PROGRESS OpenEdge

White Paper: Addressing the POODLE Security Vulnerability in Progress OpenEdge in 11.3.3.013HF

PROGRESS

Table of Contents

Copyright.....5

Chapter 1: Introduction.....7

Chapter 2: Mitigating POODLE SSL 3.0 vulnerability in Progress OpenEdge.....9

Testing for the POODLE SSL 3.0 vulnerability.....9

Chapter 3: Changing the cryptographic protocol and ciphers.....11

Cryptographic protocols and ciphers for Progress OpenEdge clients 12

Cryptographic protocols and ciphers for Progress OpenEdge servers 15

Changing the default protocols and ciphers.....16

Changing the default protocols and ciphers for Progress OpenEdge clients.....16

Changing the default protocols and ciphers for Progress OpenEdge servers.....20

Copyright

© 2015 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Easyl, Fathom, Icenium, Kendo UI, Making Software Work Together, OpenEdge, Powered by Progress, Progress, Progress Control Tower, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, SequeLink, SpeedScript, Stylus Studio, TeamPulse, Telerik, Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, AppsAlive, AppServer, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, , Future Proof, High Performance Integration, Modulus, NativeScript, OpenAccess, Pacific, ProDataSet, Progress Arcade, Progress Pacific, Progress Profiles, Progress Results, Progress RFID, Progress Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

The Release Notes can be found in the OpenEdge installation directory and online at:
https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx.

For the latest documentation updates see OpenEdge Product Documentation on Progress Communities: (https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx).

June 2015

Last updated with new content: Release 11.3.3.013HF

1

Introduction

Padding Oracle On Downgraded Legacy Encryption (POODLE) is a vulnerability that was identified in late 2014 and can affect secure communications making use of the Secure Socket Layer (SSL) 3.0 or earlier protocol. However, newer releases of secure communication protocols, including Transport Layer Security (TLS) 1.0, TLS 1.1 and TLS 1.2, are increasingly less susceptible. An OpenEdge application is therefore vulnerable to POODLE attacks as it establishes secure communications over SSL 3.0 by default. In the 11.3.3 hot fix, you can now manually configure each component individually to communicate over TLS 1.0 (versus the default SSL 3.0 protocol).

POODLE allows an attacker to decrypt cipher text using a padding oracle side-channel attack. Applications that use cipher-block chaining with the SSL 3.0 communication protocol are vulnerable to POODLE attacks. The SSL 3.0 protocol does not adequately check the padding bytes that are sent with encrypted messages. These padding bytes can be replaced by the attacker. So, an attacker can exploit this vulnerability to decrypt and extract information from inside an encrypted transaction.

Note: For more information on the encryption flaws of SSL 3.0, see https://www.openssl.org/~bodo/ssl-poodle.pdf.

2

Mitigating POODLE SSL 3.0 vulnerability in Progress OpenEdge

For your OpenEdge client and server components to be POODLE SSL 3.0 safe, you must configure them to create secure connections using TLS 1.0 only with any cipher that is TLS 1.0 compliant.

As a tradeoff to providing more security, you may find degradation in system performance.

For details, see the following topics:

- Testing for the POODLE SSL 3.0 vulnerability

Testing for the POODLE SSL 3.0 vulnerability

Using the SSLC command line client (available at `$DLC/bin`), you can determine whether a client can connect to a POODLE SSL 3.0 safe server:

```
sslc s_client -connect host_name:port -tls1
```

If the handshake between the client and server is successful, the server is POODLE SSL 3.0 safe.

3

Changing the cryptographic protocol and ciphers

Progress OpenEdge release 11.3.3 hot fix provides configuration parameters for changing the SSL protocol and cipher across all client and server components. Note that secure communication occurs only if both client and server support the same SSL protocol and cipher.

Note: If you need to debug the SSL communication, you can set the `SSL_SYS_DEBUG_LOGGING` environment variable from `1` to `5` (`5` fetches the maximum details) in the OS system variable or in `Environment` section of the `ubroker.properties` file. This generates the `cert.server.log` and `cert.client.log` files.

