CONTI'2008
The 8th INTERNATIONAL CONFERENCE ON TECHNICAL INFORMATICS, 5-6 June 2008, TIMISOARA, ROMANIA

On the security of some authentication mechanisms from Windows

Bogdan Groza*, Andrei Alexandroni**, Ioan Silea***, Victor-Valeriu Patriciu****

*Politehnica University of Timisoara, Faculty of Automatics and Computers, Timisoara, Romania, E-Mail: bogdan.groza@aut.upt.ro, WWW: http://www.aut.upt.ro/~bgroza/

**Pensive S.A., Brussels, Belgium, E-Mail: andrei@pensive.eu

***Politehnica University of Timisoara, Faculty of Automatics and Computers, Timisoara, Romania, E-Mail: ioan.silea@aut.upt.ro, WWW: http://www.aut.upt.ro/~isilea/

****Military Technical Academy, Department of Computer Engineering, Bucharest, Romania, E-Mail: vip@mta.ro, WWW: http://www.mta.ro/conducere/victor_patriciu.php

Abstract* – *The paper investigates some authentication mechanisms used in Windows. In particular, the NTLM authentication protocol, which is commonly used in several solutions from Microsoft, is analyzed. The NTLM authentication is completely unsafe in several variants of use and some of its weaknesses previously known. A critical analysis is done, the weaknesses are explained and the safe solutions are underlined. As a practical example it is shown how the NTLM authentication from SharePoint based portals can be exploited to steal passwords and how to configure the NTLM for a safe use. This analysis is relevant as SharePoint becomes widely used and NTLM is still the default option and the only authentication mechanism available when there is no support for Kerberos. Nevertheless, a comparison between the password based authentication from UNIX based OS and Windows is done.

Keywords: *authentication, cryptography, NTLM, protocol.*

I. INTRODUCTION

Authentication, and in particular entity authentication or identification, is a major security objective nowadays as proving someone's identity is the most common action in all systems. Also there is a large variety of authentication protocols based on cryptographic techniques, which have been studied for a long time by the security community, some of them are known to be secure others are known for their weaknesses. A basic textbook [3], [12] or a survey [5] can be consulted for more on authentication protocols.

It may be relevant to note that there are three types of entity authentication mechanisms: password based, challenge-response and zero-knowledge. Password based authentication is the most commonly used mechanism and it is based on the use of a secret known as password. Because passwords can be stolen, one-time passwords are an improved mechanism which makes passwords valid only once, however their use in practice is limited. Challenge-response mechanisms are also frequent in practice and offer a stronger security level. The main ideea is that the identity of a user is verified based on a response to a particullar challenge that is usually a random value, therefore, even if an attacker captures a response for some challenge, it can not be used for subsequent impersonation of the user, because future challenges will be different. Challenge-response protocols can be built on both symmetric or assymetric cryptographic primitives. Zero-knowledge protocols are the most advanced authentication protocols, however, they are more computational intensive and more difficult to implement, and because of this are not frequent in practice. As will be discused in what follows, both password based authentication and challenge-response authentication are present in Windows.

Although there are numerous solutions with a strong foundation and provable security, still protocols that are known to be insecure continue to be used in practice. One example of such protocol is the NTLM (NT LAN Manager) protocol used by some Microsoft products with the SMB protocol. NTLM is a challenge-response protocol consisting of three phases between the client and the server: the negotiation phase, when the client sends a message to the server; the challenge phase, when the server responds to the client message with a challenge and the authentication phase, which is the most important part when it comes to cryptography. In the authentication phase the client replies to the server challenge and the access to the requested resource is granted or denied. More details on the protocol will be given in the forthcoming section. Although some of its weaknesses are known [4], [10], [14] it continues to be used in several situations. One such example, that will be outlined in this paper, is the use in web portals developed through Microsoft SharePoint.

SharePoint (http://www.microsoft.com/sharepoint) is the document management and collaboration platform from Microsoft. It is a platform used by organizations to facilitate content management across the enterprise, to easily manage and track business processes and to provide collaborative spaces for different groups of users. SharePoint is becoming one of the preferred solutions for companies because of the ease of use and the large number

of issues addressed by its features. The first versions of the product were released in 2001, advancing over time to the current versions which are Windows SharePoint Services 3.0 - a free product coming as an additional component of Windows Server 2003 and Windows Server 2008, and Microsoft Office SharePoint Server 2007 (MOSS 2007) - the full-featured product coming as an additional server. SharePoint solutions can be used in a large variety of environments, starting from collaborative spaces for universities to eGovernment resource sharing platforms.

