# Global Information Assurance Certification Paper

*GSEC Certification Practical Assignment - Version 2.0*
*Joe Finamore*

**NULL Sessions In NT/2000**
**Joe Finamore**
**December 2001**

## Overview

Riddle: "When is a NULL not a null?"
Answer: "When it's a null session."

A null session is a session established with a server when no credentials are supplied. This paper is going to discuss the issue of null sessions in NT 4.0 and Windows 2000. It will investigate the uses and vulnerabilities of such sessions, and will show how to control and/or eliminate those vulnerabilities.

## LANMAN Sessions On NT 4.0

Before we can begin a discussion of null sessions, we need to find out what a session is. There is an excellent discussion of NTLM authentication at the URL:

*http://www.microsoft.com/msj/defaultframe.asp?page=/msj/0299/security/security0299.htm&nav=/msj/0299/newnav.htm*

Windows NT 4.0 uses a challenge-response protocol to establish a session with a remote machine. This session is a secure channel through which information may flow between the machines participating in the secure channel. The sequence of events goes like this:

1) The session requestor (client) sends a packet to the session acceptor (server), requesting the establishment of a secure channel.
2) The server generates a random 64-bit number (the challenge) and sends it back to the client.
3) The client takes the 64-bit number that was generated by the server and hashes it with the password of the user account that the client is trying to establish the session as. It sends this back to the server (the response).
4) The server accepts the response and passes it off to the Local Security Authority (LSA). The LSA confirms the identity of the requestor by verifying that the response was hashed with the correct password for the user that the requestor is purporting to be. This confirmation occurs locally if the requestor's account is a local account on the server. If the requestor's account is a domain account, the response is forwarded to a domain controller for verification. If the response to the challenge is confirmed to be correct, an access token is generated and sent to the client. The client then uses this access token to connect to resources on the server until the newly established session is terminated.

## LANMAN Sessions On Windows 2000 - Kerberos Authentication

Windows 2000 uses Kerberos to establish a session 'ticket'. The RFC for Kerberos version 5 can be found at:

*http://www.ietf.org/rfc/rfc1510.txt*

The sequence of events runs like this:

1) The client sends a request to the KDC (Key Distribution Center) for a TGT (Ticket-Granting Ticket). This request contains preauthentication data that is encrypted with a hash of the user's password. The preauthentication data also includes a time stamp to assure that the TGT can't be captured and then played back later. The KDC runs on a domain controller for the domain that is granting the ticket.

2) The KDC extracts a hash of the user identified in the request from its database and decrypts the preauthentication data with it. If the decryption works and has a very recent timestamp, the process continues.

3) The server generates a TGT that includes, among other things, a session key encrypted with the user's hashed password. It also includes Security Identifiers (SIDs) which identify the user and the groups belonged to.

4) The client decrypts the session key with the hash of the user's password.

5) The client uses the ticket to access resources on the server. The client has now been authenticated and a session has been established.

The ticket produced in this manner contains the following unencrypted information:

- Domain name of the Windows 2000 domain that issued the ticket
- The name of the principal the ticket identifies

The ticket also contains the following encrypted information:

- Ticket 'flags'
- The session encryption key
- The name of the domain that contains the user account that was issued the ticket
- The principal name of the user the ticket was issued to
- The session start time
- The session end time: when the ticket will expire. Tickets have a finite lifespan.
- The address(es) of the client's machine
- Authorization data which contains information about the client's allowed access

**What Is A Null Session?**

Now that we understand what a session is, and how a session includes authentication information to authorize access to resources, we can begin to unravel the mystery of null sessions. A null session is a session established with a server in which no user authentication is performed. In other words, it is anonymous access to a server. No user and password credentials are supplied in the establishment of the session. The access token ('authorization data' on Windows 2000) contains a SID of "S-1-5-7" for the user, and a username of "ANONYMOUS LOGON". This access token contains the following pseudogroups:

Everyone
NETWORK

This grants access to anything the above two pseudogroups have rights to, within the limits of the security policy.