For details, see the following topics:

- Cryptographic protocols and ciphers for Progress OpenEdge clients
- Cryptographic protocols and ciphers for Progress OpenEdge servers
- Changing the default protocols and ciphers

Cryptographic protocols and ciphers for Progress OpenEdge clients

The following table lists all the supported protocols and ciphers for Progress OpenEdge clients. When you install OpenEdge, all the default protocols are used. You can change the default to the other supported protocols (as listed in the table).

Note: All the cryptographic protocols and ciphers are supported by default. You can use either the short name or the long name of the ciphers.

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form
ABL Client	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
ABL Client (as an HTTP client) ABL Client (connecting to an external Web service)	SSLv2	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
OpenEdge WSDL Analyzer	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form
Java Open Client	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
.NET Open Client	TLSv1 SSLv3 SSLv2	All the ciphers that .NET framework 4.0 supports.	
AppServer Internet Adapter (AIA)	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
WebSpeed Agent (WSA)	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form
WebSpeed Messenger	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
REST Adapter	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
SonicESB Adapter	TLSv1 SSLv3	AES128-SHA RC4-SHA RC4-MD5 DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
OE Client	SSLv3	RC4-SHA RC4-MD5	SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5
JDBC or ODBC drivers	SSLv3	RC4-SHA RC4-MD5	SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5

Cryptographic protocols and ciphers for Progress OpenEdge servers

The following table lists all the supported protocols and ciphers for Progress OpenEdge servers. When you install OpenEdge, all the default protocols are used. You can change the default to the other supported protocols (as listed in the table).

Note: You can use either the short name or the long name of the ciphers.

Component	Cryptographic Protocols	Supported ciphers	
		Short form	**Long form**
Broker	TLSv1 SSLv3	RC4-SHA:RC4-MD5 (Default)	SSL_RSA_WITH_RC4_128_SHA: SSL_RSA_WITH_RC4_128_MD5 (Default)
		RC4-SHA RC4-MD5 AES128-SHA DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
Agent	TLSv1 SSLv3	RC4-SHA:RC4-MD5 (Default)	SSL_RSA_WITH_RC4_128_SHA: SSL_RSA_WITH_RC4_128_MD5 (Default)
		RC4-SHA RC4-MD5 AES128-SHA DES-CBC3-SHA DES-CBC-SHA EXP-DES-CBC-SHA EXP-RC4-MD5	SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_AES_128_CBC_SHA SSL_RSA_WITH_3DES_EDE_CBC_SHA SSL_RSA_WITH_DES_CBC_SHA SSL_RSA_EXPORT_WITH_DES40_CBC_SHA SSL_RSA_EXPORT_WITH_RC4_40_MD5
MS SQL server	TLSv1 SSLv3	RC4-SHA RC4-MD5	SSL_RSA_WITH_RC4_128_SHA SSL_RSA_WITH_RC4_128_MD5

Component	Cryptographic Protocols	Supported ciphers	
		Short form	Long form
OpenEdge RDBMS	`TLSv1` `SSLv3`	`RC4-SHA` `RC4-MD5`	`SSL_RSA_WITH_RC4_128_SHA` `SSL_RSA_WITH_RC4_128_MD5`
OEM/OEE (WebServer)	`TLSv1` (Default) `SSLv3`	All the cipher suites that are provided by the SSL implementation of the JSSE are supported	
OEM/OEE (Email alerts configuration)			

The following OpenEdge server components do not support the SSL 3.0 or TLS protocols:

- OpenEdge Replication
- DataServers

Changing the default protocols and ciphers

Progress OpenEdge provides a set of configuration properties for you to change the default cryptographic protocols and ciphers across all OpenEdge client and server components.

Changing the default protocols and ciphers for Progress OpenEdge clients

You can set the protocols and the ciphers for the following client components:

- **ABL Client**:
  - To set the protocol and cipher for all ABL client connections, export the following environment variables in the startup script of the client application:

Property	Description
`PSC_SSLCLIENT_PROTOCOLS`	Accepts the cryptographic protocol that is set for the client
`PSC_SSLCLIENT_CIPHERS`	Accepts the cryptographic cipher that is set for the client

    Once you export the variables, invoke a Progress OpenEdge client executable and start a session for the client application to use the set cryptographic protocols and ciphers.
  - To set the protocol and cipher for single client connection, you can set the following parameters in the connection-parameters argument of the client's `CONNECT()` method:

Connection parameter	Description
`-sslprotocols` *`SSL-protocol-names`*	Accepts the cryptographic protocol that is set for the client
`-sslciphers` *`SSL-protocol-ciphers`*	Accepts the valid cryptographic ciphers that is set for the client

Note: You can use ABL clients to create connections with different server components. If you do not set the `-sslprotocols` and `-sslciphers` connection parameters using the client's `CONNECT()` method, the protocol and cipher values are set using the `PSC_SSLCLIENT_PROTOCOLS` and `PSC_SSLCLIENT_CIPHERS` environment variables in the `ubroker.properties` file (available at *`<OpenEdge-install-directory>`*`/properties`).

- **OpenEdge WSDL Analyzer**: To set the protocol and cipher for OpenEdge WSDL Analyzer, export the following environment variables in the startup script of the client application:

Property	Description
`PSC_SSLCLIENT_PROTOCOLS`	Accepts a comma-separated list of cryptographic protocols
`PSC_SSLCLIENT_CIPHERS`	Accepts a comma-separated list of valid cryptographic ciphers

  Once you export the variables, invoke the `bprowsdldoc` client executable to use the set cryptographic protocols and ciphers.

- **Java Open Client**: In the client's `Connection` object, you can set the following properties using the `setStringProperty` method of the `RunTimeProperties` class:

Property	Description
`PROGRESS.Session.sslProtocols`	Accepts the cryptographic protocol that is set for the client
`PROGRESS.Session.sslCiphers`	Accepts the valid cryptographic cipher that is set for the client

  You can also set these properties as Java system properties.

- **.NET Open Client**: In the client's `app.config` file, you can set the `Progress.Open4GL.RunTimeProperties.SSLProtocols` runtime property with the cryptographic protocol. The cryptographic cipher is automatically set using the .NET framework.

- **AppServer Agent**: In the `[Environment.`*`<appserver_broker>`*`]` section of the `ubroker.properties` file (available at *`<OpenEdge-install-directory>`*`/properties`), you can set the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts the cryptographic protocol that is set for the agent
PSC_SSLCLIENT_CIPHERS	Accepts the valid cryptographic cipher that is set for the agent

- **WebSpeed Agent**: In the [Environment.<webspeed_broker>] section of the ubroker.properties file (available at <OpenEdge-install-directory>/properties), you can set the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts the cryptographic protocol that is set for the agent
PSC_SSLCLIENT_CIPHERS	Accepts the valid cryptographic cipher that is set for the agent

- **AppServer Internet Adapter**: In the startup script of the client application, export the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts the cryptographic protocol that is set for the AIA instance
PSC_SSLCLIENT_CIPHERS	Accepts the valid cryptographic cipher that is set for the AIA instance

After you export the variables, invoke a Progress OpenEdge client executable and start a session for the client application to use the set protocols and ciphers.

Note: Even if a secure connection is established, a known AIA logging error causes the Error Setting SSL parameters from ServerConnection error statement to appear in the log file. To avoid this problem, set the PSC_SSLCLIENT_PROTOCOLS and PSC_SSLCLIENT_CIPHERS properties in the Apache Tomcat catalina shell script file.

- **WebSpeed Messenger**: In the startup script of the client application, export the following environment variables:

Property	Description
PSC_SSLCLIENT_PROTOCOLS	Accepts the cryptographic protocol that is set for the WebSpeed client instance
PSC_SSLCLIENT_CIPHERS	Accepts the valid cryptographic cipher that is set for the WebSpeed client instance

After you export the variables, invoke an Progress OpenEdge client executable and start a session for the client application to use the set protocols and ciphers.

- **SonicESB Adapter**: Using the **Sonic Management Console**, you can set the following runtime properties:

Property	Description
`sslprotocols`	Accepts the cryptographic protocol that is set for the SonicESB adapter
`sslciphers`	Accepts the valid cryptographic cipher that is set for the SonicESB adapter

To set these properties, do the following:

1. On the **Sonic Management Console**, select the **Configure** tab.
2. Under **Services**, select **OpenEdge Native Services**.
3. Select the **dev.OpenEdge** service name.
4. Under **Init Parameters**, click **Runtime Properties**. A list of properties and their current values is displayed.
5. For the **sslProtocols** and **sslCiphers** properties, enter comma-separated values of cryptographic protocols and ciphers that you want to set for the adapter.