Regarding authentication on the SharePoint web application, which is a practical example for this paper, there are two main possibilities of configuration: NTLM authentication (which is the default), and Kerberos authentication. A great number of real world implementations of SharePoint use NTLM authentication, although the ticketing-based protocol Kerberos is known to be more secure. NTLM is the preferred choice for the authentication protocol to be used mainly because it does not require any additional configurations. On the other side, Kerberos requires a trusted connection to the Active Directory domain Key Distribution Center for both the client and the server. The client, willing to authenticate by using Kerberos, in order to get access to a protected resource, must construct a service principal name, which should be previously configured (by using the Setspn.exe tool included in the Windows Support Tools for example). NTLM is still used in some situations: client is authenticating to a server using an IP address; client is authenticating to a server that belongs to a different Active Directory; no Active Directory domain exists; if a firewall restrict the ports required by Kerberos. Consequently, in many cases, the default NTLM authentication is chosen for SharePoint implementations.

The paper is organized as follows. Section II holds an overview of the NTLM authentication protocol. In section III some design weaknesses of the used cryptographic primitives and of the protocol are presented, also a comparison between the pasword based security from Windows and Unix is done, and the insecurity of NTLM for use in SharePoint solutions is inspected. Section IV holds the conclusions of the paper.

II. AN OVERVIEW OF THE NTLM AUTHENTICATION

Besides the anonymous response variant in which no authentication takes place, five variants of authentication, based on the response for the given challenge, are available: LM Response, NTLM Response, NTLMv2 Response, LMv2 Response, NTLM2 Session Response. A complete description can be found in [9] and another good reference is [10]. All five variants are challenge-response mechanisms based on the following three round paradigm:

1. *Client* $\rightarrow$ *Server : Type 1 Message*
2. *Server* $\rightarrow$ *Client : Type 2 Message (includes the 64 bit challenge from the server)*
3. *Client* $\rightarrow$ *Server : Type 3 Message (includes the response from the client)*

The type 1 message is used only to negotiate the type of authentication and for this exposition details are not useful. The type 2 message contains a challenge from the server which is an 8 byte Nonce. The type 3 message, which is the response to the challenge, is constructed different for all five variants.

Three variants are based on the LM Response with DES [7] as the underlying cryptographic primitive. The response stage for LM Response and NTLM Response is the following:

$$Client \rightarrow Server:$$
$$DES_{K1}(challenge) \,||\, DES_{K2}(challenge) \,||\, DES_{K3}(challenge)$$

In the case of *NTLM2 Session Response* the challenge is concatenated with a *Nonce* generated by the client in order to avoid dictionary attacks. This value is hashed using MD5 [13] and only the first 64 bits are preserved to obtain the session hash $sessionHash = \lfloor MD5(challenge \,||\, Nonce) \rfloor_{64-bits}$:

$$Client \rightarrow Server:$$
$$DES_{K1}(sessionHash) \,||\, DES_{K2}(sessionHash) \,||\, DES_{K3}(sessionHash)$$

The keys for the DES encryption in step 3 are based on a 16 byte *KeyMaterial* generated from the user password. For the *LM Response* the first 14 bytes of the password are used to create a DES key (if the key is smaller than 14 bytes it is null padded). Then these two values are used as keys to encrypt the constant “KGS!@#$%” also called “magic constant”. This encrypted value is also known as the *LMHash*:

$$KeyMaterial =$$
$$DES_{KE1(password)}(\text{"KGS!@\#\$\%"}) \,||\, DES_{KE2(password)}(\text{"KGS!@\#\$\%"})$$

Here, by *KE1* and *KE2* we denote two key extraction functions which are used to extract the DES keys from the password. These functions perform some non-cryptographic operations on the password which are not relevant, except for the fact that all letters are turned to upper-case (for details see [9]). For the *NTLM Response* and *NTLM2 SessionResponse* the password bytes are simply passed through the MD4 hash function to obtain:

$$KeyMaterial = MD4(password)$$

Now these 128 bits are null-padded with 40 bits and used to create the 3 DES keys, i.e. *K1*, *K2*, *K3*, as follows:

$$K1 = Transform(KeyMaterial_{1,7})$$
$$K2 = Transform(KeyMaterial_{8,14})$$
$$K3 = Transform(KeyMaterial_{15,16} \,||\, 0_{17,22})$$

The notation $0_{17,22}$ means that the bits from 17 to 22 are all set to 0. These three mechanisms are also suggested in figure 1.

