Security Watch: The Most Misunderstood Windows Security Setting of All Time

Security WatchThe Most Misunderstood Windows Security Setting of All Time

Jesper Johansson

Almost everyone who runs a network on Windows® has heard of NTLM version 2 (NTLMv2) and the LMCompatibilityLevel setting that governs it. The setting first became available in Windows NT® 4.0 Service Pack 4 (SP4), and has been in every version of Windows based on Windows NT since then. LMCompatibilityLevel has been recommended in every security guide for Windows since 1998.

A few years ago I thought I really knew how LMCompatibilityLevel worked. I was wrong. Then for a couple of years I thought I knew how it worked again. This has happened a few times, and recently I found out that I still didn't completely understand it. This prompted me to finally go back to the code, tear through it, and figure out what really was going on. This article is about what I found. Anyone who runs Windows probably should understand this setting and how it impacts network security and stability.

Background

Windows NT-based operating systems up through and including Windows Server™ 2003 store two password hashes, the LAN Manager (LM) hash and the Windows NT hash. Starting in Windows Vista™, the capability to store both is there, but one is turned off by default.

The NT hash is a straight MD4 hash of the plaintext password. It supports all Unicode characters and passwords up to 256 characters long.

The LM hash, originally invented for use in LAN Manager over 20 years ago, is not actually a hash at all (see **Figure 1**). A hash is a mathematical function used to summarize or probabilistically identify data. LM instead uses a cryptographic one-way function (OWF). Instead of encrypting the password with some other key, the password itself is the key. This is why you will sometimes see the LM hash referred to as the LM OWF. Conversely, the NT hash is typically referred to as the NT OWF internally.

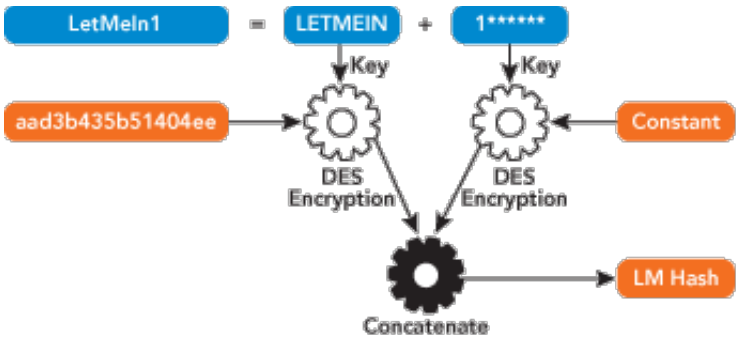

Figure 1 **LM Hash Generation**

The algorithm introduces several weaknesses that attackers can exploit. First, all lowercase characters are set to uppercase, reducing the number of possible characters. Second, it splits a long, strong, password into two seven-character chunks. Length has been shown to be the most important factor in password strength by far (see " Frequently Asked Questions About Passwords" for more on this topic). This operation reduces the strength of the password by many orders of magnitude.

Windows uses the LM OWF for the LM authentication protocol and the NT OWF for everything else. One common misconception is that the system stores a specific NTLMv2 OWF. There is not even an NTLM OWF for the NTLM protocol. The NT OWF is used for all other Windows authentication protocols in one way or another. The two OWFs are used in the authentication sequence as shown in **Figure 2**.

