

## Restrict Anonymous: Enumeration and the Null User

*Timothy M. Mullen* 2001-02-12

### Restrict Anonymous: Enumeration and the Null User

by *Timothy M. Mullen*

last updated Feb. 12, 2001

---

If you are an NT administrator, or if you provide security policies and audits for clients, then you know all about the RestrictAnonymous value in the LSA key. If not, you need to educate yourself about this setting - not so much because of what it does, but more importantly, what it doesn't do.

## An Overview of the 'Null' User

Before we discuss RestrictAnonymous, there is some basic ground we have to cover: At the core of the Windows authentication scheme lies the 'User' administrative unit. When we create users, we assign them rights, privileges, and policy restrictions to define what they can and can't do on our networks. But in addition to the standard User, Windows supports a special type of user called the 'Null' user, which is basically a pseudo-account that has no username or password, but is allowed to access certain information on the network. (Some call this feature the 'anonymous' user, but I will just call him 'Null' in order to avoid confusion with the 'anonymous' account impersonation used by IIS). By design, the Null user can enumerate account names and shares on domain controllers, member servers, and workstations alike. Therefore this Null user, a user with no credentials, can be used to glean a tremendous amount of information from your network without raising any eyebrows.

The obvious question is, "why would we ever need support for a 'Null' user?" Surprisingly, the Null user account is used by IPC (Interprocess Communications) all the time. For instance, in a multi-domain model where one-way trusts are created, the Null user is used when permissions for resources in the trusting domain need to be granted to user accounts in the trusted domain. After all, the users needing to choose from a list of available accounts are not trusted, so someone has to be able to enumerate the users. Other software like Microsoft's System Management Server uses the Null user to enumerate shares during discovery. Even the mundane action of starting a NT workstation or server creates an initial Null user logon to one of your domain controllers after its NetBIOS name is successfully registered (before you get the Ctrl + Alt + Delete logon prompt). In fact as far as NT 4.0 is concerned, without the successful

logon of the Null user, you would not be able to initialize a netlogon secure channel at all!

So what is the big deal? Well, anyone with a NetBIOS connection to your box can easily get a full dump of all your usernames, groups, shares, permissions, policies, services and more using the Null user. Needless to say, that is a tremendous amount of information for an attacker to have when attempting to breach your network's security. Programs like DumpSec (formerly DumpACL) exist solely for that purpose, and are quite popular as a result. By explicitly connecting to the target box as the Null user via the `[net use \\server "" /user:""]` command and then making NET\* enumeration API calls like NetServerGetInfo (supported by the Netapi32 library), you will be well on your way to enumerating username and policy data on the target system.

## RestrictAnonymous

This obviously did not sit well with many security folks, so Microsoft introduced support for the famed "RestrictAnonymous" registry value in SP3. With this development, an administrator could simply create the value name of "RestrictAnonymous" in the CurrentControlSet\Control\LSA key under the HKEY\_LOCAL\_MACHINE\SYSTEM hive in the system registry, plop in the DWORD value of 1, and - Presto! - no more Null user enumeration. Registries across the world were updated, administrators toasted each other's technical prowess, users cheered out their names in glory, and the world was safe from the evils of Null session enumeration!

The setting of "RestrictAnonymous" has since become standard practice for administrators. It is recommended in the securing of Web servers. It is a common setting on sensitive domain controllers. It is even a required setting if you want your systems to meet the requirements of what used to be the highest government rating for business computing products known as "C2." It also doesn't really prevent the enumeration of your usernames. That's right - in NT 4.0, the "RestrictAnonymous" setting does not prevent the Null user from getting information. As its name implies, it just restricts it a bit. Here is why:

Since Null user functionality is required for core NT domain functions, the decision was made to limit some of the things this guy could do when "RestrictAnonymous" was set rather than completely killing him off. So while certain API calls had some ACLs assigned to prevent them from being called by the Null user, some were left alone.

One of the first enumeration calls that DumpSec makes is NetServerGetInfo. When it does so,

the "RestrictAnonymous" ACLs cause it to return with failure and the program terminates. However, if you were to use programs like User2Sid and Sid2User, you would find that they function perfectly even when RA is set since they use the API calls LookupAccountName and LookupAccountSid respectively, which have no ACL's set. This is particularly dangerous since "LookupAccountName" allows you to specify any known user account, group, or domain name to return its SID. One can easily request the SID of a persistent group name like "domain admins" and concatenate the UserID of built-in accounts (like "500" for Administrator) to the Identifier Authority portion of the SID in order determine the user's full SID. LookupAccountSID will then retrieve the actual username of this user (the Administrator), even if it has been renamed from its default. When I learned this, it surprised me. I had assumed, as I think many have, that setting RA=1 kept the Null user from getting to anything. Big mistake.

## RestrictAnonymous and NetUserGetInfo

I did a little digging on my own, researched the C++ Platform SDK documentation, and found that Microsoft supposedly set ACL's on most of the NetServer\*, NetUser\*, and NetGroup\* functions in order to restrict their operation when called by the Null user. So I wrote a bunch of little C++ functoids to check this out. When this is done, NetServerGetInfo dies. NetUserEnum, used to enumerate all users, dies. NetGroupGetUsers, used to dump the usernames in a particular group, also dies. But NetUserGetInfo does not. Though the docs specify that RestrictAnonymous is enforced on this call, the NetUserGetInfo works against both Win2k and NT4.0 domain controllers, member servers, and workstations in order to enumerate information on a known user. Even a level 3 call to this API function, which returns information like password expiration date, logon hours, and workstation logon restrictions, returns successfully when called by the Null user. And if you go through the trouble of breaking out the return DWORD flags into their member bits, you can even retrieve policy data such as whether the user can change the password, whether the password ever expires, if the account is disabled, if you must have a Smartcard to logon interactively, etc. You can also dump special Operator privileges to see if the user is a Server Operator, Account Operator, or what have you. And though you must test against a valid username, it is a quick and easy way to determine if "Administrator" is both valid and the real built-in administrator account, or if users like "test" or "admin" exist out there; and it is all done anonymously, with no obvious audit trail. If you would like to test this out on your own, here is a link to the little tool I whipped up, which I called [userinfo 1.5](#).