Also, there are two variants based on the LMv2 Response, depicted in figure 2, which use HMAC-MD5 [1] as the underlying cryptographic primitive. The only difference between them is that one is using a client Nonce while the other is using a client Blob which besides the Nonce also contains more specific information such as target information and current time in milliseconds. The 3-rd round of the authentication protocol in this case is the following for *LMv2 SessionResponse* and *NTLMv2 SessionResponse* (for NTLMv2 *clientNonce* is replaced by *clientBlob*) :

$Client \rightarrow Server$:

$$HMAC-MD5_{HMAC-MD5_{MD4(password)}(user||target)}(challenge \,||\, clientNonce)$$

This completes the description of NTLM and its variants. In the next section its cryptographic vulnerabilities will be investigated.

III. CRITICAL COMMENTS AND EXPLOITING VULNERABILITIES IN SHAREPOINT PORTALS

A) Cryptographic vulnerabilities

Of course, for the variants based on the LM Response, the first obvious problem is the use of DES which is an outdated encryption function. The use of DES is of course inappropriate for these days, as attacks over this function started to be successful in the late nineties. Also there were a lot of speculations of the design of DES and some details on the design criterias were made public only in the nineties [6]. Still, the techniques of differential cryptanalysis proposed by Biham and Shamir [2] and linear cryptanalysis proposed by Matsui [11] do not represent a threath for the use in a password based authentication, as they are chosen palintext attacks wich require large amounts of plaintext-chipertext pairs. But nowadays, from March 2007, dedicated machines such as Copacobana can break DES in an average time of 6.4 Days (http://www.copacobana.org/). Also, the Deep Crack machine from Electronic Frontier Foundation is capable of testing more than 90 billions DES keys per second, which means that the entire key space can be exhausted in about 9 days, the average time for finding a key will be 4.5 days (http://w2.eff.org).

However the problem is even bigger at the generation of the DES keys since the third key, i.e. $K3 = Transform(KeyMaterial_{15,16} \,||\, 0_{17,22})$, has only 16 bits of random information while the last 40 bits are all set to 0. This finally leads to negligible entropy for the third key which can easily be found by a brute force search. This makes possible to retrieve the bytes from 17 to 19 of the key material by only testing 65536 values, this can be done in less than a second on a modern computer. Recovering these 2 bytes of the key material further makes possible to mount a more efficient dictionary attack on the password by using an offline computed dictionary of passwords and comparing the recovered last 2 bytes of the key. Although there will be many passwords that match the last 2 bytes, still the number of candidate passwords from the dictionary is reduced. As the third key is trivial to break due to its low entropy, all that is needed is to recover the first two DES keys. This should not even require twice the time needed to crack DES as the search is done over the same encrypted message, i.e. the challenge, and just the output of the DES permutation should be compared with two different values.

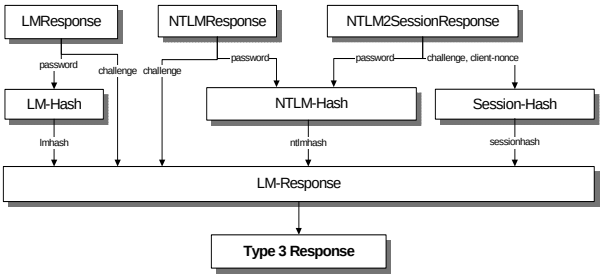

Fig. 1. Block diagram of authentication mechanism based on the LM-Response.

