1. Introduction:

This document gives an overview of the FIDO2 registration/authentication process using GreenRADIUS components - GreenRADIUS FIDO2 server and GreenRADIUS Relying Party (Client) libraries, and provides details about integrating these Relying Party libraries with a web application.

2. FIDO2 Registration/Authentication:

2.1 FIDO2 Registration:

2.1.1 Flow Diagram:

https://greenrocketsecurity-my.sharepoint.com/:i:/p/support/EX4z0XRy4SVJvD6Jxv3uyFcBqUP-8tGO0BMzSydCRGCGrQ?e=py5nQH

2.1.2 Steps:

1. On the Relying Party (RP) Web App, a user will initiate a request to register a FIDO2 authenticator.

2. The RP Web App (through the **JavaScript library**) forwards the registration request to the RP server.

3. The RP server instantiates and initializes the **PHP library** and this library will forward the registration request to the **GreenRADIUS FIDO2 server**.

4. The FIDO2 server creates an instance of **publicKeyCredentialCreationOptions**, containing User Information , Relying Party Information and a Challenge buffer. The challenge is randomly generated in order to ensure the security of the registration process.

5. The server sends the publicKeyCredentialCreationOptions instance back to the PHP library (RP server).

6. The RP server obtains this response from the PHP library and passes it back to the JS library.

7. The JS library invokes the browser's **navigator.credentials.create()** API with **publicKeyCredentialCreationOptions** as input.

8. The browser communicates with the authenticator using CTAP1/2 (when communicating with an external/roaming authenticator like Yubico,etc) or directly (if it's a platform authenticator like Windows Hello, Android Platform,Apple TouchID).

   It validates the fields in **publicKeyCredentialCreationOptions** and augments it with some extra fields like origin to create a **clientDataJSON** object.

   The browser passes the **publicKeyCredentialCreationOptions** information along with the SHA256 hash of the **clientDataJSON** object - **clientDataHash** - to the authenticator by invoking **authenticatorMakeCredential()** on the Authenticator.

9. The authenticator will ask for some form of user consent.

10. The user provides the verification.

11. After verifying consent, the authenticator then creates a new asymmetric key pair and stores the private key and a globally unique credential Id for future reference.
    The public key becomes part of the attestation, which the authenticator signs over with the attestation private key. The manufacturer’s attestation certificate chain may also be returned so that the relying party can validate the device back to a root of trust.

12. The Authenticator then returns the attestationObject to the browser.

13. The browser augments the attestationObject with the clientDataJSON to form **authenticatorAttestationResponse** and sends it to the JS library

14. The JS library will forward the authenticatorAttestationResponse to the Relying Party (RP) server

15. The RP server instantiates and initializes the PHP library and this library will forward the authenticatorAttestationResponse to the GreenRADIUS FIDO2 server.

16. The FIDO2 server performs a series of verification steps to ensure the integrity of the registration process per WebAuthn specifications and Relying Party's policies.

17. The FIDO2 server responds back to the RP Server with the status of the verification and registration.

18. The RP server sends the status of registration to the JS library.

19. The JS library calls the callback function provided to it.

2.2 FIDO2 Authentication:

2.2.1 Flow Diagram:

https://greenrocketsecurity-my.sharepoint.com/:i:/p/support/EWdG3AayUYNAqYfpCXav8x0BrlBylxWKrHEAl7AAmBE3Zg?e=sxlgEU

2.2.2 Steps:

1. On the Relying Party (RP) Web App, a user will initiate a request to authenticate using a FIDO2 authenticator.

2. The RP Web App (through the **JavaScript library**) forwards the authentication request to the RP server.

3. The RP server instantiates and initializes the **PHP library** and this library will forward the authentication request to the **GreenRADIUS FIDO2 server**.

4. The FIDO2 server creates an instance of **publicKeyCredentialRequestOptions**, containing a Challenge buffer. The challenge is randomly generated in order to ensure the security of the registration process.

5. The server sends the publicKeyCredentialRequestOptions instance back to the PHP library (RP server).

6. The RP server obtains this response from the PHP library and passes it back to the JS library.

7. The JS library invokes the browser's **navigator.credentials.get()** API with **publicKeyCredentialRequestOptions** as input.

8. The browser communicates with the authenticator using CTAP1/2 (when communicating with an external/roaming authenticator like Yubico,etc) or directly (if it's a platform authenticator like Windows Hello, Android Platform,Apple TouchID).

   It validates the fields in publicKeyCredentialRequestOptions and augments it with some extra fields like origin to create a clientDataJSON object.

   The browser passes the **publicKeyCredentialRequestOptions** information along with the SHA256 hash of the clientDataJSON object - **clientDataHash** - to the authenticator by invoking **authenticatorGetCredential()** on the Authenticator.

