

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

As the number of bad guys out there has grown so quickly in the past year, security's been on all of our minds. What can we do to make our systems a bit harder to hack? Many security fixes involve patching some hole that Microsoft inadvertently stuck in the operating system, a hole large enough for the dirtbags to crawl through. In most cases the question of "should we plug this hole?" is a pretty easy one. But this month, we'll talk about a hole that seems pretty scary to some folks -- but we'll have to be careful, as closing it can break things. Nevertheless, it's something that everyone should examine. I'm talking about something called the "null session;" it's also referred to as an "anonymous logon" and it's umm, not a security hole, it's a *feature*. (Wink wink... and note that none of this has anything to do with anonymous logins on an FTP server -- that's a completely different topic.)

What is a null session? What can it do?

A null session is a logon connection established without credentials. Yes, that's right -- the vast majority of NT 4, 2000, XP and 2003 systems let people log onto them without supplying a user name or password. Null sessions/anonymous logins can be worrisome because by default they allow *anyone* to peek into an NT 4 domain or a Windows 2000-based AD domain and dump out things including

- the list of users in your system's SAM
- SIDs for user accounts and convert SIDs to user names
- the domain's browse list
- the system or domain's password and lockout policies
- the machine's NetBIOS name and the name of the domain that it belongs to
- the list of groups in the system's SAM
- the domains that your domain trusts

But what's the harm in that? Well, there is no immediate, direct harm, as null sessions aren't about to cough up passwords. But the more that the dirtbags know, the easier it is for them to get into your system. In theory someone could retrieve user names and then just run a program to try every single possible password against the names. Of course that might take quite a long time, and with hope you'd notice in your security logs after a while that user Jane23 had had five million failed logins in the past two days. And if you had a maximum failed logon attempts setting then a dirtbag could retrieve all of your user accounts and deliberately use incorrect passwords to try to log each user on with these incorrect passwords until the dirtbag had managed to lock everyone in your domain out altogether! (In this case the default Administrator account can still log on to unlock users, but only when sitting at a domain controller.) So just letting *anyone* see your list of users is probably a bad idea.

Again, let me clarify: *anyone* can establish a null session. They don't need an account on your domain. It works even if you've disabled the Guest account.

An example null session

What's that you say? You want to try it out? Well, first of all, don't try it out on someone *else's* network -- that might not be legal, depending on where you live. And you might not want to try it out on your company's network -- that might be an RGE (Resume-Generating Event) if someone finds you do it and thinks that you've got an untoward purpose -- in other words, using a null session to suck out a list of users and shares just might get you fired. Instead, try it out on a domain that you've got authorization to hack, like a test network or virtual machine network.

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

Start with two systems, Victim and Villain. Victim can be a standalone machine or a domain controller -- you'll see different results between the two, and it's interesting to try it out both ways. You'll also see big differences in behavior between NT 4, 2000, XP and 2003.

Recall that we want to simulate a situation wherein Victim and Villain would normally not communicate with one another, so to be sure that a test doesn't succeed where it might normally fail then ensure that

- Victim and Villain are not in the same domain, and
- the user name and password that you'll use from Villain doesn't match a user name and password on Victim. For example, if you log on as Administrator at Villain then ensure that the Administrator account on Victim doesn't have the same password. If the two Administrator passwords *do* match then Villain will use that fact to log onto Victim... which would defeat the whole purpose of exploring the null session's powers.

Assuming that Villain can resolve the name "Victim," try a

```
net view \\victim
```

To ask Victim to display its shares. You should get an error like "System error 5 has occurred. Access is denied." This makes sense, as you're basically just a stranger asking for a list of shares.

Now create a null session with Victim like so:

```
net use \\victim\ipc$ /u:"" ""
```

Note the syntax -- follow the normal NET USE command with /u:"" followed by a space and then "", another pair of quotes. That says "log me on with an empty username and password" -- the sigil of the Anonymous User. You will probably see the response "The command was completed successfully." which means that you've established a null session or anonymous login. (If, on the other hand, you get the "System error 5" error message, then someone has secured Victim in some way against anonymous logins -- kudos!)

(Side note: without first creating a null session, try a net view /domain:domainname. For some reason that seems to work no matter how much I restrict the null session, oddly. Apparently anyone can get a list of the machines on a domain.)

