101 Ways Dependencies Can Ruin Your Day

Chris Thompson + Free Wortley

LunaSec

Who are we

Uber and Snapchat logos

So you want to make a website

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

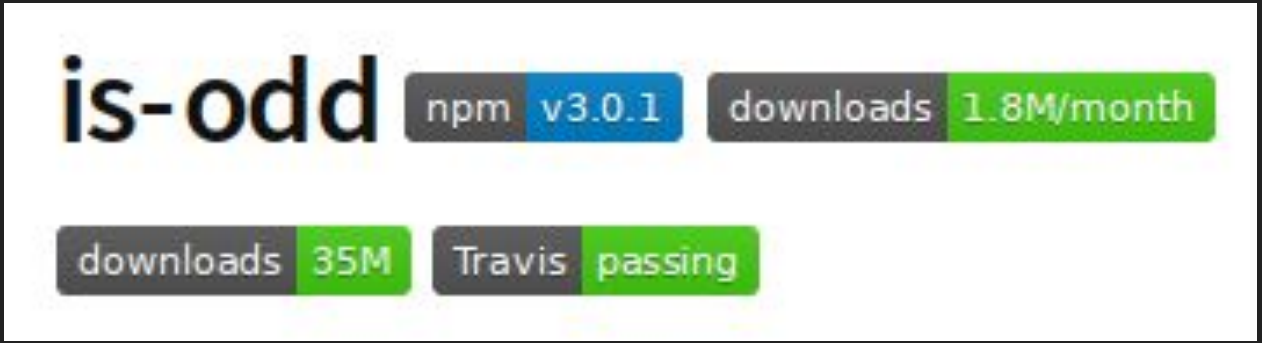
```
$ npm install express

$ pip install flask

$ gem install rails

$ go get -u github.com/gin-gonic/gin


$ echo
'<dependency>...org.springframework...</dependency>'
>> pom.xml \
  && mvn install
```

```
function isEven(n) {
  n = Number(n);
  return n === 0 || !!(n && !(n%2));
}

function isOdd(n) {
  return isEven(Number(n) + 1);
}
```

Testing whether a value is odd or even 2011 - stackoverflow

That is OK if n is with certain parameters, but fails for many scenarios.

```
Testing whether a value is odd or
even 2011 - stackoverflow
```

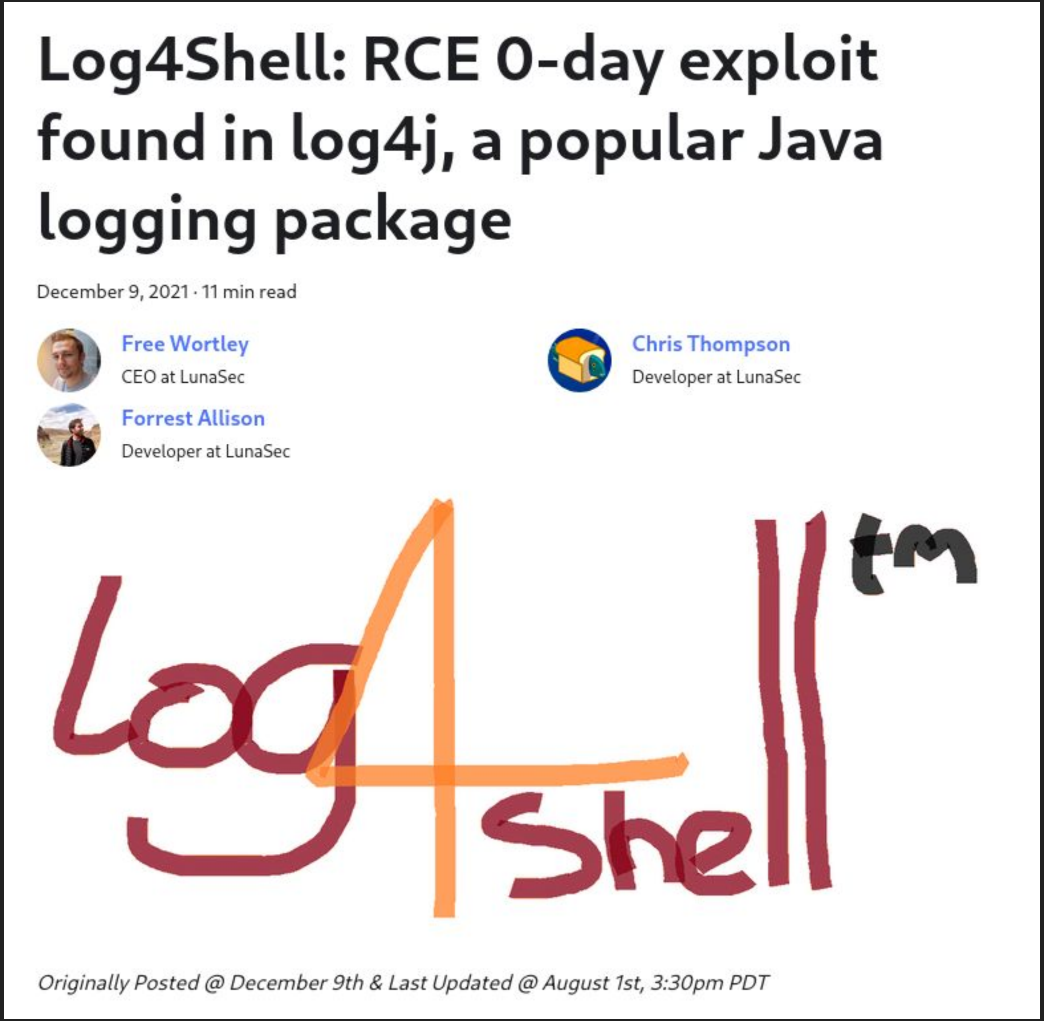


is-odd

npm v3.0.1 downloads 1.8M/month

downloads 35M Travis passing

Problem solved, right?