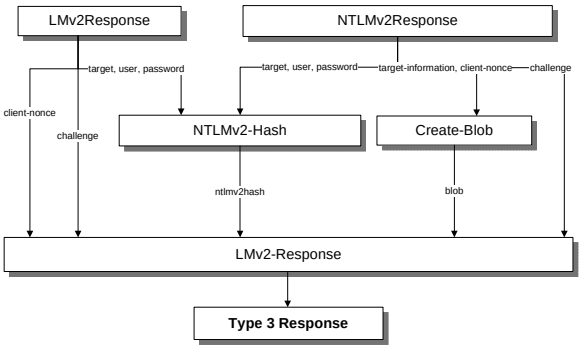

Fig. 2. Block diagram of authentication mechanisms based on the LMv2-Response

Even in the case of the *NTLM2 SessionResponse* when the client-nonce is used, it is still possible to mount dictionary attacks on the last two bytes of the key as it is easy to observe that the key material is independent on the client nonce. As for *LM Response* and *NTLM Response*, dictionary attacks can be done over the entire passwords as

there is no nonce from the client. Therefore we can conclude that all variants based on the LM Response are vulnerable to a dictionary attack. The *LMHash* itself is also insecure because the two values computed with DES are cryptographically independent and attacks can be mounted separately on each of them. For example if a password has 10 characters it will be as strong as having only 7 characters since finding the last 3 characters can be done in a matter of seconds and only finding the first 7 characters can take significant time.

Even more, specialized machines such as Copacobana or DeepCrack, which are available only for individuals with solid financial background, may not be required to crack DES in the context of the wide-spread of grid systems and as hackers could get control on hundreds to thousands computers. Therefore the use of protocols based on DES must be avoided in practice.

It is also easy to observe that for answering the challenge in the NTLM protocol it is sufficient to find the DES keys and finding the password itself may not be necessary. Therefore, because of the use of DES, the protocol can be broken no matter how strong the password is. If recovering the password itself is needed, using rainbow tables, which are becoming more popular and efficient (http://www.freerainbowtables.com/), can give good results very fast. Also, crackers such as Ophcrack (http://ophcrack.sourceforge.net/) and Cain&Abel (http://www.oxid.it/index.html) can be used for rainbow cracking LM and NTLM hashes.

Both variants based on *LMv2Response* appear to be secure, but the security level is arguably weak as MD5 hash function has several weaknesses [17], [20]. However, we do not see at this moment how the well known collision attacks on MD5 images can lead to security breaches in practice, but still we find it inappropriate to use MD5 in a contemporary solution. It is well known that new designs should use functions from the SHA-2 family [8]. Also, the use of SHA1 should be avoided as this function also has weaknesses [19].

For the LMv2 based response it is relevant to note that verifying one password could be done much faster than for LM-Response, this is because computing the HMAC-MD5 is much faster then computing 3 times the DES permutation. However, faster does not mean better, as cracking passwords that are small, for example less or equal to 6 characters, can be done faster for the *LMv2Response* than for the DES based password. We believe that it would be more efficient to take a measure such as iterating the hash over the password several times in order to increase the time for a brute force search (this measure was also used in other password based authentication systems including the one used in UNIX). Table 1 shows the computational time required by the 5 responses, the computational time was measured on the implementation from [9], which is done in Java and better timings can be achieved. By using the results from table 1, it is easy to determine for example that breaking a 6 char alphanumerical password can be done in an average time of 2 days for the *NTLMv2 Response*.

TABLE 1. Computational time for the 5 response types from NTLM (number of responses computed per second).

Response	CPU Intel T2300, 1.66Ghz	CPU Intel E6750, 2.66Ghz
LMResponse	$10\times10^3 / s$	$20\times10^3 / s$
NTLMResponse	$19\times10^3 / s$	$36\times10^3 / s$
NTLM2SessionResponse	$18\times10^3 / s$	$33\times10^3 / s$
LMv2Response	$83\times10^3 / s$	$143\times10^3 / s$
NTLMv2Response	$76\times10^3 / s$	$143\times10^3 / s$

B) A comparison between Windows and Unix password based authentication

The aforementioned *LMHash*, the 16 byte key material obtained from the DES encryption of a predefined constant, is also used in Windows NT based OS until Windows Server 2003, including Windows XP, to authenticate users on the login screen. In this section we investigate the differences bewteen the UNIX and Windows password authentication system.

