**SECURE CODE ANALYSIS REPORT ON**

**AWS SDK LIBRARY FOR CODEIGNITER 2**

1. **THE CODE REVIEW PROCESS:**

1.1 **PSEUDOCODE OF THE LIBRARY:**

1. Include Amazon Simple Storage Services
2. Create Class Aws\_sdk
3. Constructor to fetch access key and secret key from config after object instantiation of class Aws\_sdk.
4. Access key and secret key are stored in an associative array.
5. Return access key and secret key
6. Check for duplicate object in bucket.
7. If object exist, append UNIX timestamp to filename and return result.
8. Else If there is no duplicate of object then create a bucket.
9. Wait until bucket is accessible.
10. Upload an object with the following requirements: Bucket parameter, Key, ACL, Source file and Content type.
11. Wait until object is accessible.
12. Return result.

**1.2 CODE DESCRIPTION:**

The Aws sdk library for codeigniter 2 is a library that allows users to upload object to their s3 Amazon storage.

The code begins with an inclusion of the Amazon Simple Storage’s dynamic client class “use Aws\S3\S3Client;” – This allows for commands or ability to carry out specific API operation with the use of magic methods e.g. createBucket(), putObject() etc.

Class Aws\_sdk contains all methods and properties that gives the library its specific functionality for uploading object to s3 bucket.

***Methods in class Aws\_sdk:***

Access key and secret key are required for validation and access, a constructor retrieves the keys from the config file and stores keys into an array.

The **saveObject()** method takes one array parameter. This method checks for a duplicate object in the s3 bucket by using the magic method “*doesObjectExist()*” and parameter details(Bucket, key) as criteria. If there is a duplicate object, the saveObject() method appends a unix timestamp to the filename and return the result to the user with the magic method “*putObject()*”.

The **saveObjectInBucket()** method takes one array parameter. This methods contains a few other magic methods:

1. *createBucket():* This magic method creates bucket and takes one parameter for the name of bucket. The method was used in a try and catch exception in case something prevents bucket from been created.
2. *waitUntil():* This magic method waits for the created bucket or object until it is accessible. It takes two parameters. One to check the existence of bucket, and the other for the name of the bucket. This method was used in a try and catch exception in case something went wrong while polling the bucket or object.

After creating and polling bucket, information needed to save object in bucket are stored in variables. Information such as: file key, path, extension, mime and mime type. After this information are provided, the saveObject() method saves the object into the bucket. The saveObject() takes in one array parameter with the following required keys: Bucket name, key, ACL(Access Control List), Source File and Content Type. This is done in a try and catch exception in case something goes wrong while saving the object.

After object is saved, the waitUntil() magic method polls the object in a try and catch exception before returning the result.

**1.3 SECURITY CHECK:**

1. ***USER VALIDATION:*** For better security, double validation may be required. Aside the access key and secret key, a popup screen that request the login credentials of the user can greatly reduce the penetration ability of an intruder. In other words, when the access key and secret key are fetched from config directory it could be accompanied with a screen that requests the user’s AWS login credentials, more like an AWS login api request from Amazon.
2. ***ENCRYPTION:*** All methods and properties in the library that contains vital information are public. Sensitive information like the access key and secret key are better kept in private accessibility mode for less accessibility which in turn increases security. If by any reason the access key should be kept public then it can be encrypted for better security.

Better object protection also requires encryption. An inclusion of a request for the server-side encryption will encrypt data as it writes to disk and decrypts it when it is access. This feature, if added, will further increase the security and protection of uploaded object to s3 storage.

1. ***ACCESSIBILITY:*** Access key and secret key can always be accessed from the config file, there shouldn’t be any need to create a variable or array in the library for the access key and secret key. Allowing users to insert the keys directly into the library would pose a reason for it to be edited. It is better practice to protect the config file and make sure its content is only available from as an extended class.
2. ***DATA VALIDATION:*** There should be provision for notification of object extension and size that can’t be uploaded to bucket for proper data validation. This would create better user experience and also improve security.