9. The authenticator finds a credential that matches the Relying Party ID and prompts the user to consent to the authentication.

10. The user provides the verification.

11. After verifying consent, the authenticator signs the clientDataHash and authenticatorData using the previously generated private key during the registration phase, thus creating an **assertionObject**.

12. The Authenticator then returns the assertionObject to the browser.

13. The browser augments the assertionObject with the clientDataJSON to form **authenticatorAssertionResponse** and sends it to the JS library

14. The JS library will forward the authenticatorAssertionResponse to the Relying Party (RP) server

15. The RP server instantiates and initializes the PHP library and this library will forward the authenticatorAssertionResponse to the GreenRADIUS FIDO2 server.

16. The FIDO2 server performs a series of verification steps to ensure the integrity of the authentication process per WebAuthn specifications and Relying Party's policies.

17. The FIDO2 server responds back to the RP Server with the status of the assertion verification results.

18. The RP server sends the status of authentication to the JS library.

19. The JS library calls the callback function provided to it.

3. GreenRADIUS FIDO2 Relying Party Libraries:

3.1 Relying Party Client library (fido2-lib.js):

This library interacts with the **navigator.credentials.create()** and **navigator.credentials.get()** APIs defined by the Webauthn specification for creating and using credentials.

3.1.1 Integrating the Javascript library with existing application webpages:

1. Include the JS library in a web page using the <script> tag, as below:

   ```
   <script src="/path/to/library/file"></script>
   ```

e.g. `<script src="fido2-lib.js"></script>`

2. Once the library is included, the FIDO2 registration/authentication functions provided by the library can be used.

3. For the **FIDO2 registration** operation, make a call to the **registerUser()** function. This function requires the following parameters:

    a. **user**
        - the username of the user trying to login.

    b. **password**
        - the password of the user trying to login

    c. **token_label**
        - a label to identify a token

    d. **authApiUrl**
        - the URL of the authentication API of the RP application (e.g. /my-rp/auth-apis/auth.php)

    e. **callbackFunc**
        - This is a function that will get called, from within the library, once the result (success/fail) of the registration operation is obtained.

        - This callback function should have the following parameters and perform the desired action based on the values set for these parameters .
            I. authStatus - this will be set to **0**/**1** by the library based on the result of the operation.
            II. authDetails - the details regarding the result of the operation will be set in this parameter
            III. username - the username in the request will be set in this parameter

        - Consider the following example of a callback function provided to the **registerUser()** function of the library-

```
function callBackFunction(authStatus, authDetails, username)
{
        if(authStatus == 1)
        {
                window.alert('Login success for user '+username);
                window.location = "/demo-site/account.php";
        }
        else
        {
                window.alert('Login failed for user '+username+'.
                Details: '+authDetails);
        }
}
```

4. For the **FIDO2 authentication** operation, make a call to the **authenticateUser()** function. This function requires the same parameters as for the registerUser() function, except the token_label. Please refer to the above mentioned parameters.

3.2 Relying Party Server library (GrsFido2ClientService.php):

This library relays the FIDO2 request/response between the Javascript library and the GreenRADIUS FIDO2 server.

3.2.1 Requirements for the PHP library:

1. The client PHP library requires configuration values like FIDO2 server address, Relying Party identifier, Client ID, Shared Secret.

2. These values need to be maintained somewhere (file, database, etc.) by the RP application and fetched when required.

3.2.2 Integrating the PHP library with RP application authentication API:

1. Include the RP server (PHP) library in an authentication API using the following syntax:

   require ‘**/path/to/library/file**’;

   e.g. require '**./GrsFido2ClientService.php'**;

2. Create an instance of the **GrsFido2ClientService** class and pass the following initialization values:

   ```
   $fido2Service = new GrsFido2ClientService ( $fido2Server, $relyingParty, $clientId,
   $sharedSecret );
   ```

   a. **$fido2Server** - the IP address/hostname of the GreenRADIUS FIDO2 server (e.g. 10.61.0.222)

   b. **$relyingParty** - the relying party identifier, a valid domain string that identifies the FIDO2 Relying Party (e.g. fido2demo.com)

   c. **$clientId**

   d. **$sharedSecret**

3. The RP application's authentication API should fetch the above values from a config file / database / etc. and pass it to the library while creating the instance of it.

4. Once the instance (object) is created, invoke the **proxyFido2Request()** method of the class. This method requires the FIDO2 request payload as a parameter.

   ```
   $fido2Service->proxyFido2Request($reqData['fido2Payload']);
   ```

   The authentication API should extract the 'fido2Payload' parameter from the FIDO2 request and pass it on to the `proxyFido2Request` method.

5. The above method of the library will communicate with the GreenRADIUS FIDO2 server and return the appropriate response from the server.