Passwordstate

Enterprise Password Management

Encrypting Passwordstate Configuration Files

Table of Contents

1 Overview

If you are already familiar with the basics of encrypting or decrypting web.config files, skip straight to the section "**6 – Encrypting Config Files**" or "**7 – Decrypting Config Files**".

For those new to this concept, Passwordstate has many different forms of built in security to help protect your data. The Microsoft Windows operating system also has features available to help secure your environment.

Encrypting your web.config files is a Microsoft Windows feature that removes sensitive information from those files, that an attacker could use to potentially gain access to your database.

This document explains how to encrypt and decrypt your web.config files for each relevant module in Passwordstate, and how this process helps secure your data.

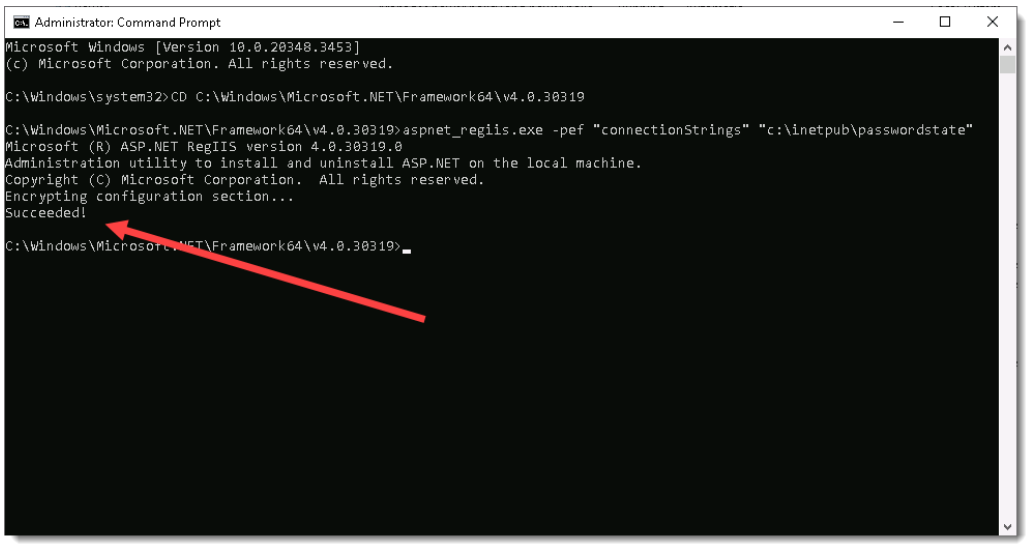
Notes:

- Encrypting or decrypting these files will terminate any active sessions, so it's best to schedule this process out of hours when no one is using the software

- If you intend to rename your server host name, or move your Passwordstate website to a different server, you should decrypt your web.config file first, and re-encrypt it again once the renaming or migration is complete

- The encryption and decryptions commands are very similar, so it's important to follow the correct section of this guide (Section 6 or 7), depending on your requirements

- When you successfully encrypt or decrypt a file in a command prompt, you will see a **Succeeded!** message as per the screenshot below:



2 How it works?

All sensitive data in the Passwordstate database such as passwords, documents and authentication options are encrypted within the database using AES 256 or FIPS encryption. In order to decrypt this data in the user interface to make it readable to an authorized user, Passwordstate uses two encryption keys.

One half of these encryption keys resides in the database, and the other half in the web.config file. When you log into Passwordstate, these keys are joined and then used to decrypt sensitive data, which makes it readable to you in the user interface.

When encrypting a web.config file using the Microsoft commands outlined later in this document, the encryption is formed using unique system keys already built into your Windows operating server. This provides an additional layer of protection, as they can only be decrypted on the same server they were initially encrypted on.

As you cannot decrypt these files on different servers, it's critical that you ensure you export your encryption keys, and store them safely outside of your Passwordstate server – without these encryption keys, if you have a server crash which is not recoverable, then it would not be possible to recover your Passwordstate instance.

