Malicious NPM Packages

Group Members: FNU Himanshu, Brian Kim, Paul Hwang,
Taylor Thackaberry, Joshua Kirkham

What Is NPM?

- Node Package Manager.
- Online repository for publishing open-source Node.js projects
- Manages dependencies
- Anybody can publish/download

```
npm install <package name>

cd project/; npm publish;
```

What are npm packages?

- The npm registry contains packages, many of which are also Node modules, or contain Node modules
- Node modules are similar to JavaScript libraries
- Repository has over a million code packages
- Packages have the ability to run preinstall/postinstall scripts

How can they be malicious?

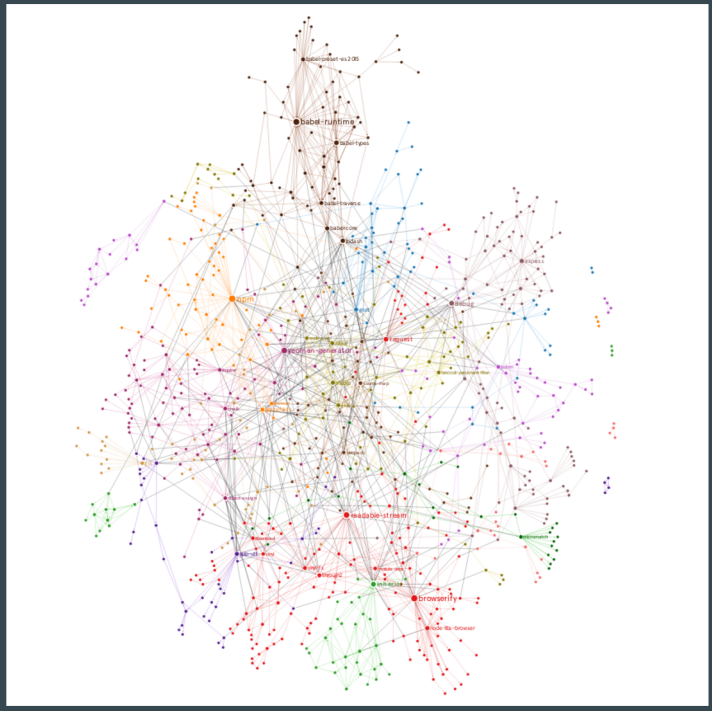
- Upon installing it, the module could have an install phase, where it could run destructive commands.
  - For example, rm -rf/
- Module could gather information from your system or network, and send it out to a 3rd party (potentially an attacker).
- Able to compromise systems by running preinstall or postinstall scripts within the package.json file.
- Can track installations for download metrics on the package that can cause potential concerns around user privacy.

The potential dangers of dependencies

- The impact is compounded by how npm is structured.
- NPM encourages small packages to solve a single problem (creates more dependencies)
- Gaining control of one of the highly-depended packages gives the attacker a greater reach.

Table 3: Characterization of package dependency graphs (without disconnected nodes)

	npm	PyPI
#Nodes	577943	84188
Avg node outdegree	4.27	2.95
Avg dependency tree size	86.55	7.33
Avg dependency tree depth	4.39	1.71



Dependency graph of top 100 npm packages

Examples

1337qq-js

- Uploaded to npm repository on December 30, 2019
- Collected sensitive information through install scripts on UNIX systems
- Collected
  - Environment variables
  - Running processes
- Environment variables can carry hard-coded passwords or API tokens in some JavaScript web/mobile apps.
- Was discovered 2 weeks after it was created and taken down

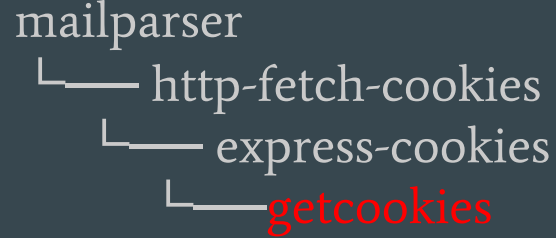
Examples (continued)

Typosquatting

- Packages with similar (but misspelled names) to popular packages designed to trick the user into installing them by accident

Backdoors

- Getcookies: contained a potential backdoor
- Users who used any packages that depended on getcookies were vulnerable even though they did not choose to use getcookies



Dependency layout of mailparser. While mailparser and the other packages above were not malicious, they depended on an insecure package.