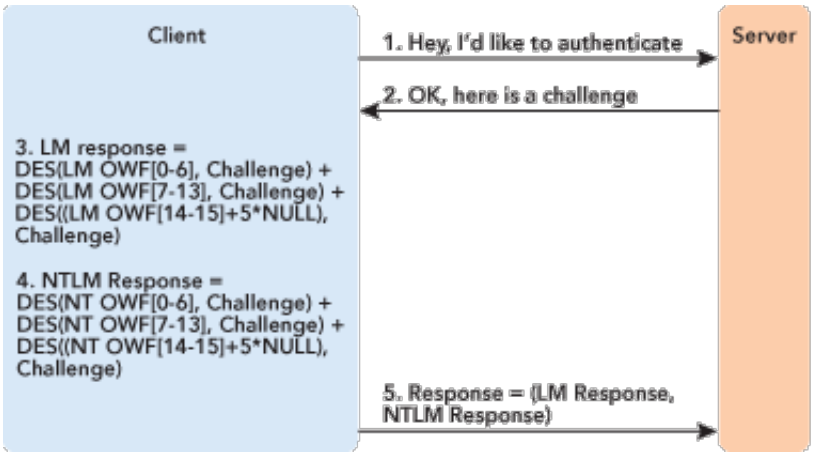

Figure 2 **LM and NTLM Response Computation**

Both the LM and NTLM protocols operate essentially the same way; the only difference is the password hash. The actual challenge-response computation is fairly simple. The client requests authentication and the authentication server responds with an 8-byte challenge. The challenge is just a piece of random data. The client encrypts the challenge using the OWF as the key and returns the result as the response to the server. The LM and NTLM responses are each always 24 bytes long. The server has the NT OWF and usually the LM OWF as well. It starts by computing the NTLM response using the same algorithm the client used, and then compares that to the

client result. If the two match, they used the correct password hash and—by an extension of logic that is not always correct—had the correct password, a successful logon results. If the results do not match, the server computes the LM response and checks if that matches what the client sent. If that fails too, the client is denied access due to a bad username or password error.

NTLMv2 was developed in response to attacks against the LM authentication protocol. The LM protocol, as the name implies, was originally used in the old LAN Manager network operating system in the mid-1980s. It was developed for the security requirements of the day, which included mostly floppy-based viruses, and hence was no match for a late 90s-style cracking attack. When the L0phtCrack password cracker from L0pht Heavy Industries, later purchased by @Stake and finally by Symantec, included a cracker against a captured LM authentication sequence, Microsoft implemented the new NTLMv2 authentication protocol to defeat L0phtCrack. (For a synopsis of the protocol and the attack that caused it to be created, see this *Windows IT Pro* article. Keep in mind, however, that this article was published in 1999 and is now fairly dated and not entirely accurate in some respects.)

NTLMv2 is turned on using the LMCompatibilityLevel switch (known as some variant on "LAN Manager authentication level" in Group Policy). LMCompatibilityLevel takes six different values, from 0 to 5. The levels are shown in Figures **3** and **4**.

Figure 4 Server-Side LMCompatibilityLevel Impact

Figure 3 Client-Side LMCompatibilityLevel Impact

Admittedly, this is a brief introduction to Windows and passwords. Interested readers should see chapter 11 in *Protect Your Windows Network*, which covers the topic in much more detail.

How NTLMv2 Works

The NTLMv2 response is very different. First, the client concatenates the user name and the logon domain name and computes an HMAC-MD5 message authentication code of those using the NT OWF as a key. The result of this keyed hash is sometimes referred to as the NTLMv2 OWF. The logon domain name is the account the client is attempting to log on to. In the case where a workstation is trying to contact a system that is not in the client's domain, this would be the server name.

The client then generates its own 8-byte challenge, which it puts into a blob that also contains a timestamp, information on the target the client is attempting to connect to, and some other data. This blob is then concatenated with the server challenge and

another HMAC-MD5 message authentication code is computed on this combined challenge using the NTLMv2 OWF as the key. The result of that computation is concatenated with the blob and returned as the NTLMv2 response buffer. The client also computes a response based only on the NTLMv2 OWF, the server challenge, and the client challenge. This response is called the LM 2 Response and is returned in the LM response field along with the client challenge, as shown in **Figure 5**.

The entire NTLMv2 response buffer is not documented publicly, but many sources have inferred a lot about it. Essentially, it contains version information, room for flags in a future implementation, room for a message from the client to the server (which is not used today), the timestamp, the client challenge, and some information about the server and share name the client is connecting to.

