psi-4ward / **psitransfer** Public

Code Issues 72 Pull requests 34 Actions Projects Security and qua

Commit 8b547bf

psi-4ward committed last week · 7 / 7

```
fix: Harden file uploads
```

master · v2.4.3 ••• 2

1 parent 22ec552 commit 8b547bf

11 files changed +377 -24

Top

Filter files…

- .github/workflows
  - ci.yaml
- lib
  - db.js
  - endpoints.js
  - store.js
  - tusboy/handlers
    - head.js
    - patch.js
  - utils.js
- package.json
- tests
  - integration
    - middleware.test.js
  - unit

store-containment.test.js

upload-id.test.js

Search within code

.github/workflows/ci.yaml

```
@@ -0,0 +1,28 @@
+ name: CI
+
+ on:
+   push:
+     branches:
+       - master
+       - main
+   pull_request:
+
+ jobs:
+   test:
+     name: Test
+     runs-on: ubuntu-latest
+     steps:
+       - name: Checkout
+         uses: actions/checkout@v4
+
+       - name: Setup Node.js
+         uses: actions/setup-node@v4
+         with:
+           node-version: 24
+           cache: npm
+
+       - name: Install dependencies
+         run: npm ci
+
+       - name: Run tests
+         run: npm test
```

lib/db.js

```
@@ -33,7 +33,7 @@ module.exports = class DB {
```

```
33  33            }
34  34          }
35  35        };
36      -     setInterval(gc, 60*1000);
    36  +     setInterval(gc, 60 * 1000).unref();
37  37
38  38      }
39  39
```

lib/endpoints.js

```
    @@ -44,6 +44,16 @@ function sha256Hex(input) {
44  44    return createHash('sha256').update(input).digest('hex');
45  45  }
46  46
    47  + /** Decoded path segment under the /files mount (must match req.params used by tusboy). */
    48  + function decodedUploadPathSegment(req) {
    49  +   const raw = req.path.startsWith('/') ? req.path.slice(1) : req.path;
    50  +   try {
    51  +     return decodeURIComponent(raw);
    52  +   } catch {
    53  +     return null;
    54  +   }
    55  + }
    56  +
47  57  const pugVars = {
48  58    baseUrl: config.baseUrl
49  59  };
    @@ -364,6 +374,14 @@ app.get(`${ config.baseUrl }files/:fid`, async (req, res, next) => {
364 374    // Download single file
365 375    debug(`Download ${ req.params.fid }`);
366 376    try {
    377 +     if (req.params.fid.includes('++') && !utils.isSafeTusUploadId(req.params.fid)) {
    378 +       return res.status(404).send(errorPage({
    379 +         ...pugVars,
    380 +         error: 'Invalid link',
    381 +         lang: req.translations,
```

```
                382 +         uploadAppPath: config.uploadAppPath || config.baseUrl,
                383 +       }));
                384 +     }
367             385       const info = await store.info(req.params.fid); // throws on 404
368             386       const safeName = utils.toSafeBasename(info.metadata.name, info.key);
369             387       res.set('Content-Disposition', contentDispositionUtf8Filename(safeName,
    info.key));
                    @@ -414,32 +432,44 @@ app.use(`${ config.uploadAppPath }files`,
414             432
415             433       if (req.method === 'GET') return res.status(405).end();
416             434
                435 +     const fid = decodedUploadPathSegment(req);
                436 +     if (fid === null) {
                437 +       return res.status(400).end('Invalid path encoding');
                438 +     }
                439 +
417             440       // Lock bucket by PATCH /files/:sid?lock=yes
418             -     const fid = req.path.substring(1);
419             -     if(!fid.includes('++') && req.method === 'PATCH' && req.query.lock) {
                441 +     if (fid && !fid.includes('++') && req.method === 'PATCH' && req.query.lock)
    {
                442 +       if (!utils.isSafeBucketFid(fid)) {
                443 +         return res.status(400).end('Invalid bucket id');
                444 +       }
420             445         await db.lock(fid);
421             446         return res.status(204).end('Bucket locked');
422             447       }
423             448
424             -     if(['POST', 'PATCH'].includes(req.method)) {
425             -       // Restrict upload to the bucket if it is locked
426             -       if(!fid.includes('++') && db.isLocked(fid)) {
                449 +     if (['POST', 'PATCH'].includes(req.method)) {
                450 +       if (fid && !fid.includes('++') && !utils.isSafeBucketFid(fid)) {
                451 +         return res.status(400).end('Invalid bucket id');
                452 +       }
                453 +       if (fid && !fid.includes('++') && db.isLocked(fid)) {
427             454           return res.status(400).end('Bucket locked');
428             455         }
429             -       try {
```

```
430       -         const info = await store.info(fid);
431       -         // Restrict upload to the bucket if it is locked
432       -         if(info.metadata.locked) {
433       -           return res.status(400).end('Bucket locked');
434       -         }
435       -         // Restrict upload to a file which upload completed already
436       -         if(!info.isPartial) {
437       -           return res.status(400).end('Upload already completed');
          456 +       if (fid) {
          457 +         if (fid.includes('++') && !utils.isSafeTusUploadId(fid)) {
          458 +           return res.status(400).end('Invalid upload id');
438       459           }
439       -       } catch(e) {
440       -         if(! e instanceof httpErrors.NotFound) {
441       -           console.error(e);
442       -           return;
          460 +         try {
          461 +           const info = await store.info(fid);
          462 +           if (info.metadata.locked) {
          463 +             return res.status(400).end('Bucket locked');
          464 +           }
          465 +           if (!info.isPartial) {
          466 +             return res.status(400).end('Upload already completed');
          467 +           }
          468 +         } catch (e) {
          469 +           if (!(e instanceof httpErrors.NotFound)) {
          470 +             console.error(e);
          471 +             return next(e);
          472 +           }
443       473           }
444       474         }
445       475       }
        @@ -452,6 +482,9 @@ app.use(`${ config.uploadAppPath }files`,
452       482         try {
453       483           assert(meta.name, 'tus meta prop missing: name');
454       484           assert(meta.sid, 'tus meta prop missing: sid');
          485 +         if (!utils.isSafeBasename(meta.sid)) {
          486 +           return res.status(400).end('Invalid bucket id');
          487 +         }
455       488           assert(meta.retention, 'tus meta prop missing: retention');
```

```
456  489            assert(Object.keys(config.retentions).indexOf(meta.retention) >= 0,
457  490              `invalid tus meta prop retention. Value ${ meta.retention } not in
                  [${ Object.keys(config.retentions).join(',') }]`);
```

