How can Windows passwords be cracked - Part 1

In this article we will show you how Windows creates and saves password hashes and how those password hashes are cracked.

In this article, I will show you how Windows creates and saves password hashes and how those password hashes are cracked.

Introduce

Password (Password) is often the main way in preventing indiscriminate intrusion. However, although an attacker does not have physical access to the computer, they can still access the server via the remote desktop protocol (remote desktop) or authenticate a service through an external web application.

The purpose of this article is to introduce you to how Windows creates and saves password hashes and how those password hashes are cracked. After introducing how to crack Windows passwords, we'll give you some tips to protect you from the vulnerabilities in these types of attacks.

How does Windows save passwords

Windows uses two methods of hashing user passwords, both of which have their own strengths and weaknesses. These are LAN Manager (LM) and NT LAN Manager version 2 (NTLMv2). The hash function (hash function) is a one-way function that puts any amount of data through this function giving a fixed length string at the output.

Hash password LAN Manager (LM)

Hash LAN Manager is one of the first password hash algorithms used by Windows operating systems, only one version is supported until NTLMv2 is used in Windows 2000, XP, Vista and Windows 7. These new operating systems still support the use of LM hash for compatibility. However, it has been disabled by default in Windows Vista and Windows 7.

This password hash is calculated by a 6-step process:

1. User passwords are converted to all uppercase characters.
2. Passwords are added with 0 characters until there are enough 14 characters.
3. The new password is divided into two 7-character hashes.
4. These values ??are used to create two DES encryption keys, each of which is added with a parity bit to create 64-bit keys.
5. Each DES key will be used to encode a predefined ASCII string (KGS! @ # $%), Resulting in two 8-byte confidential text strings.

6. These two 8-byte cryptographic text strings will be combined to form a 16-byte value, which is a complete LM hash.

In fact, the password 'PassWord123' will be converted as follows:

1. PASSWORD123
2. PASSWORD123000
3. PASSWOR and D123000
4. PASSWOR1 and D1230001
5. E52CAC67419A9A22 and 664345140A852F61
6. E52CAC67419A9A22664345140A852F61

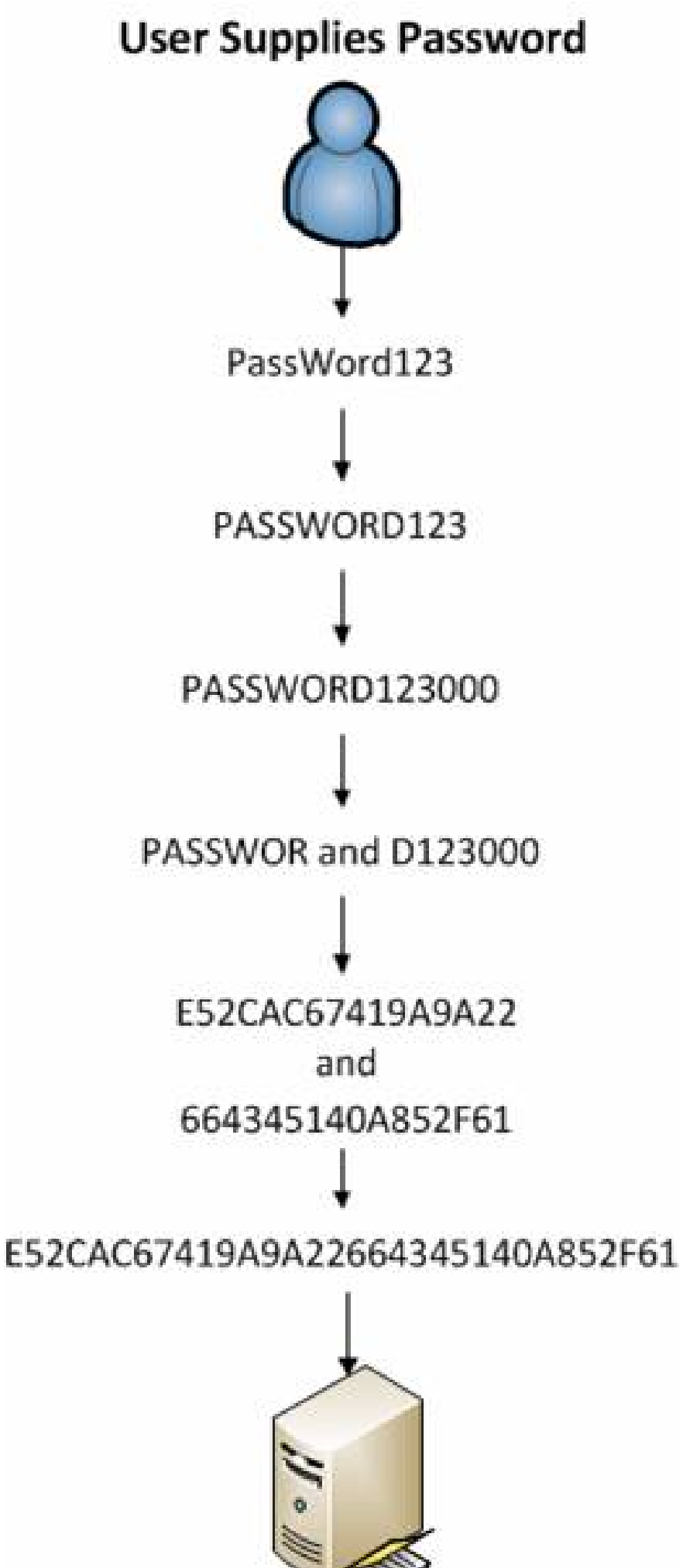
User Supplies Password
PassWord123
PASSWORD123
PASSWORD123000
PASSWOR and D123000
E52CAC67419A9A22
and
664345140A852F61
E52CAC67419A9A22664345140A852F61