Log4Shell: RCE 0-day exploit found in log4j, a popular Java logging package

December 9, 2021 · 11 min read

Free Wortley
CEO at LunaSec

Chris Thompson
Developer at LunaSec

Forrest Allison
Developer at LunaSec

Originally Posted @ December 9th & Last Updated @ August 1st, 3:30pm PDT

What did we learn?

You’re finding a Needle in a Haystack

- Finding everywhere you use log4j is hard
- Existing vendor tools failed to detect indirect uses
- Most companies resorted to scanning servers manually via SSH

You’re vulnerable…

…now what?

Upgrading Legacy Code is Hard

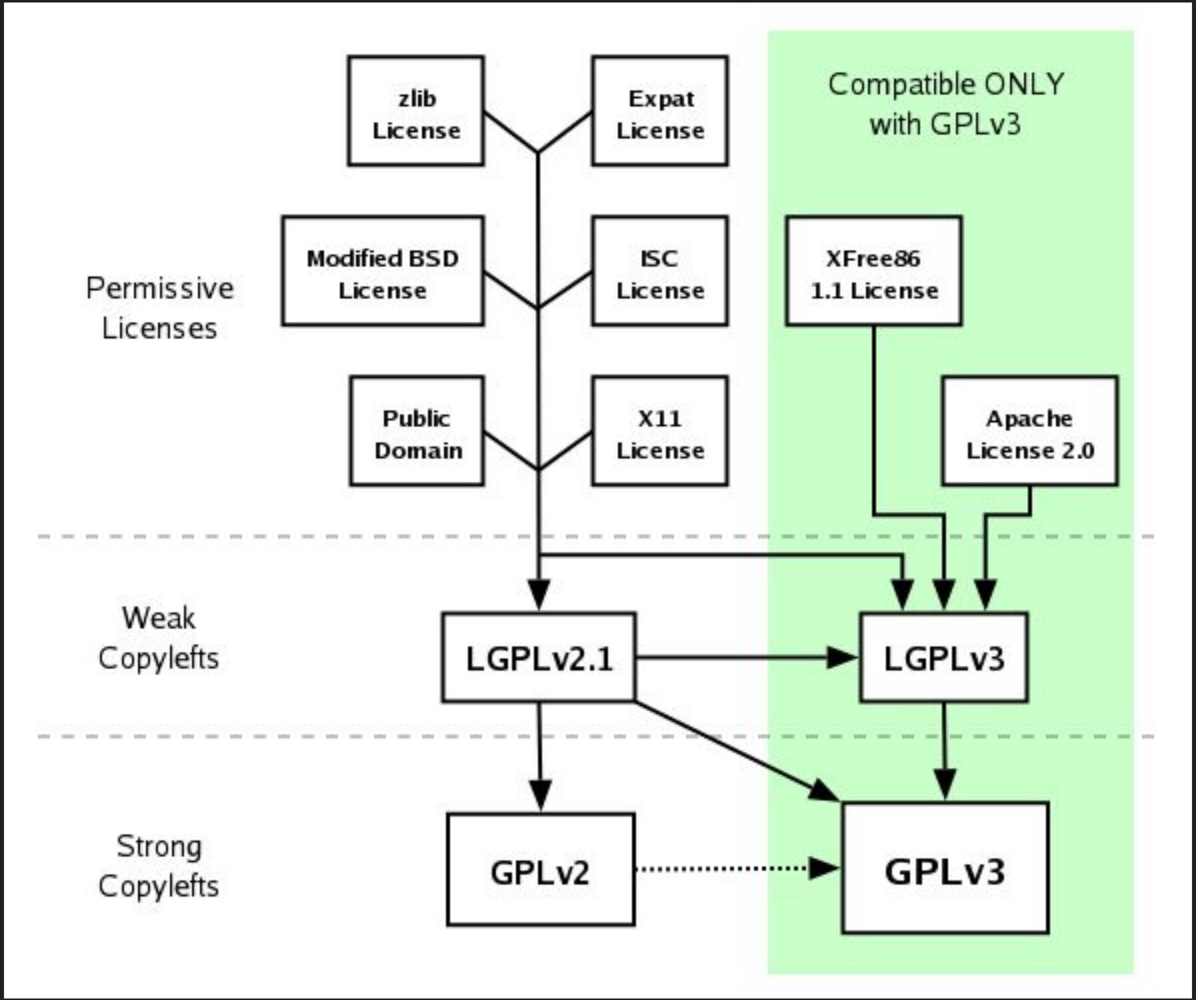
- Java has been used for decades
- Nobody knows how to build or deploy app
- Breaking changes to APIs
- Incompatible dependencies breaking build

Other Problems:

- Licensing
- Hijacked Packages
- Dependency Confusion
- Protestware
- Vulnerability in Dependency
- Targeted Attacks
- Typosquatting

Open Source Softwares Licenses

- Your devs check every dependency’s license, right?
- Copyleft licenses (GPL, etc.)
- Missing license or conflicting licenses
- License changing between releases


Compatible ONLY with GPLv3
Permissive Licenses
zlib License
Expat License
Modified BSD License
ISC License
XFree86 1.1 License
Public Domain
X11 License
Apache License 2.0
Weak Copylefts
LGPLv2.1
LGPLv3
Strong Copylefts
GPLv2
GPLv3

Hijacking Packages - Account Takeover

- Package maintainer re-uses password, gets pwned
- Attacker registers an expired email domain[1]
- Package manager (NPM) has a vulnerability[2]
- Malware on maintainer’s computer[3]

Surprise, Malware!

