



Cybersecurity

AZURE FREE DOMAIN - Day 2 Activity Guide

Secure Your Web Application with SSL Certificates (Azure Free Domain)

Today, you will secure your web application. Specifically, you will:

1. Create a key vault.
2. Create a self-signed certificate.
3. Analyze a self-signed certificate.
4. Analyze a trusted certificate.
5. Answer review questions.

⚠ Note: Since you selected a free Azure domain, Azure will provide a trusted certificate for your domain. You will analyze the difference between self-signed and trusted SSL certificates.

Resources

- [Azure Key Vaults](#)
- [What is a self signed certificate?](#)
- [Binding Certificates in Azure](#)
- [Azure App Service Managed Certificates](#)
- [Azure App Service Documentation](#)
- If Microsoft Support is needed, visit [How to open a support ticket](#)

Getting Started / Prerequisites

Before you begin Day 2, you are required to have completed the following tasks from Day 1:

- Created your own web application.
- Created your own unique domain name.

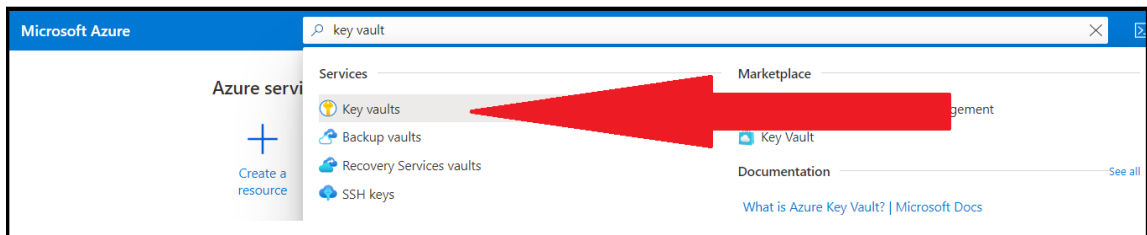
- Deployed a Docker container to your web application.
- Customized your web application with your own unique content.

Instructions

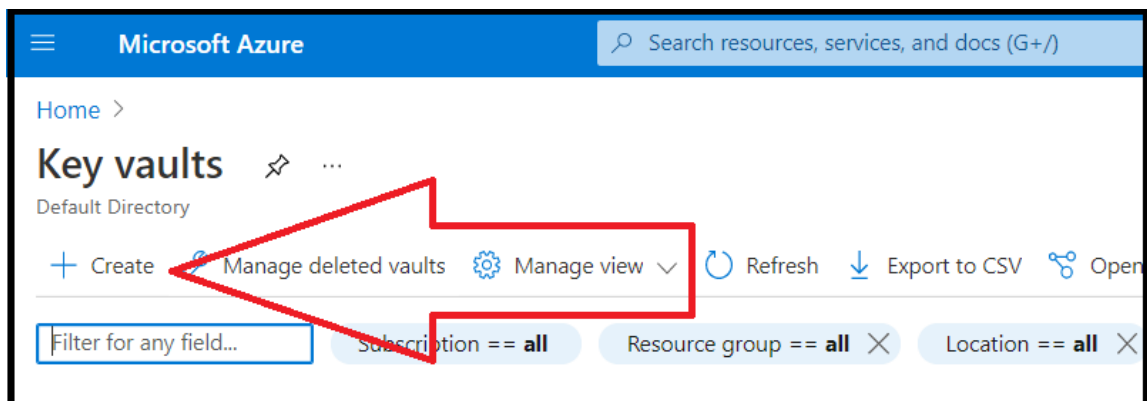
Part 1: Create a Key Vault

In this first part, you will create an Azure key vault. To do so, complete the following steps:

1. Begin by logging in to the Azure portal: <https://portal.azure.com>.
 - Make sure that you're logged in to your personal Azure account (not @Cyberxsecurity), where your Cloud Security–unit VMs are located.
2. Select “Key vaults” from the Azure search field at the top of the page, as the following image shows:



3. Select “+ Create” from the Key Vault page to create your key vault, as the following image shows:



4. On the “Create key vault” tab, make the following selections:

- Subscription/Resource Group: Select the same subscription and resource groups that you selected on Day 1.
- Key Vault Name: Choose a key vault name, such as `project1-KeyVault`. (Note: This name must be globally unique, so you will be prompted to choose a different name if the one you enter has been used before.)
- Region: Select the same region that you selected on Day 1.
- Pricing tier: Select the “Standard” tier.
- Leave the default options for all of the other tabs (Access Policy, Networking, Tags).

