Global Information Assurance Certification Paper

Windows NT LPC Privilege Escalation Vulnerability

Local Privilege Escalation Exploit Could Be Exploited Remotely
SANS GIAC Level II Practicum Option 2
August 22, 2000

Steve Manzuik

Exploit Details:

Name:	Windows NT LPC Privilege Escalation Vulnerability
Variants:	No known variants
O/S Effected:	Windows NT Server/Workstation SP6 and below
Protocols/Services:	Local exploit of LPC API function NtlmpersonateClientOfPort()
Brief Description:	By exploiting a LPC call function a local user can escalate their privileges and even access areas of the Windows NT Registry that are normally not accessible. This exploit also makes it possible for malicious users to bypass access control lists on files and directories.

Vulnerability Details:

A problem in NtlmpersonateClientOfPort system call on NT 4 was discovered in January, 2000 by Todd Sabin of Bindview RAZOR (http://razor.bindview.com).

Microsoft Windows NT includes a feature called LPC ports. This feature is mostly undocumented, but LPC ports are used for making Local Procedure Calls on a machine. A system API used with LPC ports is NtlmpersonateClientOfPort. This API allows a server to act in the security context of the client who is calling it. However, the interface to the call lets the server specify which client to impersonate based on the process and thread IDs (PID and TID).

The NT kernel does provide a checking mechanism for the parameters to verify the call is legitimate but it is possible to fool it. First, it verifies that the port you are trying to impersonate on actually has an outstanding request. This is easy to satisfy by making a request to it ourselves. Next, it checks that the message ID in the request matches the outstanding message ID in the thread you are asking to impersonate. This is also relatively easy to satisfy, as the thread is not making a request, its outstanding message ID will be zero. So, when the request comes in, we just change the PID and TID to the ones we want and change the Message ID to 0. Once this is accomplished you are able to perform any task you want as that user.

By using this exploit it is very possible that Administrative or Domain Administrative access can be obtained. An executable to demonstrate this vulnerability, named HK has been created by Todd Sabin and is available at http://www.nmrc.org/files/nt/index.html.

Pseudo Code:

There are two threads.

Server thread	Client thread
NtCreatePort	
NtReplvWaitReceivePort…	
NtReplyWaitReceivePort…	
	NtConnectPort…
(returns)	
NtAcceptConnectPort…	
NtCompleteConnectPort	
NtReplyWaitReceivePort…	
	(returns)

NtRequestWaitReplyPort…

(returns)
modify the LpcMessage received in the request so that the process and thread IDs point to the thread we want, and change the message id to 0.
NtImpersonateClientOfPort

Now we are running under the token of the thread that was specified above. For the exploit, the thread of LSASS has been impersonated due to the level of privileges that LSASS has enabled.

The impersonation token only gets the privileges that are enabled in the client at the time of impersonation. Privileges that are disabled in the client are not put into the impersonation token, even in a disabled state. LSASS has the CREATE_TOKEN privilege enabled, so impersonating LSASS allows the use of that privilege to crate a new token based on the LSASS impersonation token, but with all privileges enabled. Once this is accomplished you are able to launch another process under that token.

Pseudo Code Continued:

// get the information about the current token.
// TOKEN_USER, TOKEN_GROUP, etc. (esp. TOKEN_PRIVILEGES)
NtOpenThreadToken
GetTokenInformation // (several times)

// Add all privileges to our TOKEN_PRIVILEGES struct
// all user space, preparing for NtCreateToken

NtCreateToken // with information from the LSASS token, except that all privileges are enabled

---

Now that there is a token a CreateProcessAsUser can be used. There are a couple issues when we get to this point. CreateProcessAsUser requires more privileges than LSASS had enabled, so we do not currently have them however, the newly created token does. So, yet another token needs to be impersonated. Unfortunately, this is not enough. CreateProcessAsUser checks for the privileges in the process token, ignoring any impersonation token. So in a final step, the new token has to be changed to be the primary process token.

Pseudo Code Continued:

ImpersonateLoggedOnUser

NtSetInformationProcess (… ProcessAccessToken …)

CreateProcessAsUser

Now you have your process.

Credit for Pseudo Code: Todd Sabin, Bindview RAZOR Advisory, January 13, 2000

Exploitable Possibilities:

Bindview RAZOR and Microsoft have both labeled this issue as a locally exploitable problem only. By using the executable HK, NetCat and the RDS exploit it is possible, in my opinion, to use this exploit remotely.

Here is the scenario.

Target Box: Windows NT Server SP6a – no hotfixes
IIS 4.0 installed and WWW service enabled for web hosting
Anonymous FTP also enabled or an open writeable share present

The target box would have to be vulnerable to the RDS exploit. Unfortunately, this requirement is not unrealistic as SANS listed the RDS exploit as number four in their list of top ten security threats (www.sans.org/topten.htm). It is my opinion that the above configuration is very common and simple scans using a variety of tools like Nmap and Whisker could easily identify hosts with this configuration.

Once our target has been identified the attacker must hope that the system administrator, like many do, has ignored recommendations to not use the EVERYONE group. If this condition is met, our attacker will be able to write (upload) his attack tools, Netcat and HK. In the event that write privileges cannot be found, our attacker can simply use the RDS exploit to create a new, writeable share.