2. **REVIEW SUMMARY**

The secure code review of AWS Library for codeigniter 2 was completed on Tuesday, December 23, 2020 by Tunde Olupitan. The review was performed on code obtained from (CTO) Andrew M. Hill via github on Monday, December 22, 2020, under the address <https://github.com/fcosrno/aws-sdk-php-codeigniter>.

Information about the code structure was checked with high level overview of how things like authentication and data validation were implemented in the code. This information was used to formulate a plan for impending review.

The actual review involves a manual investigation of the PHP code in the aforementioned github address. I attempted to review the entire application and recorded my specific findings. My specific findings are presented in the next section.

3. **FINDING SUMMARY**

For this library, four high level issues were found related to the areas of user validation, encryption, accessibility and data validation. Mitigating actions should be considered to further improve the library. Details are given below.

The figures below graphically outline the review team's findings by both category and risk level.

4. **FINDING DETAILS**

*Level of risk in the library*:

***High:*** Poor Data Validation poses a high risk for the S3 pocket. This is because file extension requirement to be uploaded aren’t explicitly stated. For instance, A .json file extension that is being uploaded could contain a malicious program. Detecting this from the point of upload will be a really good practice.

Poor user validation poses a threat of allowing an unauthorized user with the secret key and access key to upload a file to a user’s s3 storage without the user being aware. A function can be included in the library to tackle this issue.

***Medium:*** exposure of the access key and secret key without encryption pose a medium risk of access. Method of access also poses a medium risk of access.

4.1 Inconsideration of the need for user login before upload.

**Category:** User validation

**Weakness:** The library allows anyone with access key and secret key to upload a file, but it does not request which user wants to make an upload.

|  |  |  |  |
| --- | --- | --- | --- |
| SOURCE FILE | LINE NUMBER | DESCRIPTION | RISK |
| Application/libraries/Aws\_sdk.php | 28 | On line 28, Instead of going straight to check if the file exist in bucket, a function that validates the user is necessary to be sure that it’s the user making an upload to his s3 storage. | High |

4.2 Validation of data issue.

**Category:** Data validation

**Weakness:** The library allows any file extension and size, but it does not prompt for extension and file size restriction.

|  |  |  |  |
| --- | --- | --- | --- |
| SOURCE FILE | LINE NUMBER | DESCRIPTION | RISK |
| Application/libraries/Aws\_sdk.php | 59 | On line 59, The file extension is retrieved and saved in a variable. There is no condition to check if the file extension is acceptable or not. This leaves the user’s s3 storage to be subject to risk of unacceptable file extension and size. | High |

4.3 Accessibility and poor transfer of sensitive information.

**Category:** Accessibility.

**Weakness:** The library allows users to directly input a secret key and access key.

|  |  |  |  |
| --- | --- | --- | --- |
| SOURCE FILE | LINE NUMBER | DESCRIPTION | RISK |
| Application/libraries/Aws\_sdk.php | 11, 12 | On line 11 and 12, In the associative array, there is a space for access key and secret key input that aren’t encrypted. This makes accessibility easy and this also leaves the library to be compulsorily edited. Malicious users could use this medium to edit other functions in the library posing a risk of an unauthorized use.  Data encryption can be | Medium |

4.4 Necessity for the protection of object by encryption

**Category:** Encryption

**Weakness:** No provision for the encryption of data to be uploaded.

|  |  |  |  |
| --- | --- | --- | --- |
| SOURCE FILE | LINE NUMBER | DESCRIPTION | RISK |
| Application/libraries/Aws\_sdk.php | 57-72 | On line 57-72, there is a function that upload object to bucket with no data encryption. Amazon provides server-side encryption for data. Inclusion of this feature to the library will encrypts object as it writes it to disks in its data centers and decrypts the object when you access it. | Medium |