Upload PATCH path traversal can create `config.<NODE_ENV>.js` and lead to code execution on restart

High **psi-4ward** published **GHSA-533q-w4g6-5586** last week

Package

psitransfer (npm)

Affected versions	Patched versions
2.4.1	2.4.3

Description

Summary

The upload PATCH flow under `/files/:uploadId` validates the mounted request path using the still-encoded `req.path`, but the downstream tus handler later writes using the decoded `req.params.uploadId`. In deployments that use a supported custom `PSITRANSFER_UPLOAD_DIR` whose basename prefixes a startup-loaded JavaScript path, such as `conf`, an unauthenticated attacker can create `config.<NODE_ENV>.js` in the application root. The attacker-controlled file is then executed on the next process restart.

Details

Observed in `2.4.1`, the upload middleware derives `fid` from `req.path.substring(1)` and calls `store.info(fid)` before handing the request to tus. For a request such as `/files/..%2Fconfig.production.js`, this outer check sees the encoded value `..%2Fconfig.production.js`. The downstream `patch('/:uploadId')` route, however, receives the decoded parameter `../config.production.js`. In the same code path, the `catch` branch uses `if(! e instanceof httpErrors.NotFound)`, which does not correctly stop execution on a missing upload target.

The write sink is `Store.getFilename(fid)`, which resolves `path.resolve(uploadDir, fid.replace('++', '/'))` and then only checks `startsWith(uploadDir)`. With a supported custom upload directory such as `<app_root>/conf`, the decoded target `../config.production.js` resolves to `<app_root>/config.production.js`, and the current string-prefix jail check still accepts it because the resolved path begins with `<app_root>/conf`.

The file creation is observable even when the request ends in failure. `store.append()` creates the target write stream first and only consults the JSON sidecar in the `finish` handler. As a result, `PATCH /files/..%2Fconfig.production.js` returns `404 Not Found` in my test, but still leaves an attacker-controlled `config.production.js` on disk.

On the next start, `config.js` executes `require(path.resolve(__dirname, \ config.${process.env.NODE_ENV}.js`))` when the file exists. I verified this in a temporary copy of the application by setting NODE_ENV=production and PSITRANSFER_UPLOAD_DIR to a custom conf directory, sending a single PATCH request that wrote JavaScript into config.production.js`, and then restarting the process. The attacker code executed during startup and created a proof file. Until a fix exists, the shortest safe workaround is to reject PATCH requests unless the expected sidecar metadata already exists and to avoid upload directory names that can prefix startup-loaded paths under the application root.

PoC

1. Start PsiTransfer `2.4.1` from source with `NODE_ENV=production` and a supported custom upload directory whose basename prefixes a startup-loaded file path, for example `PSITRANSFER_UPLOAD_DIR=/opt/psitransfer/conf`.
2. Send a PATCH request directly to the upload endpoint:

```
PATCH /files/..%2Fconfig.production.js HTTP/1.1
Host: target
Tus-Resumable: 1.0.0
Upload-Offset: 0
Content-Type: application/offset+octet-stream

module.exports = {}; require('fs').writeFileSync('/tmp/psitransfer-rce-proof', 'owned');
```

3. Observe that the response is `404 Not Found`, but `/opt/psitransfer/config.production.js` is created and contains the attacker-controlled payload.
4. Restart the PsiTransfer process, or wait for the next routine restart under the same `NODE_ENV`.
5. Observe that `/tmp/psitransfer-rce-proof` is created during startup, confirming server-side JavaScript execution from the injected `config.production.js`.

Impact

The observed result is unauthenticated creation of an attacker-controlled startup configuration file outside the intended upload directory. In affected deployments, this becomes code execution with the PsiTransfer service account on the next process restart, allowing full compromise of the application's confidentiality, integrity, and availability within that execution context. Default Docker and default source/systemd examples did not satisfy the RCE precondition in my review because their documented upload directory names do not prefix startup-loaded paths, but the vulnerable logic is still reachable.

Severity

High 7.5 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	High
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:H

CVE ID

CVE-2026-41180

Weaknesses

▶ CWE-22

Credits

pyuysig Reporter