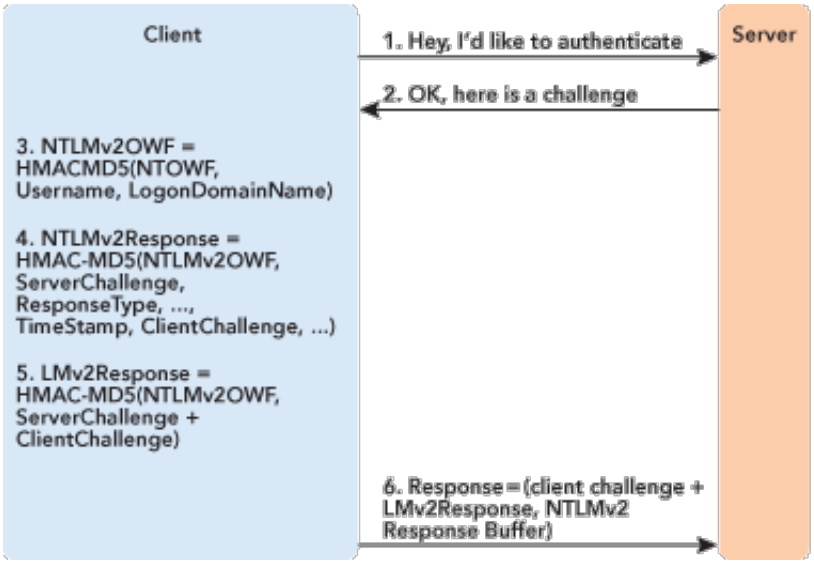

Figure 5 **LMv2 and NTLMv2 Response Computation**

As noted earlier, when NTLMv2 is used, the LM response changes as well. This is done for compatibility with servers that are domain members but do not understand NTLMv2. The problem is that the NTLMv2 response is variable in length, including items such as the server name, the share name, and so on. Older domain-joined servers, such as Windows NT 4.0 prior to SP4 and Windows 9*x* cannot pass through a variable-length response to the domain controller; they assume that both the LM and NTLM responses are exactly 24 bytes. By luck or very smart design, an HMAC-MD5 computation yields a 128-bit value which, when combined with the 8-byte client challenge, gives us 192 bits or 24 bytes—exactly the same as the three Data Encryption Standard (DES) computations used in the LM response. The older machines can thus pass the LMv2 response through to the domain controller without needing to understand the exact meaning of the bits within it. The domain controller will do the same computation it always does, checking first if the response is an NTLMv2 response, and if that fails falling back to LMv2, and then to the older protocols.

Until recently, I thought this was all there was to LMCompatibilityLevel. I conveniently pushed aside the feeling that I still did not understand the thing referred to as NTLMv2 Session Security.

Years of Confusion

As did Randy Franklin Smith in the article mentioned earlier, I believed for years that at level 1, the system was supposed to send the LM response (which is correct) and would negotiate and send the NTLMv2 response if it were possible. This is incorrect. When you set LMCompatibilityLevel to 1, it does not actually send NTLMv2 authentication.

Then another epiphany hit me. The documentation, which says that "DCs will accept" and "clients will send" is really misleading. The simpler way to think about the settings is that levels 0-3 affect the computer when acting as a client, while levels 4 and 5 affect the machine when acting as the authentication server. It is not just domain controllers that authenticate users. All systems can authenticate users against their own SAM database, so the settings really affect all of them.

Once again, I thought I knew exactly how it worked, and I wrote another article on it. Shortly after that, Seki Hidenobu wrote to me, prodding about something he called "NTLM2 Session Response," a term he picked up from Eric Glass. In fact, not only does the Ethereal network sniffer filters identify NTLM2 Session Response, it has even been documented as part of the Samba (the UNIX implementation of SMB) documentation. In addition, Christopher R. Hertel's book on the Common Internet File System (CIFS—the standardized version of the Microsoft Server Message Block, or SMB, protocol) contains a very technical and detailed description of all the protocols involved and discusses the NTLM2 Session Response. More information on this book is at www.ubiqx.org/cifs. These documents were largely developed while adding NTLMv2 support to Samba. Luke K. C. Leighton, who wrote most of the NTLMv2 code for Samba, also wrote a book on the protocols, *DCE/RPC over SMB*: *Samba and Windows NT Domain Internals* (SAMS, 1999). That book even includes source code for an implementation of NTLMv2.

NTLMv2 Session Security

Hidenobu did not know exactly what the NTLM2 Session Response was for and why it appeared, but as he pointed out, it can be cracked. Basically, when it is used, you do not pass the LM response. Instead, a client challenge is created and mixed with the server challenge using HMAC-MD5. Then the NTLM response to this combined challenge is computed the normal way, as shown in **Figure 2**, by using three DES encryptions of the challenge with pieces of the nonce, and returned to the server in the

NTLM response field. The client challenge is returned in the now unused LM response field. This modified NTLM response can be cracked using a modified precomputed hash attack even though there is now a client challenge involved. Seki exploited the fact that only two bytes of the challenge are used in the third component of the response. Once he identifies which bytes those are, he can use them as an index into a lookup table to determine all the passwords that could have possibly generated an OWF with those two bytes at the end.

