Security Advisory

Remote Desktop Protocol, the Good the Bad and the Ugly

Author: Massimiliano Montoro <mao@oxid.it>

Issue date: May, 28, 2005

Summary

The aim of this advisory is to warn Remote Desktop users about the feasibility of invisible man-in-the-middle attacks against Microsoft Terminal Services. This is an update of Erik Forsberg's advisory released in April 2003 available at the following link: http://www.securityfocus.com/archive/1/317244. In short, mitm attacks on RDP protocol are still possible and they can be completely invisible for Terminal Services users.

Systems Affected

This advisory is born after experiments and researches on the following environment:

- Terminal Server software: Microsoft Windows Terminal Services using RDP v5.2
- Terminal Server Client software: Microsoft Remote Desktop for Windows XP v5.1.2600.2180

Details

The Good

Microsoft's Windows Terminal Services (built into Windows 2000 Server and Windows Server 2003) and Windows XP's Remote Desktop, provide an easy, convenient way for administrators to implement thin computing within an organization or for users to connect to their XP desktops from a remote computer and run applications or access files. A Windows 2000 terminal server can be installed in one of two modes: administrative or application server. In administrative mode, only users with administrative accounts can access the terminal server ... this is why these sessions are so interesting.

By default, the data that travels between the terminal server and the terminal services client is protected by encryption. The RDP protocol uses the RC4 symmetric encryption algorithm which provides three levels of security:

- *High*: encrypts both the data sent from client to server and the data sent from server to client using a 128-bit key.
- *Medium*: encrypts both the data sent from client to server and the data sent from server to client using a 56-bit key if the client is a Windows 2000 or above client, or a 40-bit key if the client is an earlier version.
- *Low*: encrypts only the data sent from client to server, using either a 56-bit or 40-bit key, depending on the client version.

In administration mode, RC4 encryption keys are generated after an initial key exchange in which RSA asymmetric encryption is used.

The Bad

In April 2003 Erik Forsberg released a security advisory to the internet community http://www.securityfocus.com/archive/1/317244 explaining how mitm attacks can be performed:

"... *During extensive investigation of the Remote Desktop Protocol (RDP), the protocol used to connect to Windows Terminal Services, we have found that although the information sent over the network is encrypted, there is no verification of the identity of the server when setting up the encryption keys for the session. This means RDP is vulnerable to Man In The Middle attacks (from here on referred to as MITM attacks). The attack works as follows:*

1) The client connects to the server, however by some method (DNS spoofing, arp poisioning, etc.) we've fooled it to connect to the MITM instead. The MITM sends the request further to the server.
2) The server sends it's public key and a random salt, in cleartext, again through the MITM. The MITM sends the packet further to the client, but exchanges the public key to another one for which it knows the private part.
3) The client sends a random salt, encrypted with the server public key, to the MITM.
4) The MITM deencrypts the clients random salt with it's private key, encrypts it with the real servers public key and sends it to the server.
5) The MITM now know both the server and the client salt, which is enough information to construct the session keys used for further packets sent between the client and the server. All information sent between the parts can now be read in cleartext.

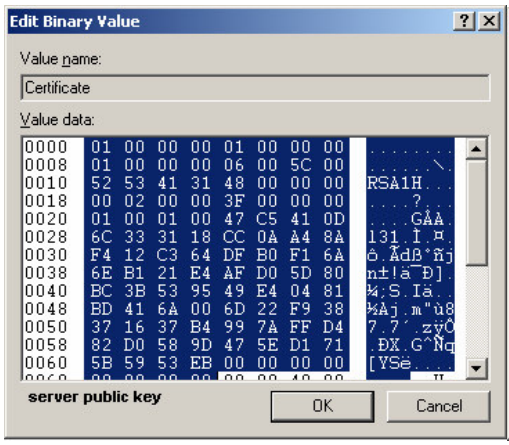
<u>The vulnerability occurs because the clients by no means try to verify the public key of the server, sent in step 2 above. In other protocols, such as the Secure Shell protocol, most client implementations solve this for example by letting the user answer a question whether a specific serverkey fingerprint is valid. ..."</u>

The Ugly
Microsoft confirmed the above problem and fixed the new versions of Remote Desktop Clients. Recent clients (mstsc.exe), including the one of version XPSP2 5.1.2600.2180, now check the Terminal Server identity verifying its public key. They solved the problem ? No, **man-in-the-middle attacks are still possible and can be really invisible for users**.