The following image shows the completed “Create key vault” tab:

Create key vault

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *

Resource group *

Instance details

Key vault name *

Region *

Pricing tier *

Recovery options
Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

Soft-delete ☐ Enabled

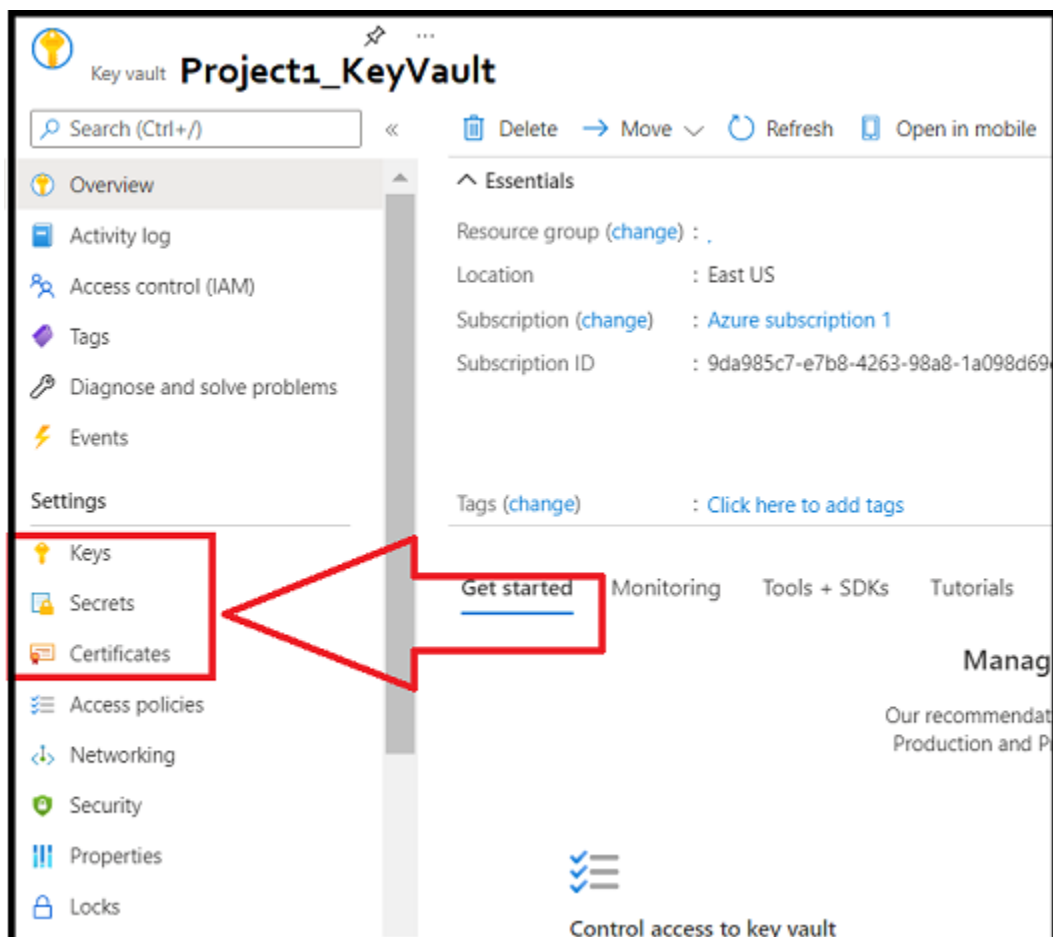
Days to retain deleted vaults *

Purge protection ☐ ☒ Disable purge protection (allow key vault and objects to be purged during retention period)
☐ Enable purge protection (enforce a mandatory retention period for deleted vaults and vault objects)

- Finally, select “Review + Create” to create your key vault.

