

Powershell Invoke-WebRequest Fails with SSL/TLS Secure Channel

Asked 8 years, 2 months ago Modified 4 years ago Viewed 467k times



I'm trying to execute this powershell command

417

```
Invoke-WebRequest -Uri https://apod.nasa.gov/apod/
```



and I get this error. *"Invoke-WebRequest : The request was aborted: Could not create SSL/TLS secure channel."* https requests appear to work ("<https://google.com>") but not this one in question. How can I get this to work or use other powershell command to read the page contents?



powershell

ssl

Share Improve this question Follow

edited Nov 30, 2017 at 4:51



polina-c

7,087

6

31

42

asked Jan 12, 2017 at 16:52



hewstone

4,715

2

24

24

2 See also [Default SecurityProtocol in .NET 4.5](#). – Franklin Yu Sep 20, 2018 at 15:41

7 Answers

Sorted by: Highest score (default)



try using this one

861

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12  
Invoke-WebRequest -Uri https://apod.nasa.gov/apod/
```



Share Improve this answer Follow



answered Jan 12, 2017 at 17:03



Chandan Rai

10.4k

2

21

29

68 By default powershell uses TLS 1.0 the site security requires TLS 1.2 – Chandan Rai Jan 12, 2017 at 17:05

3 Awh, How did you determine the TLS version of the site? – hewstone Jan 12, 2017 at 17:08

- 17 Try [SSL Labs](#) to determine information: [SSL Report: apod.nasa.gov](#) shows TLS1.1 and TLS1.2 – [Christopher G. Lewis](#) Jan 12, 2017 at 18:00
- 3 @Brandon Change it in `$env:Profile`, or better yet, [edit registry table](#). – [Franklin Yu](#) Sep 23, 2018 at 5:15
- 7 Sorry Microsoft, but what year is it? [The PCI Council suggested that organizations migrate from TLS 1.0 to TLS 1.1 or higher before June 30, 2018. In October 2018, Apple, Google, Microsoft, and Mozilla jointly announced they would deprecate TLS 1.0 and 1.1 in March 2020.](#) – [KCD](#) May 9, 2019 at 2:36



234



In a shameless attempt to steal some votes, `SecurityProtocol` is an Enum with the `[Flags]` attribute. So you can do this:

```
[Net.ServicePointManager]::SecurityProtocol =
    [Net.SecurityProtocolType]::Tls12 -bor `
    [Net.SecurityProtocolType]::Tls11 -bor `
    [Net.SecurityProtocolType]::Tls
```

Or since this is PowerShell, you can let it parse a string for you:

```
[Net.ServicePointManager]::SecurityProtocol = "tls12, tls11, tls"
```

Then you don't technically need to know the TLS version.

I copied and pasted this from a script I created after reading this answer because I didn't want to cycle through all the available protocols to find one that worked. Of course, you *could* do that if you wanted to.

Final note - I have the original (minus SO edits) statement in my PowerShell profile so it's in every session I start now. It's not totally foolproof since there are still some sites that just fail but I surely see the message in question much less frequently.

Share Improve this answer Follow

edited Mar 15, 2018 at 13:07


answered Dec 30, 2017 at 3:00



No Refunds No Returns

8,356 4 34 44

- 10 If you need to access a site that uses SSLv3, then you'll want `[Net.ServicePointManager]::SecurityProtocol = "Tls12, Tls11, Tls, Ssl3"`. Remember that SSLv3 and TLSv1.0 have been deprecated due to POODLE, so use at your own risk. – [jordanbtucker](#) Mar 1, 2018 at 19:43
- 1 isn't there a way to use reflection to just allow all `Net.SecurityProtocolType` types? there has to be – [red888](#) Feb 7, 2020 at 21:00
- 1 Why do you want to allow known-flawed protocols access by default? Regardless, I believe the imminent (as of this writing) release of PowerShell V7 will address this issue once and for all and this question will slowly fade into its rightful and well-earned oblivion. – [No Refunds No Returns](#) Feb 12, 2020 at 2:26

- 2 In a Dockerfile on Windows use RUN [Net.ServicePointManager]::SecurityProtocol = 'tls12, tls11, tls' ; single apex – [Riccardo Bassilichi](#) Apr 8, 2021 at 8:48 
-

**36**

The cause of the error is Powershell by default uses TLS 1.0 to connect to website, but website security requires TLS 1.2. You can change this behavior with running any of the below command to use all protocols. You can also specify single protocol.



