



# Responsible use of Node.js & Open Source Software

Best Practices at an Enterprise Level

Steve Husak, Distinguished Engineer - Capital One

@shusak  
#OSfinserv



# Responsible use of Node.js & Open Source Software

## Best Practices at an Enterprise Level

Stephen Husak  
Distinguished Engineer

# Who Am I?



Steve Husak  
Distinguished Engineer

- Joined Capital One in 2014
- Intent Lead for Node.js/JavaScript for Capital One
- Distinguished Engineer in our Open Source Program Office space
- Previously led the redesign and architecture of Capital One's credit card contact center software, and then worked in Capital One's Commercial Banking space
- Passionate around developer experience and engineering standards

# What's in your wallet?

Capital One

- United States based bank with presence in UK and Canada
- Credit Cards, Retail & Commercial Banking, Auto Finance
- Tech-first company
- Our mission is to “Change Banking for Good”



# Open Source Software COMMITMENT TO COMMUNITY

Capital one made an “open source first” declaration in 2014 and that’s when we made our first contributions to the open source community.

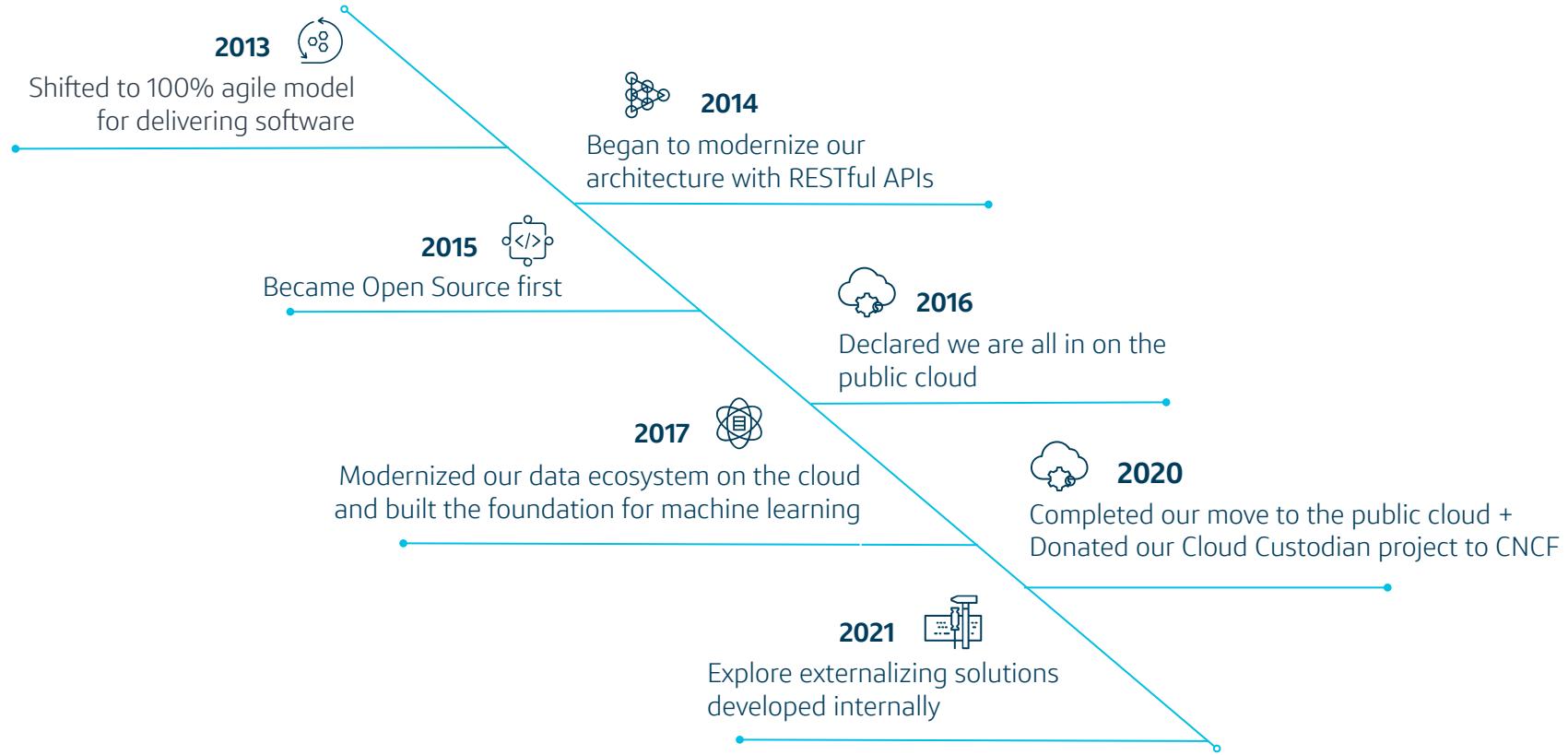
- We sponsor FINOS, OpenJS, Python, Continuous Delivery and the Cloud Native Computing Foundations to help keep open source sustainable
- Capital One’s contributions to the open source community have been significant and we’ve released more than 40 of our own software projects
- We’ve invested for years to build the culture and governance required to be open source-first in a highly regulated industry



