How NPM is Jeopardizing Open Source Projects

Eric Tolhurst

Spring 2018

Tufts University

Abstract

NPM allows developers to easily publish, update, and download JavaScript code and Node.js modules, however, NPM's lack of security can pose a threat to the integrity of many open source projects which utilize NPM packages. This is because NPM does nothing to ensure that the GitHub code posted on the NPM website is the same as the code in its corresponding NPM package. This paper outlines exactly how it is possible to publish a package to the NPM registry with different code displayed on the GitHub repository linked on NPM's website.

Introduction

Node Package Manager (NPM) is a tool which allows developers to share and reuse JavaScript code. A developer can upload JavaScript files to the NPM registry, where users can easily download packages, and developers can easily update their code. Most importantly, NPM is the default manager for Node.js, the most common environment for running JavaScript on a server. Since their founding in 2010, NPM has quickly grow to over 350,000 packages, making it by far the largest package manager (Slashdot, 2017).

Individual NPM packages can have a widespread effect on software because if a package completes a useful task in an efficient manner, many developers will download this package to save time. For example, the package left-pad, a program which simply pads a given string with a number of spaces, has been downloaded almost 2.5 million times (Williams, 2016). In 2016, the same developer of left-pad decided to pull all his 250 packages from NPM without notice. As a result, thousands of projects, including Node and Babel, crashed because they depended on one of the 250 packages. Now imagine if this same developer, whose work was used in thousands of projects, decided to update his packages with malware. Of the thousands of developers, how many would update their package without caution and jeopardize thousands of users' data? These are valid concerns when discussing the security of NPM and its quickly growing outreach.

To the Community

This topic is currently an important area of study because the use of NPM is rapidly growing the software industry, and it is used to contribute to many open source projects. NPM estimates that 75% of JavaScript developers used NPM in 2017, and NPM expects this number to rise substantially in 2018 (Voss, 2018). With the rampant use of NPM, and the ubiquity of sensitive information and payment information on web applications, it is crucial that developers become aware that NPM does not guarantee that the GitHub code on an NPM package's webpage is the same as the downloaded package. It is also worth noting that the rise of Cryptocurrency mining on web browsers has made it more beneficial to hide malicious JavaScript on webpages, even if the code does not have access to sensitive information (Lau, 2017).

On a more personal note, vulnerabilities in open source projects have a wider consequence than just the data of the users of the vulnerable app. In a society where data privacy is a huge concern in the technology world, many users make an effort to use open source software. This is not to say that open source projects are immune to data breaches and selling data, however, the availability for professionals to audit code and ensure there are no backdoors is comforting to users. If NPM's lack of security jeopardizes the integrity of open source software, users will go back to using commercial software, where privacy is often sold.

Sending Different Code to GitHub and NPM

Shipping different code to NPM and GitHub is trivial and requires no technical skill. Here is how this is done:

In order to have a GitHub repository automatically linked to a NPM package, the repository must first be initialized in the relevant directory. Next, add the file with the code you wish to hide in the same directory as the file with the code you wish to have visible on GitHub. Then, create a file called ".gitignore" containing only the name of the file which you do not wish the user to see. This will prevent GitHub from pushing this file to the public repository. You also do not want anyone to see that you are hiding files in a .gitignore file, so change into the .git directory, which was created during the initialization of your repository. Here, change to the "info" directory and add ".gitignore" to the "exclude" file (~/.git/info/exclude). Now, you are free to push whatever you want to GitHub without the hidden file's code being visible.

Next, initialize a NPM package by executing the "npm init" command and inputting the relevant information when prompted by NPM. The GitHub repository will be detected by NPM and automatically linked to the NPM package. In the "package.json" file, add a "files" field which contains the JavaScript file(s) which you want to execute in your package but do not want the user to see. Since NPM will include anything in your directory by default, this step is necessary so the public files, which only exist to deceive the user, are not published to NPM and executed in the package. Now, you are completely free to publish to NPM ("npm publish") and push to GitHub without any of the files overlapping.

