Samba and Windows NT Security Interoperability

Luke Kenneth Casson Leighton
SAMBA Team
lkcl@samba.org

Abstract

Samba provides Windows NT File, Print and Login compatibility for Unix. Starting life in 1991 as Server 0.1 with 2,000 lines of code, it now consists of just over 300,000, providing full Windows NT Domain integration - a project that began in August 1997 and is still in progress.

With the introduction of Windows 2000 (and even with Service Pack 4 for Windows NT 4.0), developers of Samba have had to address several issues related to the interoperability with Windows NT/2000 clients. This paper presents the technical security issues confronted by Samba developers during further implementation of the Windows NT 4.0 domain control protocol.

1. Introduction

Samba provides Windows NT File, Print and Login compatibility for Unix. Starting life in 1991 as Server 0.1 with 2,000 lines of code, it now consists of just over 300,000, providing full Windows NT Domain integration - a project that began in August 1997 and is still in progress.

During the development of Samba's NT Domain interoperability, there have been key sticking points that have had to be worked out, such as:

- The NTLMv2 protocol.
- The NETLOGON "Secure Channel" on DCE/RPC.
- NTLMSSP and its application on DCE/RPC.
- User-password changing.
- Administrative user-password changing.
- PDC / BDC SAM Database Synchronization.

The NTLMv2 and the NETLOGON "Secure Channel" were introduced in Windows NT Service Pack 4. Without all these key components, any server that claims to be Windows NT Domain-compatible is completely useless. Fortunately, they are now documented (ISBN 157870-150-3) to the extent that they are understood.

Worse than this, if some of these mechanisms are not implemented, a server is interoperable with Windows NT, but is insecure, as the (outdated) mechanisms are cryptographically extremely weak. It would be nice to have some official documentation on these protocols, however such an official release could potentially be used as the basis of a lawsuit claiming exposure to security threats, so unofficial will have to do (for which no direct responsibility can be claimed).

This paper outlines, for both Windows NT and Samba:

- The current level of interoperability with these security mechanisms.
- How these mechanisms are activated at the higher security levels and how they can be made mandatory (refuse low security connections).
- Where the use of these mechanisms is still insecure, and the workarounds, or The Things to Avoid.

2. NTLMv2

NTLMv2 is a much-improved CHAP system (Challenge Protocol). A server sends a challenge to the client, and the client must, to a degree of probability determined by the entropy of the challenge, prove to the server, using the challenge and the user's password, that the client does in fact know the user's password.

The NTLMv2 mechanism uses the NT password hash, not the LM hash, as the user's password in the algorithm. The response should also contain other information such as the local time, and the server should verify this time, to within +/- 30 minutes, according to the NTLMv2 specification. As it turns out, the information in the client's response is not validated on NT4 SP4, although the HMAC_MD5 signature-part of the client's response is validated.

Here follows a quote from MS KB Article Q147/7/06 which describes how to enable NTLMv2 (and optionally disable NTLMv1, or support both):

Control of NTLM security is through the following registry key:

```
HKLM
   System
      CurrentControlSet
         Control
            LSA
```

Choice of the authentication protocol variants used and accepted is through the following value of that key:

LMCompatibilityLevel
Value Type: REG_DWORD - Number
Valid Range: 0-5
Default: 0
Description: This parameter specifies the type of authentication to be used.

- *Level 0* - Send LM response and NTLM response; never use NTLMv2 session security.
- *Level 1* - Use NTLMv2 session security if negotiated
- *Level 2* - Send NTLM authentication only
- *Level 3* - Send NTLMv2 authentication only
- *Level 4* - DC refuses LM authentication
- *Level 5* - DC refuses LM and NTLM authentication (accepts only NTLMv2)

NOTE: Authentication is used to establish a session (username/password). Session security is used once a session is established using the appropriate type of authentication. Also system times should be within 30 mins of one another. Authentication can fail because the server will think the challenge from the client has expired.