Figure 1: A password is transformed into an LM hash

Passwords that follow LM hash methods have some disadvantages. The first drawback is that encryption here is based on Data Encyrption Standard (DES). DES started with an IBM project in the 1970s, the project was later modified by NIST, sponsored by the NSA and released as an ANSI standard in 1981. DES is said to be quite secure in many. years after thorough research in the 90s thanks to its 56-bit key size. However, in early 1998, the Electronic Frontier Foundation announced that it was possible to crack DES during a 23-hour period. Since then, DES has been considered obsolete and has since been replaced by Triple-DES and AES. However, these are also encryption standards that have killed victims with modern computing power and can be cracked easily.

Perhaps the biggest strength in LM hash is in the process of creating DES keys. During this process, a password issued by the user automatically converts all capitalized prints, then is inserted into a string of 14 characters long (this is the maximum length for the password by the method. LM hash), then divided into two hash 7 characters. This is a weak point when you have broken down the ciphertext string and only use uppercase ASCII characters. In essence, this algorithm makes using other characters as well as increasing the password length meaningless, which is what makes LM passwords vulnerable to repeated crack attempts. brute-force.

NTLMv2 password hash

NT LAN Manager (NTLM) is a Microsoft authentication protocol, created to succeed LM. There are many advanced downloads, NTLMv2 is accepted as a new authentication method worth choosing and implemented in Windows NT 4.

The process of creating an NTLMv2 hash (from now on we abbreviated as NT hash) is a much simpler process with what the operating system does, it relies on the MD4 hash algorithm to create the hash thanks to a series of mathematical calculations. The MD4 algorithm is used three times to create NT hash. In fact, the 'PassWord123' password will have the result '94354877D5B87105D7FEC0F3BF500B33' after using the MD4 algorithm.