Examples (continued)

mr_robot

- Inside the shrugging-logging package, adds a postinstall script that adds the package's author, "mr-robot" to every npm package owned by the user installing.

sdfjghlkfjdshlkjdhsfg

- Proof of concept of how to infect and re-publish local packages.
- Technique used for worming into any local package owned by the user installing.

Examples (continued)

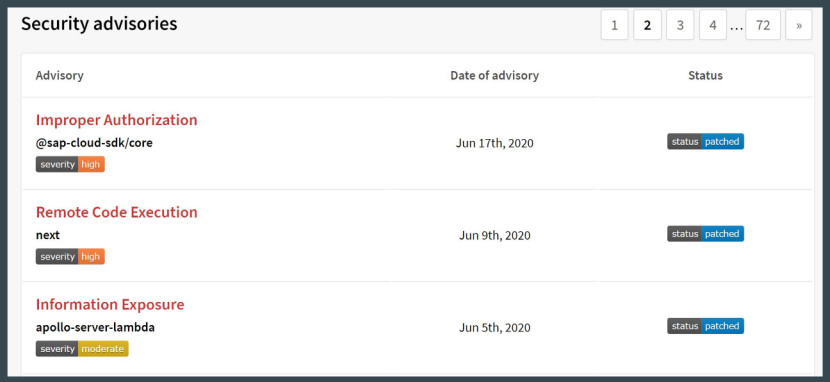
Load-from-cwd-or-npm

- This package was included with PureScript installer.
- It's purpose was to sabotage the PureScript npm installer to prevent the download.
- It returned a PassThrough stream instead of a request object, an implementation of node.js stream, that does nothing but pass bytes through unchanged.

```
purescript-installer
 └── dl-tar
                └── load-request-from-cwd-or-npm
                └──load-from-cwd-or-npm <<<<<< compromised package
```

Mitigations

- Minimize the total number of dependencies on your projects.
- Verify packages yourself
  - Authenticity, Integrity, & Security Risk
  - Stay up to date on security news
  - **Snyk:** Actively scans for and tracks known malicious/vulnerable packages
  - **NPM Shrinkwrap:** Verifies package integrity
- Central verification of package security
  - Apple app store
  - Official mainline repos for linux distributions
  - (Not a complete solution, just reduces attack surface)



Conclusion

- Npm is a powerful JS package manager
- The community-driven nature and sheer quantity of npm packages makes it difficult to discover if a package is a security risk to your project or data
- Npm does little to ensure that packages will not be harmful to those who use them
- Developers can take steps to protect themselves
- Depending on others for code == Increased attack surface

Questions for Discussion

- What could npm do to prevent malicious npm packages from being uploaded to their repository?
- What could be the implications of having a central maintainer with an approval process like the Apple App Store?
- Would this problem be worse if packages weren't open source?

Sources

1. https://www.zdnet.com/article/microsoft-spots-malicious-npm-package-stealing-data-from-unix-systems/
2. https://duo.com/decipher/hunting-malicious-npm-packages
3. https://medium.com/@liran.tal/malicious-modules-what-you-need-to-know-when-installing-npm-packages-12b2f56d3685
4. https://medium.com/@jsoverson/how-two-malicious-npm-packages-targeted-sabotaged-one-other-fed7199099c8
5. https://nakedsecurity.sophos.com/2020/01/15/malicious-npm-package-taken-down-after-microsoft-warning/
6. https://www.npmjs.com/
7. https://snyk.io/blog/how-much-do-we-really-know-about-how-packages-behave-on-the-npm-registry/