**Web.config Encryption using ASP.NET IIS Registration Tool**

Contents

[**[1] Prerequisites** 3](#_Toc170911795)

[**[2]** **Encryption workflow** 3](#_Toc170911796)

[**[3] Encryption of Web.config file the connection Strings.** 4](#_Toc170911797)

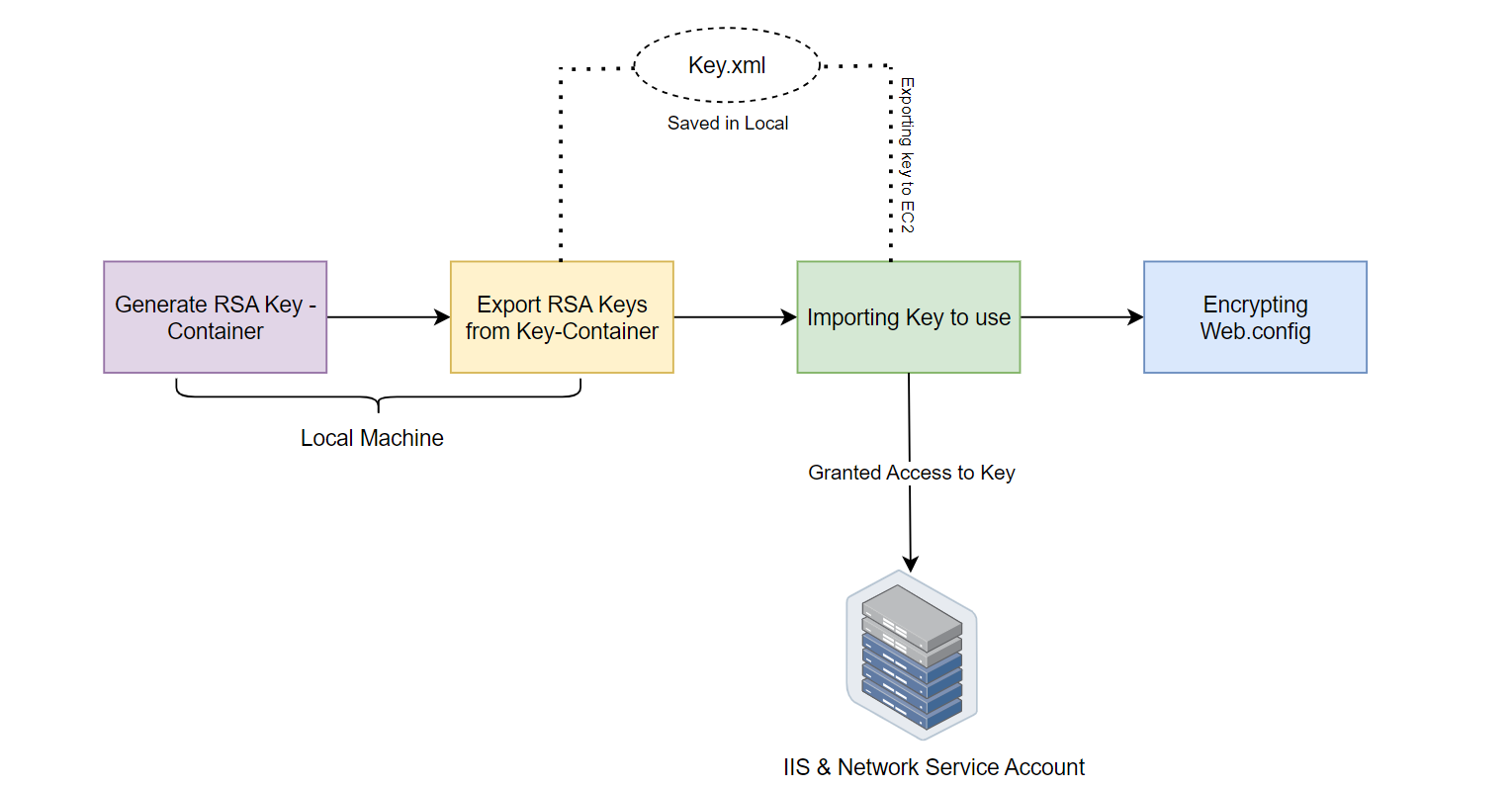
[**[4]** **Decrypting** **Web.config file** 6](#_Toc170911798)

[**[5] Safety features of ASP.NET IIS Registration tool.** 6](#_Toc170911799)

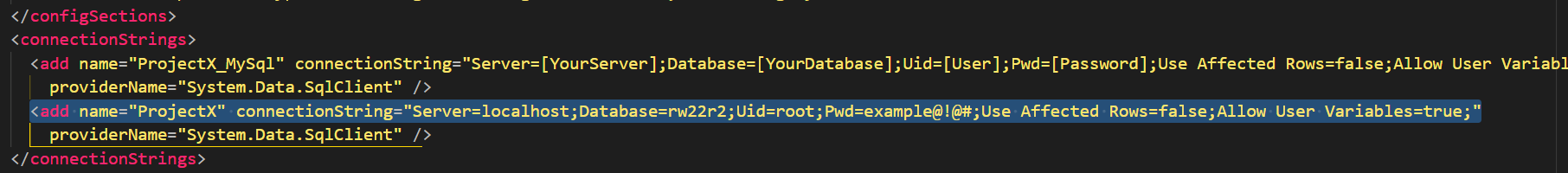
# ­­­­­**[1] Prerequisites**

1. Take a backup of web.config file.
2. Stop the Application pool before starting the process.

# **[2]** **Encryption workflow**



# **[3] Encryption of Web.config file the connection Strings.**

1. The below is the config file (before encrypting), it is evident that connection string is visible, which is not a secure way.