```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls,  
[Net.SecurityProtocolType]::Tls11, [Net.SecurityProtocolType]::Tls12,  
[Net.SecurityProtocolType]::Ssl3  
[Net.ServicePointManager]::SecurityProtocol = "Tls, Tls11, Tls12, Ssl3"
```

After running these commands, try running your command:

```
Invoke-WebRequest -Uri https://apod.nasa.gov/apod/
```

then it will work.

[Share](#) [Improve this answer](#) [Follow](#)

answered Mar 13, 2021 at 13:27



[Sidrah Madiha Siddiqui](#)

1,374 18 17

If, like me, none of the above quite works, it might be worth also specifically trying a lower TLS version alone. I had tried both of the following, but didn't seem to solve my problem:

8

```
[Net.ServicePointManager]::SecurityProtocol = "tls12, tls11, tls"
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 -
bor [Net.SecurityProtocolType]::Tls11 -bor [Net.SecurityProtocolType]::Tls
```

In the end, it was only when I targetted TLS 1.0 (specifically remove 1.1 and 1.2 in the code) that it worked:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls
```

The local server (that this was being attempted on) is fine with TLS 1.2, although the remote server (which was previously "confirmed" as fine for TLS 1.2 by a 3rd party) seems not to be.

Hope this helps someone.

Share Improve this answer Follow

edited Aug 9, 2019 at 11:20

answered Aug 9, 2019 at 10:45

 **Mark-DG1**
246 3 8

It works for me...

1

```
if (-not
([System.Management.Automation.PSTypeName]'ServerCertificateValidationCallback').Type
{
    $certCallback = @"
        using System;
        using System.Net;
        using System.Net.Security;
        using System.Security.Cryptography.X509Certificates;
        public class ServerCertificateValidationCallback
        {
            public static void Ignore()
            {
                if(ServicePointManager.ServerCertificateValidationCallback
==null)
                {
                    ServicePointManager.ServerCertificateValidationCallback +=
                        delegate
                        (
                            Object obj,
                            X509Certificate certificate,
                            X509Chain chain,
                            SslPolicyErrors errors
                        )
                        {
                            return true;
                        }
                }
            }
        }
    "@
    [Type]::GetTypeFromResource($certCallback)
}
```

```

    }
}
"@
    Add-Type $certCallback
}
[ServerCertificateValidationCallback]::Ignore()

Invoke-WebRequest -Uri https://apod.nasa.gov/apod/

```

Share Improve this answer Follow

edited Apr 2, 2019 at 12:09

answered Apr 2, 2019 at 11:18



Gowtham Balusamy

752 10 22

- 5 This answer effectively disables security checking. While it may eliminate the error it opens your device attack surface in ways that many would consider unacceptable. I would never use this on a device that I own. – [No Refunds No Returns](#) Oct 27, 2019 at 10:49

If you needed this it to make a single request, how would you reset security after the call? – [Bevan](#) Feb 8, 2023 at 7:17



Make sure you switch the SHELL first:

1

```

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop';
$ProgressPreference = 'SilentlyContinue';"]

```



```

RUN [Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12

```



```

RUN Invoke-WebRequest -UseBasicParsing -Uri 'https://github.com/git-for-
windows/git/releases/download/v2.25.1.windows.1/Git-2.25.1-64-bit.exe' -OutFile
'outfile.exe'

```



Share Improve this answer Follow

answered Mar 5, 2020 at 6:04



Michael Kang

52.9k 16 106 138

- 1 Are you suggesting this extra step as a way to protect the original PowerShell session from using weaker versions of the protocol? – [Sam Erde](#) Apr 21, 2021 at 19:45

Thanks. Now it builds on my local machine (Windows 10). "Funny" thing: Docker image was built without this fix on Azure DevOps (windows-2019 agent) – [Saibamen](#) Mar 29, 2022 at 9:11 ✎



I haven't figure out the reason but reinstalling the .pfx certificate(both in current user and local machine) works for me.

0



Share Improve this answer Follow

answered Apr 17, 2019 at 6:48


 **Spencer**
59 1 11



which .pfx did you reinstall? – [No Refunds No Returns](#) Oct 27, 2019 at 10:50

@NoRefundsNoReturns The certificate file which you want to use to send the request. – [Spencer](#) Oct 28, 2019 at 3:20

1 I'm still not clear how this is applicable to the question. – [No Refunds No Returns](#) Nov 7, 2019 at 17:43

@NoRefundsNoReturns When I `Invoke-WebRequest` locally, it works at first but will fail later. It seems like sometimes it cannot read the cert any more(I don't know the mechanism behind it. Maybe it's a setting controlled by the company). But reinstalling the cert works in this case. – [Spencer](#) Nov 11, 2019 at 5:02 

If you are losing the private key for a certificate that can certainly make the cert unusable but I doubt that's the problem here. If you are losing the key, then you need to make sure you are saving the private key and marking it persistent. Ask a new question if this is what is happening. – [No Refunds No Returns](#) Nov 20, 2019 at 20:06



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

powershell ssl

See similar questions with these tags.