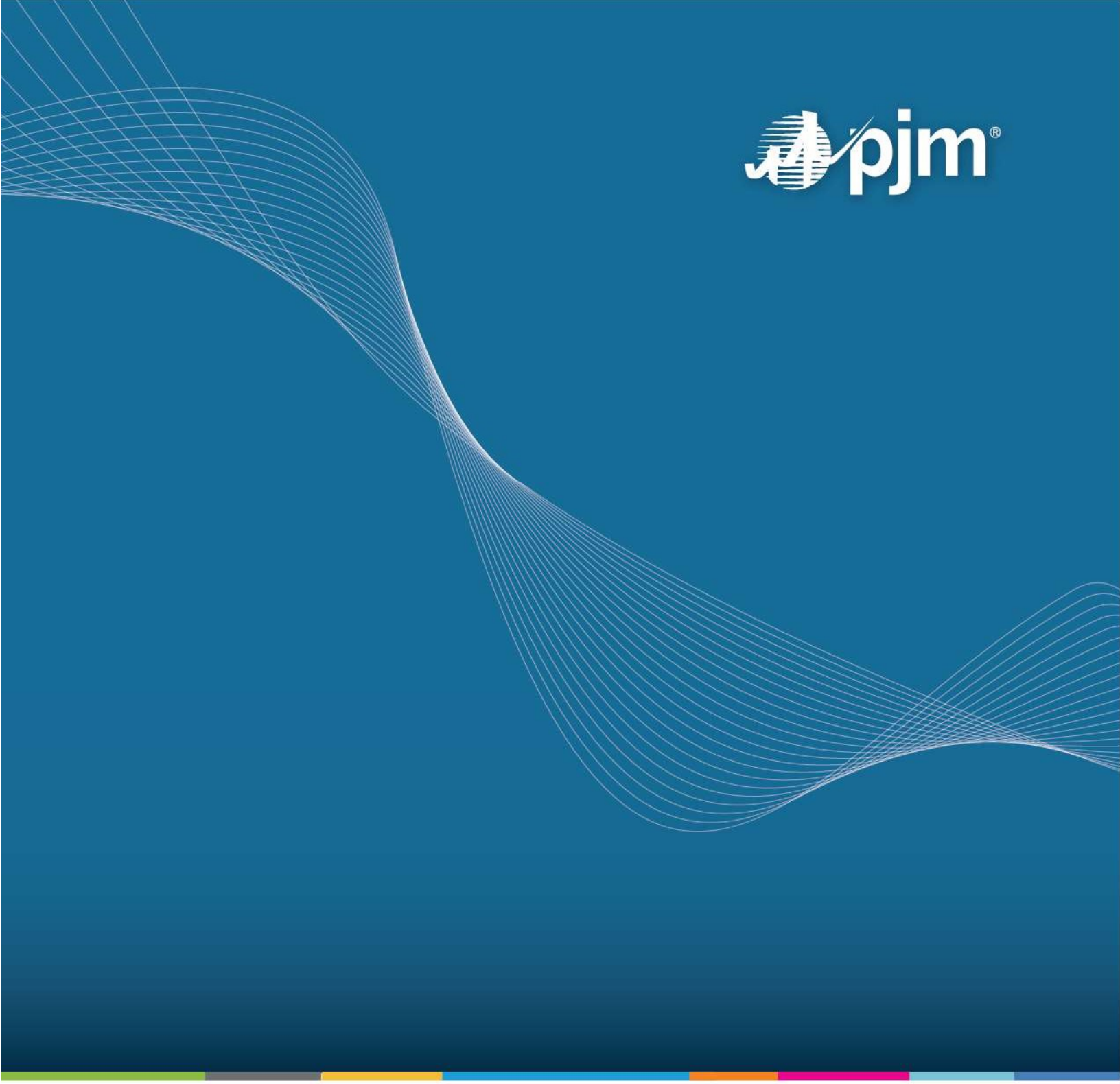


Weak Encryption Remediation Guide

Version 1.3

Enterprise Information Security Department
PJM Interconnection

For Public Use

This page is intentionally left blank.