Any password based system has to store passwords in some files. Traditionally the encrypted passwords in UNIX are stored in the file */etc/passwd*. However as this file is publicly accessible, in order to not expose the encrypted passwords of the users to other users that can attack them, the encrypted passwords are stored in the file */etc/shadow* which is accessible only by the root. For example, if we set the user name to *"user"* and the password to *"password"* the corresponding line from the file */etc/passwd* looks like following:

```
user:x:503:503:user:/home/user:/bin/bash
```

The fields of this entry corresponds to the following: nickname:password_hash:UserID:GroupID:Complete_Name:home_dir:shell_bin. What is relevant for us is the "x" after the user that indicates that the encrypted password is actually stored in */etc/shadow*. If one logs as root and has access to */etc/shadow* it will see the following line:

```
user:$1$WjmoADsj$GymnrxaSCM/EXMShtAbEn0:13536:0:99999:7:::
```

Here the value *1WjmoADsj$* is a random salting value with which the password is hashed (the salt is needed to prevent dictionary attacks). The value *GymnrxaSCM/EXMShtAbEn0* is the actual hash of the salted password. The remaining fields include: the

minimum number of days between password changes, the maximum number of days until the password must be changed, the number of days of warning given before the password must be changed, the number of days after the password must be changed when the account becomes unusable, the date (expressed as the number of days since January 1st, 1970) when the account is expired.

If one wants to generate such passwords, then the *crypt* command can be used. If a call to *crypt("password", "1WjmoADsj$")* is made then the value *1WjmoADsj$GymnrxaSCM/EXMShtAbEn0* is returned. The *1* followed by at most 8 characters indicates that the encryption is performed using an MD5-based algorithm instead of DES (*man crypt* command can be used for more details on *crypt*).

In order to contrast with table 1, by using the UNIX *time* command, we measured the time required to compute a password and the results are in table 2. It can be easily seen that computing UNIX password takes more time than for Windows.

TABLE 2. Computational time for the crypt command in UNIX based systems.

	CPU Intel T2300, 1.66Ghz	**CPU Intel E6750, 2.66Ghz**
crypt command	$1.3\times10^3 / s$	$2.5\times10^3 / s$

In Windows XP, which is for the moment the most widely used version of Windows, passwords are stored in *\system32\config\SAM* and *\system32\config\SYSTEM* from the windows directory (SAM stands for Security Accounts Manager). Although this file can not be read when Windows is running, it may be accessed by booting from a DOS system disk or its content can be dumped by some programs (for example several programs under the name *pwdump* can perform this task). The most relevant fact is that passwords are computed as the aforementioned *LMHash*. We note that the *LMHash* is disabled in Windows Vista, where is replace with the *NTHash* (computed with the MD4 hash function), but comes as a default with previous versions of Windows.

Now, some differences and weaknesses of Windows OS compared to UNIX password system can be resumed (in a potential order of relevance):

1) No salt is used in XP and this makes dictionary attacks feasible. On contrary UNIX password are computed using a salt which makes passwords not vulnerable to such an attack.

2) Using two independent DES transformations makes a brute force attack as hard as breaking only one DES transformation, which is feasible to break nowadays. UNIX does not use DES anymore, and even when it used DES in the past, it was only one DES transformation which is essentially the same as using two transformations on the same message. The fact that the two DES transformations have the same message and independent keys makes passwords of length 8, 9, 10, 11, 12 almost as secure as a 7 character password since 2, 3, 4, 5 characters are feasible to attack (as one ASCII character has 7 bits, 5 characters will lead to 35 bits which certainly can be efficiently brute-forced). This is relevant as according to [18] almost 60% of users passwords are from 8 to 12 characters and the *LMHash* will not offer more security for these users than for the 23% users that have 7 characters password. Arguably, 6 characters will also be easy to crack and therefore 13 characters passwords will not offer more security.

3) In the computation of the *LMHash* all letters are set to uppercase (see the source code from [9] for example). This makes alphabetical characters twice as easy to break.

4) The DES transformation can be computed much faster then the *crypt* command from UNIX as experimental results showed. This means that brute force attacks reveal passwords much faster in XP than UNIX. To obtain a higher lever of security the one-way function should have been iterated several times in order to compute a password.

As a partial conclusions, we can state that Windows XP offers less security than UNIX based systems (in particular our experimental results were done in Fedora 5.0 but they should be essentially the same in other versions). A recent note from Microsoft shows how to disable the *LMHash* which will force the use of the more secure *NTHash* [24]. It is somewhat disappointing that although Microsoft delivers automatic security updates quite frequently, there is no automatic update to disable the *LMHash*.

It should be also stated that another way to hack into a system would be to use a bootable disk and simply to overwrite the password file. However, although this attack is much easier to mount, and there is no cryptographic countermeasure against it, it is commonly acknowledged that it does not have the same security impact as recovering the password. This is mainly because of two reasons: first is that if the password is overwritten the user will notice that its password was changed and secondly a particular user password can help break into other accounts of the user as well (for example one user may use the same password to log on Windows XP and on Yahoo mail, and breaking it on XP will be much easier and will also give access to the mail account).