Assuming the null session worked, try a net view \\victim command once more, and you'll get a list of the shares on that system. But what else can we see? Well, to really bang on a null session you need the all-purpose null session tool "enum.exe;" you can find it at http://razor.bindview.com/tools/desc/enum_readme.html. Unfortunately it's packaged as a zipped TAR file, a common format in the Unix/Linux world for transmitting and compressing a group of files but not so common a format in the Windows world. You can, however, open the file with any recent version of PKZip. Inside you'll find a file named enum.exe, that's what you need. Run it from a command line to try to get Victim's list of users, machines in its workgroup, shares, password policy information, groups, and trusted domains. Try running this from Villain:

```
enum -U -M -S -P -G -L victim
```

When run against a basic NT 4 or 2000 system, enum gets a fair amount of information:

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

```
C:\>enum -U -M -S -P -G -L nt4basesystem
server: nt4basesystem
setting up session... success.
password policy:
  min length: none
  min age: none
  max age: 42 days
  lockout threshold: none
  lockout duration: 30 mins
  lockout reset: 30 mins
opening lsa policy... success.
server role: 3 [primary (unknown)]
names:
  netbios: NT4BASESYSTEM
  domain: WORKGROUP
quota:
  paged pool limit: 33554432
  non paged pool limit: 1048576
  min work set size: 65536
  max work set size: 251658240
  pagefile limit: 0
  time limit: 0
trusted domains:
  indeterminate
netlogon done by a PDC server
getting user list (pass 1, index 0)... success, got 2.
  Administrator Guest
enumerating shares (pass 1)... got 5 shares, 0 left:
  ADMIN$ IPC$ stuff C$ Z$
getting machine list (pass 1, index 0)... success, got 0.
Group: Administrators
NT4BASESYSTEM\Administrator
Group: Backup Operators
Group: Guests
NT4BASESYSTEM\Guest
Group: Power Users
Group: Replicator
Group: Users
cleaning up... success.
```

Running it on an unmodified XP or 2003 system (save for security patches), however, yields a lot less information -- just a bunch of "access denied" messages.

Why null sessions exist in the first place

Now, when I first read about the null session back in the NT 4.0 SP3 days, I freaked out. What, I thought, is the point of having a secure operating system with all kinds of specific lists of permissions - including a Read permission -- when it then just goes and ignores any existing permissions, allowing anyone in the world to look around my domain's insides?

The answer is that it apparently makes doing a handful of things in Microsoft networking easier for the Microsoft programmers. The classic example involves two NT 4 domains with a single one-way trust between them -- call them MASTER and RESOURCE. RESOURCE trusts MASTER, but MASTER does not trust RESOURCE.

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

Now let's suppose I'm a domain administrator of RESOURCE. There's a global group on MASTER called TRAVELERS (MASTER\TRAVELERS, more correctly) that I want to give Full Control to a share on a server in my domain. So I sit at this server, bringing up the Access Control List to the share. I click Add and would like to pick MASTER\TRAVELERS from a list of possible global groups in the MASTER domain...

... and that's where I get in trouble.

Remember, RESOURCE trusts MASTER, but not the other way around. So when the RESOURCE server that I'm sitting at asks a domain controller for MASTER to cough up the list of global groups in the MASTER domain, then the MASTER DC says "yeah, who's asking?" or, in NT-ese, "can you log on please first, so I can figure out whether or not to agree to your request?" But as MASTER doesn't trust RESOURCE, the MASTER DC doesn't want anything to do with any SIDs from RESOURCE, and so it's plainly impossible for me to log on, and so retrieving the list of global groups seems impossible.

The answer was to set up NT so that it'd reveal some of its inside info to anyone who asked. That's how the RESOURCE server gets the list of MASTER global groups. (Or users, for that matter.) Disabling the null session -- you can to a certain extent, and I'll show you how in a bit -- would make it impossible for that RESOURCE administrator to do his job.

(But here's the part of the null session story that confounds me. Why create this barn-door-sized backdoor? Why not simply modify NT so that it'll reveal group and user lists to trusting domains? It may be that I'm missing something, but this really sort of smells of "golly, it's 4 PM Friday, I purchased the airline tickets months ago and I am *going* on that vacation, dagnabbit." Merely looking at a security problem and saying "I guess we'll have to loosen things up a bit" doesn't seem like a license to just throw away security.)