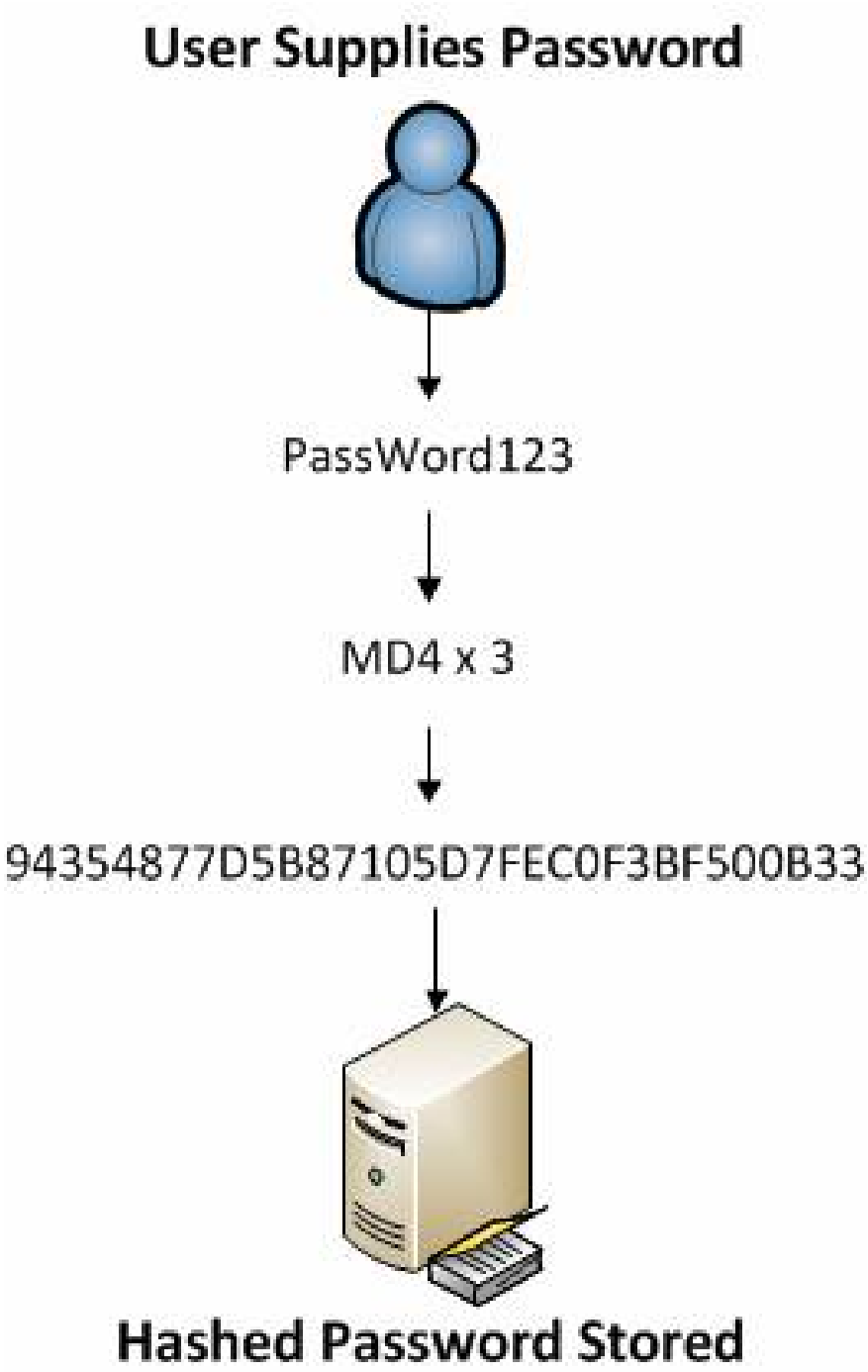

Figure 2: Password is converted to an NTLMv2 hash

MD4 is considered to be significantly more powerful than DES because it allows for a longer length of password, there is a distinction between lowercase and uppercase characters, not dividing the password into smaller parts (what makes it so Easy in cracking).

Perhaps the biggest complaint with NTLMv2 hash is that Windows does not use a technique called salting (temporarily translated as salt). Salting is a technique in which a random number is generated to calculate the hash for the password. This means that the same password can have two different hash values ??completely,

which is really ideal.

In this case, the user is able to create what are called *rainbow tables* . These *rainbow tables* are not brilliantly decorated *tables* ; they are actually tables containing hash values ??for the number of passwords possible for a certain number of characters. Using *the rainbow table* , you can simply retrieve the hash value extracted from the target computer and perform a search. When the hash value is found in the table, you will have the password. As you can imagine, a *circular bridge table* even with a small number of characters can become very large, meaning that multiplying, storing and indexing them will be a difficult task.

Conclude

In the first part of this series, we explained the password hashes and the mechanisms Windows uses to create and store those values. We also introduced the weaknesses of each method and the avenues can be used to crack those passwords. In the next part of this article series, I will show you the process of extracting and cracking these hashes to prove its weaknesses. Once proven, we will provide you with some additional security layers and create a real strong password.

HASH HAMPS

The hash function (hash function) is a one-way function that puts any amount of data through this function giving a fixed length string at the output.
For example, the word "Illuminatus" goes through the SHA-1 function, giving the result E783A3AE2ACDD7DBA5E1FA0269CBC58D.
We just need to change "Illuminatus" to "Illuminati" (converting "us" to "i") the result will be completely different (but still has a fixed length of 160 bits) A766F44DDEA5CACC3323CE3E7D73AE82.

The two important properties of this function are:

• Unidirectional: it is impossible to deduce the original data from the result, which is similar to the fact that you cannot rely solely on a strange fingerprint, but infer who is its owner.
• Uniqueness: the probability to have a collision, ie two different messages with the same hash result, is extremely small.

Some applications of hash functions:

• Intrusion prevention and detection: An anti-intrusion program compares the hash value of a file with the previous value to check if the file has been changed by someone.
• Protect the integrity of messages sent over the network by checking the hash value of messages before and after sending to detect changes even the smallest.
• Create key from password.
• Create electronic signatures.
SHA-1 and MD5 are the two most commonly used hash functions and are used in many security systems. In August 2004, at the Crypto 2004 conference, a collision was found with MD5 and SHA-0, a weaker version of the SHA-1 hash function. Shortly thereafter, around mid-February 2005, a group of three Chinese cryptographers discovered a method to find collisions with SHA-1 within only 269 computational steps (ie brute-force can be several thousand times faster.