For this reason, it is critical to keep a secure backup of your Passwordstate encryption keys - more about this can be found in the next section of this document.

3 Backup Encryption Keys

It is critical that you have a backup of the Passwordstate encryption keys in the event of a disaster.

Passwordstate has a built-in feature where you can automatically backup your encryption keys to a network share of your choice, and you can also export them at any time when logged in as a Security Administrator.

For information on how to set up the backup feature in Passwordstate, refer to either of the automatic backup guides on our documentation page located here:
https://www.clickstudios.com.au/documentation/

If you wanted to export the encryption keys at any time for the purpose of storing them somewhere safe, log into Passwordstate as a Security Administrator, and browse to **Administration** -> **Encryption Keys**, and click the **Export Keys** button on that page.

This will export the keys to a password protected zip file which we would recommend storing a safe place that you will be able to access in the event of a disaster. You could even print out the keys, and lock them in a safe if required.

In order to recover from a disaster, you will need a copy of these keys and a copy of your database, so it is critical that you keep this information stored safely outside of Passwordstate. Click Studios cannot help you recover your environment if you do not have this information.

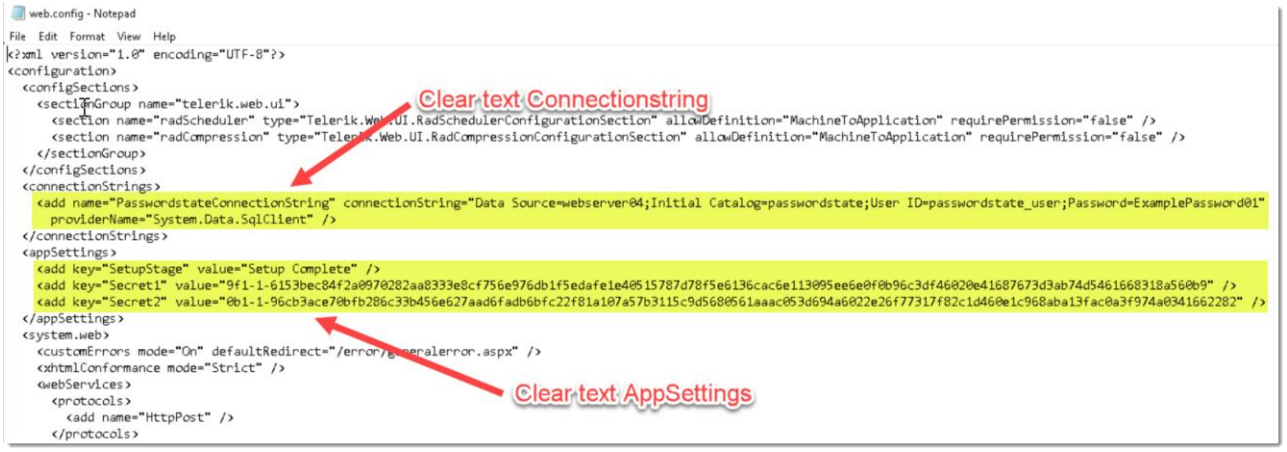
4 What an encrypted file looks like

A standard web.config file will be in clear text, and two important parts of this file with sensitive information are the "**connectionStrings**" section, and the "**appSettings**" section.