- Hijacked packages are injected with malicious code

- Code runs on package install:
  - npm install
  - python setup.py (often run as root when installing… whoops)

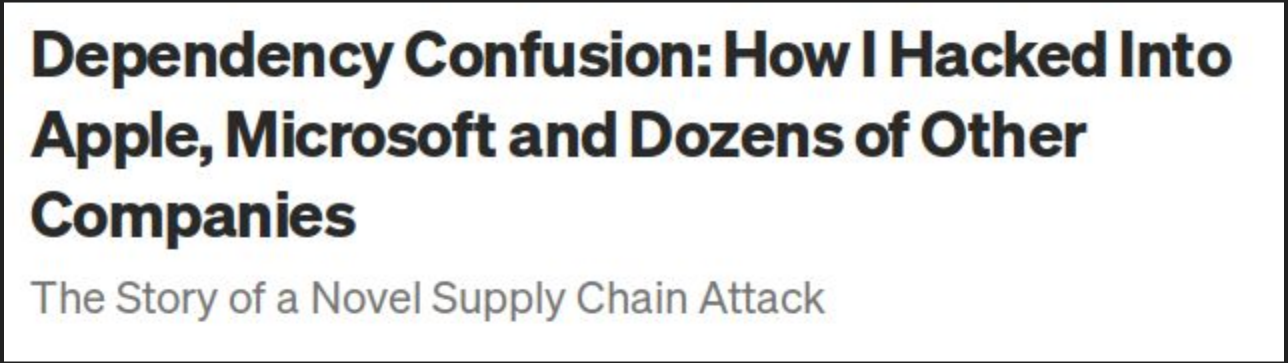
- Or when code is run:
  - Run payload when module is loaded
  - Hook some commonly used function

```
function getModulesOwned(user, cb) {
  var url = 'https://www.npmjs.org/~' + user;
  // ...
  var packages = $('.collaborated-packages a').map(function (i, el) {
    return $(this).text();
  }).get();
  // ...

function addOwner(packageName, newOwner) {
  exec('npm owner add ' + newOwner + ' ' + packageName);
}
```

shrugging-logging - npm

```
function infectModule (moduleName) {
  installModule(moduleName)
  .then(() => {
    addScript(moduleName);
    copyScript(moduleName);

    return incrementPatchVersion(moduleName);
  })
  .then(() => publishInfectedModule(moduleName))
  .catch(() => {});
}
```

sdfjghlkfjdshlkjdhsfg - npm


Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

The Story of a Novel Supply Chain Attack

- Internal package that is resolved in a public registry[20]
- The result of developers/CI having misconfigured environments

PayPal

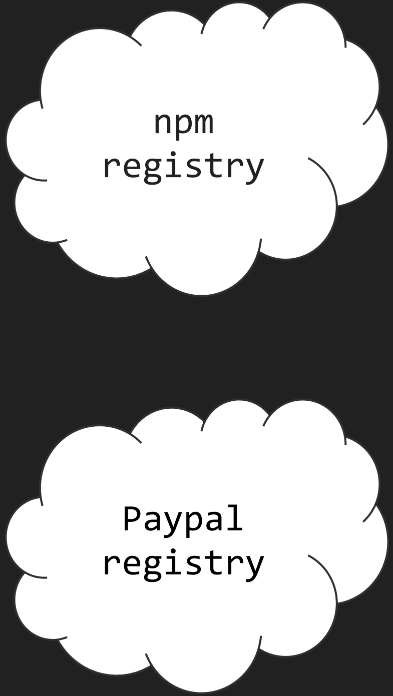
```
"dependencies": {
  "express": "^4.3.0",
  "dustjs-helpers": "~1.6.3",
  "continuation-local-storage": "^3.1.0",
  "pplogger": "^0.2",
  "auth-paypal": "^2.0.0",
  "wurfl-paypal": "^1.0.0",
  "analytics-paypal": "~1.0.0"
}
```

