Which versions of SSL/TLS does System.Net.WebRequest support?

Asked 10 years, 2 months ago Modified 8 years ago Viewed 143k times

82

Now that SSL 3 has been found to be vulnerable to the POODLE attack:

Which versions of SSL/TLS does System.Net.WebRequest use when connecting to any https Uri?

I use WebRequest to connect to several 3rd party API's. One of these has now said they will block any request that uses SSL 3. But WebRequest is part of the .Net core framework (using 4.5) so it is not obvious what version it uses.

c# ssl .net-4.5 webrequest poodle-attack

Share

Improve this question

Follow

edited Mar 23, 2015 at 22:21

Colonel Panic

137k • 96 • 416 • 478

asked Oct 16, 2014 at 21:43

JK.

21.8k • 35 • 138 • 219

Will the WebRequest fallback to SSL 3 if the server (or man in the middle) requests it? – Philip Pittle Oct 16, 2014 at 22:04

If I'm not mistaken, I think WebRequest looks at the security certificate on the server you are trying to pull up and uses whatever encryption schema is on the web server. I don't think you need to change anything on the client side. – Icemanind Oct 16, 2014 at 22:22

1 I would hope that the owners of the API that have just said they will not support SSL3 have thought to make sure their server does not request that the client use SSL3 :) – JK. Oct 17, 2014 at 0:54

@iceman any way to tell from a cert if it supports SSL3 or not? I don't see any properties related to versions. – JK. Oct 17, 2014 at 1:06

@JK. - I'm not sure about other browsers, but I'm using Chrome. In Chrome, if you go to an HTTPS website (like Wellsfargo.com for example), you'll see a green lock near the URL. Click on that lock, then click on the connection tab, and it'll show you. In the case of wellsfargo.com, Chrome says "The connection uses TLS 1.0". IE and other browsers should show the same similar thing. – Icemanind Oct 17, 2014 at 1:54

3 Answers

Sorted by: Highest score (default)

53

When using System.Net.WebRequest your application will negotiate with the server to determine the highest TLS version that both your application and the server support, and use this. You can see more details on how this works here:

http://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_handshake

If the server doesn't support TLS it will fallback to SSL, therefore it could potentially fallback to SSL3. You can see all of the versions that .NET 4.5 supports here:

http://msdn.microsoft.com/en-us/library/system.security.authentication.sslprotocols(v=vs.110).aspx

In order to prevent your application being vulnerable to POODLE, you can disable SSL3 on the machine that your application is running on by following this explanation:

https://serverfault.com/questions/637207/on-iis-how-do-i-patch-the-ssl-3-0-poodle-vulnerability-cve-2014-3566

Share Improve this answer

Follow

edited Apr 13, 2017 at 12:13

Community Bot

1 ● 1

answered Oct 17, 2014 at 19:29

Gareth Williams

607 ● 6 ● 4

21 I would suggest that to make *outbound* requests like WebRequest and HttpWebRequest safe, you should follow steps to disable SSL fallback as detailed in my question at stackoverflow.com/questions/26389899/…. Basically you run this line of code first: **System.Net.ServicePointManager.SecurityProtocol =**

SecurityProtocolType.Tls | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls12; As I understand it, the registry fix detailed in this answer only applies to inbound connections. – Jordan Rieger Feb 16, 2015 at 20:17

1 Note that you only have the option of `SecurityProtocolType.Tls` (and therefore TLS 1.0) in .net framework 3.5 or lower – James Westgate Oct 15, 2015 at 11:45

Could I get higher TLS support by ditching System.Net.WebRequest and using another library such as RestSharp or is it the actual .net framework that needs to be upgraded? – The Muffin Man Nov 4, 2015 at 2:04

I believe this is the actual .net framework that needs to be upgraded. www.passionatecoder.ca – Ehsan Jan 6, 2016 at 20:54

1 @JamesWestgate I was under the impression that `TLS` was something that would actually be implemented at the OS/machine level? Isn't that where the actual HTTP traffic is occuring? So I don't think it makes sense to try to tell your code to use a particular TLS that's really determined by the OS. – Don Cheadle Mar 8, 2016 at 22:45

53

This is an important question. The SSL 3 protocol (1996) is irreparably broken by the Poodle attack published 2014. The IETF have published "SSLv3 MUST NOT be used". Web browsers are ditching it. Mozilla Firefox and Google Chrome have already done so.

