bitovi / launchpad Public

<> Code ⊙ Issues 26 Pull requests 9 ⊙ Actions Projects Wiki ···

New issue

Jump to bottom

Security Fix for Remote Code Execution - huntr.dev #124

Merged **bmomberger-bitovi** merged 8 commits into `bitovi:master` from `418sec:master` on Jun 2

Conversation 5 Commits 8 Checks 0 Files changed 2

huntr-helper commented on Jul 3, 2020

https://huntr.dev/app/users/Mik317 has fixed the Remote Code Execution vulnerability 🔨. Mik317 has been awarded $25 for fixing the vulnerability through the huntr bug bounty program 💵. Think you could fix a vulnerability like this?

Get involved at https://huntr.dev/

Q | A
Version Affected | ALL
Bug Fix | YES
Original Pull Request | 418sec#1
GitHub Issue URL | #123
Vulnerability README | https://github.com/418sec/huntr/blob/master/bounties/npm/launchpad/1/README.md

User Comments:

📊 Metadata *

Please enter the direct URL for this bounty on huntr.dev. This is compulsory and will help us process your bounty submission quicker.

Bounty URL: **https://www.huntr.dev/app/bounties/open/1-npm-launchpad**

⚙️ Description *

The issue arised in multiple locations, so I to validate the type of data the functions were going to use (like the `paths`) and since there were some `multi commands` I didn't use the `execFile` function, which should have had stored many new variables because we wouldn't have been to concatenate and return results that would have been used in a second/third command correlated to the first one executed. In this case I simply made a functionality that `deletes every quote` from the variable, making impossible threat the `variables concatenated` as commands, but only as `arguments` of the specific command.

💻 Technical Description *

The fix has been applied in 3 different ways inside 3 different files, so I'll comment each one.

1. The issue arises firstly here: https://github.com/bitovi/launchpad/blob/master/lib/local/instance.js#L12 because of the fact the `name` variable is `concatenated` inside the various commands without being sanitized. Since the `name` is inside some `single-quotes` it would have been useless split the 3 different commands inside 3 different `execFile` that would have used more resources to store the content of the singular commands that should be concatenated again ... instead I introduced the `safe` function which deletes the `quotes` from the `name` in order to make it to be only an argument not escapable from quotes.

```
var safe = function (str) {
   // Avoid quotes makes impossible escape the `multi command` scenario
   return str.replace(/['"]+/g, '');
}
```

Note I've used the `execFile` function later in this file in the following lines: https://github.com/Mik317/launchpad/blob/master/lib/local/instance.js#L104 and https://github.com/Mik317/launchpad/blob/master/lib/local/instance.js#L110, in order to avoid `commands` could be executed in a dangerous context.
2. The 2' issue arised inside the following line: https://github.com/bitovi/launchpad/blob/master/lib/local/version.js#L21
In this case it I used `execFile` in order to avoid concatenation of other strings containing dangerous characters.
Patched with:

```
    exec(command.split(' ')[0], command.split(' ').slice(1), function(error, stdout) {
```

In this case the first part of the `command` is taken as `command to execute` (surely a path since the `command` variable is made through

```
    var command = path.join('"' + __dirname, '..', '..', 'resources', 'ShowVer.exe" "' + browser.command + '"');
```

), while the second part are the `arguments` .

3. The last issue arised here: https://github.com/bitovi/launchpad/blob/master/lib/local/version.js#L50
   In this case the `browser path and filename` weren't checked completely, and even if the execution of malicious code would have been possible only if the `default browser` of the victim has a badly crafted `filename` , I inserted a check to see if the `path+filename` pointing to the browser is a valid `path` .
   The issue has been fixed through this function:

```
// Validate paths supplied by the user in order to avoid "arbitrary command execution"
var validPath = function (filename){
  var filter = /[`!@#$%^&*()_+\-=\[\]{};':"\\|,<>\/?~]/;
  if (filter.test(filename)){
    console.log('\nInvalid characters inside the path to the browser\n');
    return
  }
  return filename;
}
```