paypal - github

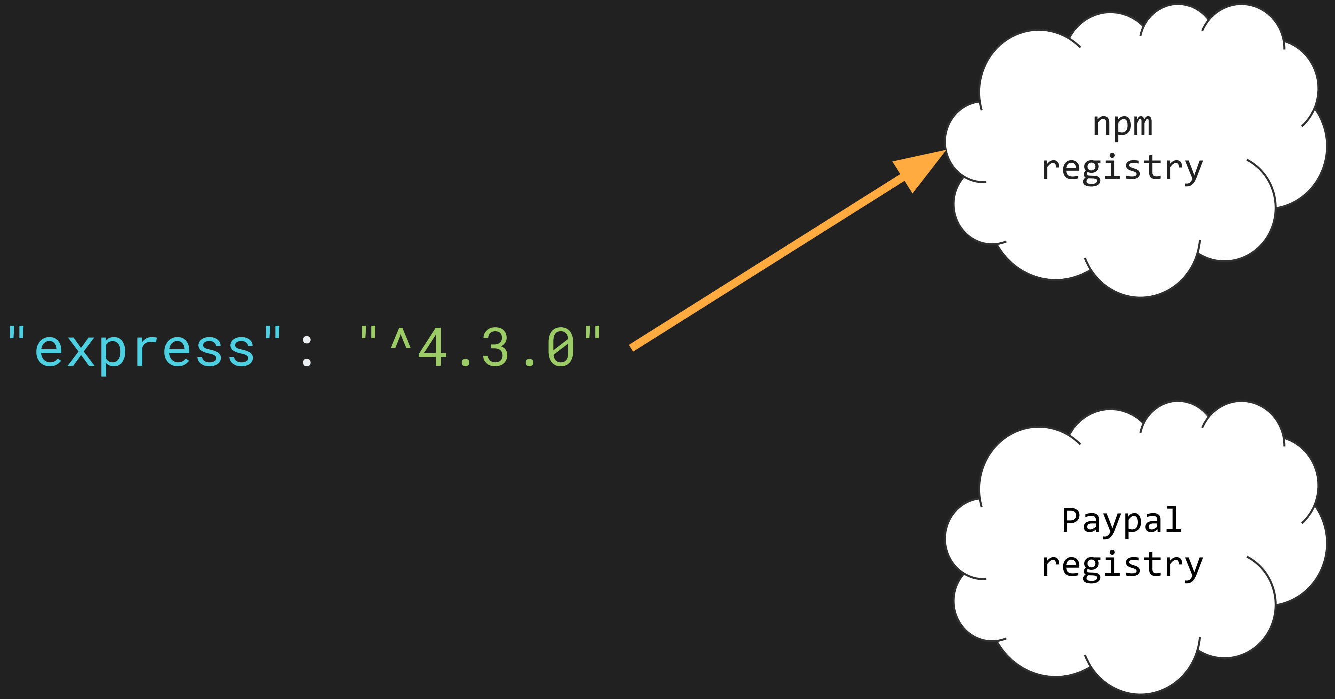
```json
"dependencies": {
  "express": "^4.3.0",
  "dustjs-helpers": "~1.6.3",
  "continuation-local-storage": "^3.1.0",
  "pplogger": "^0.2",
  "auth-paypal": "^2.0.0",
  "wurfl-paypal": "^1.0.0",
  "analytics-paypal": "~1.0.0"
}
```