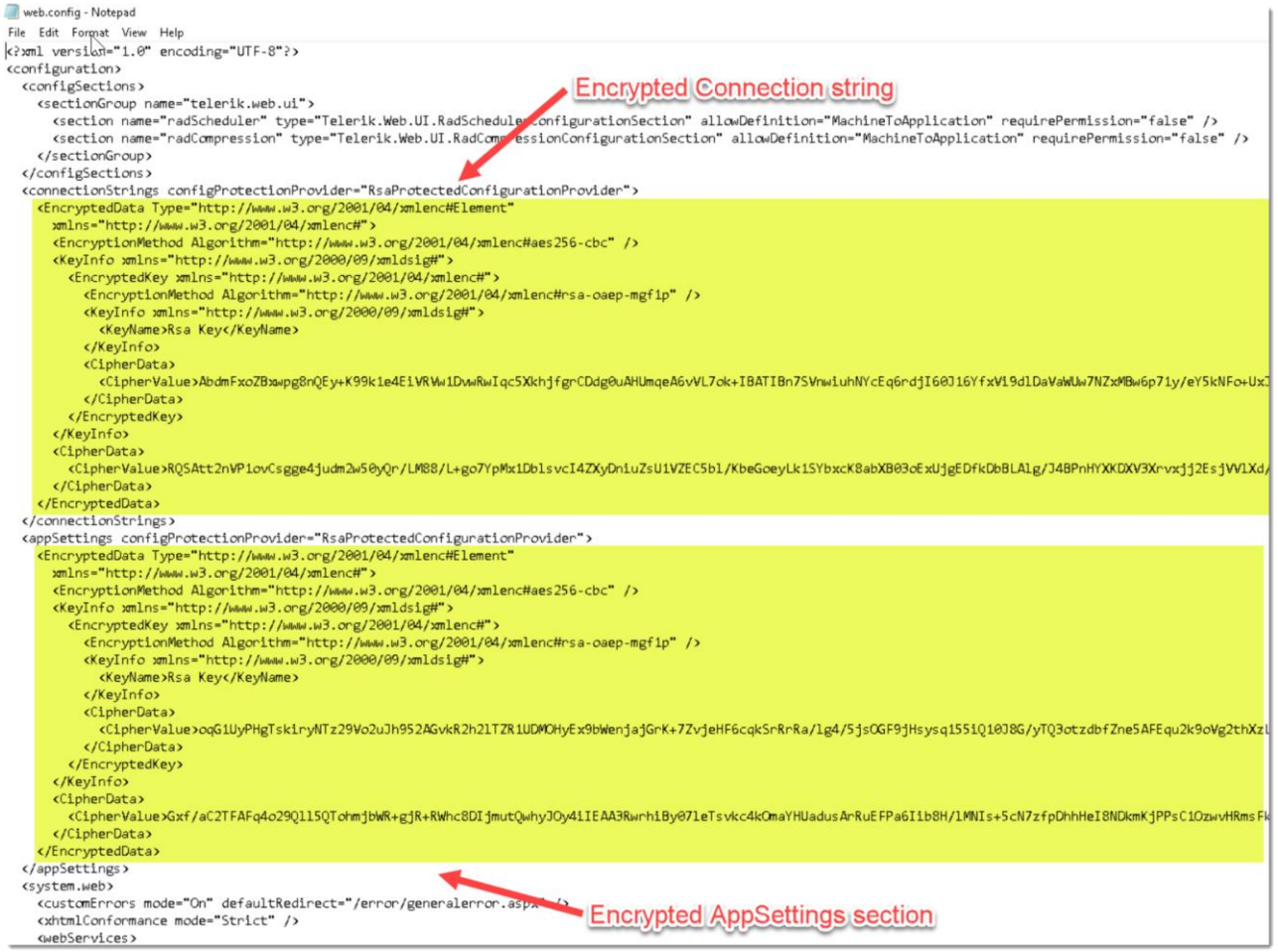
The **connectionStrings** section holds the credentials that your Passwordstate website uses to connect to your database. For example, it will contain the **server name**, the **database name** and **database instance** if it is applicable, and the **SQL username** and **password**.

The **appSettings** section contains the two split **Secret Encryption Keys**.

A clear text web.config file looks like this:



An encrypted web.config file looks like this:



As you can see, the encrypted web.config file is not readable, and references “RSAProtected”.

5 Where are the config files located?

Primary Passwordstate website

Server Location: Check **Administration** -> **Authorised Webservers** page

Default Location: c:\inetpub\Passwordstate**web.config**

Purpose: Main Passwordstate website for daily activity, storing and viewing Passwords etc.