5. After your key vault has been created, select your new resource to view your new key vault.
6. Preview the options available on your key vault to store secure information, including:
 - Keys
 - Secrets
 - Certificates

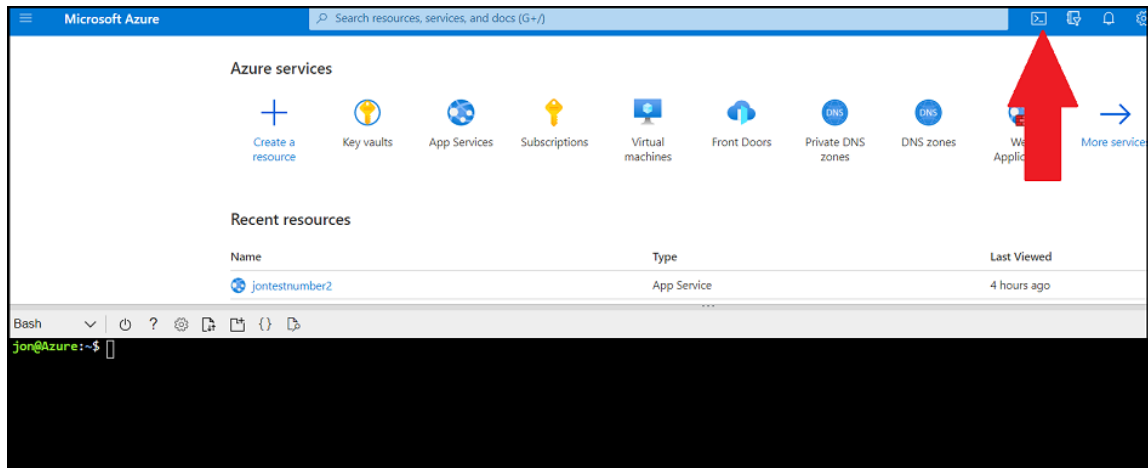
The following image shows these options:



Part 2: Create a Self-Signed Certificate

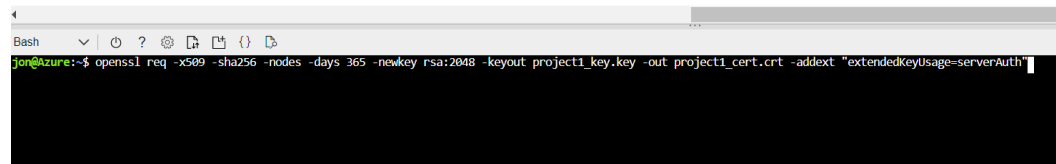
In this second part, you will return to the command line to create a self-signed certificate using OpenSSL. To do so, complete the following steps:

1. From your Azure portal, access the same Cloud Shell that you accessed on Day 1 to load the Docker container, as the following image shows:



- From this command line, you will now use the open source cryptography and SSL/TLS “toolkit” OpenSSL (it is preinstalled).
 - Recall that during Cryptography week, we used OpenSSL to generate keys and an IV to encrypt a message.
2. Next, you will use OpenSSL to generate a self-signed certificate.
 - A self-signed certificate is a certificate that has not been signed by a certificate authority.
 - These certificates are simple to create and have no expense.
 - We will explore their advantages and disadvantages in today's review questions.
 3. From the command line, enter the following command: `openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout <privatekeyname.key> -out <certificatename.crt> -addext "extendedKeyUsage=serverAuth"`
 - For example: `openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout project1-key.key -out project1-cert.crt -addext "extendedKeyUsage=serverAuth"`