Nor is that the only case of "we need information even if we can't log on." Any Windows 9x system trying to retrieve a browse list from an NT, 2000, XP or 2003-based browse master would lack any credentials and therefore would be unable to request the browse list, and so Network Neighborhood on the 9x systems would be empty. So Microsoft modified the NT family (NT 3.x, 4, 2000, XP, and 2003) to permit anonymous null session users to request and then get browse lists. If you chose to restrict null session access in your network, therefore, then the Browser -- Network Neighborhood -- wouldn't work in some situations, particularly situations involving (by some reports) NT4 or Win 9x systems browsing a domain. Now, *that* might get the attention of your users! Worse yet, plenty of applications over the years have come to depend on the Browser's existence. For example, BackupExec users will know that BE lets you back up remote systems -- you can run BE on Server 1 and tell it to back up Server 2 just as if the tape drive were sitting on Server 2. But BackupExec versions 8.5 and earlier will completely fail when trying to back up a remote server if that server has restricted null session access.

That has led to some strange workarounds for those wanting to restrict null sessions. One network administrator solved The Case Of the Empty Network Neighborhood by forcing all of his NT-based OS systems (NT 4, 2000, XP, 2003) *not* to be browse masters so that the Windows 9x systems (who never give a hoot *who* they tell about the browse list) become his browse masters!

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

Restricting null sessions

As Windows became popular, hackers noticed the null session and used it to create a variety of annoying tools like RedButton, which would use a null session to identify your system's local Administrator account even if you *did* rename it -- but again it couldn't crack its password -- and so Microsoft changed the NT family to allow you to dial back the anonymous user's powers.

First, there's the absolute "null killer:" block ports 139 and 445. Null sessions just plain can't happen without them.

On NT 4 running SP3 or later, Microsoft added a Registry entry RestrictAnonymous, a REG_DWORD entry that goes in HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA. It takes two values, 0 or 1. 0 is the default and leaves NT 4 in the out-of-the-box -- relatively open -- state. Set RestrictAnonymous to 1 and reboot the system, however, and you'll find the NT 4 system considerably less generous with information for anonymous visitors; an enum gets a small amount of data, but not much. A NET USE for a null session works -- "the command completed successfully" -- but doesn't yield anything beyond an "access denied" when asked for information on shares.

On Windows 2000, Microsoft redefined RestrictAnonymous. Now it has three possible values, zero to two. Under Windows 2000, 0 means no restrictions on the null session, as before. But 1's been redefined. 2 now does what 1 used to do -- it basically shuts down most access for null sessions. the "new 1," in contrast, only keeps null sessions from seeing the list of users and shares.

Microsoft also exposes this Registry entry through Group Policies -- look in Computer Configuration / Windows Settings / Security Settings / Local Policies / Security Options and the first entry is labeled "Additional restrictions on anonymous connections." It offers three values: "None. Rely on default permissions," "Do not allow enumeration of SAM accounts and shares," and "No access without explicit anonymous permissions." You've probably already noticed that they correspond exactly to values 0, 1 and 2 for the Windows 2000 version of RestrictAnonymous.

Windows 2000 has an advantage over NT 4 in its RestrictAnonymous in another way as well; if you set RestrictAnonymous using the Group Policy Editor then you needn't reboot the system to see the change take effect; a simple "secdit /refreshpolicy machine_policy" will bring about the RestrictAnonymous change immediately.

XP and 2003 offer five group policies, increasing your degree of control over the anonymous users out there:

- Network Access: Allow anonymous SID/Name translation
- Network Access: Do not allow anonymous enumeration of SAM accounts
- Network Access: Do not allow anonymous enumeration of SAM accounts and shares
- Network Access: Let Everyone permissions apply to anonymous users
- Network Access: Named Pipes that can be accessed anonymously

The first, Allow anonymous SID/Name translation, puts a bullet in RedButton's head. (Well, so did RestrictAnonymous=1 in NT days, but this one's more targeted, so to speak.) RedButton and similar tools were able to find your system's Administrator account because the SID of the Administrator account is fixed. You see, every SID in your domain looks like "S-1-5-21-X-Y-Z-RID," where X, Y, and Z are 32-bit numbers specific to your domain or SAM. If your domain's X equaled 23, Y was 88 and Z 900, then that would mean that every single SID in your domain would be S-1-5-21-23-88-900-

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

something. The only difference between your account and mine would be the "something," a 32-bit value called the *relative* ID or RID. Where this gets interesting is that the Administrator account *always* has the same RID: 500. It's relatively (no pun intended) easy to get X, Y and Z for a domain; stick a 500 on the end and you have the SID of the Administrator account. Anonymous folks can then ask, "what's the user name for S-1-5-21-23-88-900-500?" and by default 2000 and NT will promptly reply. Turning off SID/name translation makes that impossible on a 2003 system.

The second and third are just more specific restrictions on anonymous users; where 2000 only let you bar anonymous folks from seeing both user accounts and shares or let them see both, 2003 offers you the option to let outsiders shares but not user accounts.