High Availability Passwordstate website

Server Location: Check **Administration** -> **Authorised Webservers** page

Default Location: c:\inetpub\Passwordstate**web.config**

Optional Install: **Yes**

Additional License: **Yes**

Purpose: Second Passwordstate website used for disaster recovery purposes in the event your main Passwordstate website becomes unavailable.

Password Reset Portal

Server Location: Check **Administration** -> **Password Reset Portal Administration** -> **System Settings** -> **Miscellaneous** to obtain your Password Reset Portal URL. Perform an NSLookup on this URL in a command prompt to find server location. Example: ***nslookup passwordresetportal.contoso.com***

Default Location: c:\inetpub\PasswordstateResetPortal**web.config**

Optional Install: **Yes**

Additional License: **Yes**

Purpose: Self Service Active Directory password reset portal, so end users can unlock or reset their own domain passwords.

AppServer Module

Server Location: Check **Administration** -> **Authorised Webservers** page

Default Location: c:\inetpub\PasswordstateAppServer**web.config**

Optional Install: **Yes**

Additional License: **No**

Purpose: Used for iOS and Android mobile Apps, Browser Extensions, or Self Destruct Messages.

Remote Site Locations Agents

Server Location: Check **Administration** -> **Remote Site Locations** page

Default Location: C:\Program Files (x86)\Passwordstate Agent**PasswordstateAgent.exe.config**

Optional Install: **Yes**

Additional License: **No**

Purpose: Agent that can be installed on disconnected networks, with the purpose of performing Privileged Account management over on those networks. Multiple agents can be deployed to multiple networks.

Self Destruct Web Site (Push/Pull)

Server Location: Check **Administration** -> **System Settings -> Self Destruct Messages** tab to obtain your "**Separate Site URL**". Perform an NSLookup on this URL in a command prompt to find server location. Example: ***nslookup selfdestruct.contoso.com***

Default Location: C:\inetpub\PasswordstateSelfDestruct**web.config**

Optional Install: **Yes**

Additional License: **No**

Purpose: An additional website that can be optionally installed which allows users to read Self Destruct messages. Typically, this module be installed in something like your DMZ to allow secure access from outside the network.

6 Encrypting Config Files

Each config file may be slightly different, so first determine which file you are performing this task on, and follow the instructions below.

Step 1: Log into the machine where you intend on encrypting the files
Step 2: Open a command prompt as **Administrator**
Step 3: Change directories by pasting the following code into your command prompt, and hit enter:

CD C:\Windows\Microsoft.NET\Framework64\v4.0.30319

Depending on which server you are actioning this on, choose the appropriate section below:

Option #1: Primary Passwordstate website

Step 1: Stop the **Passwordstate Service**
Step 2: To encrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pef "connectionStrings" "c:\inetpub\passwordstate"

Step 3: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "c:\inetpub\passwordstate"

Step 4: Start the **Passwordstate Service**

Option #2: High Availability Passwordstate website

Step 1: Stop the **Passwordstate Service**
Step 2: To encrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pef "connectionStrings" "c:\inetpub\passwordstate"

Step 3: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "c:\inetpub\passwordstate"

Step 4: Start the **Passwordstate Service**

Option #3: Password Reset Portal

Step 1: Stop the **Passwordstate Reset Portal Service**
Step 2: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "c:\inetpub\passwordstateresetportal"

Step 3: Start the **Passwordstate Reset Portal Service**

Option #4: AppServer Module

Step 1: Stop the **Passwordstate App Sever Service**
Step 2: To encrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pef "connectionStrings" "c:\inetpub\passwordstateappserver"

Step 3: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "c:\inetpub\passwordstateappserver"

