byte hash value together with 5 null bytes to create three 56-bit DES keys (16 bytes hash + 5 bytes null == 21 bytes * 8 == 168 bits == 3 * 56-bit DES key). The 8-byte random challenge is DES-encrypted with each of the three DES keys thereby generating a 24-byte response. This response is returned to the server. The server pulls the user's hash values from it's password database and performs the same calculation. If the server's 24-byte answer is the same as the client's 24-byte response the logon to the server is allowed.

Someone sniffing the intermediate network only sees the 8-byte challenge and the 24-byte response. To obtain the original hash values the attacker must perform three brute force DES attacks which is very difficult (although not impossible). However, this protocol is vulnerable to an attack with a modified client (will be explained later).

What would you do with a Windows/NT password?

An important question we have to ask ourselves is why people want to steal Windows/NT passwords? At first sight this may seem a strange question. Wasn't stealing passwords what hacking was all about? The answer is: it depends. Hackers want to gain access to your machine in order to do something what is normally not allowed. This could be:

- Rebooting your machine
- Modifying application to their (financial) advantage
- Impersonating another user

Since you need administrative access in order to acquire a copy of the Windows/NT password information, why bother with cracking the passwords at all! Once you have administrative access you can basically do what you want (although Windows/NT is a bit more secure there then UNIX because it allows for a clear distribution of administrative sub-priviliges and administrators can not covertly modify user's files without the attention-paying user noticing).

Indeed, most hackers will probably stop after having obtained administrative access to your NT machine because this allows them to do whatever they want. However, it might still be very tempting for them to try and crack the password database for the following reasons:

- Most people use the same password on more then one machine. Cracking one or more NT passwords will probably give the hacker access on other machines.
- Windows/NT hashed passwords are *password equivalent*. More about password equivalency in the next section.
- It allows them easy access to the user's applications and data files. As I stated before in Windows/NT it is impossible for a system administrator to modify correctly protected user files (mainly due to the fact that system administrators do not automatically have access to all files and there is no NT equivalent of the UNIX chown command).
- Passwords can be traded on the black market!

Password Equivalence

Due to the way LanManager authenticates remote users over the network, a hashed password (uncracked) can be abused by hackers. This is because the LanManager password is a *password equivalent*.

Remember that the LanManager authentication mechanism is a challenge-response protocol. The server sends a challenge, which is encrypted with the hashed password by the client and returned to the server (see a previous section). An important aspect of this protocol is that the client operating system does not actually need to know the password. It only needs to know the *hashed password*! The regular LanManager client software has the enduser input his/her password and calculates the hashed pasword from the manually entered original password.

Now suppose you have a modified client which has access to a stolen copy of a cracked Windows/NT password database (containing the hashed LanManager and Windows/NT passwords). When opening a network connection, the server generates the challenge and sends it to the client. The client looks up the hash value to use (in the copy of the NT password database) and calculates the response. The server concludes that the response has been generated with the same hash value as is stored in its password database and therefore authenticates the client. The client has accomplished this without knowing the original (clear text) password!

This means that even a good password (dictionary attack resistent) can be misused!

Creating such a modified client is not so hard as it looks. There are several good public domain LanManager compatible clients that come with source code included (such as the Linux smbfs or the SAMBA smbclient program). Modifying those is not within everyone's reach, but as soon as someone has done it this program will spread and become available to the average copycat hacker. The morale of this is that you must protect your NT password database with vigour! Even if your passwords are resistant against a dictionary attack they can be abused by the modified client!

The Microsoft HotFix

In response to all this, Microsoft has released a HotFix which (amongst other things) attempts to solve these problems. This HotFix can be downloaded from Microsoft's FTP server. It will also be a part of Service Pack 3 for Windows/NT Server and Workstation 4.0. For sites determined to secure their NT computers now, the HotFix can be installed at your own risk. Microsoft has not fully tested the fix. Incompatibilities with existing systems and applications may exist.

The fix contains the following parts:

- Strong encryption of the NT password hash values
- SMB Signing
- Protection of "anonymous connections"

Strong encryption of the NT password hash values

As has been stated before in this article, in an effort to protect the password hash values, Windows/NT encrypts these hash values before they are stored in the SAM. Unfortunately, the encryption chosen is pretty trivial: the hash values are DES encrypted with a key easily derived from a user attribute (the RID). Jeremy Allison has reverse engineered the algorithm, and his pwdump.exe program undoes this encryption.

The Microsoft HotFix enables the system administrator to configure strong encryption for the password hashes stored in the SAM. This feature uses a 128 bit system key which is unique for each NT computer. With the HotFix comes a separate utility SYSKEY.EXE which is used to enable strong encryption and manage the system key. Once strong encryption is enabled, the pwdump.exe program becomes useless because knowledge of the system key is nescessary in order to decrypt the password hash values.