Contents

Background

Transport Layer Security (TLS) is the most widely deployed security protocol that encrypts data transmitted between two endpoints, most commonly when loading websites over HTTPS. Over time, new attacks against TLS and the encryption algorithms it uses have been discovered. Network connections employing obsolete protocols are at an elevated risk of exploitation by adversaries. PJM has announced in the tech change forums that PJM will be retiring obsolete TLS protocol configurations in PJM internet-facing websites. PJM wants to stop supporting TLS 1.0 or TLS 1.1 protocols and certain insecure ciphers such as 3DES cipher and the TLS_RSA_* ciphers in TLS 1.2.

The purpose of this document is to provide details of the TLS protocol configurations that PJM will support after obsolete configurations are disabled on PJM internet-facing websites. The document also provides guidance that administrators can use to update the noncompliant source device (browser or browserless) deployments, so that obsolete TLS protocol configurations can be disabled expeditiously.

PJM is providing this document to aid PJM stakeholders in discontinuing the use of obsolete TLS protocol configurations. PJM will provide assistance to stakeholders seeking to understand or clarify details in this document. However, due to the varied browser and browserless environments that PJM stakeholders use, PJM is unable to provide additional technical support.

Recommended TLS Configurations

PJM will continue to support TLS version 1.2.

PJM will continue to support the following cipher suites used in TLS 1.2, and PJM customers should make necessary changes to only use these cipher suites.

Cipher Suite ID	Name
0x009E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
0x009F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
0xC02B	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
0xC02C	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
0xC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
0xC030	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Browser Compatibility

PJM has disabled TLS 1.0 and 1.1 and weak ciphers in TLS 1.2 in the PJM Train environment for browser and browserless connections. PJM stakeholders can use the Train environment to determine if their browsers are using supported configurations.

The latest versions of web browsers, such as Microsoft Edge, Google Chrome, Mozilla Firefox and Apple Safari, have TLS 1.2 enabled by default. To enable TLS 1.2 protocols on web browser versions where TLS 1.2 is not enabled by default, please refer to the respective vendor support documentation.

Guidelines to Determine the Actual Protocol and Cipher Suite Negotiated by the Browser

Refer to the guidelines below:

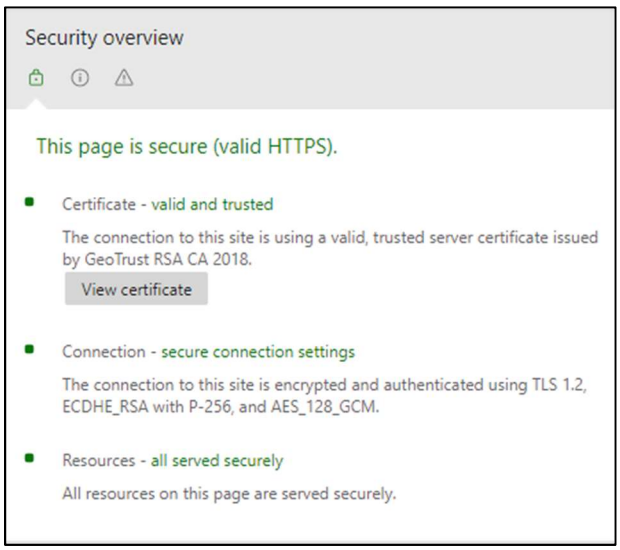
Microsoft Edge

1 | Launch Microsoft Edge.

2 | Enter the URL you wish to check in the browser.

3 | Press **CTRL+SHIFT+i**.

4 | Click on the "**Security**" Tab.