C) Some remarks on the design of the protocol

So far we have investigated cryptographic weaknesses of the NTLM. However, it should be also stated that protocols

can hide design weaknesses which makes possible to break them even if the underlying cryptography has no weaknesses. Such an attack, known as a man-in-the-middle attack, was suggested in [4] and is feasible on some variants of NTLM.

For a complete knowledge of all the attacks that are feasible and to validate the variants of NTLM which are secure a formal verification should be done. This is subject of future work for us. However there is a formal analysis of the MS-CHAPv2 protocol done under the AVISPA project and no vulnerabilities were found [23]. MS-CHAP [21], [22] is another challenge-response authentication protocol from Microsoft and is similar to NTLM. Since NTLMv2 is somewhat analogous to MS-CHAPv2, and no attacks were found on the MS-CHAPv2 [23], it is likely that NTLMv2 also has no flaws. It should be also stated that there are two versions of MS-CHAP and both of them were analyzed by Schneier et al. [15], [16] who found several cryptographic weaknesses.

D) The practical use of NTLM in SharePoint Solutions

The NTLM authentication protocol is frequently used in Microsoft Windows networks and its use is not restricted to SharePoint solutions; it can also be used as an authentication mechanism for other applications, such as a custom web application. The NTLM security levels can be accessed and modified in the Windows Registry by changing the *lmcompatibilitylevel* DWORD value located at the following registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa as shown in figure 3. A technical article on the *lmcompatibilitylevel* setting can be found in [10].

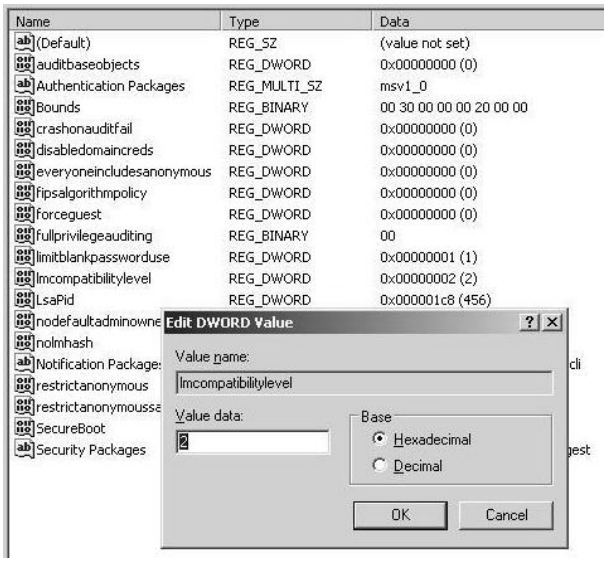

Fig. 3. Setting the *lmcompatibilitylevel* in Windows registry

The available levels set by the *lmcompatibilitylevel* variables are:

1) Level 0: is the less secure option, allowing only the LM and NTLM responses to be sent by clients, and disallowing the use of NTLM2 Session Response.

2) Level 1: with this option clients can use LM, NTLM, and NTLM2 Session Response.

3) Level 2: indicates that clients can use NTLM and NTLM2 Session Response.

4) Level 3: indicates that clients can use only the NTLMv2 response.

5) Level 4: allows clients to use only the NTLMv2 response, and explicitly denies LM responses to be accepted by the authenticating server.

6) Level 5: allows clients to use only the NTLMv2 response, and explicitly denies both the LM and NTLM responses to be accepted by the authenticating server.

The default installation of Windows Server 2003 comes with the default level 2, which is completely unsafe as it uses older versions of the responses: NTLM and NTLM2 Session Response. The newest operating systems from Microsoft (Windows Vista and Windows Server 2008) resolve this issue by enforcing clients to use only the NTLMv2 response. However, Windows Server 2003 is still largely used in production, being currently the most used Microsoft server operating system.

With respect to the discussion on NTLM from section II, we can establish the recommended security levels. Unsafe configurations are levels 0, 1, or 2. This is because the clients would always attempt to use the unsafe versions of NTLM: LM, NTLM, or NTLM2 Session Response. Safe configurations are levels 3, 4, or 5. This is because clients are required in these cases to use the NTLMv2 response, which offers a much stronger authentication.