**Steps to Encrypt web.config:**

**Step 1**: **Login to EC2-Server/Machine other than where encryption needs to be performed:**  
 Open command prompt as an `Administrator` and run below command  
 **cd c:\Windows\**[**Microsoft.NET**](http://microsoft.net/)**\Framework\v4.0.30319\**

**Step 2: Creating RSA Key(s):**

**aspnet\_regiis.exe -pc "<container\_Name>" -exp**

**-exp:** option ensures that the newly created key container is exported to a file for backup purposes. This file will contain the necessary information about the key container, including its public key, which can be used to import the key container onto other machines or to recover it if needed.

**Step 3:** **Exporting RSA key from container** :  
 **aspnet\_regiis -px "<container\_Name>" c:\RSAkeys\keys.xml -pri**

**-pri:** This option indicates that you want to include the private key information along with the public key information in the exported XML file. Including the private key is necessary if you intend to use this key container for encryption or decryption.

**NOTE**: Keep the key.xml file safe as this will be used to encrypt/decrypt web.config.

**Step 4: Copy PX.Data.dll from the bin folder of the site to**:

**c:\Windows\**[**Microsoft.NET**](http://microsoft.net/)**\Framework\v4.0.30319\**

**Step 5: Importing Keys to Live EC2-Server:**Copy-Paste the keys.xml from EC2 where Keys were created in Step-3 to EC2 instance where the application is hosted and import it using the below command:

**aspnet\_regiis -pi "<container\_name\_live>" c:\keys.xml**

**Step 6:** **Modifying web.config to use the key for encryption:**  
Add the <configProtectedData> section to web app’s web.config file, just above <connectionStrings> is preferred.

**<configProtectedData>**

**<providers>**

**<add name="ConfigCryptoProvider"**

**type="System.Configuration.RsaProtectedConfigurationProvider,System.Configuration, Version=2.0.0.0,**

**Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a, processorArchitecture=MSIL"**

**keyContainerName=""< container\_name\_live>"**

**useMachineContainer="true" />**

**</providers>**

**</configProtectedData>**

**Step 7:** The account used by your IIS worker process will need to be granted access to the RSA keypair that you imported in the prior step. On every web server, run the “aspnet\_regiis -pa” command to grant access to the RSA keypair.

For providing **access to the RSA Key pair** imported use the below commands:

1. **aspnet\_regiis -pa "<container\_name\_live>" "NT AUTHORITY\NETWORK SERVICE"** To provide access to built-in windows network service account.
2. **aspnet\_regiis -pa "<container\_name\_live>" "IIS APPPOOL\DefaultAppPool"**

This provides access to the AppPool .   
‘**DefaultAppPool’** Should be replaced with the name of app-pool if the application/site is separate from DefaultAppPool

**Step 8:** The web.config file is by default located in **C:\Path\_to\webconfig\file.config**

Enter the following command to encrypt the connectionStrings:

**ASPNET\_REGIIS -pef "connectionStrings" "C:\Path\_to\webconfig\file.config" -prov ConfigCryptoProvider"**

**Note**: **1.** The parameter "connectionStrings" is case sensitive  
 **2.** Only give file path till root directory, do not enter full path till .config file.

A screenshot of a computer program

Description automatically generatedAfter following all the above steps correctly your web.config file should look like below, hiding all the crucial information from unwanted guests.­­  
Restart the IIS application and check if Instance is up and running.

# **[4]** **Decrypting** **Web.config file**

To decrypt the web.config file use the below command in CMD (admin)

**ASPNET\_REGIIS -pdf "connectionStrings" "C:\Path\_to\webconfig\file.config"**

NOTE: Encrypt the keys.xml file exported from the RSA container and keep it safe. These keys are generated from machine-level RSA containers so will only be accessible to ‘administrator’ account while decrypting.so make sure you are logged in using the same user who generated and encrypted the web.config file in 1st place.

# **[5] Safety features of ASP.NET IIS Registration tool.**

The **ASP.NET IIS Registration Tool (Aspnet\_regiis.exe)** is official tool by Microsoft for encryption or decryption of sections of a Web configuration file. ASP.NET will automatically decrypt encrypt configuration elements when the Web.config file is processed. Below are some safety features of this tool which make it secure.

1. **Data Protection API (DPAPI) Integration:  
   aspnet\_regiis.exe** leverages the Data Protection API (DPAPI) provided by Windows to encrypt and decrypt sensitive data. DPAPI ensures that the encryption keys are securely managed and protected by the operating system. This integration enhances the overall security of the encryption process.
2. **Machine-Specific Encryption Keys:**  
   When encrypting configuration sections without specifying a key, **aspnet\_regiis.exe** generates machine-specific encryption keys. These keys are unique to the machine on which the encryption operation is performed, ensuring that the encrypted data can only be decrypted on the same machine. This feature adds an extra layer of protection against unauthorized access to encrypted data.
3. **Administrator Privileges Requirement:**  
   To use **aspnet\_regiis.exe**, users typically need administrative privileges. This requirement helps restrict access to the tool and prevents unauthorized users from encrypting or decrypting sensitive data in the **web.config** file.
4. **Configuration Section Granularity:**   
   **aspnet\_regiis.exe** allows users to encrypt specific sections of the **web.config** file, such as connection strings or application settings. This granularity enables users to selectively encrypt only the sensitive data while leaving other configuration settings unencrypted, providing a more focused approach to security.
5. **Backup and Restore Options:** The tool provides options to backup and restore encryption keys, allowing users to recover encrypted data in case of key loss or corruption. These backups and restore features help ensure the availability and integrity of encrypted data.