- **Web Services Adapter**: In the `default.props` file (available at *`<OpenEdge-install-directory>`*`/webapps/wsa/wsa1`), set the following properties:

Property	Description
`sslprotocols`	Accepts the cryptographic protocol that is set for the WSA application
`sslciphers`	Accepts the valid cryptographic cipher that is set for the WSA application

To set the default protocol and cipher values for a WSA client, set the properties as `Default`. The following code shows how to set the properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<ApplicationRuntimeProperties xmlns="urn:schemas-progress-com:WSAD:0011"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
<sslCiphers>Default</sslCiphers>
<sslProtocols>Default</sslProtocols>
...
</ApplicationRuntimeProperties>
```

- **REST Adapter**: In the `runtime.props` file (available at `$CATALINA_HOME/webapps/application_name/WEB-INF/adapters`), set the following properties:

Property	Description
`sslProtocols`	Accepts the cryptographic protocol that is set for the REST application
`sslCiphers`	Accepts the valid cryptographic cipher that is set for the REST application

Changing the default protocols and ciphers for Progress OpenEdge servers

You can set the protocol and the cipher for the following server components:

- **AppServer Broker**: In the `[Environment.<appserver_broker>]` section of the `ubroker.properties` file, you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts the cryptographic protocol that is set for the AppServer instance
`PSC_SSLSERVER_CIPHERS`	Accepts the valid cryptographic cipher that is set for the AppServer instance. The default value is `RC4-SHA:RC4-MD5`.

  Once you make a change in the `ubroker.properties` file, you must restart the AppServer broker for the new values to take effect.

  Note: Ensure that when you set `PSC_SSLCLIENT_CIPHERS` in the `[AppServer.<broker_name>]` section of the `ubroker.properties` file, you must also set `PSC_SSLSERVER_CIPHERS`.

- **WebSpeed Broker**: In the `[Environment.<webspeed_broker>]` section of the `ubroker.properties` file, you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts the cryptographic protocol that is set for the Web server instance
`PSC_SSLSERVER_CIPHERS`	Accepts the valid cryptographic cipher that is set for the Web server instance. The default value is `RC4-SHA:RC4-MD5`.

  Once you make a change in the `ubroker.properties` file, you must restart the WebSpeed broker for the new values to take effect.

- **OpenEdge Database Server**: In the startup script of the server application, export the following environment variables.

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the database server instance
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the database server instance. The default value is RC4-SHA:RC4-MD5.

Once you export the variables, invoke a database server executable and start a session for the client application to use the set protocols and ciphers.

- **OpenEdge RDBMS**: In the ubroker.properties file (available at *<OpenEdge-install-directory>*/properties), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the instance
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the instance. The default value is RC4-SHA:RC4-MD5.

- **MS SQL Server**: In the ubroker.properties file (available at *<OpenEdge-install-directory>*/properties), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the instance
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the instance. The default value is RC4-SHA:RC4-MD5.

- **AppServer Agent**: In the [Environment.*<appserver_broker>*] section of the ubroker.properties file (available at *<OpenEdge-install-directory>*/properties), you can set the following environment variables:

Property	Description
PSC_SSLSERVER_PROTOCOLS	Accepts the cryptographic protocol that is set for the agent
PSC_SSLSERVER_CIPHERS	Accepts the valid cryptographic cipher that is set for the agent. The default value is RC4-SHA:RC4-MD5.

- **WebSpeed Agent**: In the `[Environment.<webspeed_broker>]` section of the `ubroker.properties` file (available at `<OpenEdge-install-directory>/properties`), you can set the following environment variables:

Property	Description
`PSC_SSLSERVER_PROTOCOLS`	Accepts the cryptographic protocol that is set for the agent
`PSC_SSLSERVER_CIPHERS`	Accepts the valid cryptographic cipher that is set for the agent. The default value is `RC4-SHA:RC4-MD5`.

- **OpenEdge Business Process Management**:
  - **EJB server**: In the `server.xml` file (available at `<OEBPM_HOME>/jboss/server/ejbServer/deploy/jbossweb.sar`), you can set the following properties:

Property	Description
`protocol`	If you want to change the cryptographic protocol for secure communication with an AppServer, set this property to `TLS`.
`sslProtocol`	Accepts the cryptographic protocol that is set for connecting to the AppServer

The following code shows how to set the properties.