Consequently, it is recommended to set the *lmcompatibilitylevel* value to a value greater or equal to 3 whenever possible. The most common restriction of using the recommended security level is when needed to accommodate legacy clients that do not support the NTLMv2 response (such as Windows 95/98, or non-Windows computers). However, still a great number of organizations overlook this security breach and do not change the default security level even when this is feasible. This is mostly because authentication protocols are widely misunderstood or their importance underestimated. As practice shows, in most cases, user consider to increase security levels only after critical situations occur due to lack of security. Therefore it is likely that weak security levels will continue to be used in Windows.

E) A practical example: attacks over SharePoint based portals

A possible attack on the authentication from SharePoint applications is done by capturing network traffic passing between a client requesting a resource on a SharePoint site and the authenticating server. This could be done for example by an intruder in the company's local area network, or even by an employee of the company trying to steal sensitive information. After capturing packets, the intruder can mount an attack to find the password of the victim in order to illegitimately authenticate afterwards to a protected resource by using the victim's username and password.

To simulate a practical attack, Wireshark (http://wireshark.org) has been used as the tool for capturing network protocol traffic between a client and a server. The target environment tested is the most typical scenario used by companies to implement SharePoint solutions. It is composed of: a Microsoft Office SharePoint Server (MOSS) 2007 hosted on a Windows Server 2003 R2 SP2 machine, acting also as a domain controller and DNS server, and a Windows XP SP2 client machine. The target environment has been virtualized by using Microsoft's Virtual PC 2007. This configuration was set up only for experimental purposes, but it simulates with high fidelity a real world environment.

The first step taken in the set up of the test environment was the initial installation of the server operating system. After all the initial configurations were done and the network interface has been connected to the local area network and assigned a private IP address, the server was promoted to a domain controller and the DNS server role was added for internal name resolution. Further, the prerequisite software necessary for MOSS 2007 was installed: .NET Framework 2.0, Internet Information Server (IIS) 6.0 with the World Wide Web service, and SQL Server 2005 for hosting the MOSS databases. Then, after the initial configuration of the SharePoint farm and after starting the SharePoint services, a new web application has been created to host a collaborative SharePoint site. The configuration for the SharePoint web application and site collection has been done through the MOSS Central Administration console on the server.

Then, after the configuration of the SharePoint farm, a new web application has been created to host a collaborative SharePoint site. A domain user has been created on the domain controller, and this user has been imported to the previously created SharePoint site.

Using the Wireshark tool, a new network traffic capture was started for the network interface between the two machines involved in the communication. When the user attempts to authenticate to a SharePoint resource he will need to enter his password on a logon window, or he can be automatically allowed access on the SharePoint site due to his domain credentials he is logged on with on the domain.

From this point on, the relevant parts of the captured packets can be easily extracted for further analysis. For example, when a password set to *"poli"* was used (the password is indeed small, but we use it just as a proof of concept) for authenticating a user on a SharePoint site using the default security level of 2, a packet containing the following information was captured:

Session nonce = challenge || clientNonce
=0x56c873557e88dcb2 || 0x08433915f46b868b
=0x56c873557e88dcb208433915f46b868b
27d151b1037f79c2a5c2646826de164e

NTLM Response=
$DES_{K1}(sessionHash) \| DES_{K2}(sessionHash) \| DES_{K3}(sessionHash)$
=0x86ff70f0483b5de2fd31c2b42fec17f80
d98bb9b887dab6b

Since the NTLM2 Session Response was used, the value of sessionHash was computed as shown previously $sessionHash = MD5(challenge \| clientNonce)$. The packet fragments from above are the most important parts of the messages exchanged between the client and the authentication server, which could then be used for revealing the user's password. Subsequently, the password was found by an exhaustive search in a matter of hours. Although this password is small, and useful just as an artificial example, it may be meaningful as several papers point out that user still choose small passwords. For example according to [18] 65% precent of the users use passwords of 8 characters or lower. Because of the low entropy, as ASCII digits loose one bit of information on each byte, these passwords can be easily cracked. For example a password of 6 chars can be cracked in a matter of days and about 15% of users choose passwords of this size according to [18].

IV. CONCLUSIONS

As a final conclusion we believe that using any of the variants based on the LM Response must be avoided. This is because as DES can be cracked in present days by a brute force search on the DES key-space. Also, the adversary does not even need to recover the password itself in order to respond the challenge, the LM Response based variants could be all cracked, no matter what is the entropy of the password and its length.