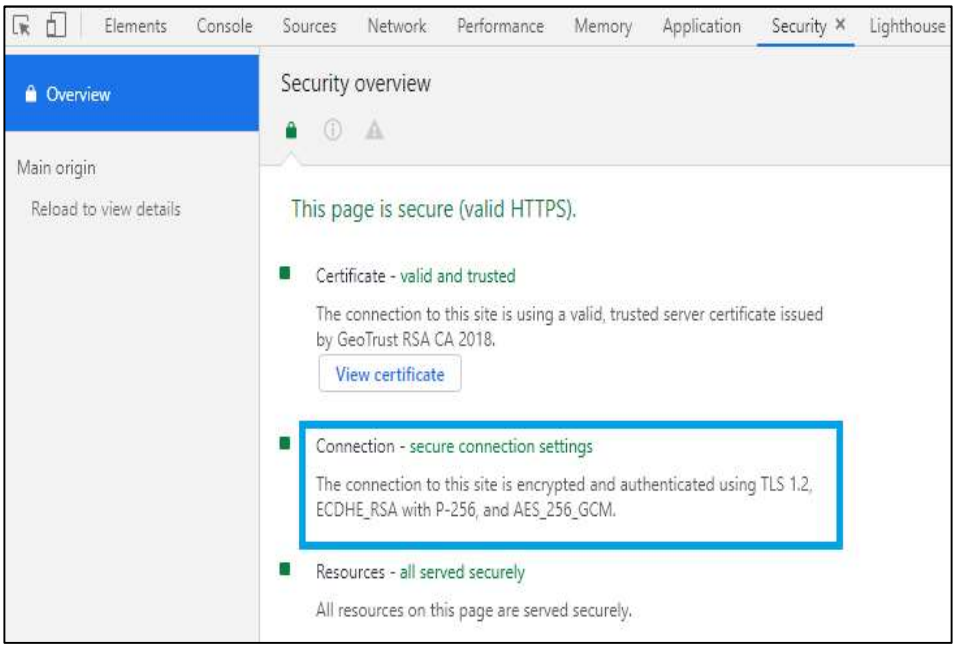


Google Chrome

If you are using Google Chrome version 50 or greater, use the following steps to determine the TLS version and negotiated cipher suite.

1 | Open the Developer Tools with Crtl+Shift+I or by clicking on the ⋮ on the Chrome menu > **More tools** > "Developer tools," and then click on the **Security** tab.

2 | The information has now been added to the 🔒 **Overview** section without reloading and includes the key exchange group.



Mozilla Firefox

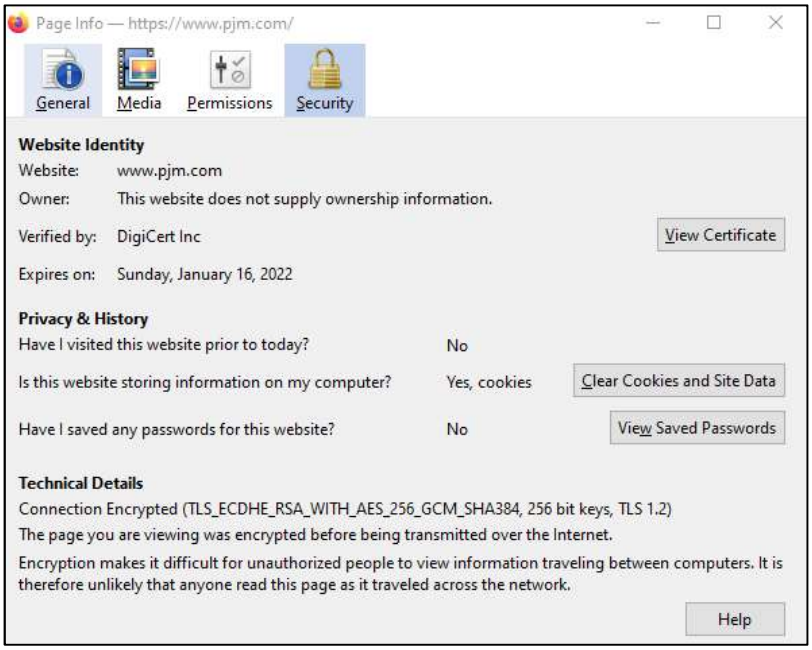
1 | Open Firefox.

2 | Enter the URL you want to check in the browser.

3 | Click on the 🔒 lock icon in the location bar.

4 | Click on the arrow next to connection secure, and then click on more information.

5 | In the new window, look for the **Technical Details** section.



Guidelines to Configure Browserless APIs to Use TLS 1.2

Refer to the guidelines below:

Java (Oracle)

Java 8 (1.8) and higher	Compatible with TLS 1.2 by default.
Java 7 (1.7) and higher, Java 6u121 and higher	Enable TLS 1.2 using the https.protocols Java system property for HttpsURLConnection. To enable TLS 1.2 on non-HttpsURLConnection connections, set the enabled protocols on the created SSLSocket and SSLEngine instances within the application source code. We recommend testing this change before deploying it to your production servers.
Below Java 6u121	Not compatible with TLS 1.2.