**How To Create A Null Session**

From a user's point of view, a session is established with a server either at login time, or at some other time when access is needed to a server for some resource. If, for example, a user named "BOB" wishes to access some files on a share named "DATA" which resides on a server named "DATASTORE" which he is not already authenticated on, he would issue a command similar to:

net use * \\DATASTORE\DATA * /user:BOB

He would be prompted for his password and the appropriate authentication method would kick in. Assuming he is properly authenticated, he will be granted an 'access token' or a 'ticket' and, using that, will be connected to the desired share.

If, on the other hand, null sessions are allowed, and the "DATA" share is setup as a 'null share', he could simply type:

net use * \\DATASTORE\DATA "" /user:""

This will connect Bob, as an anonymous user, to the "DATA" share without requiring him to supply a username or a password...a cracker's dream!

Null sessions can also be established at the API level with languages such as C++. A good discussion of establishing null sessions using the WIN32 API calls can be found at:

*http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=494*

Null sessions can be used to establish connections to 'null session pipes', if the server is so configured. A 'pipe' is a facility that allows a process on one system to communicate with a process on a different system. Null sessions can also be used to establish connections to shares, including such system shares as \\servername\IPC$. The IPC$ is a special hidden share which allows communication between two processes on the same system (Inter Process Communication). The IPC$ share is an interface to the 'server' process on the machine. It is also associated with a pipe so it can be accessed remotely.

**Why Were Null Sessions Created?**

The question that logically raises itself at this point is: "Why did Microsoft create support for null sessions?" Microsoft touts the security of NT and Windows 2000. Isn't a null session more-or-less a bypass of authentication security?

Well, in a general sense, "Yes". Null sessions do tend to undermine the underlying security structure of the operating system. There were, however, compelling reasons to incorporate them into the design of Microsoft Networking. The original purpose of null sessions was to allow unauthenticated machines to obtain browse lists from servers. It should be remembered that both NT and Windows 2000 organize machines into groups known as "domains". Domains are collections of machines that share the same security boundaries. That is to say, they share the same database of user and machine accounts, as well as the passwords connected to each. A 'user' password is used to authenticate a user in a domain. A 'machine' password is used to maintain a 'secure channel' between a machine and the domain controller. In both cases, the password is used to establish a measure of trust between the machine/user and the domain server.

If all communication were intra-domain, null sessions would be largely unnecessary. This is not the case, however. There is often the need to use inter-domain communication to perform such tasks as:

Obtain a browse list from a server in a different domain
Authenticate a user in a different domain

This problem is addressed partially by the concept of inter-domain 'trust relationships'. A trust relationship is a relationship negotiated between two different domains whereby a domain agrees to trust the security integrity of the other domain and, therefore, the domain controllers in the two domains allow the flow of information between them. A password is negotiated at the establishment of the trust relationship, and that password is used to establish a secure channel between the domain controllers. So, in essence, a trust relationship is an authenticated relationship between domains.

The problem is that not even trust relationships address all issues of inter-connectivity that may be needed at a site. Consider, for example, the process of establishing a trust relationship in the first place. If "DOMAIN1" wishes to establish a trust relationship with "DOMAIN2", it needs to contact the PDC for that domain and negotiate a password for the secure channel. In order to do that, it needs to enumerate the machines in the domain and determine the name of the PDC for "DOMAIN2". There are various methods for finding this name (and subsequently, address), including WINS, DNS, LMHOSTS, AD (Active Directory), etc. Null sessions make this process much easier to accomplish, however, because they allow direct enumeration of machines and resources in a domain from an unauthenticated machine with little prior knowledge.

There are other scenarios that benefit greatly from null sessions. For example, consider an administrator in a multi-domain site in which trust relationships have not been setup between all domains, for whatever reason. It is often necessary, in the course of an administrator's job, to connect to resources in all domains. Null sessions make the enumeration of users, machines, and resources easier.