The Results:

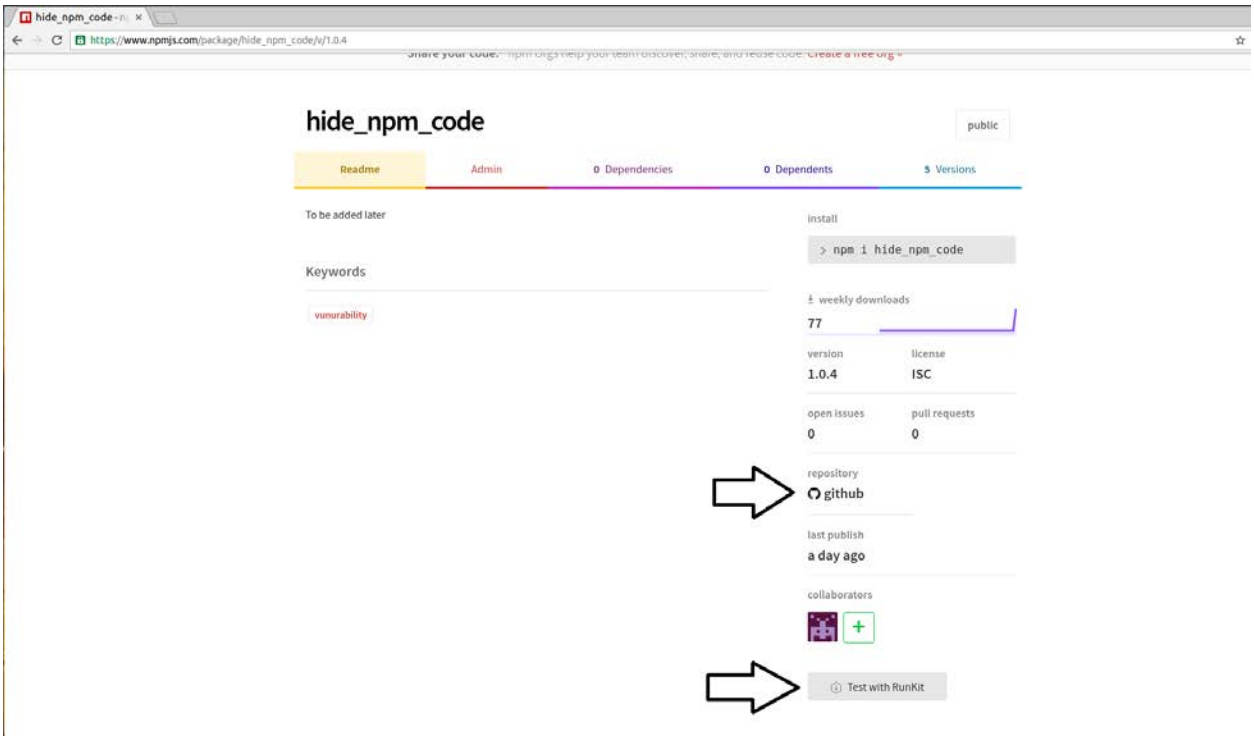

Figure 1: NPM Homepage of the package: "hide_npm_code." Note that there are both links to GitHub and Runkit (for testing)