lib/store.js

```
     @@ -22,16 +22,21 @@ class StreamLen extends Transform {
22   22   class Store {
23   23
24   24     constructor(targetDir) {
25      -     this.dir = path.normalize(targetDir);
        25   +     this.dir = path.resolve(targetDir);
26   26     }
27   27
28   28
29   29     getFilename(fid) {
30      -     let p = path.resolve(this.dir, fid.replace('++', '/'));
31      -     if(!p.startsWith(this.dir)) {
        30   +     if (typeof fid !== 'string' || fid.includes('\0')) {
32   31         throw new Error('file name not in jail path. aborting');
33   32       }
34      -     return p;
        33   +     const base = this.dir;
        34   +     const resolved = path.resolve(base, fid.replace(/\+\+/g, '/'));
        35   +     const rel = path.relative(base, resolved);
        36   +     if (rel === '' || rel.startsWith('..') || path.isAbsolute(rel)) {
        37   +       throw new Error('file name not in jail path. aborting');
        38   +     }
        39   +     return resolved;
35   40     }
36   41
37   42
```

lib/tusboy/handlers/head.js

```
     @@ -14,11 +14,16 @@
14   14   //
15   15
16   16   const encodeMetadata = require('../tus-metadata').encode;
```

```
      17  + const utils = require('../../utils');
      18  + const errors = require('../errors');
17    19
18    20    module.exports = (store) => (
19    21      async (req, res) => {
20    22        res.set('Cache-Control', 'no-store')
21    23        const { uploadId } = req.params
      24  +     if (!utils.isSafeTusUploadId(uploadId)) {
      25  +       throw errors.unknownResource(uploadId)
      26  +     }
22    27        const upload = await store.info(uploadId)
23    28        // The Server MUST always include the Upload-Offset header in the
24    29        // response for a HEAD request, even if the offset is 0, or the upload
```