paypal - github

```
"express": "^4.3.0"
```



paypal - github

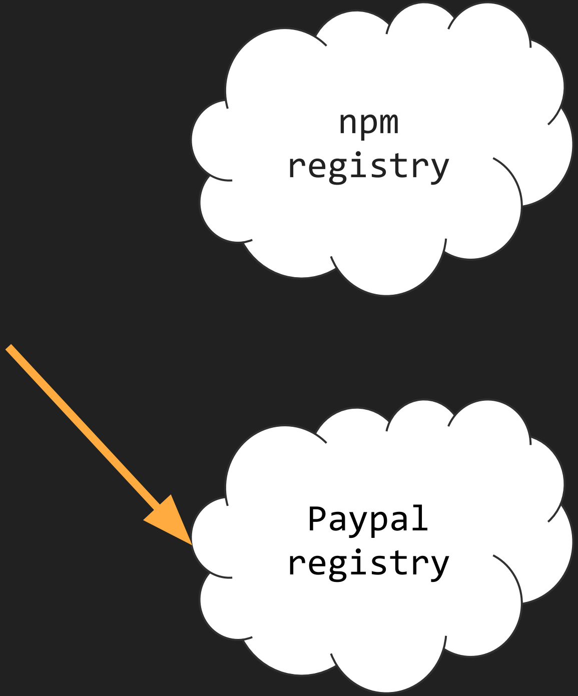


```
"express": "^4.3.0"
```

paypal - github

developer/ci with valid auth

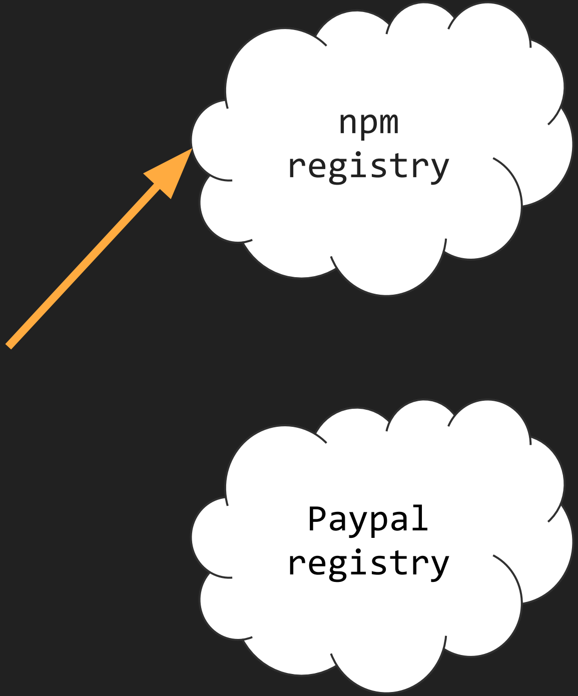
```
"auth-paypal": "^2.0.0"
```



paypal - github

developer/ci with **invalid** auth

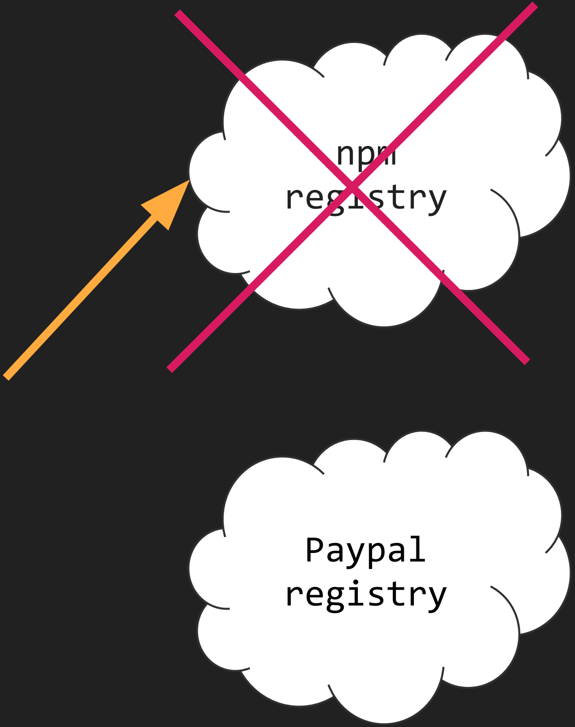
```
"auth-paypal": "^2.0.0"
```



paypal - github

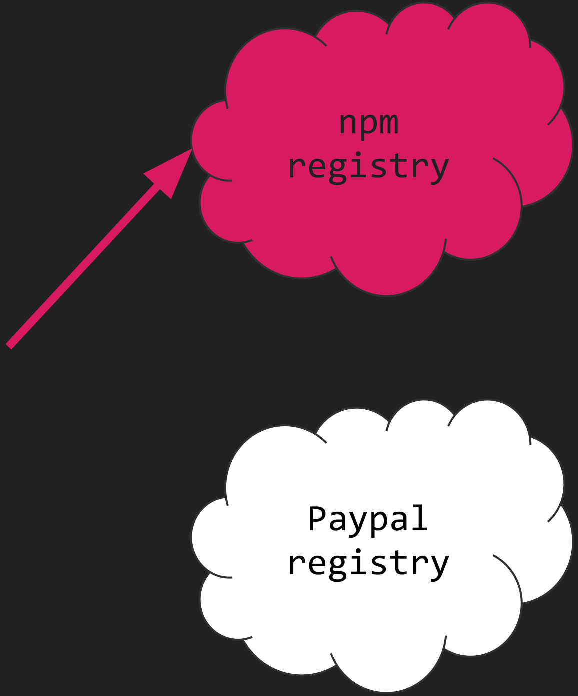
developer/ci with **invalid** auth

```
"auth-paypal": "^2.0.0"
```



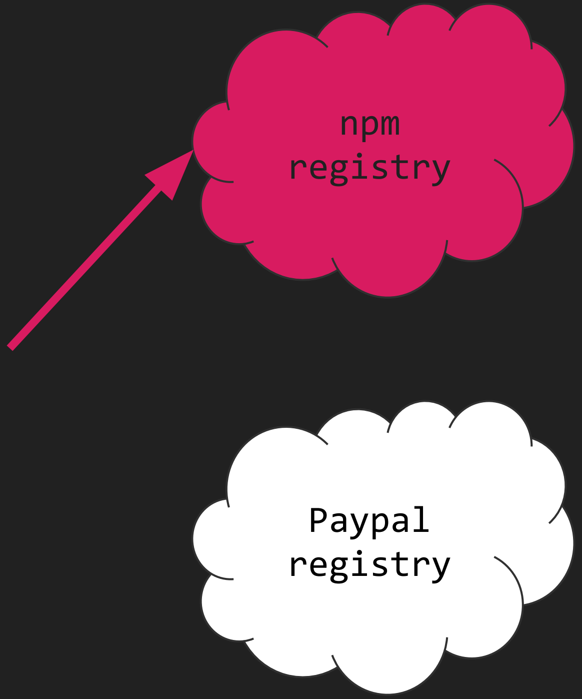
developer/ci with **invalid** auth

```
"auth-paypal": "^2.0.0"
```



developer/ci with **invalid** auth

```
"auth-paypal": "^2.0.0"
```



Pwned!