Featured Open Source Projects: Data Profiler, Rubicon-ML and Hygieia



# Capital One's technology transformation



# Security landscape for Node.js & npm has changed dramatically over the past few years

Malicious NPM

Node.js  
RCE, H

## Node.js prototype pollution is bad for your app environment

[https://www.theregister.com/2022/07/25/nodejs\\_prototype\\_pollution/](https://www.theregister.com/2022/07/25/nodejs_prototype_pollution/)

## Malicious NPM packages are part of a malware “barrage” hitting repositories

<https://arstechnica.com/information-technology/2021/12/malicious-packages-sneaked-into-npm-repository-stole-discord-tokens/>

Attack  
A Large-Scale

<https://thehackernews.com/2021/12/npm-supply-chain-attack-hits-hundreds-of-websites-and-apps.html>

## NPM supply-chain attack impacts hundreds of websites and apps

<https://www.bleepingcomputer.com/news/security/npm-supply-chain-attack-impacts-hundreds-of-websites-and-apps/>



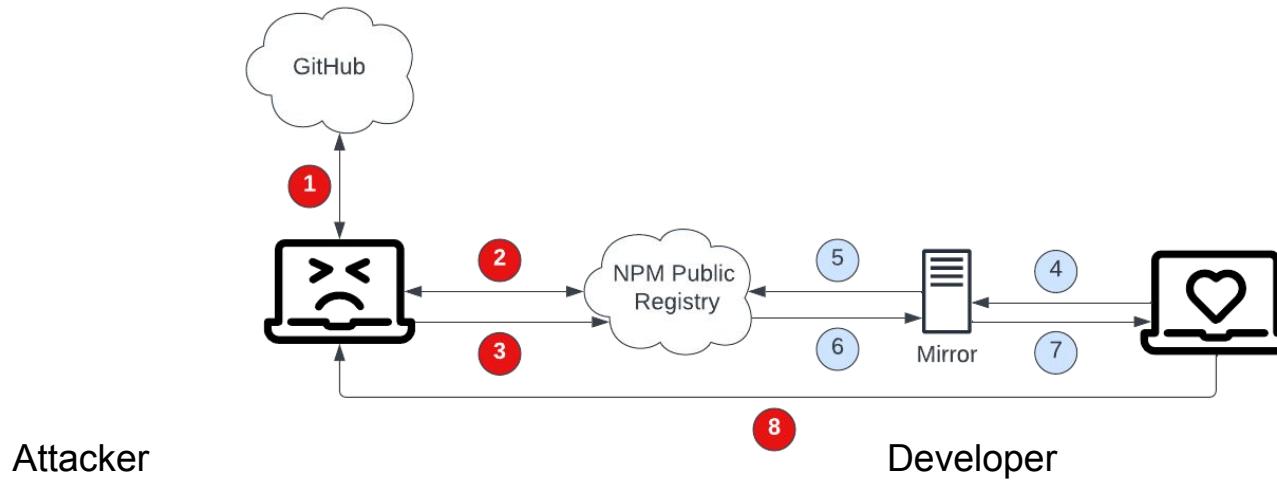
# Attacking npm Packages

## Classes of attacks

| Type  | Description  |
|---|--|
| Cryptojacking                               | Code is inserted to use computing resources to mine for crypto currencies (E.g. ua-parser-js)  |
| Typosquatting                               | Utilizes common misspellings of popular packages to install malicious code (E.g. react-model vs. react-modal)                                      |
| Dependency Confusion (Substitution Attacks) | These can occur when a project relies on private packages from an internal registry and attacker published same named package on external registry |
| Data harvesting                             | Code executed on install or invocation scrapes local computer environment and sends data to hacker   |
| Ransomware                                  | Code is blocked from executing until other conditions are met  |
| Hijacking                                   | Account or code to publish modules is compromised and tampered with to distribute malicious intent   |
| Author's own attacks (Denial of Service)    | Authors publish nefarious versions to block usage for payment demands and/or to promote political views (E.g. node-ipc, colors, faker)             |