lib/tusboy/handlers/patch.js

```
          @@ -31,6 +31,7 @@
31    31    // data as possible.
32    32    const storeErrors = require('../store/errors');
33    33    const errors = require('../errors');
      34  + const utils = require('../../utils');
34    35
35    36    module.exports = (store, {
36    37      onComplete,
          @@ -56,6 +57,10 @@ module.exports = (store, {
56    57
57    58      const uploadId = req.params.uploadId
58    59
      60  +   if (!utils.isSafeTusUploadId(uploadId)) {
      61  +     throw errors.unknownResource(uploadId)
      62  +   }
      63  +
59    64      try {
60    65        const {
61    66          offset,
```

lib/utils.js

```
          @@ -40,9 +40,36 @@ function isSafeBasename(name, opts = {}) {
```

```
40  40      return toSafeBasename(name, '', opts) === name;
41  41    }
42  42
    43  + // Node randomUUID() v4 shape (case-insensitive).
    44  + const UUID_V4_RE =
    45  +   /^[0-9a-f]{8}-[0-9a-f]{4}-4[0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}$/i;
    46  +
    47  + /**
    48  +  * Tus resume URL id: exactly one "++", safe bucket sid, UUID key.
    49  +  */
    50  + function isSafeTusUploadId(uploadId) {
    51  +   if (typeof uploadId !== 'string' || uploadId.includes('\0')) return false;
    52  +   const parts = uploadId.split('++');
    53  +   if (parts.length !== 2) return false;
    54  +   const [sid, key] = parts;
    55  +   if (!sid || !key) return false;
    56  +   return isSafeBasename(sid) && UUID_V4_RE.test(key);
    57  + }
    58  +
    59  + /**
    60  +  * Bucket-only id (lock PATCH, middleware checks without "++").
    61  +  */
    62  + function isSafeBucketFid(fid) {
    63  +   if (typeof fid !== 'string' || fid.includes('++') || fid.includes('\0'))
        return false;
    64  +   return isSafeBasename(fid);
    65  + }
    66  +
43  67    module.exports = {
44  68      toSafeBasename,
45  69      isSafeBasename,
    70  +   isSafeTusUploadId,
    71  +   isSafeBucketFid,
    72  +   UUID_V4_RE,
46  73    };
47  74
48  75
```

package.json

```
@@ -35,7 +35,10 @@
```

```
35  35      "scripts": {
36  36        "start": "NODE_ENV=production node app.js",
37  37        "dev": "NODE_ENV=dev DEBUG=psitransfer:* nodemon -i app -i dist -i data
                app.js",
38      -     "debug": "node --inspect app.js"
    38  +     "debug": "node --inspect app.js",
    39  +     "test": "node --test 'tests/**/*.test.js'",
    40  +     "test:unit": "node --test tests/unit/*.test.js",
    41  +     "test:integration": "node --test tests/integration/*.test.js"
39  42      },
40  43      "engines": {
41  44        "node": ">= 24"
```