The "weak" part of this scheme is that the system needs to know the system key in order to validate user logons. Therefore, the system administrator must make the system key available to the system during boot time. The HotFix gives you three possibilities (through the SYSKEY.EXE utility):

1. Store the key somewhere on the system
   A randomly generated key can be stored somewhere on the system. Microsoft claims that the system key is stored using a complex obfuscation algorithm. This is off course not very secure. My guess is that the obfuscation algorithm will be broken pretty soon. As soon as this happens, a modified version of pwdump.exe will be created which uses the system key to decrypt the password hashes, basically undoing the strong encryption feature of the HotFix.

1. Store the key on a floppy disk
   The SYSKEY.EXE utility can store a randomly generated key in a file called called "StartKey.Key" on a floppy. The floppy disk must be present during the boot process in order

to load the system key. A major nuisance of this mechanism is that either the floppy must be present in the drive at all times, or the floppy must be inserted manually at system boot time. If the floppy is always available (and the startup sequence of the computer excludes A:), security is completely absent. So the floppy must be removed from the drive after system boot. However, this inhibits an automatic reboot after a system crash! The system administrator must manually insert the floppy disk in order to allow the reboot to proceed. The current implementation of SYSKEY.EXE does not allow you to "feed" multiple NT computers with the same system key floppy. It is needless to say that a lost or corrupt system key floppy renders your NT system inoperable.

1. Enter the system key manually at boot time
   During the boot process a popup box appears inviting you to enter a password. A message digest function is executed on the password in order to determine the 128-bit key. Basically, the password is the system key. This method is quite secure but it requires manual intervention at every system boot. An "advantage" is that you can use the same password for multiple NT computers.

According to Microsoft, future implementations of the strong encryption feature will allow pulling the system key from external devices (i.e. smart cards). If you really need this feature I would suggest entering the password manually or storing the system key on a floppy.

SMB Signing

Another feature implemented with the HotFix is "SMB Signing". The technotes accompanying the HotFix describe how to implement SMB Signing (by setting registry keys), but does not explain what SMB Signing is! I think that SMB Signing has to do with placing digital signatures on all or part of the SMB protocol exchanges. In this way, clients and servers can be sure that both parties in the communication are who they claim to be. If implemented as I think it is, this will inhibit the modified client attack described earlier. I will look into this in greater detail.

Protection of "anonymous connections"

The third feature of the HotFix has got to do with protecting "anonymous connections". These are SMB connections that specify no userid, but which are allowed by Windows/NT under some circumstances. Anonymous connections and their implications are not a topic of this article. For more information, see the article on "RedButton" on http://www.ntsecurity.com.

Conclusion

The original article that spawned all this interest in NT password security made some pretty wild allegations. Luckily, this story contained mostly nonsense. Things are not as bad as they wanted us to believe. However, the article raised a few interesting points and a lot of attention.

It is certainly true that Windows/NT systems are vulnerable to various kinds of password related attacks. UNIX has been investigated for more than 20 years and a lot of security enhancements have been implemented by various UNIX vendors. Windows/NT has only just been getting the same sort of attention that UNIX has "enjoyed" for so long. Windows/NT users better pay attention to current NT security developments. New security bugs and fixes are bound to pop up!

The conclusions of this article in short:

- Windows/NT is basically vulnerable to a dictionary attack. Because the Windows/NT hashing functions do not use a "salt", such an attack will be much faster then the already popular UNIX password dictionary attacks.
- In order to perform a dictionary attack, a hacker must obtain a dump of the Windows/NT password database. We currently believe that you need administrative privileges to make such a dump, although other ways exist to obtain password hash values.
- Modern Internet services that allow unidentified programs to execute on the client computer could be subverted to make an NT password database dump and mail it to the hacker. Examples of these services are ActiveX controls and programs hidden in MIME email messages. It is never a good idea for a system administrator to run a UPO (Unidentified Program Object).
- The best protection against dictionary attacks is using passwords that are difficult to guess. Try

to convince your user population of this fact!
- A copied password database can be abused without cracking the passwords, although this requires some effort by the hacker.
- Microsoft has released a hotfix (which will be part of Service Pack 3) which solves most of these problems, against a cost of increased system administration overhead.

In general I advise the following:

- Protect administrative access to your Windows/NT computer as best as you can. Create only a small (manageable) amount of administrators and teach them the ways of security.
- Do not make a Windows/NT server accessible from the Internet for common file and print server access. This decreases the vulnerability for attacks with modified clients and other SMB-related programs.
- Only logon as an administrator in order to perform specific administrative tasks. Do not logon as an administrator for tasks such as mail reading or web browsing.
- Enable the security features of your browsers and mail readers. Do not run UPO's (such as ActiveX controls). Java applets seem secure enough, but ActiveX is a gaping security leak, waiting to be exploited!
- Have your users choose good passwords. Run Crack yourself to verify this (there might be ethical or legal aspects involved in doing so, check this with your local legal department)!
- In a secure environment, apply Microsofts HotFix.

About the Open Solution Providers

This article was brought to you as a public service announcement of the Open Solution Providers. The Open Solution Providers is a dutch consultancy specialising in UNIX, Windows/NT and Internet consultancy, software development and education. For more information check our home page at http://www.osp.nl.