In Samba, the equivalent options are:

```
client NTLMv2 = yes, no or auto
```

which enables whether either samba itself or a client-side program such as smbclient or samedit uses NTLM or NTLMv2 for outgoing connections.

```
server NTLMv2 = yes, no or auto
```

which enables whether the samba server accepts or refuses either NTLM, or NTLMv2, or both, on incoming connections.

3. NETLOGON "Secure Channel"

The NETLOGON "Secure Channel" provides a means to negotiate the level of security to be used for the communication of User Authentication requests and SAM Database Replication. Unfortunately, the default level of security in NT Service Pack 3 and below, all ports of NT to Unix (AS/U by AT & T; AFPS by SCO; Cascade by Sun Microsystems) and all production releases of Samba (which has unofficial PDC support) is, as is already known, pretty much a token effort.

With the introduction of Service Pack 4, it has been possible to negotiate a more secure NETLOGON channel, the name of which is not publicly known, but is negotiated by setting bit 30 of the neg_flags parameter in a NetrAuthenticate2 DCE/RPC function call on the NETLOGON pipe. NT 5 negotiates an entire army of encryption mechanisms, which are presumably intended to mirror either the Crypto API or the same kinds of authentication mechanisms that can be negotiated for use with Kerberos 5.

In order to enable the NETLOGON Secure Channel, here follows a quote from MS KB Article Q183/8/59:

A fix to Windows NT 4.0 Netlogon service has been designed that will allow or integrity checking. After applying this fix, the following Registry values can be used to modify the behavior of the secure channel between the client and domain controller. All of the following values can be found in the registry under the following key:

```
HKLM
   CurrentControlSet
      Services
         Netlogon
            Parameters
```

SignSecureChannel

Value Type: REG_DWORD - Boolean
Valid Range: 0 (FALSE) or 1 (TRUE)
Default: TRUE
Description: This parameter specifies that all outgoing secure channel traffic should be signed. If SealSecureChannel is also TRUE, it will override any setting for this parameter and force it to TRUE.

SealSecureChannel

Value Type: REG_DWORD - Boolean
Valid Range: 0 (FALSE) or 1 (TRUE)
Default: TRUE
Description: This parameter specifies that all outgoing secure channel traffic should be encrypted.

RequireSignOrSeal

Value Type: REG_DWORD - Boolean
Valid Range: 0 (FALSE) or 1 (TRUE)
Default: FALSE
Description: This parameter specifies that all outgoing secure channel traffic must be either signed or sealed. Without this parameter, this is negotiated with the Domain Controller. This flag should only be set if ALL of the domain controllers in ALL the trusted domains support signing and sealing. If this parameter is TRUE, SignSecureChannel is implied to be TRUE.

In Samba, the equivalent options are:

```
client schannel = yes, no or auto
```

which enables whether either samba itself (as a Domain Member, BDC or for verifying Inter-Domain Trust Relationships) or a client-side program such as samedit uses the encrypted form of the NETLOGON Secure Channel or not, for outgoing connections.

```
server schannel = yes, no or auto
```

which enables whether the samba server accepts or refuses either unencrypted or encrypted NETLOGON sessions, or both, on incoming connections.

Failure to use the NETLOGON encrypted channel implies that any user logons or SAM Database replications will be insecure. And unfortunately, that means:

Windows NT 4.0 Service Pack 4 and below Advanced Server for Unix (AS/U) and all other ports of Windows NT to Unix, including the new Sun Microsystems' Cascade product. The unofficial PDC support in all Samba production releases.

Samba does have a 98% complete implementation of the encrypted channel - sufficient to make one NetrSamSync or NetrSamLogon, after which the connection must be dropped because there is a sequence of eight bytes that is attached to the Sealing-part of the channel, and only the first exchange (request / response) is currently known! This is sufficient for the purposes of making a single SAM Database replication request (using the samsync sub-command in samedit), but not sufficient, server-side, to service multiple incoming requests, as a PDC.