The fourth one's nitroglycerine: "Let Everyone permissions apply to anonymous users." Fortunately, 2003 disables it by default. You probably know the Everyone group, which contains every user account in the domain as well as user accounts for all trusted domains. That's a pretty large group, so giving it access to *anything* is a little scary. That's prompted many administrators to yank the Everyone group from things. Anyway, this policy makes things much worse; when set, it lets anyone anonymously logging in -- any null session user -- to act as a member of the Everyone group.

Finally, developers often need to allow one program to talk directly to another program without using intermediate files. So the NT family has always had the notion of a "named pipe," something that I first saw in OS/2. The idea is (roughly -- I'm not a programmer and us VBscript retards don't get to use named pipes) that Program A creates a named pipe in a manner very similar to creating a file. Program B, which is designed to talk directly to Program A, connects to that named pipe (by name, not surprisingly) and can then send data to program A by simply "writing" to the pipe, much as it would write to a file.

The operating system and OS applications make use of what might be called "well-known named pipes," and named pipes have permissions on them just as files and directories do. This policy lets you grant anonymous users the ability to access particular named pipes. By default null sessions do *not* have access to named pipes; this policy changes that for seven named pipes (comnap, comnode, spoolss, epmapper, locator, trkwks, and trksvr). As I'm not a coder I can't offer much insight here, *except* to say that the one named pipe that you probably do not want to grant access to is one called "winreg," which grants access to your Registry.

By the way, if you ever want to see what named pipes are running on your system, visit www.sysinternals.com and download "pipelist.exe." Again the good Dr. Russinovich comes to our aid!

What breaks when you restrict anonymous access

So what settings should you use? Well, that, umm, depends. Clearly it'd be nice to set RestrictAnonymous to 1 in an NT 4 network, "Additional restrictions on anonymous connections" to "No access without explicit anonymous permissions" on a 2000-based network, and then in a 2003-based network

- Set "Network Access: Allow anonymous SID/Name translation" to Disabled
- Set "Network Access: Do not allow..." both to Enabled
- Set "Network Access: Let Everyone permissions apply to anonymous users" to Disabled

All About Null Sessions or Anonymous Logons

By Mark E. Donaldson

And who knows, that might even work. But it may not because so many things in the Microsoft world depend on null sessions. It seems that basically that over the years when Microsoft has faced a number of "how do we get this to work?" security problems then they've solved them by simply given some new powers to the null session. So if we decide to take away some of those powers then we've got to be prepared to either cook up some workarounds, or simply accept that some things just plain won't work any more.

No one seems to have a complete list of things that don't work with null session restrictions and, just to make things more complex, there are varying levels of restrictions. So the cardinal rule here is **test it and test it again**. I wish I had the ability to tell you that such-and-such will definitely work and so-and-so won't. But here's the list of things that have seen problems:

- In general, Microsoft seems pretty busy finding things that depend on null sessions and getting rid of them, but the "getting rid of" part only applies to their latest OSes. Turning on all of the anti-null stuff in version X seems to work if your clients are all running version X, but often causes trouble with anything running X-1 or earlier systems.
- You can't change a password from a Windows system if the password has expired.
- Macintosh users can't change passwords at all.
- Trusting domains, as I've explained, can't establish a connection with their trusted domains, or so many say -- that didn't match my experience.
- Migration tools seem to need pretty loose null connections; in particular the SID/name translation seems essential.

Here's what I found when testing. I created a small network with Windows Server 2003-based Active Directory domain controllers. I then set the domain policies as I suggested above, disabling SID/name translation and Everyone permissions, and then enabling the two "do not allow..." policies. Then I joined Windows XP, 2000 and NT 4 systems to that domain. I was surprised to find that they all joined without trouble and had no logon problems. I was even able to build a two-way trust between an NT 4 domain and the AD domain. I suspect this was easy because I was working with 2003; I get the impression that Microsoft heard from a lot of Windows 2000 admins that they really wanted to restrict anonymous access but that it broke too many things, so perhaps 2003 works a mite better with NT 4 domains.

Here's the bottom line. If your network is sitting behind a firewall that blocks 139 and 445 then you don't have nearly as much to worry about as a network that's not firewalled. But even a firewalled network has to worry about hacks from insiders, so no matter how your network's set up it's worth looking into restricting null session access. Start from this article, build a small test network and find out how much null restriction you can tolerate. Best of luck and please, I'd love to hear what's working and not working for you!