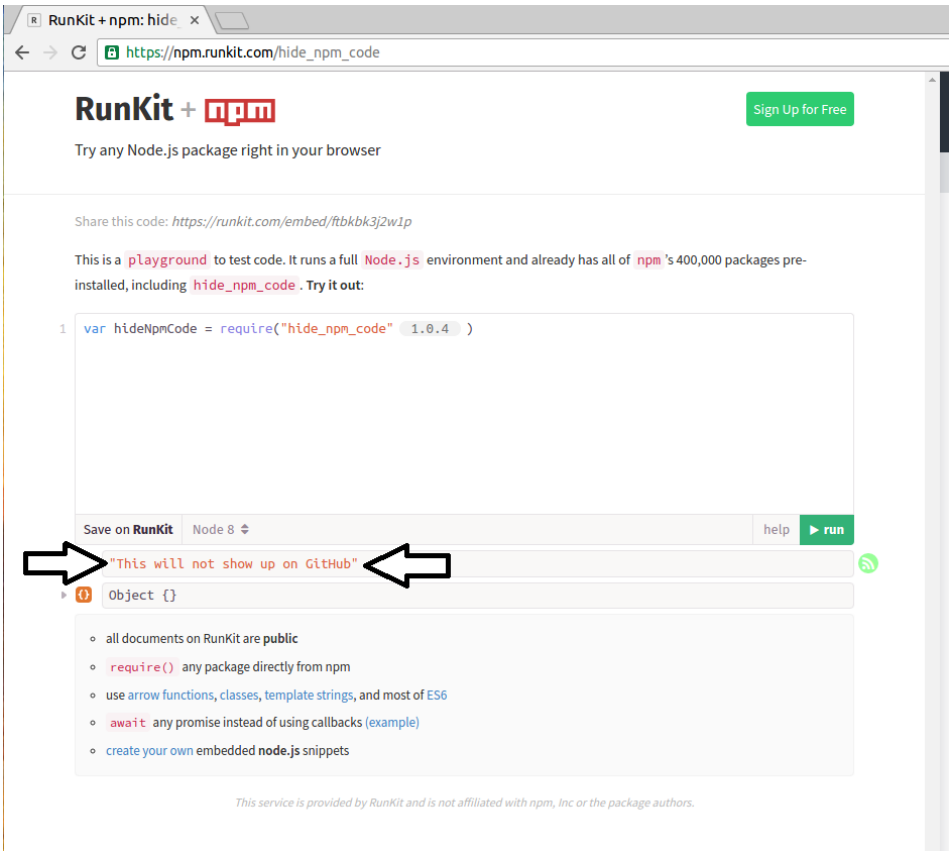

Figure 2: Runkit of "hide_npm_code." Console reads: "This will not show up on GitHub"