4.NTLMSSP

The NTLM Secure Service Provider is documented in the MSDN at a high-level, client-side. None of the internal workings is publicly available. NTLMSSP is a means by which the NTLM CHAP Protocol (and now the NTLMv2 CHAP Protocol) can be leveraged to provide signed and sealed communication. Typically this is used to securely exchange "authentication tokens" (which is the only [unconfirmed] acceptable excuse to use strong crypto). ISBN 157870-150-3 provides an annotated example of how NTLMSSP negotiation is performed and used. The example only covers NTLMv1 at 40-bit, as this is the only mechanism that is currently fully understood.

In order to mandate certain levels of security, should a client actually request it, there follows a quote from MS KB Article Q147/7/06:

Control over the minimum security negotiated for applications using NTLMSSP is through the following key:

```
HKLM
   System
      CurrentControlSet
         Control
            LSA
               MSV1_0
```

The following values are for this key:

NtlmMinClientSec

Value Type: REG_DWORD – Number
Default: 0
Valid Range: the logical 'or' of any of the following

values:

0x00000010
0x00000020
0x00080000
0x20000000

NtlmMinServerSec

Value Type: REG_DWORD – Number
Valid Range: same as *NtlmMinClientSec*
Default: 0
Description: This parameter specifies the minimum security to be used.

0x00000010	Message integrity
0x00000020	Message confidentiality
0x00080000	NTLMv2 session security
0x20000000	128 bit encryption

NtlmMinClientSec and *NtlmMinServerSec*

- If the bit with value 0x00000010 is set in the *NtlmMinClientSec* or *NtlmMinServerSec* value, the connection will fail if message integrity is not negotiated.
- If the bit with value 0x00000020 is set in the *NtlmMinClientSec* or *NtlmMinServerSec* value, the connection will fail if message confidentiality is not negotiated.
- If the bit with value 0x00080000 is set in the *NtlmMinClientSec* or *NtlmMinServerSec* value, the connection will fail if NTLMv2 session security is not negotiated.
- If the bit with value 0x20000000 is set in the *NtlmMinClientSec* or *NtlmMinServerSec* value, the connection will fail if 128-bit encryption is not negotiated.

NOTE: These settings will not guarantee that the NTLM SSP is actually used by every application, or that message integrity or confidentiality will actually be used by an application even when they are negotiated.

What this last comment means, for example, is that when, say, you change your password, you expect the password change to be encrypted using 128-bit-based NTLMv2 encryption. However, because of a bug, the wrong Domain and password is used (you changed the password from someone else's workstation whilst that other person was logged in. Your domain and username is used with their password, in the NTLMSSP encryption negotiation, which of course fails unless you both have the same password!). The workstation then falls back to not using encrypted NTLMSSP, which is accepted by the server, bypassing any attempts to use higher security settings. No warning is issued at either the client or the server end that this is occurring.

At present, there are no equivalent options in Samba, as Samba only supports 40-bit NTLMSSP using NTLM only - not NTLMv2. This is because NTLMSSP with NTLM is quite complicated to work out without a specification, and NTLMv2 even more so even with a partial specification: there are some key constants missing, without which the (otherwise useful) specification is only of academic interest. This is a pity, because the use of NTLMv2 in NTLMSSP looks like it has been well thought out, cryptographically.

5. User-password changes

User-password changing uses the old user's password hash to encrypt the new Unicode plaintext password, in an undocumented function call, SamrChangeUserPassword. The old password is used as an RC4 cypher key to encrypt the new password.

This implies that if the encrypted DCE/RPC negotiation fails, and the client falls back to using unencrypted DCE/RPC to transfer the SamrChangeUserPassword's function arguments (one of which is the encrypted password block mentioned above), then if two users ever have the same password, or the same user ever reuses a password, an attacker can XOR these two blocks together to obtain information about the two new plaintext passwords.