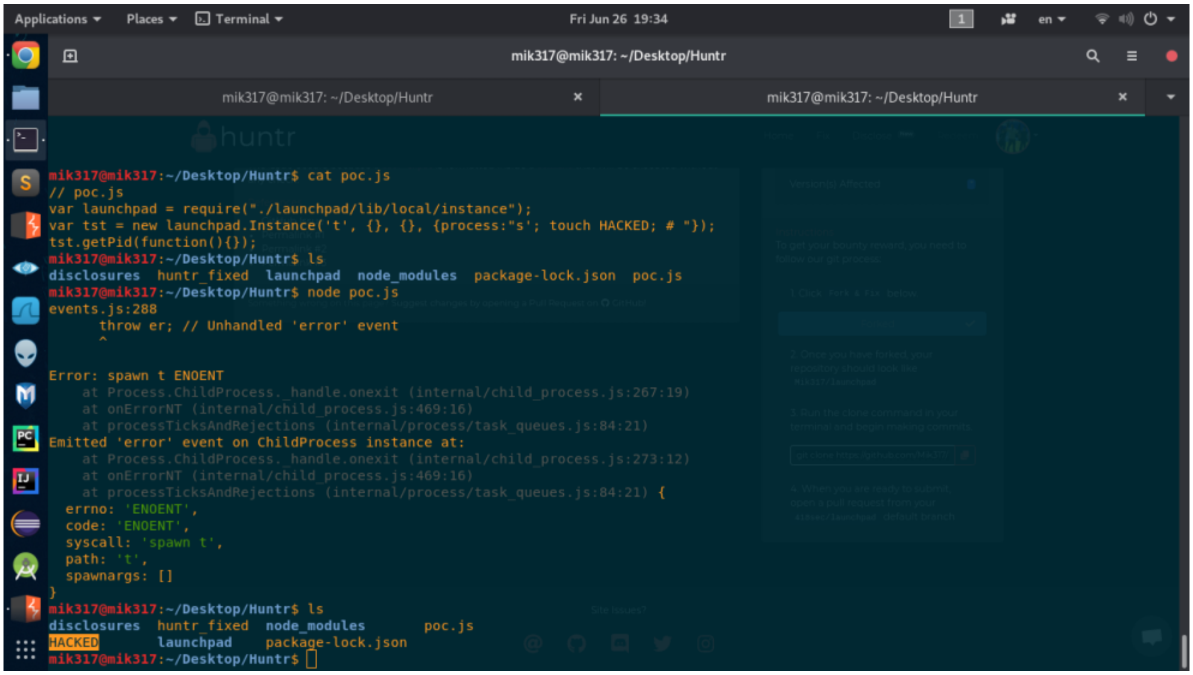
🐛 Proof of Concept (PoC) *

1. Download the JS library (launchpad)

2. Go inside the path you downloaded it and make the following `poc.js` file:

```
// poc.js
var launchpad = require("./launchpad/lib/local/instance");
var tst = new launchpad.Instance('t', {}, {}, {process:"s'; touch HACKED; # "});
tst.getPid(function(){});
```