Eric Glass, who did not have the luxury or the curse (depending on how you look at it) of Windows source code access, stated that the "NTLM2 Session Response can be used in conjunction with NTLM2[sic] Session Security...." This is, in fact, almost exactly the correct description. To produce an answer for Hidenobu I ended up having to go back to the source code. It turns out that what the Samba documents call the "NTLM2 Session Response" is actually a slightly modified version of the NTLM response, shown in **Figure 6**.

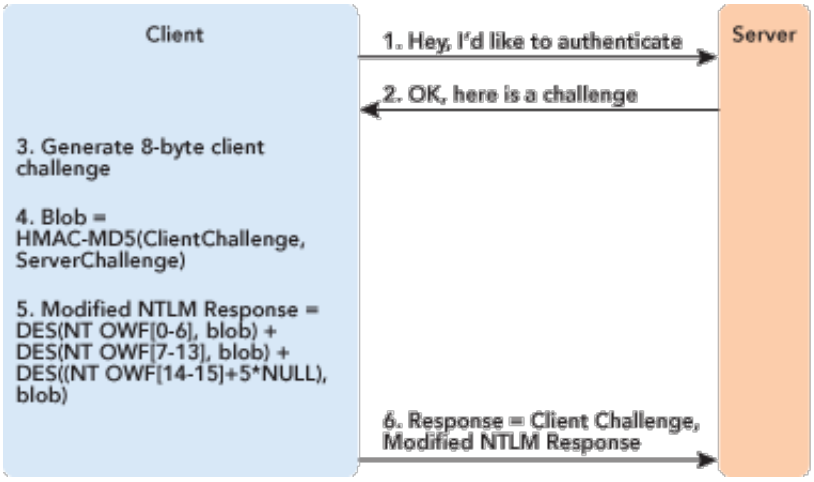

Figure 6 **Response Computation for LMCompatibilityLevel 1 or 2**

This modified response is used when the client knows it is talking to an uplevel server that can handle this new response. This means that, as Glass pointed out, it is seen when you use NTLMv2 Session Security. It was created to make certain man-in-the-middle attacks more difficult by introducing a client-side challenge, while still retaining the ability for downlevel clients to pass the authentication through to an upgraded domain controller. This new response does not really even have a name, and has never been documented by Microsoft. In the source code the term "NTLM2" sometimes shows up, contrasted with "NTLM3," which is really NTLM version 2. It is possible that the use of the term NTLM2 Session Response originated from someone who was looking at debug symbols and saw that term.

NTLMv2 Session Security has been another enigma. In fact, "NTLMv2 Session Security" is really a misnomer. It really should be referred to as "NTLMv2-style session key computation," although one can easily understand why that term never caught on.

NTLMv2 Session Security is related to how session keys are computed. When using the original NTLM protocol, the session key is based on the user's NT hash (the NT OWF). When NTLMv2 Session Security is used, the session key is based on not only the NT OWF but also on the client and server challenges. Those session keys are not used during the actual authentication sequence, but when an application requests security by calling the EncryptMessage or SignMessage APIs.

In other words, you can think of NTLMv2 Session Security as having two components: a stronger response computation and a better session key generation algorithm. The response computation, however, can be either the mysterious NTLM2 Session Response or a real NTLMv2 response.

NTLMv2 Session Security can be forced by setting the LMCompatibilityLevel setting, but it is also negotiated between the client and the server. Starting with Windows 2000 Security Rollup Pack 1 (SRP1) all systems will attempt to use NTLMv2 Session Security if both sides agree. If either side disagrees they will fall back to the old protocols if they are configured to allow that.

LMCompatibilityLevel and Security

As mentioned earlier, starting with Windows 2000 SRP1, including Windows 2000 SP3, all versions of Windows XP and all versions of Windows Server 2003, NTLMv2 Session Security is always negotiated. Therefore, there is no longer any functional difference between LMCompatibilityLevel 0 and 1. They generate exactly the same results on all supported platforms.