Java (IBM)

Java 8	Compatible with TLS 1.2 or higher by default. You may need to set com.ibm.jsse2.overrideDefaultTLS=true if your application or a library called by it uses SSLContext.getinstance ("TLS").
Java 7 and higher, Java 6.0.1 service refresh 1 (J9 VM2.6) and higher, Java 6 service refresh 10 and higher	Enable TLS 1.2 using the https.protocols Java system property for HttpsURLConnection and the com.ibm.jsse2.overrideDefaultProtocol Java system property for SSLSocket and SSLEngine connections. You may also need to set com.ibm.jsse2.overrideDefaultTLS=true. We recommend testing this change before deploying it to your production servers.

.NET

.NET 4.6 and higher	Compatible with TLS 1.2 by default.
.NET 4.5 to 4.5.2	.NET 4.5, 4.5.1 and 4.5.2 do not enable TLS 1.2 by default. Two options exist to enable these, as described below. **Option 1:** .NET applications may directly enable TLS 1.2 in their software code by setting System.Net.ServicePointManager.SecurityProtocol to enable SecurityProtocolType.Tls12 and SecurityProtocolType.Tls11. The following C# code is an example: System.Net.ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls; **Option 2:** It may be possible to enable TLS 1.2 by default without modifying the source code by setting the SchUseStrongCrypto DWORD value in the following two registry keys to 1, creating them if they don't exist: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319" and "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319." Although the version number in those registry keys is 4.0.30319, the .NET 4.5, 4.5.1 and 4.5.2 frameworks also use these values. Those registry keys, however, will enable TLS 1.2 by default in all installed .NET 4.0, 4.5, 4.5.1 and 4.5.2 applications on that system. It is advisable to test this change before deploying it to your production servers. These registry values, however, will not affect .NET applications that set the System.Net.ServicePointManager.SecurityProtocol value.
.NET 4.0	.NET 4.0 does not enable TLS 1.2 by default. To enable TLS 1.2 by default, it is possible to install .NET Framework 4.5, or a newer version, and set the SchUseStrongCrypto DWORD value in the following two registry keys to 1, creating them if they don't exist: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319" and "HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\v4.0.30319." Those registry keys, however, may enable TLS 1.2 by default in all installed .NET 4.0, 4.5, 4.5.1 and 4.5.2 applications on that system. We recommend testing this change before deploying it to your production servers. These registry values, however, will not affect .NET applications that set the System.Net.ServicePointManager.SecurityProtocol value.
.NET 3.5, .NET 3.0 and .NET 2.0	We recommend upgrading to .NET 4.6 or higher. If you cannot upgrade, to use TLS 1.2 in .NET 2.0, .NET 3.0 and NET 3.5, the value 3072 needs to be set in the SecurityProtocol. We recommend testing this change before deploying it to your production servers.

Guidelines When Using HTTPS Proxy Server

Some networks intercept outbound HTTPS traffic by using a proxy server that creates its own certificates, so that the unencrypted communications with PJM and other websites can be inspected. Those proxy servers create their own TLS connections to PJM websites. Networks that use this type of proxy server need to ensure that they support TLS 1.2 and prefer TLS 1.2 when connecting to PJM websites. Irregular behavior may be observed if the proxy server either does not support TLS 1.2 or prefers TLS 1.1 over TLS 1.2 when connecting to PJM websites.

The general configuration recommendations for intercepting HTTPS proxy servers regarding the TLS 1.1 disablement are the following:

- If HTTPS interception is required by the company's policy, or otherwise cannot be removed or exempted, update that proxy server to a newer version that supports TLS 1.2.
- If the intercepting HTTPS proxy server does support TLS 1.2, but prefers TLS 1.1 by using it in its initial ClientHello messages, update the proxy server's configuration to prefer TLS 1.2 over TLS 1.1 when connecting to PJM websites *.pjm.com.

Common Errors When Using Unsupported TLS Configuration

When a noncompliant source device tries to connect to a PJM application that has weak TLS configurations disabled, the following errors are observed at the source device.

Browsers	ERR_SSL_VERSION_OR_CIPHER_MISMATCH
Browserless Java API	javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure
Browserless .NET API	system.Net.Http.HttpRequestException: The SSL connection could not be established