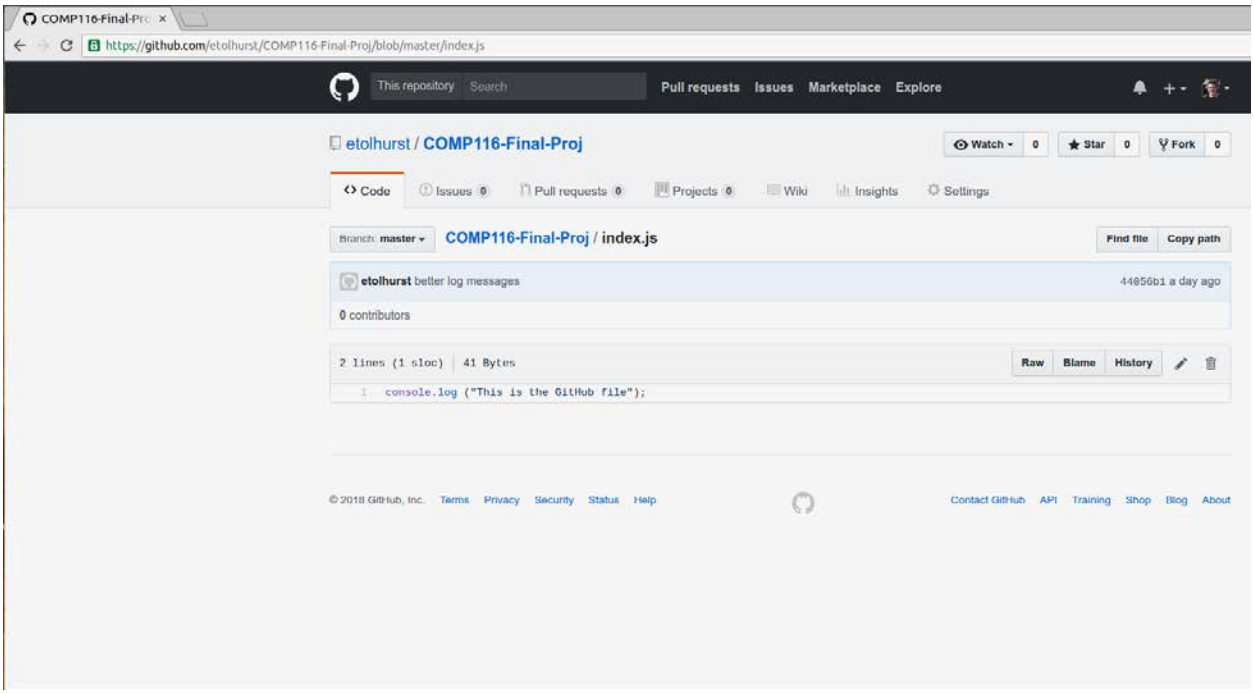

Figure 3: GitHub repository of "hide_npm_code." Note that there is no sign of the log in Figure 2 in this repository.

Defense

One direct solution to the NPM issue displayed above is the use of a tool which can verify that the code on NPM is identical to the code in the GitHub repository. Although this tool can detect differences in code on a GitHub pull request and the NPM package, it does not defend against all potential vulnerabilities in third party code on an open source project. The best method of defense in this situation is to treat third party code similar to how a user's input should be handled: kept away from all sensitive information.

This problem seems to be a NPM specific issue, rather than an issue pertaining to all package managers. The manager Bower, for example, directly installs packages from a provided link, rather than a central registry (Example: "bower install git://github.com/user/package.git"). (Bower, n.d.) This method makes it easier to confirm that the downloaded code will be the same as the visible code. This is not necessarily a better method than NPM's, as NPM's massive directory and its ease of publishing and updating packages to a central directory are what makes NPM so useful and popular.

Conclusion

There is no valid reason that NPM provides a link to a GitHub repository when they do nothing to validate that the code on GitHub is identical to the code uploaded to NPM. The easiest solution to this problem would be to remove the GitHub link from the NPM webpage altogether, however, it is annoying to the user if he has to download the entire package before he can look over the code. It is much easier to click a GitHub link and quickly look over the contents of the download. A better solution would be to check the contents of the GitHub repository linked in the package.json file and validate that it matches the code being published to NPM. If it does not, raise an error and prevent the user from publishing his package to NPM.

Some may suggest that projects which handle sensitive information should never utilize or accept outside contributions. This can never be a valid solution, as third-party contribution is in the nature of open source projects. In the end, it is up to the developer to ensure his software is not infected by third-party code. Third-party code should never be trusted, just like user input. Modules created by outside parties should never overlap with modules which contain the user's private data (addresses, credit card information, password files).

Bibliography

Bower. (n.d.). *Web sites are made of lots of things — frameworks, libraries, assets, and utilities. Bower manages all these things for you.* Retrieved from Bower: https://bower.io/#getting-started

Lau, H. (2017, December 19). *Browser-Based Cryptocurrency Mining Makes Unexpected Return from the Dead*. Retrieved from Symantec: https://www.symantec.com/blogs/threat-intelligence/browser-mining-cryptocurrency

Slashdot. (2017, January 14). *Node.js's npm Is Now The Largest Package Registry in the World*. Retrieved from https://developers.slashdot.org/story/17/01/14/0222245/nodejss-npm-is-now-the-largest-package-registry-in-the-world: https://developers.slashdot.org/story/17/01/14/0222245/nodejss-npm-is-now-the-largest-package-registry-in-the-world

Voss, L. (2018, January 3). *The State of JavaScript Frameworks, 2017*. Retrieved from npm: https://www.npmjs.com/npm/state-of-javascript-frameworks-2017-part-1

Williams, C. (2016, March 23). *How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript*. Retrieved from The Register: http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/