Could a few lines of code F@#k it all up?
Dependencies

- Probably 90% of commercial application make use of Open Source Software
- Open Source is great but....
- Node.js is quickly becoming a leading framework for web development
- Some features of node’s repository are concerning from the security point of view
A fairly anonymous developer at that time decided to “Liberate” his modules following a disagreement with NPM staff.

Among them was a module named Left-Pad
The Left-pad Saga

npm ERR! 404 Registry returned 404 for GET on https://registry.npmjs.org/left-pad

npm ERR! 404 'left-pad' is not in the npm registry.

npm ERR! 404 You should bug the author to publish it (or use the name yourself!)

When building projects on travis, or when searching for left-pad on npmjs.com, both will report that the package cannot be found.

Here is an excerpt from the travis build log:

```sh
npm ERR! Linux 3.13.0-40-generic
npm ERR! argv "/home/travis/.nvm/versions/node/v4.2.2/bin/node" "/home/travis/.nvm/versions/node/v4
npm ERR! node v4.2.2
npm ERR! npm  v2.14.7
npm ERR! code E404
```

npm ERR! 404 'left-pad' is not in the npm registry.

npm ERR! 404 You should bug the author to publish it (or use the name yourself!)
The Left-Pad Saga

- Left-Pad was used by ~40 npm modules – up to 370 now
  - including React and Babel (used by FB, AirBnB and others)

- First of all Azer is no longer anonymous.

- He actually triggered an important discussion within the community:
  - Should an author be able to un-publish his work without a process?
  - What happens to the available module names?
The NPM Platform
npm – Node.js Package Manager

- NPM is the node.js open source package manager, command-line client, and a centralized registry of Node.js modules

- “Find, share, and reuse packages of code from hundreds of thousands of developers”

- Over 435,000 modules used by over 6.5 million developers
Some points to note about npm registry

- Semantic Versioning – npm encourages the use of semver, or semantic versioning.
  - Dependencies are not locked to a certain version by default.
  - For any dependency, the dependency author can push a new version of the package.
Some points to note about npm utility

- Persistent authentication – npm utilizes persistent authentication to the npm server.
  - Users are not logged out until they manually do so
  - Typing ‘npm install’ may allow any module to execute arbitrary publish commands

Of course
I am Superman.
Some points to note about npm repo

- Centralized registry – NPM utilizes a centralized registry
  - Typing `npm publish` ships your code to this registry server, where it can be installed by anyone.
Let's look at an example: activedirectory

“activedirectory”

LDAP client for AuthN and AuthZ

4 Dependencies?

~3k weekly downloads

4 Dependencies?
So simple - npm install <module name>
Let's take an example npm

4 Dependencies?

Underscore

Idapjs

bunyan

async

lodash

dtrace-provider

verror

vasync

dashdash

once

backoff

asn1

moment

Safe-json-stingify

mv

Dtrace-provider

4+10+4+1=19
What about lodash?

~71k Dependents!

lodash

42,866,114
downloads last
month!

0 Dependencies?
lodash v4.15.0

A modern JavaScript utility library delivering modularity, performance, & extras.

```
_.assign({ 'a': 1 }, { 'b': 2 }, { 'c': 3 });
// → { 'a': 1, 'b': 2, 'c': 3 }
_.map([1, 2, 3], function(n) { return n * 3; });
// → [3, 6, 9]
```
Let’s take a look at some potential scenarios
Ways to cause damage

- Create a useful module
  - Use good old marketing
  - Update it after it gets adoption

- Create a module named similarly to another popular module (Typo attacks)
  - Packages are identified by names (No unique identifier/key)

- Take control of a legit account

- Create a self replicating worm
Creating a self replicating NPM worm (Lifecycle Scripts)

- Socially engineer a npm module owner to npm install an infected module on their system.
- Worm creates a new npm module

```
"scripts": {
    "start": "node create malicious_npm_module",
    "predeploy": "echo im about to deploy",
    "postdeploy": "echo ive deployed",
    "prepublish": "coffee --bare --compile --output lib/foo src/foo/*.coffee"
}
```

Full reports by Sam Saccone
https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/6eacfaeb94596f5852569290066a50b/018dbb99def69801852013f175/$FILE/npmwormdisclosure.pdf
Creating a self replicating NPM worm (Persistent Auth)

- Worm sets lifecycle hook on the new module to execute the worm on any install
- Worm publishes the new module to the user's npm account

Full reports by Sam Saccone
https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/6eacfaeb94596f5852569290066a50b/018dbb99def6980185257f820013f175/$FILE/npmwormdisclosure.pdf
Creating a self replicating NPM worm (Semver)