Depending on the ability of the attacker he may now do one of two things. He could generate his own script to exploit the RDS vulnerability or he could simply point his web browser to www.wiretrip.net/rfp and download a working PERL script. For the purpose of this hypothesis we will assume that the attacker is using the PERL script provided by Rain.Forrest.Puppy (RFP).

By launching the RDS script successfully the attacker is able to assume the he has the ability to launch a command. The attacker would then launch the following command:

HK NC –l –p 23 –t –e cmd.exe

This command will use the Local Promotion Vulnerability proof of concept code (HK.EXE) to launch NetCat with elevated privileges. NetCat will then create a Telnet server on port 23. Our attacker now has to simply telnet to port 23 of the target box and he will be presented with a command prompt.

From here the possibilities are endless. The attacker can simply add himself a user to the administrators by using the built in NET USER commands. Or, the attacker may simply choose to steal/delete or modify files. The second scenario can obviously be taken advantage of by simply using the RDS exploit but gaining full administrative rights remotely definitely adds a higher level of danger.

Other possibilities once the attacker reaches this point are registry edits, rootkit installations, back door creation, DDoS client installation etc. Once this attack has been carried out, the attacker owns the target machine and can do anything he pleases.

Defenses:

This paper has discussed two separate attacks, a locally exploitable one and a scarier remotely exploitable one.

To defend against the locally exploitable attack Microsoft has released a hotfix. We must also be sure to not overlook the fact that not only do administrators need to install this hotfix but they also need to be sure to review the physical security of their servers, especially the ones containing sensitive data.

The Microsoft hotfix is available at:

Intel: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=17382
Alpha: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=17383

The second attack scenario is a little more difficult to defend against, as it requires defense from multiple threats. Defending against this attack however, is not impossible.

First, administrators need to defend themselves from the RDS vulnerability. Advice on correcting this problem is available from both Microsoft and RFP, who discovered the problem.

http://www.wiretrip.net/rfp/p/doc.asp?id=29&iface=2
http://support.microsoft.com/support/kb/articles/q184/3/75.asp
http://www.microsoft.com/technet/security/bulletin/ms98-004.asp
http://www.microsoft.com/technet/security/bulletin/ms99-025.asp

Information is also available at www.sans.org/topten.html

Second, administrators need to install the Microsoft hotfix to correct the NtlmpersonateClientOfPort vulnerability (Local Promotion Vulnerbility).

And third, to protect against the threat of NetCat being used administrators need to have all Internet exposed hosts behind a firewall and only allow connections to ports that are required for business operation. These exposed hosts then need to be periodically port scanned and the process listening on each port must be verified. It is my understanding that NetCat will listen on a port that is already in use by another process so it is important to physically verify each listening port by performing a simple telnet to each one.

Attack Detection:

Detecting this attack while it is being carried out is difficult and chances are, most administrators will not notice the extra traffic caused by the exploits. The use of an Intrusion Detection System (IDS) might assist in detecting at a minimum the use of RFPs RDS script.

After the fact however, there are a number of things that would leave clues as to what was done to the attacked server. Logging can be your best friend when it comes to detecting attacks. In this case, logging can only partly help you but it is a start.

The RDS exploit as created by RFP will make GET requests to /msadc/msadcs.dll which will show in the IIS logs. This is a tip that someone is at least looking at attempting the RDS exploit on your machine.

As explained by RFP the script will then begin to make RDS queries using POST requests to one of the following URLs:

Normal query:
/msadc/msadcs.dll/ActiveDataFactory.Query

VbBusObj to bypass custom handlers:
/msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetRecordset

Query VbBusObj for NetBIOS name:
/msadc/msadcs.dll/VbBusObj.VbBusObjCls.GetMachineName

If you happen to be actually using RDS functionality on your web site ActiveDataFactory.Query will appear in your logs but requests to the other two URLs should set off alarms.

It is important to note that a skilled attacker may have created his own custom exploit script for the RDS vulnerability and the above listed log entries may only exist if the PERL script by RFP is used.

To detect the use of HK.EXE you could turn on auditing for processes and objects but it is fairly impractical to do so without the use of third party logfile analyzers.

Of course periodically checking for open ports on exposed hosts and verifying what is listening on those ports will help detect the use of NetCat.

Conclusion:

It is my opinion that a remote attack as described in this paper could be carried out by an attacker with basic skills. All of the tools and scripts to perform this attack are readily available on the Internet and very simple to carry out.

I would also consider this type of attack to be high risk as even SANS has identified that there are a lot of misconfigured Windows NT Servers exposed to the Internet.

Obviously better education, especially in security and maintenance is needed for our system administrators. In order to defend from this attack system administrators do not need a lot of skill, just a little knowledge.

Credits:

Local Promotion Vulnerability in Windows NT 4.0 – Todd Sabin, Bindview RAZOR – http://razor.bindview.com

RDS Exploit – Rain.Forrest.Puppy (RFP) – www.wiretrip.net/rfp

NetCat for NT – Weld Pond, L0pht, www.l0pht.com/~weld/netcat

SANS Top 10 List – The Sans Institute – www.sans.org

Special thank you to: Simple Nomad, Todd Sabin, and Bob Keyes all of BindView RAZOR for discussing the feasibility of this attack with me.