nearform / **fast-jwt** Public

Code | Issues 7 | Pull requests 6 | Actions | Security and quality 5 | Ins

Commit de12105

antoatta85 and **Copilot** authored yesterday · 6 / 6 · Verified

```
fix: cache confusion risk with composite-key approach (#587)

* fix: GHSA-rp9m-7r4c-75qg cache confusion risk with composite-key approach

* Revert "fix: GHSA-rp9m-7r4c-75qg cache confusion risk with composite-key approach"

This reverts commit e12fe37.

* add warning on custom cacheKeyBuilder function in the README

* added a warning when custom cache key builder method is used

* fix lint errors

* Apply suggestion from @Copilot

Co-authored-by: Copilot <175728472+Copilot@users.noreply.github.com>

* resolving copilot comments

---------

Co-authored-by: Copilot <175728472+Copilot@users.noreply.github.com>
```

master (#587) · v6.2.0

1 parent 80b49c7 commit de12105

3 files changed +64 -3 lines changed

Top

Filter files...

- README.md
- src
  - verifier.js
- test

verifier.spec.js

3 files changed +64 -3 lines changed

Search within code

README.md

```
@@ -176,7 +176,7 @@ Create a verifier function by calling `createVerifier`
and providing one or more
176 176
177 177  - `clockTolerance`: Timespan in milliseconds is the tolerance to apply to the
        current timestamp when performing time comparisons. Default is `0`.
178 178
179     - - `cacheKeyBuilder`: The function that will be used to create the [cache's key]
        (#caching) for each token. To mitigate the risk of leaking sensitive
        information and generate collisions, [a hashing function](./src/utils.js) is
        used by default.
    179 + - `cacheKeyBuilder`: The function that will be used to create the [cache's key]
        (#caching) for each token. To mitigate the risk of leaking sensitive
        information and reduce the risk of collisions, [a hashing function]
        (./src/utils.js) is used by default. When using a custom function, users must
        be aware of the possible risks. Check the [Caching section](#caching) for more
        information.
180 180
181 181  The verifier is a function which accepts a token (as Buffer or string) and
        returns the payload or the sections of the token.
182 182
@@ -280,7 +280,13 @@ The default `cacheKeyBuilder` is a function that hashes
the token. This provides
280 280
281 281  For a detailed discussion about it, take a look at [this issue]
        (https://github.com/nearform/fast-jwt/issues/503).
282 282
283     - > **_Note:_** Errors are not cached by default, to change this behaviour use
        the `errorCacheTTL` option.
    283 + > [!NOTE]
    284 + > Errors are not cached by default, to change this behaviour use the
        `errorCacheTTL` option.
    285 +
    286 + > [!WARNING]
```

```
287 + > Setting up a custom `cacheKeyBuilder` method which does not properly create unique keys for different tokens can lead to cache collisions. This could cause tokens to be mis-identified during the verification process. For more information check [this GHSA](https://github.com/advisories/GHSA-rp9m-7r4c-75qg).
288 + > Users willing to set up a custom cache key builder method have to make sure it works properly.
289 +
284 290
285 291   ## Token Error Codes
286 292
```

src/verifier.js

```
@@ -516,6 +516,18 @@ module.exports = function createVerifier(options) {
516 516
517 517     const allowedCritHeadersSet = new Set(allowedCritHeaders || [])
518 518
519 +   const cache = createCache(cacheSize)
520 +
521 +   if (cache && options?.cacheKeyBuilder) {
522 +     process.emitWarning(
523 +       'A custom cacheKeyBuilder is in use with caching enabled. ' +
524 +         'Cache key collisions can lead to identity/authorization bypass. ' +
525 +         'Make sure your cacheKeyBuilder generates unique keys for different tokens. ' +
526 +         'See https://github.com/nearform/fast-jwt/security/advisories/GHSA-rp9m-7r4c-75qg',
527 +       { code: 'FAST_JWT_CACHE_KEY_BUILDER_SECURITY_RISK' }
528 +     )
529 +   }
530 +
519 531     // Add validators
520 532     const validators = []
521 533
@@ -581,7 +593,7 @@ module.exports = function createVerifier(options) {
581 593       isAsync: keyType === 'function',
582 594       validators,
583 595       decode: createDecoder({ complete: true }),
```

```
584       -     cache: createCache(cacheSize),
      596 +     cache,
585   597       requiredClaims,
586   598       allowedCritHeaders: allowedCritHeadersSet,
587   599       cacheKeyBuilder
```

test/verifier.spec.js

```
@@ -1196,6 +1196,49 @@ test('caching - sync - custom cacheKeyBuilder', t
=> {
1196  1196    t.assert.ok(verifier.cache.get(invalidToken)[0] instanceof TokenError)
1197  1197  })
1198  1198
      1199 + test('caching - sync - custom cacheKeyBuilder emits security warning', async
               t => {
      1200 +   let onWarning
      1201 +   const warningPromise = new Promise(resolve => {
      1202 +     onWarning = w => {
      1203 +       if (w.code === 'FAST_JWT_CACHE_KEY_BUILDER_SECURITY_RISK') {
      1204 +         resolve(w)
      1205 +       }
      1206 +     }
      1207 +     process.on('warning', onWarning)
      1208 +   })
      1209 +
      1210 +   t.after(() => process.off('warning', onWarning))
      1211 +
      1212 +   createVerifier({ key: 'secret', cache: true, cacheKeyBuilder: id => id })
      1213 +
      1214 +   const timeout = new Promise((_, reject) =>
      1215 +     setTimeout(() => reject(new Error('Timed out waiting for
               FAST_JWT_CACHE_KEY_BUILDER_SECURITY_RISK warning')), 1000)
      1216 +   )
      1217 +
      1218 +   const warning = await Promise.race([warningPromise, timeout])
      1219 +   t.assert.equal(warning.code, 'FAST_JWT_CACHE_KEY_BUILDER_SECURITY_RISK')
      1220 +   t.assert.ok(warning.message.includes('cacheKeyBuilder'))
      1221 + })
      1222 +
```

```
+ test('caching - sync - default cacheKeyBuilder does not emit security warning', async t => {
+   let warningReceived = false
+   const onWarning = w => {
+     if (w.code === 'FAST_JWT_CACHE_KEY_BUILDER_SECURITY_RISK') {
+       warningReceived = true
+     }
+   }
+   process.on('warning', onWarning)
+   t.after(() => {
+     process.off('warning', onWarning)
+   })
+
+   createVerifier({ key: 'secret', cache: true })
+
+   // Wait a tick to allow any potential warning to be emitted
+   await new Promise(resolve => setImmediate(resolve))
+   t.assert.equal(warningReceived, false)
+ })
+
```

```
1199 1242   test('caching - async', async t => {
1200 1243     const token = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhIjoxfQ.57TF7smP9XDhIexBqPC-F1toZReYZLWb_YRU5tv0sxM'
1201 1244     const invalidToken = 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhIjoxfQ.aaa'
```

Comments 0

Please sign in to comment.