Unfortunately, there is no way to mandate, on NT, that unencrypted DCE/RPC sessions be refused, so the only guaranteed secure way to change user passwords is to use the ntpass sub-command of the samedit tool. Unfortunately, Samba's encrypted DCE/RPC channel negotiation only supports 40-bit NTLMv1, which is better than nothing.

6. Administrative user-password changes

Administrative user-password changes, which are also carried out when an account is created, use an undocumented function call, SamrSetUserInfo. The password is in Unicode plaintext, and is encrypted with the Administrator's "User Session Key". When

NTLMv1 is used, a User Session Key is calculated from the password - and does not change.

For secure, remote Account Management using USRMGR.EXE, is therefore vital that NTLMv2 is used, and even then, used only once and the connection to the DC torn down. This is of course completely impractical in a large-scale environment, so other measures are required, such as ensuring that the remote-access consoles from which USRMGR.EXE are on secure, private networks to the Domain Controller, and that those consoles are physically secure as well.

Alternatively, the sub-command samuserset of the samedit tool could be used [note: this is not currently implemented - however the code is publicly available :)]. Although this would only be at the current level of security - 40-bit - it is better than giving away plain-text passwords and a cryptographic hint on the Administrator's password as well!

7. SAM Database Replication

Samba has manually-controllable SAM Database replication. Given that Unix has cron (a mechanism to run commands at certain time intervals) this is not such a great loss. Samba must first be configured as a BDC:

```
[global]
   workgroup = NTDOMAIN
   password server = pdcname
   ; because the pdc is the
   ; domain master!
   domain master = no
   domain logons = yes
   security = user
   encrypt passwords = yes
   server schannel = yes
   client schannel = yes

[netlogon]
   ; stores login scripts
   read only = yes
   path = /usr/local/samba/netlogon
```

Once this is done, and Samba is joined to the domain, the samedit tools' sub-command, samsync, can be used to contact the PDC, obtain the user accounts and update the BDC's local copy of the SAM database. Here follows a transcript of a Samba Server being joined to an NT Domain, with samedit. It is important that the NETLOGON Secure Channel be encrypted for this, otherwise the SAM database pretty much goes over-the-wire in cleartext (note the prevalent usage of the password "test" in the example below, a particularly secure password, this is):

```
# bin/samedit -S changeme-nt4s -U root%test \
> -l log
Server: \\CHANGEME-NT4S: User: root Domain:
Connection: OK

[root@CHANGEME-NT4S]$ use \\knight -Uroot%test
-W knight

use \\knight -Uroot%test -Wknight

Server: \\KNIGHT: User: root Domain: knight
Connection: OK

[knight\root@CHANGEME-NT4S]$ createuser
knight$ -s -j domain

createuser knight$ -s -j domain

SAM Create Domain User
Domain: DOMAIN Name: knight$ ACB: [S]
Create Domain User: OK
Join KNIGHT to Domain DOMAIN
Set $MACHINE.ACC: OK

[knight\root@CHANGEME-NT4S]$ lsaquery

lsaquery

LSA Query Info Policy
Domain Member     - Domain: DOMAIN (S-1-5-21-
4070507235-114175824-2771791698)
Domain Controller - Domain: DOMAIN (S-1-5-21-
4070507235-114175824-2771791698)

[knight\root@CHANGEME-NT4S]$ samsync

samsync

   SAM Database Sync
   -----------------
Domain: DOMAIN
Group: Domain Admins
Group: Domain Users
Group: Domain Guests
Group: testgroup
Group: testgroup2
Group: testgroup3
Group: testgroup5
Group: testgroup6
Group: testgroup7

Account: Administrator
{
   0x01,0xFC,0x5A,0x6B,0xE7,0xBC,0x69,0x29,
   0xAA,0xD3,0xB4,0x35,0xB5,0x14,0x04,0xEE
};
{
   0x0C,0xB6,0x94,0x88,0x05,0xF7,0x97,0xBF,
   0x2A,0x82,0x80,0x79,0x73,0xB8,0x95,0x37
};
Account: Guest
{
   0xB3,0xCC,0x5A,0x77,0xA6,0x8F,0x64,0x77,
   0x61,0x2A,0x53,0xE1,0x2D,0xFC,0x18,0x3B
};
{
   0xB3,0xCC,0x5A,0x77,0xA6,0x8F,0x64,0x77,
   0x61,0x2A,0x53,0xE1,0x2D,0xFC,0x18,0x3B
};
Account: CHANGEME-NT4S$
{
   0x17,0x47,0xDB,0xE6,0x1B,0xA8,0x60,0x32,
```

```
   0x1D,0x1A,0xEE,0x2B,0x53,0xF6,0x29,0xEA
};
{
   0x5E,0x6A,0xBA,0x10,0xF7,0xA2,0x3F,0xDC,
   0xEF,0x50,0xBA,0x30,0x62,0x75,0xBF,0x53
};
Account: NT4-1$
{
   0x8F,0xCA,0x67,0xCF,0x5A,0x9F,0xEB,0x7D,
   0xB0,0x6F,0xDA,0xCB,0xE2,0xEF,0xDE,0xAB
};
{
   0x6D,0x60,0xD6,0x79,0x43,0xE7,0x2C,0xE3,
   0x46,0xC3,0x4C,0xD1,0xD4,0xC9,0xD6,0x2C
};
Account: root
{
   0x01,0xFC,0x5A,0x6B,0xE7,0xBC,0x69,0x29,
   0xAA,0xD3,0xB4,0x35,0xB5,0x14,0x04,0xEE
};
{
   0x0C,0xB6,0x94,0x88,0x05,0xF7,0x97,0xBF,
   0x2A,0x82,0x80,0x79,0x73,0xB8,0x95,0x37
};
Account: knight$
{
   0xBF,0xFB,0x57,0x74,0x20,0x86,0xF0,0x83,
   0x1A,0xD1,0x2E,0xDD,0xA1,0x3A,0x11,0xFC
};
{
   0x92,0x3A,0x73,0x26,0xCA,0xFC,0x62,0xAD,
   0x7E,0x25,0x04,0x32,0x56,0x2D,0x2A,0x41
};
```