tests/integration/middleware.test.js

```
    @@ -0,0 +1,136 @@
1   + 'use strict';
2   +
3   + const fs = require('fs');
4   + const path = require('path');
5   + const os = require('os');
6   + const http = require('node:http');
7   +
8   + const tmpDir = fs.mkdtempSync(path.join(os.tmpdir(), 'psitransfer-int-'));
9   + process.env.PSITRANSFER_UPLOAD_DIR = tmpDir;
10  +
11  + const { test, after } = require('node:test');
12  + const assert = require('node:assert');
13  +
14  + const tusMeta = require('../../lib/tusboy/tus-metadata');
15  + const app = require('../../lib/endpoints');
16  +
17  + const TUS = { 'Tus-Resumable': '1.0.0' };
18  + const UUID = '00000000-0000-4000-8000-000000000001';
19  +
20  + function request(port, method, pathname, headers = {}, body = null) {
21  +   return new Promise((resolve, reject) => {
22  +     const req = http.request(
23  +       {
24  +         hostname: '127.0.0.1',
```

```
25 +         port,
26 +         method,
27 +         path: pathname,
28 +         headers,
29 +       },
30 +       (res) => {
31 +         const chunks = [];
32 +         res.on('data', (c) => chunks.push(c));
33 +         res.on('end', () => {
34 +           resolve({
35 +             status: res.statusCode,
36 +             body: Buffer.concat(chunks).toString(),
37 +           });
38 +         });
39 +       }
40 +     );
41 +     req.on('error', reject);
42 +     if (body != null) req.write(body);
43 +     req.end();
44 +   });
45 + }
46 +
47 + after(() => {
48 +   fs.rmSync(tmpDir, { recursive: true, force: true });
49 + });
50 +
51 + test('store.info error that is not NotFound yields 500 (instanceof guard)',
   async (t) => {
52 +   const sid = 'corruptsid';
53 +   const bucketDir = path.join(tmpDir, sid);
54 +   const fileBase = path.join(bucketDir, UUID);
55 +   fs.mkdirSync(bucketDir, { recursive: true });
56 +   fs.writeFileSync(`${fileBase}.json`, '{ not json', 'utf8');
57 +   fs.writeFileSync(fileBase, '', 'utf8');
58 +
59 +   const server = http.createServer(app);
60 +   await new Promise((resolve) => server.listen(0, '127.0.0.1', resolve));
61 +   const port = server.address().port;
62 +
63 +   try {
```

```
64  +     const fid = `${sid}++${UUID}`;
65  +     const res = await request(port, 'PATCH',
      `/files/${encodeURIComponent(fid)}`, {
66  +       ...TUS,
67  +       'Upload-Offset': '0',
68  +       'Content-Type': 'application/offset+octet-stream',
69  +     });
70  +     assert.strictEqual(res.status, 500);
71  +   } finally {
72  +     await new Promise((resolve) => server.close(resolve));
73  +     fs.rmSync(bucketDir, { recursive: true, force: true });
74  +   }
75  + });
76  +
77  + test('PATCH with encoded traversal in upload id is rejected before Tus
      handler', async (t) => {
78  +   const server = http.createServer(app);
79  +   await new Promise((resolve) => server.listen(0, '127.0.0.1', resolve));
80  +   const port = server.address().port;
81  +
82  +   try {
83  +     const pathname =
84  +       '/files/foo%2B%2B' + encodeURIComponent('../..') + '%2Fetc%2Fpasswd';
85  +     const res = await request(port, 'PATCH', pathname, {
86  +       ...TUS,
87  +       'Upload-Offset': '0',
88  +       'Content-Type': 'application/offset+octet-stream',
89  +     });
90  +     assert.ok(res.status === 400 || res.status === 404);
91  +   } finally {
92  +     await new Promise((resolve) => server.close(resolve));
93  +   }
94  + });
95  +
96  + test('POST with malicious meta.sid is rejected', async (t) => {
97  +   const server = http.createServer(app);
98  +   await new Promise((resolve) => server.listen(0, '127.0.0.1', resolve));
99  +   const port = server.address().port;
100 +
101 +   try {
```

```
102 +     const meta = {
103 +       name: 'x',
104 +       sid: '../../etc',
105 +       retention: '3600',
106 +     };
107 +     const res = await request(port, 'POST', '/files/', {
108 +       ...TUS,
109 +       'Upload-Length': '10',
110 +       'Upload-Metadata': tusMeta.encode(meta),
111 +       'Content-Type': 'application/offset+octet-stream',
112 +     });
113 +     assert.strictEqual(res.status, 400);
114 +     assert.match(res.body, /bucket|Invalid/i);
115 +   } finally {
116 +     await new Promise((resolve) => server.close(resolve));
117 +   }
118 + });
119 +
120 + test('lock PATCH with traversal bucket id is rejected', async (t) => {
121 +   const server = http.createServer(app);
122 +   await new Promise((resolve) => server.listen(0, '127.0.0.1', resolve));
123 +   const port = server.address().port;
124 +
125 +   try {
126 +     const res = await request(
127 +       port,
128 +       'PATCH',
129 +       '/files/' + encodeURIComponent('..') + '%2F' + encodeURIComponent('..') +
    '%2Fetc?lock=yes',
130 +       TUS
131 +     );
132 +     assert.strictEqual(res.status, 400);
133 +   } finally {
134 +     await new Promise((resolve) => server.close(resolve));
135 +   }
136 + });
```

