oauth2-server SOS Fund Audit Fix Log

Identified Vulnerabilities

A: Authorization codes are authenticated with RSA in “ECB mode” (High)

Alex Bilbie (oauth2-server): This has been addressed in two different ways:

For developers who upgrade to 6.0.* of the library an encryption key is now required and that is used to encrypt the payload using the Defuse/Crypto library. This has completely resolved the issue for 6.0.* users.

For 5.1.* developers who make a single line of code update to pass in an encryption key into the AuthorizationServer instance benefit from the above too.

For 5.1.* developers who don’t pass in an encryption key the library now adds a random number of bytes of padding to the array then the contents is shuffled before it is encrypted using the old RSA key encryption method. This has mitigated the specific issue highlighted in the audit.

Taylor: I’m glad Defuse/Crypto is helpful here. :-) The use of Defuse/Crypto’s API looks good, although you should be aware that “encryptWithPassword” is intentionally slow--it runs some tens of thousands of rounds of PBKDF2 in each call to generate a key from the password. So if performance is an issue it would be better to generate and save a “Key” object.

B: Insufficient validation of code_challenge field in src/Grant/AuthCodeGrant.php (Medium)

Alex Bilbie (oauth2-server): During the request validation phase the code_challenge parameter is checked against a regular expression to ensure it conforms to the character types and length as required in RFC7636.

Taylor: The fix looks good, although it would be more correct to use \A and \z instead of ^ and $, since with $ it allows a newline at the end (e.g. str_repeat(“A”, 43) . “\n”).

C: Invalid/rejected “scope” names are reflected into the output (Low)

Alex Bilbie (oauth2-server): This has been resolved as of release 5.1.4 by passing the scope field through the following code: `htmlspecialchars($scope, ENT_QUOTES, 'UTF-8', false)`

Taylor: This looks good, as long as the error string is only used for outputting to HTML and never used for something insane that would need the string to be escaped differently (e.g. database query etc).

D: Keys are saved to a temporary directory using predictable filenames (Medium)

Alex Bilbie (oauth2-server): This has been resolved as of release 5.1.4 by ensuring the process has exclusive ownership (600 chmod permission) of the file or throwing an error.

Taylor: There's a potentially-exploitable race condition in the code that saves the file (saveKeyToFile() in CryptKey). The file is created, the key is written, and *then* the permissions are set. Instead, create the file, set the permissions (and make sure that worked), and finally write the key. Otherwise the key might be accessible in the moment of time between when the key is written and the permissions are set. On a more pedantic note, the filename of the key is its SHA1 hash, which is safe as long as the key is used for Defuse/Crypto (since that library never uses SHA1(key)), but another crypto library could conceivably use SHA1(key) as its actual key, which would be exposed in the filename. It would be best to not to expose any function (even a cryptographic one) of the key.

E: Leakage of code_challenge field (Medium)

Alex Bilbie (oauth2-server): If the user has opted into the improved encryption through the use of an encryption key (see A) the field will be better encrypted inside the authorization code payload.

If the user has not opted into improved encryption then the authorization code payload is now shuffled and has a random amount of padding included to ensure the structure of the payload is no longer guessable.

Taylor: The new Defuse/Crypto code looks good, but I don't think the shuffling and random padding help mitigate this issue. The problem in this issue was that the private key was used when encrypting and the public key was used while decrypting (instead of how it should be, encrypt to the public key, decrypt with the private key). Obtaining the public key (which is not intended to be hard, it's supposed to be public) would let you decrypt the messages just like normal, and any randomization underneath wouldn't get in the way. The best solution is probably to keep driving everyone to update to the newer crypto.

Miscellaneous Issues

1: Validate expected fields are being parsed (info)

Alex Bilbie (oauth2-server): This has not been addressed. Whilst the expected fields are indeed defined in the IANA registry, the OAuth 2.0 specification is an open framework which is used as a building block for other standards (such as OpenID Connect) and custom implementations and the library author does not want to unnecessarily restrict the ability to build on top of this library by restricting the fields available in a request.

2: Add code to catch bad returns from getRedirectUri() (Info)

Alex Bilbie (oauth2-server): This has not been addressed.