paypal - github

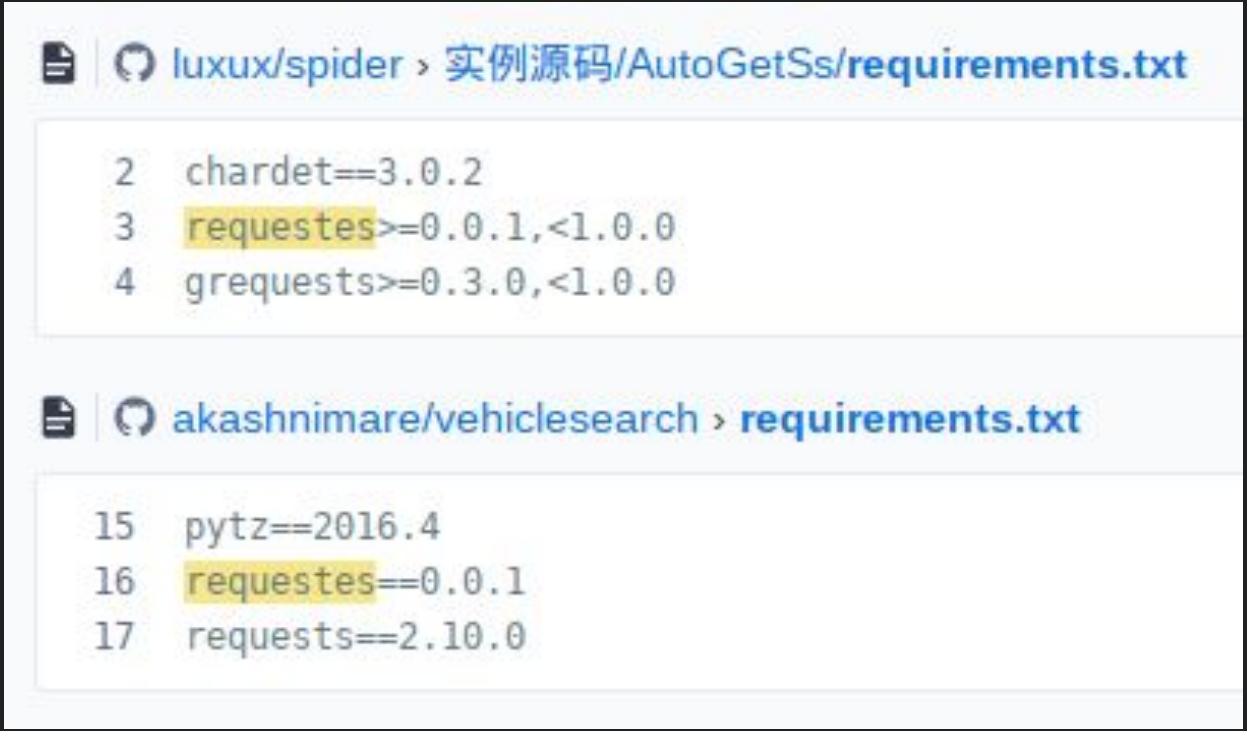
Typosquatting

- Like DNS typosquatting, but for packages
  - google.com -> gooogle.com, gogle.com, gewgle.com

- Common package mispells
  - pip install smb <- should be pysmb[9]

- Standard library dependencies
  - “import json”[10]

- Package install command typo
  - pip install requirements-dev.txt[11]



luxux/spider › 实例源码/AutoGetSs/**requirements.txt**

```
chardet==3.0.2
requestes>=0.0.1,<1.0.0
grequests>=0.3.0,<1.0.0
```

akashnimare/vehiclesearch › **requirements.txt**

```
pytz==2016.4
requestes==0.0.1
requests==2.10.0
```

“requestes” search - sourcegraph

Unpublishing

- NPM left-pad[7]
  - Npm changed their unpublish policy in response[8]

- NPM Faker.js / Colors.js[6]

Protestware

- Political activism by authors modifying their projects
- Many updates are harmless, printing out a log message
- npm package: node-ipc[12]
    - Attempts to geo-locate where the code is being run
    - If located in Russia or Belarus, then replaces every file with a ❤
    - Many projects affected, most namely Vue.js[14]

```
if (countryName.includes("russia") ||
countryName.includes("belarus")) {
    getFiles("./");
    getFiles("../");
    getFiles("../../");
    getFiles("/");
}
// ...
fs.writeFile(combinedPath, "❤️", function () {});
```

node-ipc - npm

Adding a “helpful” feature

- PR made by trusted, public users…
    - Includes a covert, malicious package (flatmap-stream)[13]
    - PR included a *non-malicious* package, but was later updated to be malicious[14]