- Worm walks all user’s npm modules (publish permissions) and adds new module as a dependency in package.json.
- Worm publishes new versions to each of the modules with a “bugfix” level semver bump.

```
"dependencies": {
  "primus": "*",
  "async": "~0.8.0",
  "express": "4.2.x",
  "malicious_npm_module": ""
}
```

Full reports by Sam Saccone
https://www.kb.cert.org/CERT_WEB/services/vul-notes.nsf/6eacfaeab94596f58525692900666a50b/018dbb99def6980185257f820013f175/$FILE/npmwormdisclosure.pdf
Less Malicious – more Careless – still Vulnerable
return function middleware (req, res, next) {
    // Strip any null bytes from the url
    while(req.url.indexOf('%00') !== -1) {
        req.url = req.url.replace(/\%00/g, '');
    }
}
What we did

- Scan for security issues
  - Top 50 popular packages
  - Top 50 dependent-upon packages
  - Other popular packages
- Analyze results
- Responsible Disclosure
  - Contact dev
  - Wait for patch
  - Publish
Top 50 NPM packages

https://www.npmjs.com/

Packages people 'npm install' a lot

- browserify
- grunt-cli
- bower
- express
- npm
- cordova
- forever

Most depended-upon packages

- lodash
- request
- async
- underscore
- express
- commander
- bluebird
- chalk
- debug

https://www.npmjs.com/
## Scan for security issues

<table>
<thead>
<tr>
<th>PROJECT NAME</th>
<th>LAST SCAN DATE</th>
<th>TEAM</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>acorn-master</td>
<td>6/22/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>97564</td>
</tr>
<tr>
<td>ansi-regex-master</td>
<td>6/22/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>414</td>
</tr>
<tr>
<td>esprima-master</td>
<td>6/22/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>75907</td>
</tr>
<tr>
<td>inherits-master</td>
<td>6/23/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>47</td>
</tr>
<tr>
<td>isarray-master</td>
<td>6/23/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>25</td>
</tr>
<tr>
<td>lodash-master</td>
<td>6/23/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>128337</td>
</tr>
<tr>
<td>object-keys-master</td>
<td>6/23/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>419</td>
</tr>
<tr>
<td>private-master</td>
<td>6/23/2016</td>
<td>CxServer\SP\Company\npm project</td>
<td>198</td>
</tr>
</tbody>
</table>
What is wrong with this picture?
What is wrong with this picture?

ecstatic

```javascript
return function middleware (req, res, next) {
  // Strip any null bytes from the url
  while(req.url.indexOf('%00') !== -1) {
    req.url = req.url.replace(/\%00/g, '');
  }
}
```

- 150k weekly downloads
- **263** other npm packages are dependent on ecstatic
Developer Response

- PoC:
  - 22kb payload - 1 sec lag
  - 35kb payload - 3 sec lag
  - 86kb payload - server crashed

http://www.checkmarx.com/%00%00%00%00%00%00 (...
```
return function middleware (req, res, next) { 
  // Strip any null bytes from the url
  // This was at one point necessary because of an old bug in url.parse
  //
  // See: https://github.com/jfhb/ndoe-ecstatic/issues/16#issuecomment-3039914
  // See: https://github.com/jfhb/ndoe-ecstatic/commit/43f7e72a31524f88f47e367c3cc3af710e67c9f4
  //
  // But this opens up a regex dos attack vector! D:
  //
  // Based on some research (ie asking #node-dev if this is still an issue),
  // it's *probably* not an issue. :)
  /*
  while(req.url.indexOf('%00') !== -1) {
    req.url = req.url.replace(/\%00/g, '');
  }
  */
```
Other Scan Results

- **Command Injection**
  - Variable from user input was used as an argument for an OS command.
  - Dev response: “The flaw exists because the original author used it... A possible solution is to delete the vulnerable file”.
Other Scan Results

- Command Injection
- Stored XSS
- Denial of Service by Loop
- Denial of Service by Regex (ReDoS)
- CSV Injection
- Insecure Randomness
- Open Redirect
So how do I protect myself?
Be a Safe User!

- Check if there are any hooks: `npm show $module scripts`
- Inspect module’s code - and check out its dependencies
- Don’t allow scripts to execute automatically: `npm install --ignore-scripts`
- Use `npm shrinkwrap` to lock down your own dependencies
- Sometimes it’s better to write your own functions!
- Analyze your own code *together* with its dependencies!
- Enable 2FA and/or log out!
Be a Safe Corporate User!

- Run a local NPM Registry
- Replicate official registry ... or not
- Prevent installing from main registry
Thank You.