Another situation, one that requires null sessions, is the circumstance where the LMHOSTS.SAM file uses the "#INCLUDE <filename>" tag. The share point that contains the included file must be setup as a null session share. An article that discusses this topic can be found at:

*http://support.microsoft.com/default.aspx?scid=kb;EN-US;q121281*

This is an older article, originally published in 1994, but it was updated on 8/8/2001. It should be noted that the sample LMHOSTS.SAM files installed with NT 4.0 and Windows 2000 discuss this situation in the comments. A lot of the LANMAN legacy has survived into Windows 2000.

It should be noted here, as well, that some vendors have advocated the use of null sessions with their software. An interesting article on this is found at:

*http://www.dcs.ed.ac.uk/home/archives/bugtraq/msg00784.html*

This article refers to an installation procedure from a vendor that created null shares on a

server to perform its tasks. It is not inconceivable that a server is at risk, unbeknownst to the administrator. (The author has personally received recommendations from a vendor in a similar situation to solve the problem in likewise fashion.)

One final example of the 'usefulness' of null sessions is the situation where a service, running under the local "SYSTEM" account, needs access to some network resource. This is only possible if the resource is accessible through a null session. There is an article at Microsoft that discusses this:

*http://support.microsoft.com/default.aspx?scid=kb;EN-US;q124184*

Microsoft does not recommend opening up null sessions to get around this problem, however. Their recommendation is to use a user-specific account instead to run the service.

**What Are the Vulnerabilities of Null Sessions?**

So, now that we have a better understanding of sessions, in general, and null sessions, in particular, what sorts of vulnerabilities do null sessions present? There are several reasons why they can cause security concerns. As we are undoubtedly aware, Access Control Lists (ACLs), which are lists of ACEs (Access Control Entries), control access to resources in an NT or Windows 2000 domain. An ACE specifies a user/group by SID, and enumerates which rights the user/group is granted or denied. The problem lies in ACEs that grant rights to the built-in group "Everyone". "Everyone", in NT parlance, means literally everyone. If the group "Everyone" has rights to a resource through an ACE, and the means to access that resource, such as a pipe or a share, is also open to "Everyone", then that resource is anonymously accessible to anyone.

Well then, what sorts of things might this include? If you perform an out-of-the-box install of NT 4.0, you'll notice lots of things are accessible to "Everyone"; notably the root of the system drive (usually C:\). A particularly compelling area that is open to "Everyone" is the folder that contains repair information:

%SystemRoot%\Repair (Usually: "C:\Winnt\Repair")

A couple of the more sensitive files, such as "sam._" have more restrictive security on them, but most of the files located here are world-readable. If, for some reason, a share point is available in a parent folder of this one, and the share is a null session share, these files would be readily available to an anonymous intruder. You'll also notice that many areas of the registry are accessible to "Everyone". This makes it possible to connect to the IPC$ share of a server and run the registry editor (REGEDT32.EXE) to view, and even change, some registry keys...all from the comfort of anonymity.

Another vulnerability exposed by null sessions is the enumeration of user accounts in a domain. Why is this a problem? It is a problem because it removes half of the barrier in place to prevent breaking into a domain account. In order to masquerade as someone else, you need two pieces of information:

username
password

Once you know the username, it's just a matter of guessing, or cracking, the password.

This vulnerability is at its greatest when it exposes the username of the renamed 'Administrator' account in a domain (You did rename your administrator account, didn't you?). All an intruder has to do is connect a null session, and enumerate the users, looking for a user with a SID of "500". An example of the code that can be used to accomplish this can be found at:

*http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=494*

This vulnerability was a crucial component of the classic "Red Button Attack" which was addressed and fixed by Microsoft in Service Pack 3 for NT 4.0.

The enumeration of machines and resources in a domain also makes it easier for someone to break in. If someone can anonymously obtain the names of all of the machines in a domain, and then list the resource shares on those machines, it becomes a simple matter to try all of them until one is found which is open to "Everyone". Given that the root of the system drive is open to "Everyone" by default, and that the default share-level security applied to a newly created share is to grant "Full Access" to "Everyone", the problem is obvious.