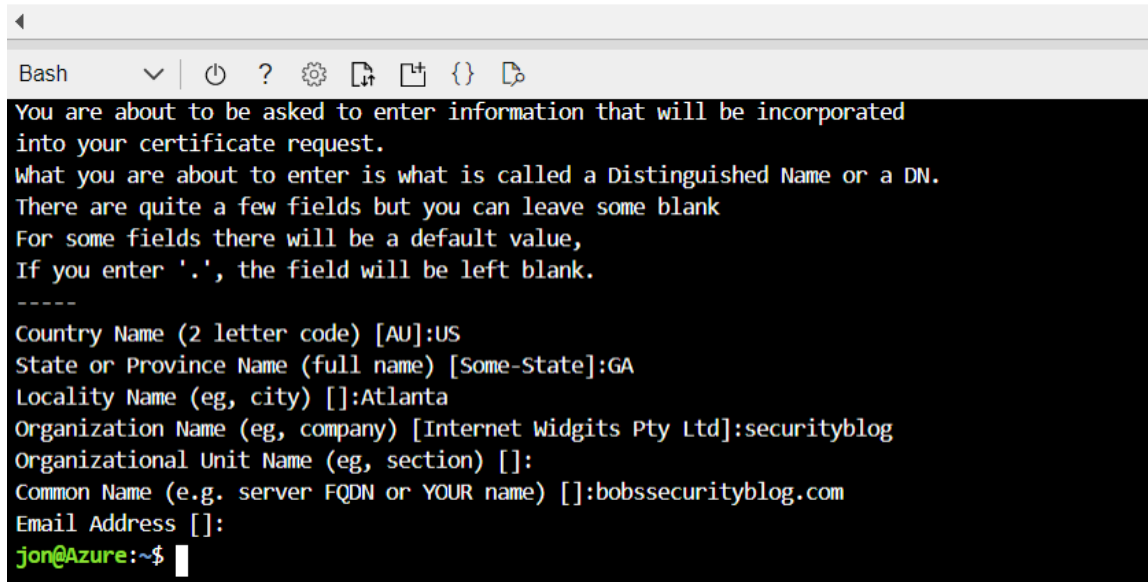
- The following image shows this step:

A terminal window screenshot showing a command being entered. The prompt is 'jon@Azure:~\$'. The command is 'openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout project1_key.key -out project1_cert.crt -addext "extendedKeyUsage=serverAuth"'. The command is partially entered, with the cursor at the end of the string.

- We added the following options:
 - `-x509`: Indicates for OpenSSL to create an SSL certificate.
 - `-sha256`: Uses the sha256 hashing algorithm.
 - `-nodes`
 - `-days 365`: States the certificate will be valid for one year.
 - `-newkey rsa:2048`: Uses a 2048-bit RSA key.
 - `-keyout project1-key.key`: The outputted name of the private key.
 - `-out project1-cert.crt`: The outputted name of the certificate.
 - `-addext "extendedKeyUsage=serverAuth"`: Indicates how a public key can be used.
 - Refer to the following [document](#) for additional information on OpenSSL options.
4. After pressing Enter, you will be asked several questions about your certificate. Answer the following:
- **Country Name (2 letter code) [AU]**: Enter your country.
 - **State or Province Name (full name) [Some State]**: Enter your state.
 - **Locality Name (e.g., city) []**: Enter your city.
 - **Organization Name (e.g., company) [Internet Widgits Pty Ltd]**: Enter "Student".
 - **Organizational Unit Name (e.g., section) []**: Leave blank by pressing Enter.
 - **Common Name (e.g., server FQDN or YOUR name) []**: Enter your full domain name, such as "bobsblog.com".

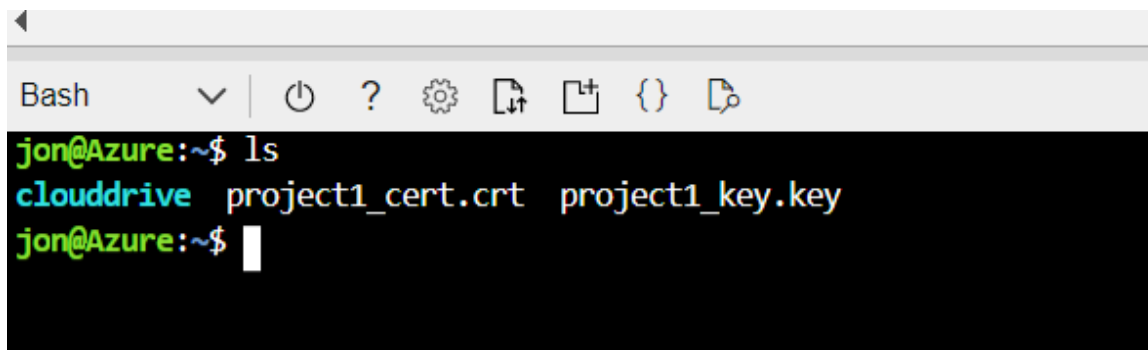
- **Email Address []:** Leave blank by pressing Enter.