During the initial key-exchange phase, the terminal server sends to the client a server certificate created at the start up of Terminal Server services. This certificate is stored in the registry of the server under the following key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermService\Parameters\Certificate

It contains an RSA public key and its digital signature as illustrated below:

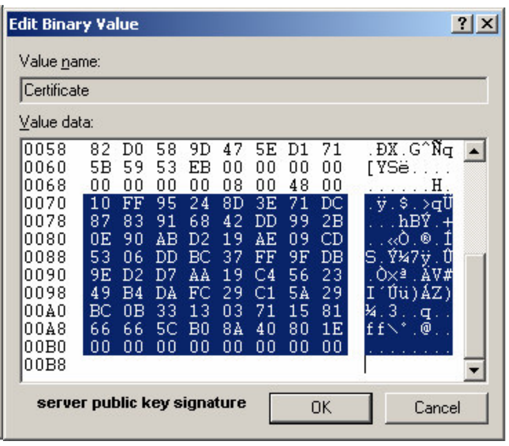

Edit Binary Value

Value name: Certificate

Value data:

```
0000  01 00 00 00 01 00 00 00  ........
0008  01 00 00 00 06 00 5C 00  ......\.
0010  52 53 41 31 48 00 00 00  RSA1H...
0018  00 02 00 00 3F 00 00 00  ....?...
0020  01 00 01 00 47 C5 41 0D  ....GÅA.
0028  6C 33 31 18 CC 0A A4 8A  l31.Ì.¤.
0030  F4 12 C3 64 DF B0 F1 6A  ô.ÃdB°ñj
0038  6E B1 21 E4 AF D0 5D 80  n±!ä¯Ð].
0040  BC 3B 53 95 49 E4 04 81  ¼;S.Iä..
0048  BD 41 6A 00 6D 22 F9 38  ½Aj.m"ù8
0050  37 16 37 B4 99 7A FF D4  7.7´.zÿÔ
0058  82 D0 58 9D 47 5E D1 71  .ÐX.G^Ñq
0060  5B 59 53 EB 00 00 00 00  [YSë....
```

server public key


Edit Binary Value

Value name: Certificate

Value data:

```
0058  82 D0 58 9D 47 5E D1 71  .ÐX.G^Ñq
0060  5B 59 53 EB 00 00 00 00  [YSë....
0068  00 00 00 00 08 00 48 00  ......H.
0070  10 FF 95 24 8D 3E 71 DC  .ÿ.$.>qÜ
0078  87 83 91 68 42 DD 99 2B  ...hBÝ.+
0080  0E 90 AB D2 19 AE 09 CD  ..«Ò.®.Í
0088  53 06 DD BC 37 FF 9F DB  S.Ý¼7ÿ.Û
0090  9E D2 D7 AA 19 C4 56 23  .Ò×ª.ÄV#
0098  49 B4 DA FC 29 C1 5A 29  I´Úü)ÁZ)
00A0  BC 0B 33 13 03 71 15 81  ¼.3..q..
00A8  66 66 5C B0 8A 40 80 1E  ff\°.@..
00B0  00 00 00 00 00 00 00 00  ........
00B8
```

server public key signature

The public key modulus (n) is the same as the one present in the RSA2 key stored in the LSA Secret "L$HYDRAENCKEY" of the server; the signature is the information used by the client to verify the server identity.

From a man-in-the-middle attacker's point of view, the public key signature must be modified on the fly to trick the client into verifying the new Mitm public key that will be replaced into the network packet directed to the client. But … what is used to produce this signature ?

Well, a digital signature is noting more nothing less than a hash of something (in this case a server public key) encrypted using a private key and an asymmetric encryption algorithm. This is exactly what is done by the terminal server. At the client-side, this signature is decrypted using a public key and the result is compared with a new hash of the received server public key calculated by the client; if the two hashes match the identity of the server is proven.

Here the ugly part ...
Microsoft use another RSA private key to sign the Terminal Server public key and this private key is public ! It could sound strange but this is only the truth, the private key used for the signature creation is hard-coded into mstlsapi.dll and it is dynamically created, used and de-allocated into a subroutine of the "TLSInit" API. Every Windows user has this file ... is this a new kind of public-private key (PPK) ?!?