**How Do I Protect Against This Vulnerability?**

The best way to prevent this is to disallow null sessions to the fullest extent possible. In order to do this, an assessment of the vulnerability is a good starting point. The 'DumpSec' tool will enumerate shares on a system, as well as the security tied to each. It will also dump registry permissions and perform a plethora of other useful security auditing tasks. DumpSec is available at:

*http://www.systemtools.com/somarsoft/*

There are a couple of registry keys that are pertinent here as well:

**HKLM\System\CurrentControlSet\Control\Lsa\RestrictAnonymous**

"HKLM" refers to the hive "HKEY_LOCAL_MACHINE". If this is set to "1" anonymous connections are restricted. An anonymous user can still connect to the IPC$ share, but is restricted as to which information is obtainable through that connection. A value of "1" restricts anonymous users from enumerating SAM accounts and shares. A Value of "2", added in Windows 2000, restricts all anonymous access unless explicitly granted.

The other keys to inspect are:

**HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\NullSessionShares**

and:

**HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\NullSessionPipes**

These are MULTI_SZ (multi-line string) registry parameters that list the shares and pipes, respectively, that are open to null sessions. Verify that there are no shares and pipes open that you do not want open. Put security on the above keys as well so they can't be easily modified. Verify that only "SYSTEM" and "Administrators" have access to these keys.

In Windows 2000, security policy is the place to setup the protection. The policy is set by launching the appropriate MMC (Microsoft Management Console) snap-in. On a Domain Controller (DC), launch the "Domain Security Policy" MMC applet from the "Administrative Tools" program group. On non-DCs, launch the "Local Security Policy" MMC applet. You will find an entry labeled:

"Additional restrictions for anonymous connections"

There are 3 possible values to set:

"None. Rely on default permissions."

"Do not allow enumeration of SAM accounts and shares."

"No access without explicit anonymous permission."

The last value is the most secure. It corresponds to the value of "2" in the registry value:

**HKLM\System\CurrentControlSet\Control\Lsa\RestrictAnonymous**

discussed above. Be sure to check out the "effective setting". Policies set at another level can affect the "effective setting".

Another step that is advisable is to restrict remote registry access. Only Administrators and Backup Operators have network access to the registry by default in Windows 2000 and later. It's a good idea to check the remote registry access settings on any server. This is accomplished by verifying security permissions on the registry key:

**HKLM\System\CurrentControlSet\Control\SecurePipeServers\winreg**

When a user attempts to connect to the registry of a remote computer, the 'server' service on the target computer checks for the presence of the above (winreg) key. If the key does not exist, the user is permitted to connect to the target computer's registry. If the key does exist, the ACL on the key is checked. If the ACL gives the user read or write access, that user may connect to the registry. Once a user is allowed to remotely connect to a registry, the ACLs on the individual keys then come into effect. If, for example, security on any given key is set to allow 'read' access for "Everyone", anonymous 'read' access to the registry through a null session is possible; not a good idea! It would be a daunting, and dangerous, task to attempt to remove security for "Everyone" from all registry keys, so the better idea is to disallow remote access completely except for specific accounts and groups. There are several values under the 'winreg' key that apply as well. The values on NT 4.0 are:

**HKLM\...\winreg\AllowedPaths\Machine**

**HKLM\...\winreg\AllowedPaths\Users**

Both of these values are of the type MULTI_SZ. In Windows 2000, there is no "Users" value by default. These keys enumerate which registry keys are open to remote access by Machines and Users, respectively. They can be used to override the security on the 'winreg' key itself. Machines may need access for such services as directory replication and spooling. There are a couple of Microsoft Knowledge Base articles on this topic which comprehensively outline the particulars of remote registry access:

*http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q186433*

*http://support.microsoft.com/directory/article.asp?ID=kb;en-us;Q153183*

It's a good idea, after installing software on a server, to verify that the installation program did not enable any null session shares or pipes. In fact, as always, it's a good idea to test

the installation on a non-production server before executing it on the 'real' server, especially if the server is on a DMZ or external LAN. Anonymous access to a machine that sits on the public, or semi-public side, could be disastrous.