flatmap-stream heist - the setup

1. GitHub user **right9ctrl** creates a PR resolving a feature request in the widely used package **event-stream**

2. **right9ctrl** creates the package **flatmap-stream** and uses it in their PR

3. **flatmap-stream**’s code is published to GitHub, no malicious code

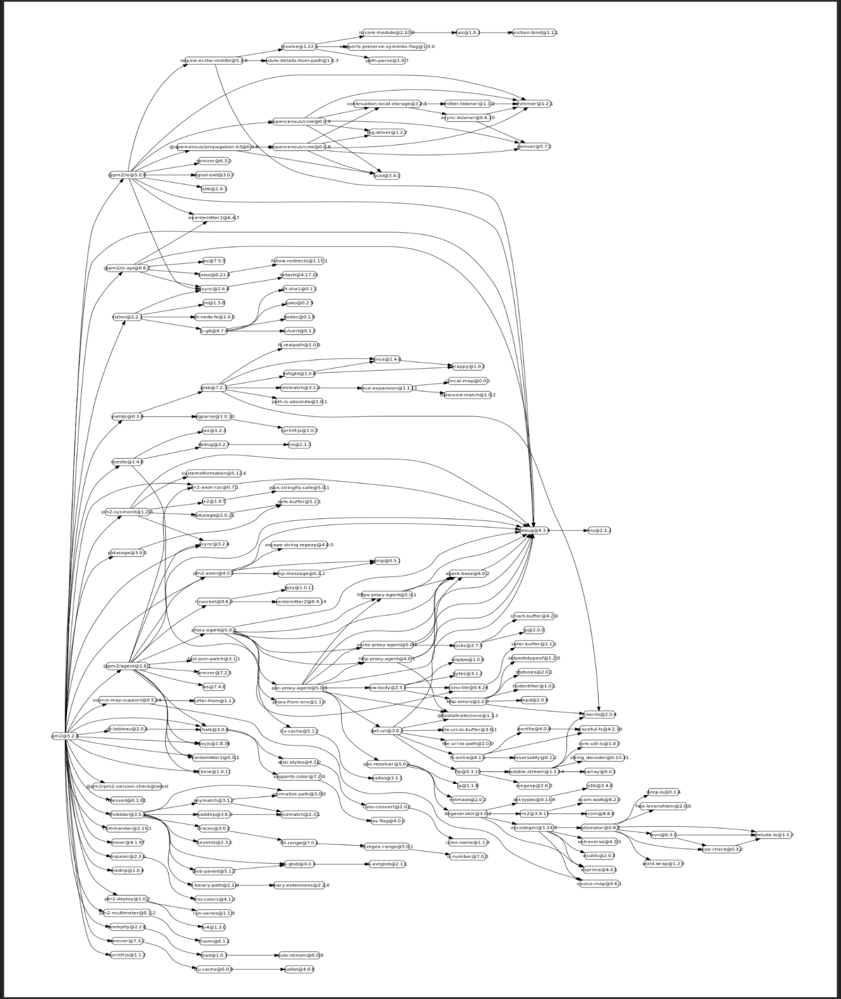
flatmap-stream heist - covert ops

- **flatmap-stream** had different code compiled in the release package on NPM
- Code was encrypted
- Decryption key was the npm package description of **bitpay/copay**, a Bitcoin wallet

@right9ctrl If you removed flatmap-stream because your realized it was an injection attack why didn't you yank event-stream@3.3.6 from npm and put a PSA? If you didn't know, why did you choose to use a completely unused/unknown library (0 downloads on npm until you use it)?

#116 event-stream - github

If it couldn’t get any worse…



pm2 - npm

A meaningful bottom line

- Safety in numbers (popular frameworks)
  - React, Vue.js, Express, Pandas, Rails, etc.
- Prefer the standard library (when possible)
- Think critically before importing 3rd party code
- Support the packages you rely on
  - Sponsorship
  - Developer time

Most automation is misleading

- Too many false positives

- Not all security products actually improve security
  - npm audit

- Incentives are in the wrong place
  - “Critical vulnerability” = Red alert
  - Red alert = Fear
  - Fear = Sales

Next-Gen Dependency Analysis

- Surface only *meaningful* vulnerabilities
  - Is the vulnerable function ever called?
  - Is the vulnerability in a package of a package of a package…
- Trivially updated packages are automatically bumped
  - Need reliable CI + testing to avoid outages
- Sandboxing dependencies
  - Restrict permissions by default
- Inventory dependencies
  - SBOM Executive Order[19]

Thank You!

Twitter: @LunaSecIO

GitHub: github.com/lunasec-io/lunasec

Blog: log4shell.com

LunaSec

References

[1] https://thehackerblog.com/zero-days-without-incident-compromising-angular-via-expired-npm-publisher-email-domains-7kZplW4x/

[2] https://web.archive.org/web/20161031001622/http://webinos.org/2013/06/17/reflections-on-nodejs-malware/

[3] https://contolini.com/building-an-npm-worm

[4] https://duo.com/decipher/hunting-malicious-npm-packages

[5] https://www.kb.cert.org/vuls/id/319816/

[6] https://snyk.io/blog/open-source-npm-packages-colors-faker/

[7] https://web.archive.org/web/20210327093144/https://kodfabrik.com/journal/i-ve-just-liberated-my-modules

[8] https://docs.npmjs.com/policies/unpublish

[9] http://blog.fatezero.org/2017/06/01/package-fishing/

[10] https://pytosquatting.overtag.dk/

[11] https://github.com/pylola/requirements.txt/

References

[12] https://www.lunasec.io/docs/blog/node-ipc-protestware/

[13] https://github.com/cncf/tag-security/blob/main/supply-chain-security/compromises/2018/event_stream.md

[14] https://github.com/zlw9991/node-ipc-dependencies-list

[15] https://log4shell.com/

[16] https://heartbleed.com/

[17] https://imagetragick.com/

[18] http://newosxbook.com/ent.jl

[19] https://www.cisa.gov/sbom

[20] https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

References

[21] https://npms.io/

[22] https://snyk.io/advisor/npm-package/express

[23] https://medium.com/checkmarx-security/new-protestware-found-lurking-in-highly-popular-npm-package-d46f8ba67e36

Writing an HTTP Parser is hard

- Good libraries extend the standard library of the language
- Ocaml had a weak standard library, Jane Street made one, Ocaml is popular now
- npm’s left-pad package solved a common problem
- Until generics, you had to code your own “find a needle in a haystack” loop for every type

Why are there so many?

- Software deliverable timelines based on using dependencies
- Valuing speed over “correctness”
- Solution to a problem is a package, not lines of code
  - Complicated/challenging language semantics
  - Common practice, “just install X”
  - Low upfront cost, higher maintenance cost

TODO Free's story

```
Maybe have Free tell story about Uber using npm 2 vs 3?
```

Insert stern sounding security stuff here…

– David Fischer

`requestes` README.md - pypi 2013

Hijacking Packages - Modifying Package

- Repository
  - Push payload directly to code
  - Modify cached dependencies (ex. binary blob in vendor folder, modify cache for Node PNP package, etc.)
- Registry
  - Push an “updated” version
  - Push new packages as trusted user (user may have push access to a company’s package scope, ex. @lunasec/…) [4][5]

```
{
  "name": "some-package",
  "scripts" : {
    "install" : "scripts/install.js",
    "postinstall" : "scripts/postinstall.js",
    "uninstall" : "scripts/uninstall.js"
  }
}
```

package.json - node

```go
package harmlesspackage

init() {
    steal("~/.aws/creds")
}
```

harmless.go - go

```
> npm install styled-components@5.3.5
```

```
A message from the styled-components core team: If you are
seeing this,
your environment is set to Russian locale. By now it is our
hope that you
have seen the devastation, horrors, and complete disregard
the Russian
military has for Ukrainian civilians...
```

