Problems with PSQL and Windows 10 Release 1803

A White Paper From

For more information, see our web site at
http://www.goldstarsoftware.com

Problems with PSQL and Windows 10 Release 1803

Last Updated: June 26, 2018 (See Last Page)

In April, Microsoft released a new version of Windows 10, known as Release 1803, and this release has been slowly rolling out to users over the last few weeks. This particular release of the popular operating system has been particularly rife with issues, and we do recommend putting off this release as long as possible, in the hopes that they will be fixing some of the bigger issues.

One issue is particularly nasty – this release can prevent database applications, such as those built on PSQL, to fail to make any outbound network connections if they are loaded from a remote server. This paper explains the problem in more detail, and provides several possible workarounds.

Symptoms of the Problem

Users will generally start to complain when they launch an application (after upgrading to release 1803) and the application fails. With PSQL applications, this can cause Status 3103, 94, or other errors. Some applications may post these status codes, but others may simply indicate that they cannot contact the database. Other applications may simply refuse to do anything or may crash completely. The wide variety of possible symptoms makes this issue particularly difficult to resolve.

The Root Cause of the Problem

The real cause of the problem is a change in the new release – possibly in the OS itself, but more likely in the Windows Defender Firewall – that prevents certain applications from opening up ANY network connections. However, it is much more complicated than this simple description.

To better understand this issue, we have to discuss some networking acronyms first. In older versions of Windows networking, Microsoft used a communications protocol for file sharing called Server Message Block, or SMB. The SMB protocol is the only option for systems running on Windows XP, Windows Server 2003, or older. With Windows Vista and Windows Server 2008, Microsoft "improved" the SMB protocol, adding some features including directory entry caching, and called it SMB2. Sadly, the caching feature was slightly less than stable, and many applications had problems with this particular implementation, causing many users to eschew the SMB2 protocol and disable it on their systems. Fast forward to Windows 8 and Windows Server 2012, and Microsoft released SMB3, which addressed those caching issues, making SMB3 much more reliable.

Now, with the progress of the OS, each version of Windows has its own "base" version of SMB, but the systems are typically backward-compatible, meaning that if a Windows 8 machine talks to a Windows Server 2012 machine, they will use SMB3, but if it talks to a Windows Server 2008 machine, it will use SMB2 instead. Along that same line, a Windows XP workstation (which ONLY understands SMB) will use SMB to talk to any server OS, and a machine running ANY OS will only be able to use SMB to talk to a Windows Server 2003 machine. Until now, this hasn't caused any issues.

For reasons not yet known – it could be a simple bug, closing a security hole, or simply a desire to force users to upgrade to a new operating system – Microsoft is now detecting when an application executable file is loaded across an SMB file sharing connection. When such an event occurs, the executable program is completely blocked from opening up ANY network sockets. As you can

imagine, network connections are used by database applications to communicate with the database engine. Network connections are ALSO used to contact other services, such as fax services, Email systems, web sites, and more – much of what we use computers for today!

Note that this issue does NOT arise if you are loading the EXE from an SMB2 or SMB3 network connection, or if you are running the EXE from a local hard disk! This means that this issue will impact users with older file servers (or NAS devices) running applications that are exclusively stored on the file server and run across the network. (A majority of older, network-based applications do exactly this!) Because of the specificity of the issue, it can seem that some applications work just fine, while others strangely fail for no other obvious reason.

Confirming That You Have the Problem

Obviously, this issue ONLY applies to Windows 10 1803 for right now, but this could change in the future. What makes it difficult to diagnose is that there is no "easy" way to see what protocol you are using to load an EXE. One way to check is to download a tool called Wireshark (www.wireshark.org) , grab a network capture while running the EXE file, and look for "SMB" as the protocol.

However, there is an easier way. Goldstar Software has built a simple test application called Test1803 that attempts to open up a network socket and displays an error if it fails. You can download this tool from this web page:

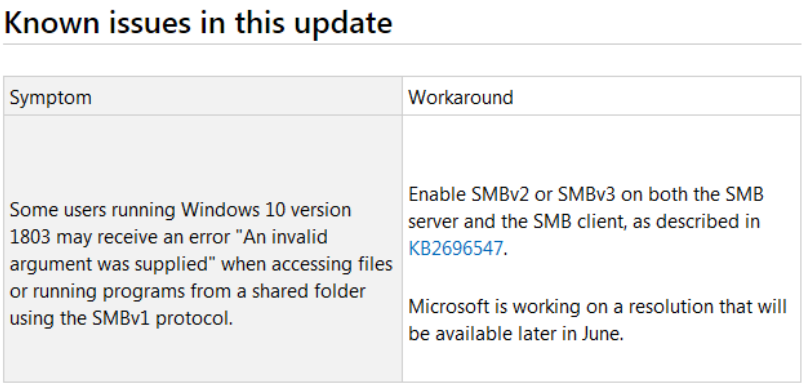
http://www.goldstarsoftware.com/tools.asp

To use it, simply open up the ZIP file and copy the TEST1803.EXE file over to your file server where your failing application resides. The, from a command prompt on your Windows 10 machine, run the TEST1803.EXE application from the server. You will get an instant notification as to whether you can open a socket or not.