For those of you that want a little more meat with your enumeration, I wrote another

application called [UserDump](#) that is based on the same NetUserGetInfo call, but with an interesting twist: this tool combines LookupAccountName, LookupAccountSID, and NetUserGetInfo together into a tool that iterates through the user directory SID by SID (starting with the Administrator SID of 500), grabs the account name, and then sticks it into NetUserGetInfo to dump all the info for you. In a single command line, you can literally dump the entire domain user-base to peruse at your leisure - all anonymously, and all with RA set on the target boxes!

Remember, these functions will fail if your current logon credentials do not match those of a domain or local user, or if you are not the Null user. That is why it is important to explicitly connect up to the IPC\$ share as the Null user before you rush out and test this, or else you will get confusing results. I had a version that explicitly ran the thread under the anonymous token, but those would only run on a Win2k box, so I pulled it.

So, what do you do? Unfortunately, the short answer is "nothing". Currently, Windows NT 4.0 only supports a RestrictAnonymous value of 1 (and I hear that it will stay that way). Short of removing the IPC\$ share there is not much you can do to prevent this. Of course, you can disable the WINS client (TCP/IP) bindings on the NIC's of your Internet-facing machines (or use IPSec in Win2k) and/or block NetBIOS at your firewall, which you should all be doing anyway if possible. This will keep Internet-based traffic from calling enumeration functions, but it still leaves the boxes on your intranet exposed to enumeration from within. For some of us, this is not a big deal, but for others it is; while we would like to think that we can trust our users on the inside, we know that in some cases we can't. But, by the same token, if you can't trust the folks on the inside, then you have bigger problems than anonymous enumeration. This is why a policy of auditing, log review, and strong password enforcement is important. You will find that the latter, strong password enforcement, is key to many of the potential security issues you will face. The trick is to make users choose passwords that are strong enough to withstand cracking for a reasonable time period, but not so complex that they end up writing them on sticky notes and plastering them to their monitors.

There is, however, a little light at the end of this tunnel, and that is Windows 2000. But as the old adage says, that light may actually be a train coming in the other direction?

## **RestrictAnonymous Value '2'**

Windows 2000 introduced support for a new value in RestrictAnonymous - '2'. This is pretty

cool because it prevents all anonymous calls (as far as I can tell) that are not explicitly granted to the null user. When Windows 2000 builds the access token for the Null user, he is specifically not included in the 'Everyone' group when RA is set to 2, and that is the key. In reality, the only reason Null can do so much in NT land is not because he is somebody special, but because he is part of the Everyone group. Since Windows 2000 allows us to remove him from this group, we can effectively lock-down what he can do. In fact, this setting prevents him from using the IPC\$ share in the first place. I tried going back to an NT box and setting RA to 2 just to see if there was some super-secret undocumented support for this, but no joy.

If you are one of the many admins out there concerned about the power of Null, I suggest you read up on your new options in Windows 2000. However, be warned that you should pay particular attention to the words "undesirable behavior" when you come across them in your research. As you may know, "undesired behavior" is Microsoftese for "breaks everything." You see, while setting RA to 2 will indeed keep a Null user from talking to your server, it will also keep any NT 4.0 box from doing so as well. So, before you rush out and set your Domain Controller Policy to use "No access without explicit anonymous permissions" (RA=2), make sure that you understand that doing so will keep all of your NT member servers and workstations from finding any domain controller using this setting. It will also keep down-level servers in trusted domains from talking to you, and any browser service out there that tries to get a domain or server list from a RA=2 box will fail miserably. Due to these different "results", you are not advised to utilize this setting in mixed-mode environments unless you have done more testing than any of us ever has time to do.

## Conclusion

I, of course, researched everything beforehand and had it all planned out so that the implementation of the new policy went without incident. Well, maybe there was a little trial and error. Now that I think about it, I guess there could have been a few issues on the back-end. OK, so I really broke everything, but it was a good learning experience anyway.

The point is, know what RestrictAnonymous does, and more importantly, what it doesn't do. And if you go the Windows 2000 route, be careful you don't end up restricting a lot more than you planned to when you set out to simply restrict anonymous logons. I still wonder what business RA=1 has being part of the C2 specification, but that is another article.

*Timothy M. Mullen is CIO of AnchorIS.Com, a software development company specializing in manufacturing*

*and enterprise based-accounting systems and solutions.*

## Relevant Links

[userinfo 1.5](#)

*Timothy M. Mullen*

[userdump 1.11](#)

*Timothy M. Mullen*

[MSKB Q143474: Restricting Information Available to Anonymous Logon Users](#)

*Microsoft Product Support Services*

[MSKB Q129457: RestrictAnonymous Access Enabled Lets Anonymous Connections Obtain the Password Policy](#)

*Microsoft Product Support Services*

[MSKB Q246261: How to Use the RestrictAnonymous Registry Value in Windows 2000](#)

*Microsoft Product Support Services*

[Privacy Statement](#)

Copyright 2006, SecurityFocus