styled-components - npm

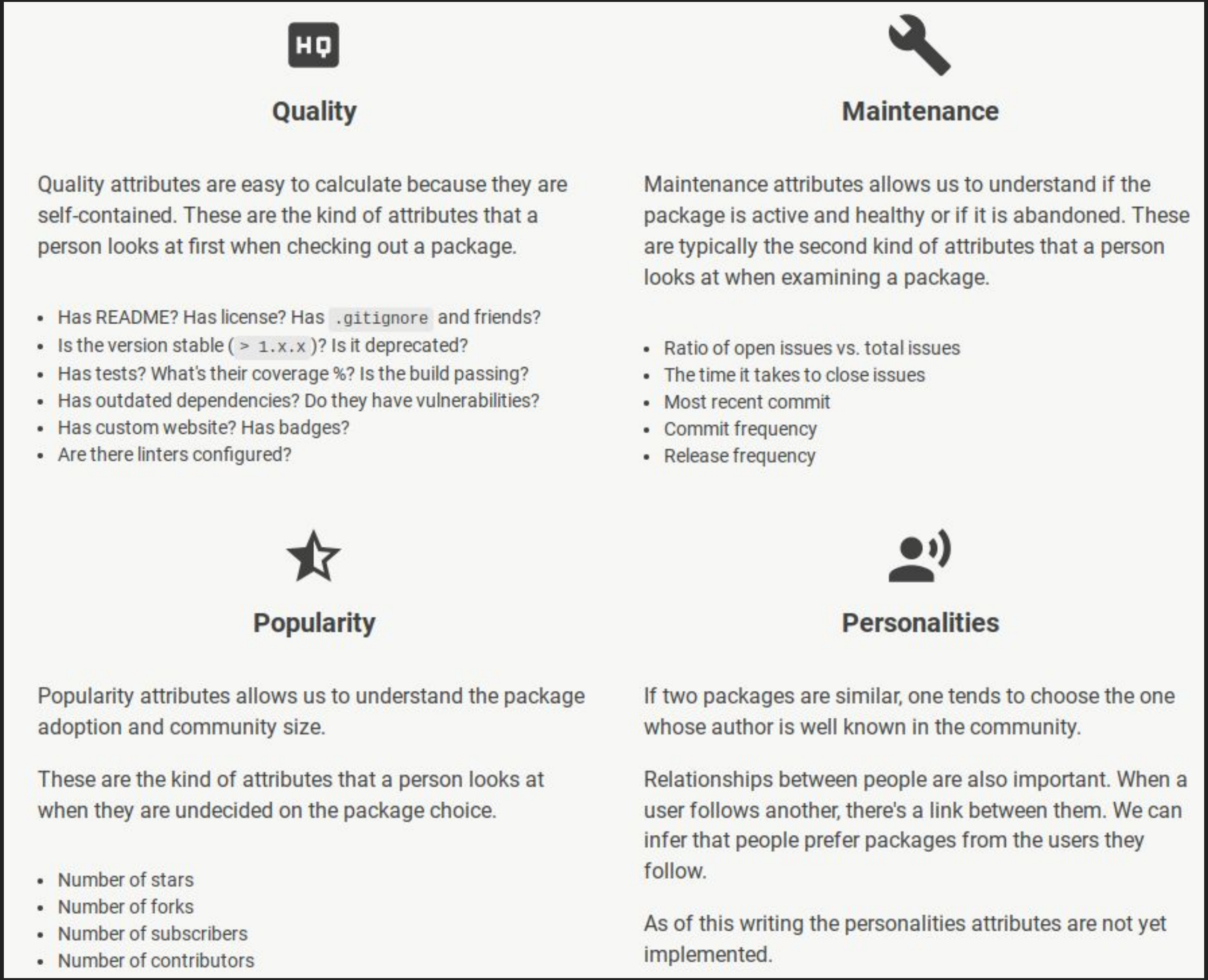
Vulnerability in a Dependency

- Your logging library can let someone pop a shell on your server [15]
- The SSL library your web server uses lets anyone on the internet read your process’ memory[16]
- The library you are using to resize a profile picture can give someone remote code execution to your cluster[17]

Limitations

- We only have heuristics
  - npms.io health score[21]
  - snyk package score[22]

- Does the proxy block the addition of a random transitive dependency?



Quality

Quality attributes are easy to calculate because they are self-contained. These are the kind of attributes that a person looks at first when checking out a package.

- Has README? Has license? Has `.gitignore` and friends?
- Is the version stable (`> 1.x.x`)? Is it deprecated?
- Has tests? What's their coverage %? Is the build passing?
- Has outdated dependencies? Do they have vulnerabilities?
- Has custom website? Has badges?
- Are there linters configured?

Maintenance

Maintenance attributes allows us to understand if the package is active and healthy or if it is abandoned. These are typically the second kind of attributes that a person looks at when examining a package.

- Ratio of open issues vs. total issues
- The time it takes to close issues
- Most recent commit
- Commit frequency
- Release frequency

Popularity

Popularity attributes allows us to understand the package adoption and community size.

These are the kind of attributes that a person looks at when they are undecided on the package choice.

- Number of stars
- Number of forks
- Number of subscribers
- Number of contributors

Personalities

If two packages are similar, one tends to choose the one whose author is well known in the community.

Relationships between people are also important. When a user follows another, there's a link between them. We can infer that people prefer packages from the users they follow.

As of this writing the personalities attributes are not yet implemented.

npms.io