The first command is createuser, with the -s option denoting that knight is a Server (a BDC). The second command, lsaquery, is a clunky, (now old-fashioned) way to manually obtain the Domain name of the PDC, and the third command, samsync, actually obtains the SAM database. Examining the smbpasswd file on knight would show that these accounts have been added or updated.

8. Conclusion

With the right knowledge, NT Domain interoperability is feasible. Without this knowledge, not only can there be no possibility of interoperability, but also it cannot be guaranteed that the, up until now proprietary, methods are secure. As it turns out, in this case, Windows NT Domain systems are not secure even in the default configuration, as they still allow, for backwards-compatibility reasons, the lowest security settings. Hopefully, now that it is known that these systems are so insecure it will encourage the use of a more responsible and already-proven (time and time again) attitude to security. Namely that the only safe way to develop security measures is to publish their workings and let experts analyze it - before they are actually used in production environments.

9. References

Subject: NT admin password change algorithms expose user plaintext passwords
Date: Mon, 5 Jun 2000 06:33:36 +1000
From: Luke Kenneth Casson Leighton <lkcl@samba.org>
To: NT Bug Traq List <NTBUGTRAQ@LISTSERV.NTBUGTRAQ.COM>, bugtraq@securityfocus.com

Subject: Why You Should Upgrade To NT4 SP4 or NT5
Date: Mon, 5 Jun 2000 05:37:57 +1000
From: Luke Kenneth Casson Leighton <lkcl@samba.org>
To: NT Bug Traq List <NTBUGTRAQ@LISTSERV.NTBUGTRAQ.COM>, bugtraq@securityfocus.com

L. Leighton, DCE/RPC over SMB: Samba and Windows NT Domain Internals, Publisher: Macmillan Technical Publishing, ISBN 15787015