Step 4: Start the **Passwordstate App Sever Service**

Option #5: Remote Site Locations Agents

Step 1: Stop the **Passwordstate Agent Service**
Step 2: Rename the file **PasswordstateAgent.exe.config** to **web.config**
Step 3: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "C:\Program Files (x86)\Passwordstate Agent"

Step 4: Rename the web.config file back to **PasswordstateAgent.exe.config**
Step 5: Start the **Passwordstate Agent Service**

Option #6: Self Destruct Website (Push/Pull)

Step 1: To encrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pef "appSettings" "C:\inetpub\PasswordstateSelfDestruct"

Step 2: On your core Passwordstate webserver, restart the **Passwordstate Windows Service**

7 Decrypting Config Files

Each config file may be slightly different, so first determine which file you are performing this task on, and follow the instructions below.

Step 1: Log into the machine where you intend on encrypting the files
Step 2: Open a command prompt as **Administrator**
Step 3: Change directories by pasting the following code into your command prompt, and hit enter:

CD C:\Windows\Microsoft.NET\Framework64\v4.0.30319

Depending on which server you are actioning this on, choose the appropriate section below:

Option #1: Primary Passwordstate website

Step 1: Stop the **Passwordstate Service**
Step 2: To decrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pdf "connectionStrings" "c:\inetpub\passwordstate"

Step 3: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "c:\inetpub\passwordstate"

Step 4: Start the **Passwordstate Service**

Option #2: High Availability Passwordstate website

Step 1: Stop the **Passwordstate Service**
Step 2: To decrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pdf "connectionStrings" "c:\inetpub\passwordstate"

Step 3: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "c:\inetpub\passwordstate"

Step 4: Start the **Passwordstate Service**

Option #3: Password Reset Portal

Step 1: Stop the **Passwordstate Reset Portal Service**
Step 2: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "c:\inetpub\passwordstateresetportal"

Step 3: Start the **Passwordstate Reset Portal Service**

Option #4: AppServer Module

Step 1: Stop the **Passwordstate App Sever Service**
Step 2: To decrypt the **connectionStrings** section, execute this line of code:

aspnet_regiis.exe -pdf "connectionStrings" "c:\inetpub\passwordstateappserver"

Step 3: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "c:\inetpub\passwordstateappserver"

Step 4: Start the **Passwordstate App Sever Service**

Option #5: Remote Site Locations Agents

Step 1: Stop the **Passwordstate Agent Service**
Step 2: Rename the file **PasswordstateAgent.exe.config** to **web.config**
Step 3: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "C:\Program Files (x86)\Passwordstate Agent"

Step 4: Rename the web.config file back to **PasswordstateAgent.exe.config**
Step 5: Start the **Passwordstate Agent Service**

Option #6: Self Destruct Website (Push/Pull)

Step 1: To decrypt the **appSettings** section, execute this line of code:

aspnet_regiis.exe -pdf "appSettings" "C:\inetpub\PasswordstateSelfDestruct"

Step 2: On your core Passwordstate webserver, restart the **Passwordstate Windows Service**

8 Troubleshooting

Issue 1:

When encrypting or decrypting files, and you receive and err in the command prompt stating the following:

or

Failed to decrypt using provider 'RsaProtectedConfigurationProvider'. Error message from the provider: The RSA key container could not be opened. (c:\inetpub\passwordstate\web.config line 10)

Fix:

You must run the command prompt as administrator when performing the encryption/decryption tasks. Ie, right click the command prompt shortcut, and select "**Run as Administrator**"

Issue 2:

If you try to decrypt a web.config file on any other machine that it was originally encrypted on, then you will get the following error:

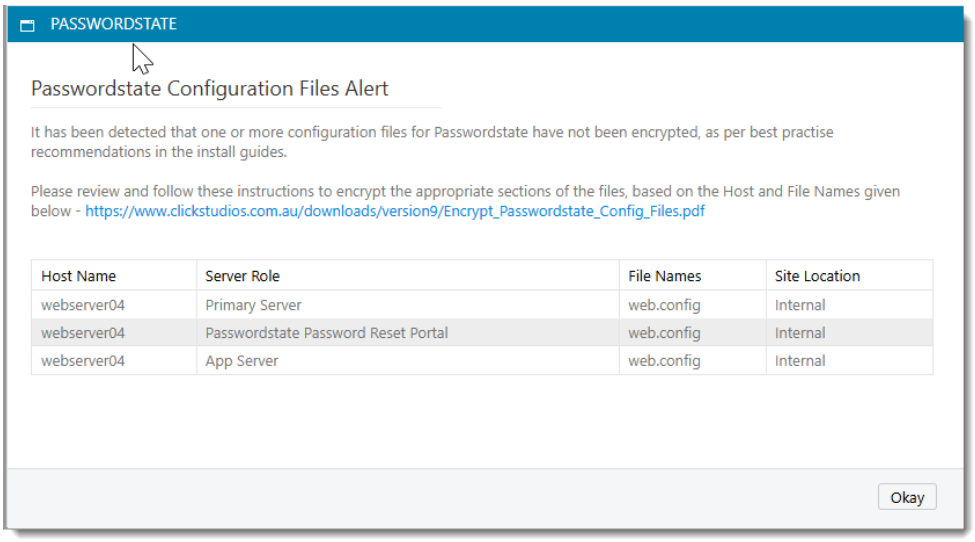
Failed to decrypt using provider 'RsaProtectedConfigurationProvider'. Error message from the provider: Error occurred while decoding OAEP padding. (c:\inetpub\passwordstate\web.config line 10)

Fix:

You must decrypt the file on the same server where it was encrypted. If this isn't possible due to a server failure, you will need to rebuild the web.config file by following the section "**5.1.3 Rebuilding the Web.config File**" in the Security Administrators manual:
https://www.clickstudios.com.au/downloads/version9/Passwordstate_Security_Administrators_Manual.pdf

Issue 3:

If you have encrypted your files, but you are still getting warnings when logging into Passwordstate stating that your web.config file are not encrypted like the following example:



Fix:

There are several suggestions below that can help with this:

1. Ensure your Passwordstate Windows Service is started on the server
2. Check the Application Event logs on your server and look for any errors related to the Passwordstate Windows Service. If you find any errors and are unsure what they mean, forward to Click Studios Support for analysis
3. Check the identity of the account that your Passwordstate Windows Service runs under. By default, it runs under an account called "**Local System**". Whatever account you find assigned to your application pools, ensure that it has read permissions to the **C:\ProgramData\Microsoft\Crypto\RSA\MachineKeys** folder on the server.
4. Ensure both the **appSettings** and **connectionStrings** sections of the web.config file are encrypted. If only the connectionStrings section is encrypted, but the appSettings section is not, then the pop-up warning will still alert you when you log into Passwordstate.
5. If the pop-up warning is related to your High Availability site and it is running **in Passive mode** (read only), ensure the Base URL under **Administration** -> **System Settings** -> **Miscellaneous** page is correct.
6. Again, if the warning if for the High Availability website and you have that site running in Passive mode, check the API on the primary website is functioning correctly. You can test this by logging into that primary site, and try generating a random password from the password generator feature up the top right-hand corner of your page. If a random password generates, then the API is working.
7. On the **Administration** -> **Authorized Web Servers** page, ensure you have only one server that has the **Primary Server Role**. You should have only one server on this page that has the role. If you don't have any servers with that role, edit the appropriate server and assign the role to it, and the restart the Passwordstate Windows Services on that server
8. Try restarting the Passwordstate Windows Services, and this should query the files again to see if they are encrypted.

Notes:

- The Passwordstate Services will check the web.config files to see if they are encrypted each time they are started, or they will automatically check once per day
- If this troubleshooting section does not help, please log a support call with Click Studios who will do a number of checks on your system. Make sure you mention which files are being reported as not being encrypted: https://www.clickstudios.com.au/support.aspx