Also, for LMv2 based response it is relevant to note that verifying one password could be done much faster than for LM-Response and therefore, cracking passwords that are small, for example less or equal to 6 characters, can be done very fast even for the LMv2 Response. In large, we

think that LM Response based variants must disappear from practice, while the LMv2 will need to be replaced by a new design in the near future.

As an immediat measure, the *LMHash* must be disabled and the *lmcompatibilitylevel* value should be set to the highest level possible. Also, even it is not a consequence of this paper, but a general aspect, passwords of less than 8 characters or with low entropy must be avoided. As SharePoint starts to be widely used, user sensitive information may be exposed by the weaknesses of an authentication protocol such as NTLM. For the long term developing stronger authentication mechanisms, such as two factor authentication should be considered.

REFERENCES

[1] M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication", Advances in Cryptology – CRYPTO 96, LNCS vol. 1109, Springer-Verlag, 1996.

[2] E. Biham, A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", Technical report CS90-16, Weizmann Institute of Science, CRYPTO'90 & Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.

[3] Colin Boyd, Anish Mathuria, Protocols for Authentication and Key Establishment, Springer Verlag, 300 pages, ISBN-13: 978-3540431077, 2003.

[4] Jesse Burns, Information Security Partners, "NTLM Authentication Un-safe", http://www.isecpartners.com/files/NTLM_Unsafe_0.pdf, 2004.

[5] John A. Clark, Jeremy L. Jacob, "A survey of authentication protocol literature", available at http://www.cs.york.ac.uk/jac/papers/drareview.ps.gz, 1997.

[6] D. Coppersmith, "The Data Encryption Standard (DES) and its strength against attacks", IBM Journal of Research and Development, 1994.

[7] FIPS 46, Data Encryption Standard (DES), National Institute of Standards and Technology (NIST), U.S. Department of Commerce, 1976.

[8] FIPS 180-1, (1995), 180-2, Announcing the Secure Hash Standard., National Institute of Standards and Technology (NIST)., U.S. Department of Commerce, 2002.

[9] E. Glass, "The NTLM Authentication Protocol and Security Support Provider", http://davenport.sourceforge.net/ntlm.html.

[10] Jesper Johansson, "Security Watch: The Most Misunderstood Windows Security Setting of All Time", http://technet.microsoft.com/en-us/magazine/cc160954.aspx, 2006

[11] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", Advaces in cryptology, EUROCRYPT'03, LNCS 765, 1993.

[12] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.

[13] R. Rivest, "The MD5 Message-Digest Algorithm", MIT Laboratory for Computer Science and RSA Data Security, RFC 1321, 1992.

[14] D. Sanai, H. Seki, "Optimized Attack for NTLM2 Session Response", http://www.blackhat.com/presentations/bh-asia-04/bh-jp-04-pdfs/bh-jp-04-seki.pdf, 2004.

[15] B. Schneier and Mudge, "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)," Proceedings of the 5th ACM Conference on Communications and Computer Security, ACM Press, pp. 132-141.

[16] Schneier, B., Mudge and D. Wagner, "Cryptanalysis of Microsoft's PPTP authentication extensions (MS-CHAPv2)", in: R. Baumgart, editor, CQRE, Lecture Notes in Computer Science 1740 (1999), pp. 192-203.

[17] B. Schneier, "Cryptanalysis of MD5 and SHA: Time for a New Standard", http://schneier.com/essay-074.html, 2004.

[18] B. Schneier, "Real-World Passwords", 2006, http://www.schneier.com/blog/archives/2006/12/realworld_passw.html.

[19] X. Wang, Y.L. Yin, H. Yu, "Collision search on SHA1", http://theory.csail.mit.edu/ ~yiqun/shanote.pdf, 2005.

[20] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions", Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 19-35, 2005.

[21] G. Zorn, S. Cobb, "Microsoft PPP CHAP Extensions", RFC 2433, Microsoft Corporation, 1998.

[22] G. Zorn, "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, Microsoft Corporation, 2000.

[23] The AVISPA Library of Protocols, CHAPv2, http://avispa-project.org/library/CHAPv2.html.

[24] Microsoft, How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases, http://support.microsoft.com/kb/299656, 2007.