```
<Server>
...
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
<Connector protocol="TLS"
           sslProtocol="TLSv1,TLSv1.1,TLSv1.2"
           ... />
-->
...
```

  - **Portal server**: In the `server.xml` file (available at `<OEBPM_HOME>/jboss/server/portalServer/deploy/jbossweb.sar`), you can set the following properties:

Property	Description
`protocol`	If you want to change the cryptographic protocol for secure communication with an AppServer, set this property to `TLS`.
`sslProtocol`	Accepts the cryptographic protocol that is set for connecting to the AppServer

The following code shows how to set the properties.

```
<Server>
...
<!-- SSL/TLS Connector configuration using the admin devl guide keystore
<Connector protocol="TLS"
           sslProtocol="TLSv1,TLSv1.1,TLSv1.2"
           ... />
-->
...
```

- **OpenEdge Management and OpenEdge Explorer**:
  - **WebServer**: In the `fathom.properties` file (available at `$DLC/properties`), you can set the following environment variables:

Property	Description
`HttpsEnabled`	Enables you to change the cryptographic protocols and ciphers for secure communication with a WebServer. If you enable SSL for the WebServer in OpenEdge Management and OpenEdge Explorer, this property is set to `true`.
`SSLEnabledProtocols`	If you want to change the default cryptographic protocol for the WebServer, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic protocols that are set for secure communication.
`SSLEnabledCipherSuites`	If you want to change the default cryptographic ciphers for the WebServer, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic ciphers that are set for secure communication.

Additionally, in the `fathom.init.params` file (available at `$DLC`), you can set the the property `ssl.KeyManagerFactory.algorithm=IbmX509`.

Note: You can set the `ssl.KeyManagerFactory.algorithm=IbmX509` property only on AIX, AIX (64-bit), and LinuxPPC systems.

Note: In OpenEdge 11.3.3, OpenEdge Management is configured to use Jetty Web server version 7.6.3. Even if you successfully enable secure communications using the Webserver, you may see the following warning message about the secure connection closing occuring repeatedly in the AdminServer log file:

```
...
Invoking E-mail action test. Action: Default_Clear_Action, From:
admin@domain.com, To: user@domain.com (9614)
2015-06-19 18:36:27.088:WARN:oeji.nio:javax.net.ssl.SSLException:
Inbound closed before receiving peer's close_notify: possible truncation
 attack?
Invoking E-mail action test. Action: Default_Clear_Action, From:
admin@domain.com, To: user@domain.com (9614)
2015-06-19 18:36:27.518:WARN:oeji.nio:javax.net.ssl.SSLException:
Inbound closed before receiving peer's close_notify: possible truncation
 attack?
...
```

This behavior is expected and these messages can be ignored.

- **Email alerts configuration**: In the `fathom.properties` file (available at `$DLC/properties`), you can set the following environment variables:

Property	Description
`SmtpSSLEnabledProtocols`	If you want to change the default cryptographic protocol for the email alerts configuration, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic protocols that are set for secure communication.
`SmtpSSLEnabledCipherSuites`	If you want to change the default cryptographic ciphers for the email alerts configuration, enter this property in the `fathom.properties` file. The property accepts a comma-separated list of valid cryptographic ciphers that are set for secure communication.

Note: Both the AppServer Broker and Agent inherit environment variables (`PSC_SSLCLIENT_PROTOCOLS`, `PSC_SSLCLIENT_CIPHERS`, `PSC_SSLSERVER_PROTOCOLS`, and `PSC_SSLSERVER_CIPHERS`) both from the AdminServer process and the `[Environment.<broker_name>]` sections of the `ubroker.properties` file. The variables in the `[Environment.<broker_name>]` sections of the `ubroker.properties` file supercede the variables set by the AdminServer process. If you set these variables in the `ubroker.properties` file and then use the shell to manually start the AdminServer process, the brokers and agents inherit the variables set in the `ubroker.properties` file.