The Microsoft Windows Terminal Server PPK follows:

```
public exponent: e
0x5B,0x7B,0x88,0xC0

public modulus: n
0x3D,0x3A,0x5E,0xBD,0x72,0x43,0x3E,0xC9,0x4D,0xBB,0xC1,0x1E,0x4A,0xBA,0x5F,0xCB,
0x3E,0x88,0x20,0x87,0xEF,0xF5,0xC1,0xE2,0xD7,0xB7,0x6B,0x9A,0xF2,0x52,0x45,0x95,
0xCE,0x63,0x65,0x6B,0x58,0x3A,0xFE,0xEF,0x7C,0xE7,0xBF,0xFE,0x3D,0xF6,0x5C,0x7D,
0x6C,0x5E,0x06,0x09,0x1A,0xF5,0x61,0xBB,0x20,0x93,0x09,0x5F,0x05,0x6D,0xEA,0x87,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00


private exponent: d
0x87,0xA7,0x19,0x32,0xDA,0x11,0x87,0x55,0x58,0x00,0x16,0x16,0x25,0x65,0x68,0xF8,
0x24,0x3E,0xE6,0xFA,0xE9,0x67,0x49,0x94,0xCF,0x92,0xCC,0x33,0x99,0xE8,0x08,0x60,
0x17,0x9A,0x12,0x9F,0x24,0xDD,0xB1,0x24,0x99,0xC7,0x3A,0xB8,0x0A,0x7B,0x0D,0xDD,
0x35,0x07,0x79,0x17,0x0B,0x51,0x9B,0xB3,0xC7,0x10,0x01,0x13,0xE7,0x3F,0xF3,0x5F,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

secret prime factor: p
0x3F,0xBD,0x29,0x20,0x57,0xD2,0x3B,0xF1,0x07,0xFA,0xDF,0xC1,0x16,0x31,0xE4,0x95,
0xEA,0xC1,0x2A,0x46,0x2B,0xAD,0x88,0x57,0x55,0xF0,0x57,0x58,0xC6,0x6F,0x95,0xEB,
0x00,0x00,0x00,0x00

secret prime factor: q
0x83,0xDD,0x9D,0xD0,0x03,0xB1,0x5A,0x9B,0x9E,0xB4,0x63,0x02,0x43,0x3E,0xDF,0xB0,
0x52,0x83,0x5F,0x6A,0x03,0xE7,0xD6,0x78,0x45,0x83,0x6A,0x5B,0xC4,0xCB,0xB1,0x93,
0x00,0x00,0x00,0x00

d mod (p-1): dmp1
0x65,0x9D,0x43,0xE8,0x48,0x17,0xCD,0x29,0x7E,0xB9,0x26,0x5C,0x79,0x66,0x58,0x61,
0x72,0x86,0x6A,0xA3,0x63,0xAD,0x63,0xB8,0xE1,0x80,0x4C,0x0F,0x36,0x7D,0xD9,0xA6,
0x00,0x00,0x00,0x00

d mod (q-1): dmq1
0x75,0x3F,0xEF,0x5A,0x01,0x5F,0xF6,0x0E,0xD7,0xCD,0x59,0x1C,0xC6,0xEC,0xDE,0xF3,
0x5A,0x03,0x09,0xFF,0xF5,0x23,0xCC,0x90,0x27,0x1D,0xAA,0x29,0x60,0xDE,0x05,0x6E,
0x00,0x00,0x00,0x00

q^-1 mod p: iqmp
0xC0,0x17,0x0E,0x57,0xF8,0x9E,0xD9,0x5C,0xF5,0xB9,0x3A,0xFC,0x0E,0xE2,0x33,0x27,
0x59,0x1D,0xD0,0x97,0x4A,0xB1,0xB1,0x1F,0xC3,0x37,0xD1,0xD6,0xE6,0x9B,0x35,0xAB,
0x00,0x00,0x00,0x00
```

The knowledge of the PPK key lets the attacker calculate a valid signature for the mitm public key generated on the fly during the mitm attack; the client will verify the mitm signature correctly and it will accept the session **without informing the users that the server key is changed from the usual one**.

The signature is calculated encrypting, with the private part of the PPK key, the MD5 hash of the server public key for a total of 108 bytes hashed.

Impact

By default, in administrative mode, only users with administrative accounts can access the terminal server. An attacker which can perform a successful mitm attack on the RDP sessions could easily obtain administrative credentials and compromise the server. Be warned that the attack could be completely invisible because the Remote Desktop client software does not inform the user about changed server keys.

Solution

More than a solution I can suggest to Microsoft to keep a list of known server keys at the client-side, alerting the user if one of the expected key is changed. Although this does not prevent mitm attacks, it can give to the user the opportunity to check for unusual activities.

Tools and exploits

The attack described above has been successfully implemented into the software Cain & Abel available at http://www.oxid.it. From version 2.7 the program can now perform man-in-the-middle attacks against RDP protocol sessions decrypting all the information that travels from client to server in both directions. The program try also to recognize the keyboard activity at the client-side providing some kind of password interception.

References

- *Microsoft Terminal Services vulnerable to MITM-attacks*: http://www.securityfocus.com/archive/1/317244
- *rdesktop* - A Remote Desktop Protocol Client: http://www.rdesktop.org
- *rdpproxy* - An RDP mitm proxy: http://cvs.sourceforge.net/viewcvs.py/rdesktop/rdpproxy/
- *Cain & Abel v2.7* - Provides the ability to perform invisible mitm attacks against RDP: http://www.oxid.it

Disclaimer

The information in this advisory are provided "AS IS" without warranty of any kind. In no event shall the authors be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages due to the misuse of any information provided in this advisory.

Massimiliano Montoro
mao@oxid.it

Microsoft and Microsoft Terminal Services are all registered trademarks of the Microsoft Corporation.