Two excellent tools for checking protocol support in browsers are SSL Lab's client test and

https://www.howsmyssl.com/ . The latter does not require Javascript, so you can try it from .NET's HttpClient:

```
// set proxy if you need to
// WebRequest.DefaultWebProxy = new WebProxy("http://l

File.WriteAllText("howsmyssl-httpclient.html", new
HttpClient().GetStringAsync("https://www.howsmyssl.com

// alternative using WebClient for older framework ver
// new WebClient().DownloadFile("https://www.howsmyssl
webclient.html");
```

The result is damning:

> Your client is using TLS 1.0, which is very old, possibly susceptible to the BEAST attack, and doesn't have the best cipher suites available on it. Additions like AES-GCM, and SHA256 to replace MD5-SHA-1 are unavailable to a TLS 1.0 client as well as many more modern cipher suites.

That's concerning. It's comparable to 2006's Internet Explorer 7.

To list exactly which protocols a HTTP client supports, you can try the version-specific test servers below:

```
var test_servers = new Dictionary<string, string>();
test_servers["SSL 2"] = "https://www.ssllabs.com:10200
test_servers["SSL 3"] = "https://www.ssllabs.com:10300
test_servers["TLS 1.0"] = "https://www.ssllabs.com:103
test_servers["TLS 1.1"] = "https://www.ssllabs.com:103
```

```
test_servers["TLS 1.2"] = "https://www.ssllabs.com:103

var supported = new Func<string, bool>(url =>
{
    try { return new HttpClient().GetAsync(url).Result
    catch { return false; }
});

var supported_protocols = test_servers.Where(server =>
supported(server.Value));
Console.WriteLine(string.Join(", ", supported_protocol
```

I'm using .NET Framework 4.6.2. I found HttpClient supports only SSL 3 and TLS 1.0. That's concerning. This is comparable to 2006's Internet Explorer 7.

Update: It turns HttpClient does support TLS 1.1 and 1.2, but you have to turn them on manually at `System.Net.ServicePointManager.SecurityProtocol`. See https://stackoverflow.com/a/26392698/284795

I don't know why it uses bad protocols out-the-box. That seems a poor setup choice, tantamount to a major security bug (I bet plenty of applications don't change the default). How can we report it?

Share Improve this answer

Follow

edited Oct 7, 2021 at 7:34

Community Bot

1 • 1

answered Mar 23, 2015 at 22:13

Colonel Panic

137k • 96 • 416 • 478

Which ***protocol*** *must I use* ? – Kiquenet Apr 12, 2018 at 18:20

Are there any servers that test for TLS1.3 ? – sgmoore Feb 18, 2021 at 14:40

8

I also put an answer there, but the article @Colonel Panic's update refers to suggests forcing TLS 1.2. In the future, when TLS 1.2 is compromised or just superceded, having your code stuck to TLS 1.2 will be considered a deficiency. Negotiation to TLS1.2 is enabled in .Net 4.6 by default. If you have the option to upgrade your source to .Net 4.6, I would highly recommend that change over forcing TLS 1.2.

If you do force TLS 1.2, strongly consider leaving some type of breadcrumb that will remove that force if you do upgrade to the 4.6 or higher framework.

Share Improve this answer

Follow

answered Mar 7, 2016 at 18:32

Prof Von Lemongargle

3,758 ● 32 ● 31

2 "Negotiation to TLS1.2 is enabled in .Net 4.6 by default" do you have documentation for this? – felickz Jun 29, 2016 at 19:25

2 At the time, we found inconclusive documentation that hinted at this feature (see msdn.microsoft.com/en-us/library/… just above the examples). However, we verified this by creating a test program with a server that was locked down to TLS 1.2 and used a webrequest to try and download a page. Then we

changed framework versions on the program. All downlevel versions failed to connect, but .Net 4.6 connected without a problem. – Prof Von Lemongargle Jul 1, 2016 at 17:19

Thanks for the details, i see so many API providers pointing out that 4.5 doesn't connect to TLS 1.2 but i struggled to provide any official documentation and had to rely on the community :) – felickz Jul 11, 2016 at 20:41

This is correct and the way to go. We've got a number of older servers talking to web endpoints that are now insisting on TLS. Web service references started to fail with SSL/TLS errors. You can hack the registry but upgrading the server to the latest .NET version is easier and safer. – Neil Laslett Jun 21, 2017 at 14:16