The following image shows this step:



```
Bash
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:GA
Locality Name (eg, city) []:Atlanta
Organization Name (eg, company) [Internet Widgits Pty Ltd]:securityblog
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:bobssecurityblog.com
Email Address []:
jon@Azure:~$
```

5. Now, view your newly created key (.key) and certificate (.cert) by running `ls`, as the following image shows:



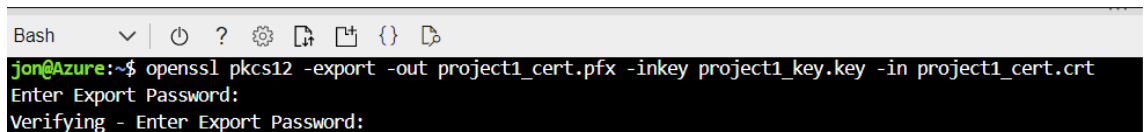
```
Bash
jon@Azure:~$ ls
cloudrive  project1_cert.crt  project1_key.key
jon@Azure:~$
```

- Note that Azure requires a PFX format for its certificates.
 - The PFX format is the server certificate and the private key combined into a single encrypted file.
6. To create a PFX format, run the following command: `openssl pkcs12 -export -out <new_certificatename.pfx> -inkey <keyname.key> -in <certificename.crt>`

- For example: `openssl pkcs12 -export -out project1-cert.pfx -inkey project1-key.key -in project1-cert.crt`
- We added the following options:
 - `pkcs12`: Indicates for OpenSSL to create a PFX certificate.
 - `-export -out project1-cert.pfx`: States what to name the PFX file.
 - `-inkey project1-key.key`: This is the current private key that you are importing.
 - `-in project1-cert.crt`: This is the current certificate that you are importing.


7. After pressing Enter, you will be prompted for a password to encrypt your PFX key.

- Don't forget your password, as you will be prompted for it again shortly!
- The following image shows this step:



```
Bash
jon@Azure:~$ openssl pkcs12 -export -out project1_cert.pfx -inkey project1_key.key -in project1_cert.crt
Enter Export Password:
Verifying - Enter Export Password:
```

8. View your new PFX certificate by running `ls`, as the following image shows:



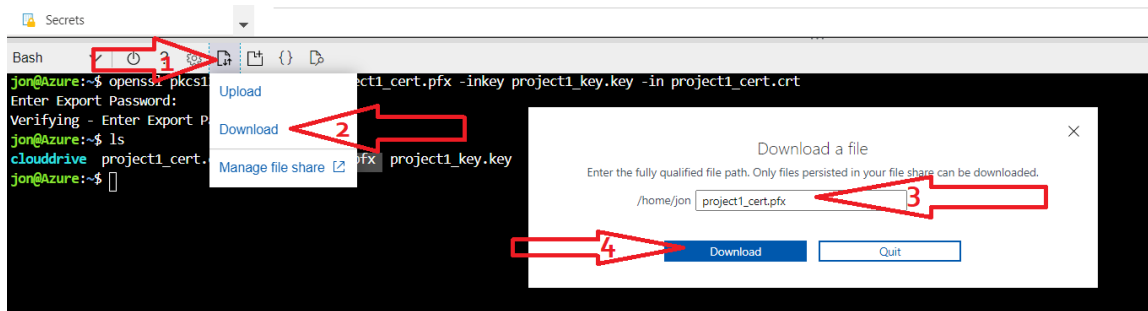
```
jon@Azure:~$ ls
clouddrive project1_cert.crt project1_cert.pfx project1_key.key
jon@Azure:~$
```

9. To download your new PFX certificate, complete the following four steps:

- (1) Click the “Upload/Download” icon in the toolbar above your Cloud Shell window.
- (2) Select “Download.”

- (3) Enter the name of your PFX certificate in the "Download a file" window.
- (4) Click "Download."