The difference between LMCompatibilityLevel set to 0 or 1 and setting it to 2 is that with level 2 the client refuses to send the LM response. This is why level 2 is the default setting on Windows Server 2003. It also means that Windows Server 2003 cannot be a client to a Windows 9*x* system unless it passes the authentication through to a domain controller. When acting as the authentication server, however, Windows Server 2003 accepts LM responses and authenticates against the LM hashes stored locally or in Active Directory®.

This meant I had to modify my tables showing how the various levels of LMCompatibilityLevel work. **Figure 3** shows the settings that affect how the system acts as a client. **Figure 4** shows how the settings affect the system when acting as an authentication server.

Levels 0 and 1 are identical on systems with Windows 2000 SRP1 or higher. The Security Rollup Pack is included in Windows 2000 SP3 and, of course, the same functionality with respect to NTLMv2 Session Security is turned on by default in Windows XP and Windows Server 2003.

In terms of actual security impact, negotiating NTLMv2 Session Security ranges from significant to minimal, depending on the environment. Theoretically, NTLMv2 Session Security can be used with any protocol. But used with NTLM version 1, it also improves the challenge response computation by including the client-side challenge, similar to how NTLMv2 itself works. This makes a precomputed hash attack against captured challenge-response pairs more difficult, but certainly not impossible. More significantly, it makes man-in-the-middle attacks much more difficult.

NTLMv2 Session Security is still negotiated. Therefore, if the attacker has the ability to act as the server itself or to modify the transaction in any way, the attacker can downgrade the authentication protocols to the older versions, enabling both types of attack once again. Only if LMCompatibilityLevel is set to 3 on the client are these attacks actually stopped.

Recommendations

Although Windows Vista has not been released yet, it is worthwhile to point out some changes in this operating system related to these protocols. The most important change is that the LM protocol can no longer be used for inbound authentication—where Windows Vista is acting as the authentication server. Windows Vista will no longer store the LM hash by default. Acting as a client, Windows Vista also makes a change to outbound protocols by setting LMCompatibilityLevel to 3 by default. In other words, NTLMv2 will finally be the default protocol for non-domain authentication. In the next scheduled release of the Windows Server platform, code-named "Longhorn Server," a lot of work has been done to reduce the need for NTLM altogether. In Windows Server 2003, NTLM, and sometimes even LM, is used in many cases, such as in clusters. In the next version of the operating systems many of these protocols will finally be turned off by default.

Please note that these statements are based on prerelease versions of the operating systems. It is possible that compatibility testing during the beta cycle will force these settings to change.

So what should you do now? In all environments you should consider setting the LMCompatibilityLevel as high as possible. The decision needs to take into account the types of systems used in the environment and how high the threat of attack is. For instance, inside a datacenter where all communication lines are secured, the threat of man-in-the-middle attacks is relatively small. In an unsecured or WEP-secured wireless network the risk is significant. Attackers commonly sniff these networks in the hopes of catching the authentication sequence and cracking it.

Generally, the setting should be as high as the needs of the environment allows. In a

network that consists of only Windows 2000 SP3 and higher systems, and has no third-party devices that require LM or NTLM authentication, LMCompatibilityLevel can be safely set to 3 or higher on all systems. Problems may occur when level 5 is used on systems running operating systems prior to Windows Server 2003 SP1. Routing and Remote Access Services will fail if the RRAS server or the domain controller is running with LMCompatibilityLevel 5.

It is also important to consider the error messages you get when you have a conflict in the LMCompatibilityLevel capabilities of different systems. Effectively, the only negotiation in this entire set of protocols is NTLMv2 Session Security. Downlevel systems will ignore that flag. Therefore, the only error message you get if there is a settings conflict between client and server is that access was denied due to a bad username or password. Neither side to the transaction has the ability to determine that the issue was due to unsupported authentication protocols. This makes troubleshooting somewhat challenging.

Jesper Johansson, a senior security strategist in the Microsoft Security Technology Unit and contributing editor for TechNet Magazine, focuses on how customers should best deploy Microsoft products more securely. He has a PhD in IS and has delivered speeches on security at conferences all over the world.