**TLS 1.2 Cipher Suite check and mitigation**

# Introduction

This document describes a PowerShell script used for checking compliance with TLS 1.2 security settings plus creation of a mitigation script for use on client machines connecting to VestasOnline Business (VOB) server version 3.28.

The analytics script (“AzureDevOpsTls12Analysis.ps1”) aims to help customers in preparation to deprecation of TLS 1.0 and TLS 1.1 protocols and weak cipher suites by Azure DevOps Services and the same applies to VOB 3.28 server.

The script performs read-only analysis, does not execute any mitigations by itself, but only creates one or more mitigation script(s) **which can be viewed and evaluated before executing it manually**.

**To** **ensure you have fallback option** you can inspect the mitigation scripts and check registry settings for the keys which are affected by mitigation script(s) **before executing the mitigation script(s)**.

The analytics script runs on Windows client / server OS and detects well-known causes of TLS 1.2 and cipher suite incompatibilities.

Lowest OS version where this script has been tested on: Windows Server 2008 R2.

**NOTE**: Scripts must be run with Administrator privileges as full registry access is required.

# Description of script

The PowerShell script "AzureDevOpsTls12Analysis.ps1" was taken from Microsoft GitHub at https://github.com/microsoft/azure-devops-tls12, which is made to ensure client settings are correct for using TLS 1.2 versus Microsoft Azure DevOps environment, and the requirements for this is similar to the requirements we have for using TLS 1.2 in Vestas.

The script checks the TLS settings and prerequisites and checks the Cipher Suites for TLS 1.2 and check the configuration for TLS for the .NET components, and based on the analysis it makes, it will generate a script with mitigation steps to correct settings in registry which are required for TLS 1.2 use. **The analysis script does not itself make any changes**.

This mitigation script(s) (e.g. "Mitigation-NetFramework.ps1") is not run automatically but must be executed manually once the analysis script has finished. The analytics script will list the mitigation script(s) it has generated in PowerShell output windows when executed.

The steps involved in using the script is to:

1. Open PowerShell as administrator
2. Run the script (".\AzureDevOpsTls12Analysis.ps1")
3. If settings are not ok, the result list in section "Analysis of TLS 1.2 compatibility: .NET Framework" shows in purple text, and the text is shown in green if test is ok - see example below in section 3
4. If settings were not as required, one or more mitigation script(s) are generated and should be executed (e.g. ".\Mitigation-NetFramework.ps1" as in example run below in section 3).

The content of the mitigation script typically looks like below, where .NETFramework settings in registry have keys added to ensure secure cryptography and correct TLS version is used:

# This PowerShell script was generated as a mitigation by Azure DevOps TLS 1.2 transition readiness checker.

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319", "SchUseStrongCrypto", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319", "SystemDefaultTlsVersions", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319", "SchUseStrongCrypto", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319", "SystemDefaultTlsVersions", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v2.0.50727", "SchUseStrongCrypto", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v2.0.50727", "SystemDefaultTlsVersions", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v2.0.50727", "SchUseStrongCrypto", 1)

[microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v2.0.50727", "SystemDefaultTlsVersions", 1)

'Done!'

After execution of mitigation script, a reboot may be required, but the analysis script can be run to check again.

# Example run on a laptop where .NET Framework settings were not ok

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

The purple text in above output lists the problems found by the analysis script.

Above example produced a Mitigation script ”Mitigation-NetFramework.ps1” with content as shown below, generally setting two key values for 32 bit .NETFrameworks and same two key values for 64 bit .NETFrameworks.

The key values which are added are the following:

**SchUseStrongCrypto:**

The KEY\_LOCAL\_MACHINE\SOFTWARE\[Wow6432Node\]Microsoft\.NETFramework\<VERSION>: SchUseStrongCrypto registry entry has a value of type DWORD. A value of 1 causes your app to use strong cryptography. The strong cryptography uses more secure network protocols (TLS 1.2 and TLS 1.1) and blocks protocols that aren't secure. A value of 0 disables strong cryptography. This registry setting affects only client (outgoing) connections in your application.

**SystemDefaultTlsVersions:**

The KEY\_LOCAL\_MACHINE\SOFTWARE\[Wow6432Node\]Microsoft\.NETFramework\<VERSION>: SystemDefaultTlsVersions registry entry has a value of type DWORD. A value of 1 causes your app to allow the operating system to choose the protocol. A value of 0 causes your app to use protocols picked by the .NET Framework.

The <VERSION> must be v4.0.30319 (for .NET Framework 4 and above) or v2.0.50727 (for .NET Framework 3.5).

|  |
| --- |
| **Generated file, ”Mitigation-NetFramework.ps1” content:** |
| # This PowerShell script was generated as a mitigation by Azure DevOps TLS 1.2 transition readiness checker. |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319", "SchUseStrongCrypto", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319", "SystemDefaultTlsVersions", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319", "SchUseStrongCrypto", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319", "SystemDefaultTlsVersions", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v2.0.50727", "SchUseStrongCrypto", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\.NETFramework\v2.0.50727", "SystemDefaultTlsVersions", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v2.0.50727", "SchUseStrongCrypto", 1) |
| [microsoft.win32.registry]::SetValue("HKEY\_LOCAL\_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v2.0.50727", "SystemDefaultTlsVersions", 1) |
| 'Done!' |

After executing above mitigation script, the alanysis script was run again, and now shows all ok (see next page):

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

The green text in above output lists that settings were ok as found by analysis script.

# Additional check/modification if above does not solve issues

If above does not solve issue, the script “Check\_and\_update\_TLS\_1.2\_registry.ps1” should be run.

This script makes further checks and changes on below registry settings:

Check if registry path exists: HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client

Check if: DisabledByDefault is already set to 0 - if not, it will be set to 0

Check if: Enabled is already set to 1 - if not, it will be set to 1

Check if registry path exists: HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\WinHttp

Check if DefaultSecureProtocols is already set to 2560 – if not, it will be set to 2560.

Check if registry path exists: HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server

Check if: DisabledByDefault is already set to 0 in – if not, it will be set to 0.

Check if: Enabled is already set to 1 – if not, it will be set to 1.

# References

These links may provide some background information:

[Enable TLS 1.2 support as Microsoft Entra TLS 1.0/1.1 is deprecated - Azure | Microsoft Learn](https://learn.microsoft.com/en-us/troubleshoot/azure/entra/entra-id/ad-dmn-services/enable-support-tls-environment?tabs=azure-monitor#enable-tls-12-on-client-or-server-operating-systems-)

[Transport Layer Security (TLS) best practices with .NET Framework | Microsoft Learn](https://learn.microsoft.com/en-us/dotnet/framework/network-programming/tls#audit-your-code-and-make-code-changes)

[How to enable Transport Layer Security (TLS) 1.2 on clients - Configuration Manager | Microsoft Learn](https://learn.microsoft.com/en-us/mem/configmgr/core/plan-design/security/enable-tls-1-2-client)

[Cipher Suites in TLS/SSL (Schannel SSP) - Win32 apps | Microsoft Learn](https://learn.microsoft.com/en-us/windows/win32/secauthn/cipher-suites-in-schannel?redirectedfrom=MSDN)