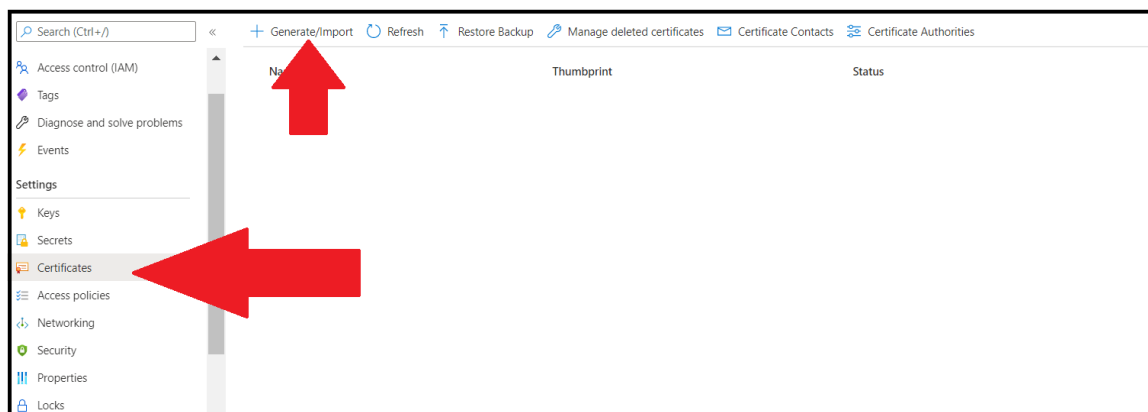
The following image shows these steps:



Part 3: Analyze a Self-Signed Certificate

In this part, you will use Azure to import and analyze self-signed certificates. To do so, complete the following steps:

1. From the Azure Portal, select "Key Vaults."
2. From your key vault, select "Certificates" and then "+ Generate/Import," as the following image shows:



3. On the "Create a certificate" page, select the following:


- **Method of Certificate Creation:** Import
- **Certificate Name:** project1PFX-cert
- **Upload Certificate File:** Select your PFX certificate (it's likely in your Downloads folder)
- **Password:** Enter the password that you created in Part 2

The following image shows these steps:

4. Select “Create” to upload your certificate.

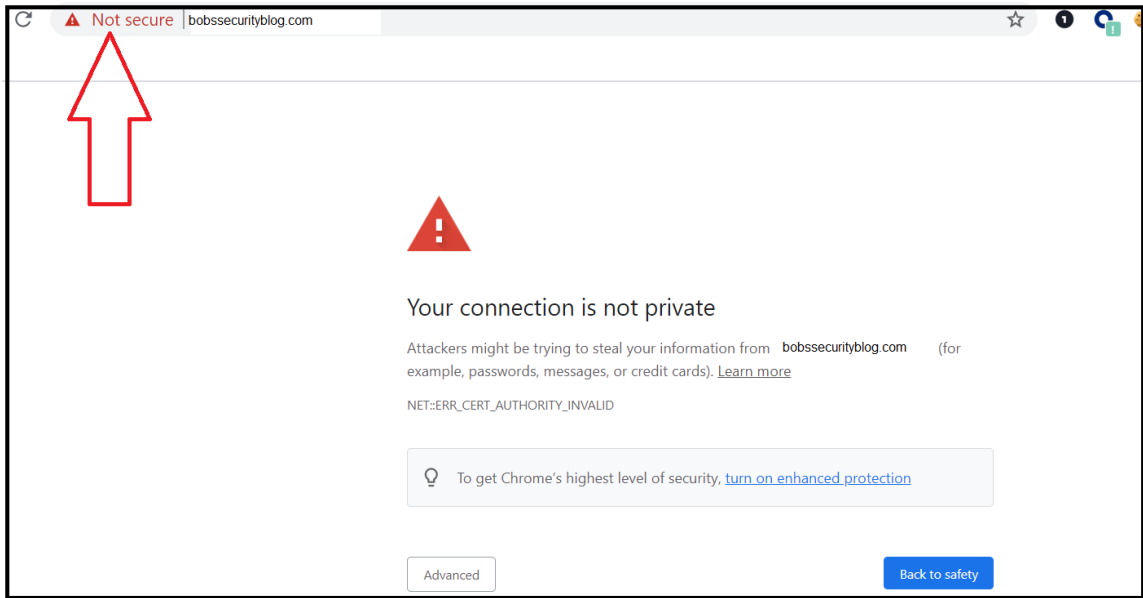
- The following success message should appear to confirm that your PFX certificate has been uploaded to your key vault:

Name	Thumbprint	Status	Expiration date
Completed			
project1PFX-cert	0AB42D8ECD6F62AA7E6AE8A2492A77D45E233E91	✓ Enabled	8/11/2022

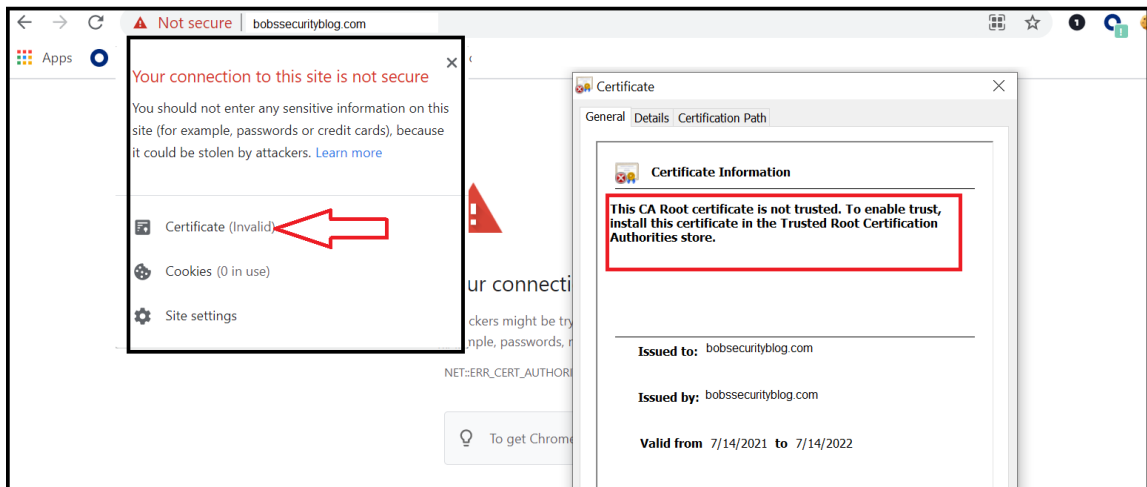
5.  Normally, after uploading a certificate, you would add it to your web application. Since you have selected the free domain option and Azure has already provided a certificate, you will instead analyze a mock self-signed certificate.

6. Navigate to the following webpage: <https://self-signed.badssl.com/>.

- This webpage represents how your application would operate if you had added your self-signed certificate to it.
- Did your browser return a “Your connection is not private” error like the one shown in the following image?



- Note that this image is from the Chrome browser; the message may look slightly different depending on your browser.
7. Let's examine this webpage's certificate. Click “Not secure” in the search bar if you are in Chrome, or a similar message depending on your browser, as shown in the following image:



- After selecting “Not secure,” select “Certificate (Invalid)” from the menu to examine the certificate.
- Note the reason for the error based on the message on the certificate. This is due to the fact that the certificate was not created by a trusted CA—it would have been created by you.

⚠ Checkpoint ⚠

Before continuing, make sure that you have completed the following critical tasks:

- ✓ Created your Azure key vault.
- ✓ Created a self-signed certificate using Open SSL.

Part 4: Analyze a Trusted SSL Certificate

In this part, you will analyze a trusted SSL certificate. An advantage of using Azure's free domain, “azurewebsites.net,” is that Microsoft provides the secure SSL certificate. To analyze this certificate, complete the following steps:

1. Open a browser, and access the domain that you created on Day 1 (*webpage@azurewebsites.net*).
 - Did you encounter any errors, like you did with the self-signed certificate?

2. Click on the lock icon in the top left of your browser (the left-hand side of the search bar) to analyze your certificate and its details.

You will answer review questions about your certificate in the next part of the activity.

Part 5: Answer Review Questions

- Open your copy of the [Project 1 Technical Brief](#) review questions, and answer the Day 2 review questions.
 - Note that you will submit this document as one of your deliverables at the end of the project.

Day 2 Milestone

In today's class, you:

1. Created a key vault.
2. Created a self-signed certificate.
3. Analyzed a self-signed certificate.
4. Analyzed a trusted certificate.
5. Answered review questions.

Completing these steps required you to leverage your terminal, systems administration, cloud, cryptography, and networking skills. This is an impressive set of tools to have in your toolkit!