tests/unit/store-containment.test.js

```
••• @@ -0,0 +1,70 @@
1 + 'use strict';
```

```
+
+ const { test } = require('node:test');
+ const assert = require('node:assert');
+ const fs = require('fs');
+ const path = require('path');
+ const os = require('os');
+
+ const Store = require('../../lib/store');
+
+ const SAMPLE_UUID = '00000000-0000-4000-8000-000000000001';
+
+ test('getFilename rejects prefix false-positive (startsWith bypass)', () => {
+   const jailParent = fs.mkdtempSync(path.join(os.tmpdir(), 'psi-jail-'));
+   try {
+     const base = path.join(jailParent, 'conf');
+     fs.mkdirSync(base, { recursive: true });
+     fs.writeFileSync(path.join(jailParent, 'config.production.js'), 'x');
+     const store = new Store(base);
+     assert.throws(() => store.getFilename('..++config.production.js'), /jail/);
+   } finally {
+     fs.rmSync(jailParent, { recursive: true, force: true });
+   }
+ });
+
+ test('getFilename rejects traversal via multiple ++ segments', () => {
+   const base = fs.mkdtempSync(path.join(os.tmpdir(), 'psi-store-'));
+   try {
+     const store = new Store(base);
+     assert.throws(() => store.getFilename('legit++..++..++outside'), /jail/);
+   } finally {
+     fs.rmSync(base, { recursive: true, force: true });
+   }
+ });
+
+ test('getFilename rejects NUL in fid', () => {
+   const base = fs.mkdtempSync(path.join(os.tmpdir(), 'psi-store-'));
+   try {
+     const store = new Store(base);
+     assert.throws(
+       () => store.getFilename(`legit++${SAMPLE_UUID}\0/../../../etc/passwd`),
```

```
+       /jail/
+     );
+   } finally {
+     fs.rmSync(base, { recursive: true, force: true });
+   }
+ });
+
+ test('getFilename rejects resolved path equal to jail root (rel empty)', () => {
+   const base = fs.mkdtempSync(path.join(os.tmpdir(), 'psi-store-'));
+   try {
+     const store = new Store(base);
+     assert.throws(() => store.getFilename('bucket++..'), /jail/);
+   } finally {
+     fs.rmSync(base, { recursive: true, force: true });
+   }
+ });
+
+ test('getFilename accepts valid sid++uuid path under jail', () => {
+   const base = fs.mkdtempSync(path.join(os.tmpdir(), 'psi-store-'));
+   try {
+     const store = new Store(base);
+     const fid = `mybucket++${SAMPLE_UUID}`;
+     const got = store.getFilename(fid);
+     const expected = path.join(base, 'mybucket', SAMPLE_UUID);
+     assert.strictEqual(got, expected);
+   } finally {
+     fs.rmSync(base, { recursive: true, force: true });
+   }
+ });
```

tests/unit/upload-id.test.js

```
@@ -0,0 +1,41 @@
+ 'use strict';
+
+ const { test } = require('node:test');
+ const assert = require('node:assert');
+
+ const utils = require('../../lib/utils');
+
+ const UUID = '00000000-0000-4000-8000-000000000001';
```

```
+
+ test('isSafeTusUploadId rejects sid with path traversal', () => {
+   assert.strictEqual(utils.isSafeTusUploadId(`../../evil++${UUID}`), false);
+ });
+
+ test('isSafeTusUploadId rejects non-UUID key', () => {
+   assert.strictEqual(utils.isSafeTusUploadId('legit++not-a-uuid'), false);
+ });
+
+ test('isSafeTusUploadId rejects extra ++ separators', () => {
+   assert.strictEqual(utils.isSafeTusUploadId('a++b++c'), false);
+ });
+
+ test('isSafeTusUploadId rejects empty sid or key', () => {
+   assert.strictEqual(utils.isSafeTusUploadId(`++${UUID}`), false);
+   assert.strictEqual(utils.isSafeTusUploadId('sid++'), false);
+ });
+
+ test('isSafeTusUploadId accepts valid compound id', () => {
+   assert.strictEqual(utils.isSafeTusUploadId(`abc12++${UUID}`), true);
+ });
+
+ test('isSafeBucketFid rejects traversal-shaped id', () => {
+   assert.strictEqual(utils.isSafeBucketFid('../etc'), false);
+ });
+
+ test('isSafeBucketFid rejects compound id', () => {
+   assert.strictEqual(utils.isSafeBucketFid(`x++${UUID}`), false);
+ });
+
+ test('isSafeBucketFid accepts plain bucket id', () => {
+   assert.strictEqual(utils.isSafeBucketFid('abc123'), true);
+ });
```

Comments 0

Please sign in to comment.