3. Execute through `node poc.js`
4. The `HACKED` file will we created

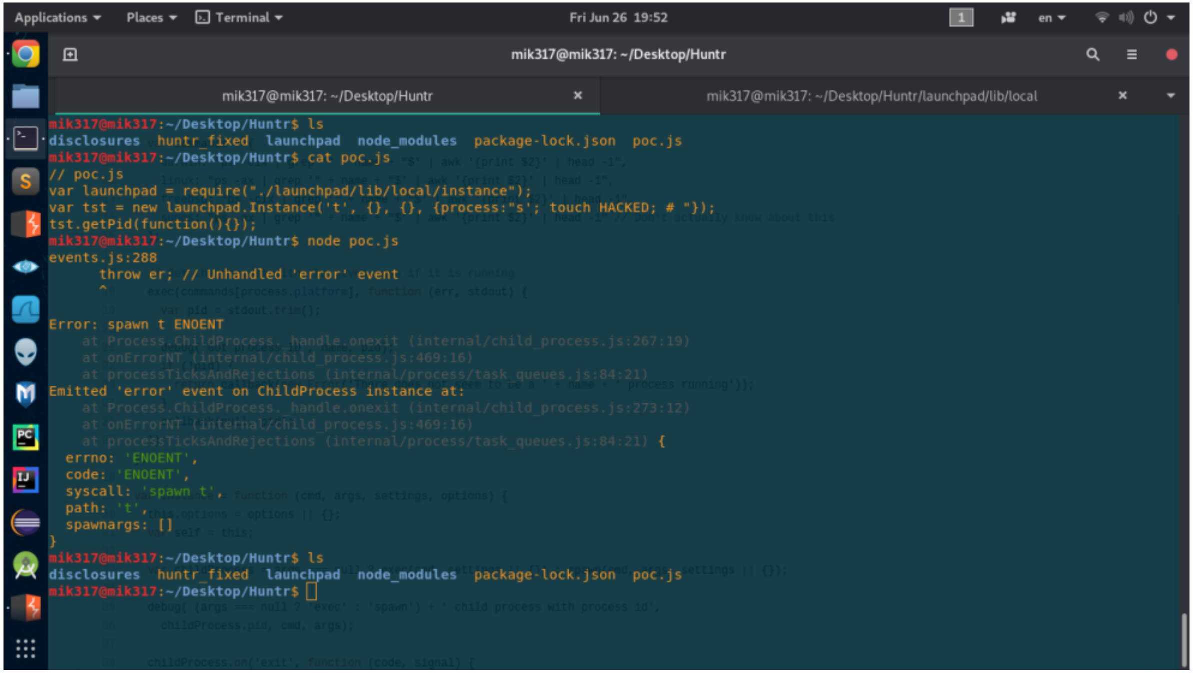

Applications
Places
Terminal
Fri Jun 26 19:34
en
mik317@mik317: ~/Desktop/Huntr
mik317@mik317: ~/Desktop/Huntr
mik317@mik317: ~/Desktop/Huntr

```
mik317@mik317:~/Desktop/Huntr$ cat poc.js
// poc.js
var launchpad = require("./launchpad/lib/local/instance");
var tst = new launchpad.Instance('t', {}, {}, {process:"s'; touch HACKED; # "});
tst.getPid(function(){});
mik317@mik317:~/Desktop/Huntr$ ls
disclosures  huntr_fixed  launchpad  node_modules  package-lock.json  poc.js
mik317@mik317:~/Desktop/Huntr$ node poc.js
events.js:288
      throw er; // Unhandled 'error' event
      ^

Error: spawn t ENOENT
    at Process.ChildProcess._handle.onexit (internal/child_process.js:267:19)
    at onErrorNT (internal/child_process.js:469:16)
    at processTicksAndRejections (internal/process/task_queues.js:84:21)
Emitted 'error' event on ChildProcess instance at:
    at Process.ChildProcess._handle.onexit (internal/child_process.js:273:12)
    at onErrorNT (internal/child_process.js:469:16)
    at processTicksAndRejections (internal/process/task_queues.js:84:21) {
  errno: 'ENOENT',
  code: 'ENOENT',
  syscall: 'spawn t',
  path: 't',
  spawnargs: []
}
mik317@mik317:~/Desktop/Huntr$ ls
disclosures  huntr_fixed  node_modules       poc.js
HACKED       launchpad    package-lock.json
mik317@mik317:~/Desktop/Huntr$
```