Remember, also, that the security token for a null session contains the two pseudogroups "Everyone" and "NETWORK". You can protect files by locating every location on a volume where these groups have access to and then modifying those permissions. This assumes, of course, that the volume is NTFS. Unless you absolutely need null session access to a resource, replace the group "Everyone" with the built-in group "Authenticated Users" on the resource in question. Use a utility such as XCACLS, or one of the third-party ACL-management utilities, to accomplish this. Proceed with caution, however, any time you attempt an operation like this, particularly when you use it to recurse into subfolders. Always use the "/E" option with XCACLS to verify that you are Editing the ACLs rather than replacing them. Remove the group "Everyone" every place you find it, and add, instead, the same rights to the group "Authenticated Users". Also, any time you create a share, remove "Everyone" from the ACL for that share. Use "Authenticated Users", or whatever user or group is appropriate, instead to grant share-level access.

One final step that can be taken is to set the policy for:

"Access this computer from the network"

by removing this privilege from "Everyone" if that group enjoys that right. Be sure to add this privilege to the appropriate users and groups before revoking this right from "Everyone". This is accomplished in NT 4.0 in "User Manager" on a non-DC and in "User Manager For Domains" on a Domain Controller. This is accomplished in Windows 2000 by modifying this policy under "User Rights Assignment" in the appropriate MMC security policy snap-in.

**Conclusion:**

Null sessions were created in order to facilitate inter-platform communication, especially at the service level. Use of null sessions, however, can expose information to an anonymous user that could compromise security on a system. If possible, eliminate null sessions entirely from your system. If, for whatever reason, this is not possible, take all possible precautions to ensure that the only information exposed is the information you want exposed. If not, null sessions could provide a convenient ingress point into your system that could lead to a security compromise.

**References:**

"About NULL Sessions." 28 June 1998. URL: http://downloads.securityfocus.com/library/null.sessions.html (6 Dec. 2001)

"Berbee: Security - ChkLock." 2001. URL: http://www.berbee.com/security/techover.html (6 Dec. 2001)

"Dr. Solomon's" - Possible Hole." 15 June 1998. URL:

http://www.dcs.ed.ac.uk/home/archives/bugtraq/msg00784.html (6 Dec. 2001)

Chappell, David. "Exploring Kerberos, the Protocol for Distributed Security in Windows 2000 -- MSJ, August 1999." Aug. 1999. URL: http://www.microsoft.com/msj/defaultframe.asp?page=/msj/0899/kerberos/kerberos.htm &nav=/msj/0899/newnav.htm (6 Dec. 2001)

"LMHOSTS #Include Directives Requires Null Session Support." 8 Aug. 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q121281 (7 Dec. 2001)

"Security Briefs Notes & Errata (Feb '99)." Feb 1999. URL: http://www.develop.com/kbrown/securitybriefs/sb9902.htm (5 Dec. 2001)

Brown, Keith. "Security Briefs Q&A, MSJ February 1999." Feb. 1999. URL: http://www.microsoft.com/msj/defaultframe.asp?page=/msj/0299/security/security0299.h tm&nav=/msj/0299/newnav.htm (7 Dec. 2001)

"NT Null Session Admin Name Vulnerability." 28 June 1999. URL: http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=494 (7 Dec. 2001)

"Service Running as System Account Fails Accessing Network." 8 Aug. 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q124184 (7 Dec. 2001)

"Clarification of Winreg Operation in Windows NT." 9 Aug. 2001. URL: http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q186433&ID=KB;EN-US;Q186433 (7 Dec. 2001)

"How to Restrict Access to the Registry from a Remote Computer." 19 Oct. 2001. URL: http://support.microsoft.com/default.aspx?scid=kb;en-us;Q153183&ID=kb;en-us;Q153183 (7 Dec. 2001)