Working Around the Problem

Once you have confirmed that your system is impacted by this problem, you have to find a solution. Here are several possible workarounds and solutions that we've found thus far.

1. **Wait for Microsoft to Fix It**: Since we don't know yet if this is a bug or an intentional change, you could try holding your breath and hoping that it goes away. However, you are probably reading this because your application is failing, so this likely won't be viable. As of June 12, 2018, Microsoft has confirmed this issue (https://support.microsoft.com/en-nz/help/4284835) and is supposed to be working on a solution:



Known issues in this update

Symptom	Workaround
Some users running Windows 10 version 1803 may receive an error "An invalid argument was supplied" when accessing files or running programs from a shared folder using the SMBv1 protocol.	Enable SMBv2 or SMBv3 on both the SMB server and the SMB client, as described in KB2696547. Microsoft is working on a resolution that will be available later in June.

2. **Roll Back to Release 1709**: The previous Windows 10 update did not exhibit this issue, so rolling back your system to a previous version (and delaying the update indefinitely) is a simple option. Sadly, Microsoft does not allow you to delay updates forever, and it may slip it in again when you are not paying attention, so you may have to roll back several times.
3. **Upgrade Your File Server**: If you upgrade the server on which the shared application is installed to a newer OS (that supports SMB2 or newer), then you'll be OK. This option can be quite expensive and time consuming, though, and is not a good "quick fix", but it is certainly recommended by Microsoft, who would love to have more licensing dollars.
4. **Enable SMB2**: If you are already on Windows Server 2008 or newer, but you have disabled SMB2 because of the caching options, try enabling it again. Note, though, that this may just change you back to the cache issues from yesteryear. Better yet, make sure that ALL of your workstations are Windows 8+ and that you have Server 2012+, and stick with SMB3.
5. **Move Your Application**: If you move your application to a different location – either on a newer file server, NAS, or even to a local hard disk volume – then you can easily avoid the problem.
6. **Replace Windows Defender Firewall**: This is the most interesting option, and indicates that the issue may be with Defender instead of the core OS itself. Simply install a third-party firewall solution, and the Defender firewall will be disabled. (Some have reported that Avast Free AntiVirus is a good option.) Note that if you have Defender AV disabled, but Defender Firewall ENABLED, then you may still experience the issue. It is also important to note that just disabling the Defender firewall is NOT enough!

A Possible Fix?

On June 26, 2018, Microsoft released a new Windows 10 operating system, Build 17134.137, which is available through patch KB4284848. A full description of this update can be found at https://support.microsoft.com/en-nz/help/4284848/windows-10-update-kb4284848, but the important change appears to be this one:

- Addresses an issue where some users may receive an error when accessing files or running programs from a shared folder using the SMBv1 protocol. The error is "An invalid argument was supplied".

Initial reports from users indicates that this issue IS FIXED after applying the patch! You can use Windows Update to apply these fixes, or you can download the 600+MB update manually from http://www.catalog.update.microsoft.com/Search.aspx?q=KB4284848 if you prefer.

I have not yet tested this myself, but hope to do that in the next few days as time allows.

Finding More Help

If you have other problems getting this to work, I recommend first contacting Microsoft to express your displeasure in this change. However, that isn't likely to help get you up and running right away. If you need some additional hand-holding, Goldstar Software may be able to assist you as well. You can contact us at 1-708-647-7665 or via the web at http://www.goldstarsoftware.com.

Appendix A: Source Code for Test1803

In the interest of full disclosure, I have included the source code for Test1803 here, which was built with Microsoft Visual Studio 2013. If you don't trust running an application from a third party, then you can certainly build your own testing tool using these same functions.

```
// Test1803.cpp : Defines the entry point for the console application.
//

#include "StdAfx.h"
#pragma comment( lib, "wsock32.lib" )

int _tmain(int argc, _TCHAR* argv[])
{
	WSADATA WsaData;

	printf("Starting my test run now....\n");
	WSAStartup(MAKEWORD(2, 2), &WsaData);
	int sockethandle = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
	if (sockethandle > 0)                                  //Only continue if no error!
		printf("Call to socket succeeded! You are OK running database applications from this location.\n.",
WSAGetLastError());
	else
		printf("Call to socket failed with Status %d. You should not run database applications from this
location.\n", WSAGetLastError());
	WSACleanup();
	return 0;
}
```