🔥 Proof of Fix (PoF) *

1. Download the fixed version
2. Use the same POC previously indexed
3. The `HACKED` file is NOT created anymore


Applications
Places
Terminal
Fri Jun 26 19:52
en
mik317@mik317: ~/Desktop/Huntr
mik317@mik317: ~/Desktop/Huntr
mik317@mik317: ~/Desktop/Huntr/launchpad/lib/local

```
mik317@mik317:~/Desktop/Huntr$ ls
disclosures  huntr_fixed  launchpad  node_modules  package-lock.json  poc.js
mik317@mik317:~/Desktop/Huntr$ cat poc.js
// poc.js
var launchpad = require("./launchpad/lib/local/instance");
var tst = new launchpad.Instance('t', {}, {}, {process:"s'; touch HACKED; # "});
tst.getPid(function(){});
mik317@mik317:~/Desktop/Huntr$ node poc.js
events.js:288
      throw er; // Unhandled 'error' event
      ^

Error: spawn t ENOENT
    at Process.ChildProcess._handle.onexit (internal/child_process.js:267:19)
    at onErrorNT (internal/child_process.js:469:16)
    at processTicksAndRejections (internal/process/task_queues.js:84:21)
Emitted 'error' event on ChildProcess instance at:
    at Process.ChildProcess._handle.onexit (internal/child_process.js:273:12)
    at onErrorNT (internal/child_process.js:469:16)
    at processTicksAndRejections (internal/process/task_queues.js:84:21) {
  errno: 'ENOENT',
  code: 'ENOENT',
  syscall: 'spawn t',
  path: 't',
  spawnargs: []
}
mik317@mik317:~/Desktop/Huntr$ ls
disclosures  huntr_fixed  launchpad  node_modules  package-lock.json  poc.js
mik317@mik317:~/Desktop/Huntr$
```

👍 User Acceptance Testing (UAT)

It doesn't introduce any error (at least using the module through the PoC I crafted)

Regards,
Mik

Mik317 and others added 8 commits 2 years ago

`Update version.js` d430b5d

`Update instance.js` 09ce4fa

Update version.js	d3993fc
Update instance.js	abf3dbc
Update instance.js	68518b2
Update version.js	e711d07
Update version.js	a3ff180
Merge pull request #1 from Mik317/master …	de5aca1

JamieSlome commented on Jul 3, 2020 Contributor

@daffl - let me know your thoughts! 🍰

frank-dspeed commented on Apr 13, 2021

@matthewp **@justinbmeyer** this should get merged i think it looks pritty good to me.

bmomberger-bitovi merged commit `e2f506b` into `bitovi:master` on Jun 2

bmomberger-bitovi commented on Jun 2 • edited Contributor

After this PR, the browsers do not exit as expected during the test suite. Prior to the PR during the test battery browsers would exit. I don't know the source of the issue yet but am investigating. **@JamieSlome** and **@Mik317** I would appreciate your help on this.

bmomberger-bitovi commented on Jun 2 Contributor

I am seeing the above problem on MacOS 10.15.7 (my dev machine). Other platforms are as yet untested.

bmomberger-bitovi commented on Jun 2 Contributor

Root cause: Because execFile does not create a subshell, quoting does not work the same way when constructing the command to terminate a process:
Not working:
`execFile('osascript', ['-e', '\'tell', 'application', '"Opera"', 'to', 'quit\'']); // this is the result of splitting the original line on spaces`

Also not working:
`execFile('osascript', ['-e', ''tell application' "Opera"' to quit'']);

In each of these cases, the single quote at the beginning and end of the AppleScript snippet is passed to the script interpreter as a literal quote, and it is not accepted. (The error returned is `Syntax Error: A unknown token can’t go here`)

Working:
`execFile('osascript', ['-e', 'tell application "Opera" to quit']`

I have made a PR to address this. No further action is required. Thanks for the security fix!

bmomberger-bitovi mentioned this pull request on Jun 21

Security Notice & Bug Bounty - Remote Code Execution - huntr.dev #123
Closed

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.

None yet

5 participants