# Example of a Supply Chain Attack

## Substitution Attack



1. Finds package.json files that reference unscoped packages on github.com

2. Verifies packages don't exist on npm public registry

3. Publishes modified package incrementing version on npm public registry

8. Malicious code does its work

4. Queries local mirror for latest version

5. Mirror determines that latest version is on npm public registry

6. Mirror pulls and caches latest version overwriting legitimate version

7. Receive malicious version

# Reduce Risks by being Well-Managed

- Be intentional on the usage of Node.js
- Use “Golden” Images
- Centers of Excellence can help drive culture of responsible use
- Track package usage through SBOMs
- Push “left” the detection of vulnerable versions and block releases at appropriate CVE levels
- Adopt best practices around security, training, and usage of all components in the software lifecycle

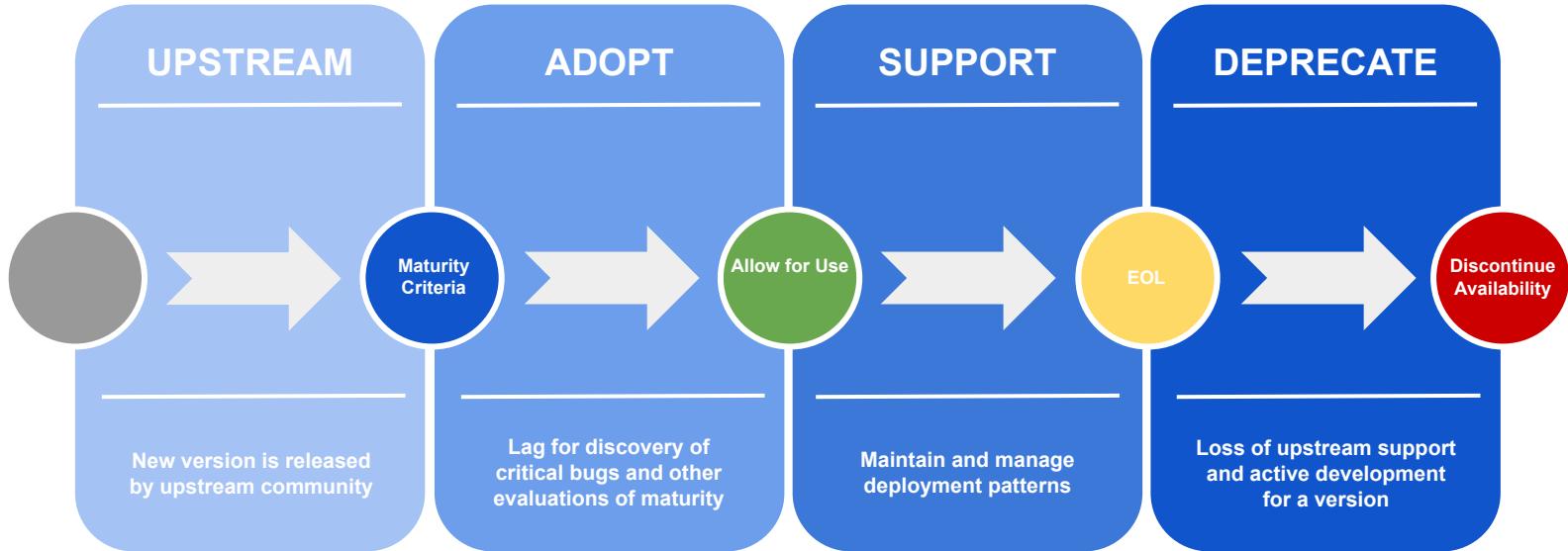


Tip: Metrics and associated dashboards can help raise the visibility of your security stance.



# Be Intentional on Node.js Version Usage

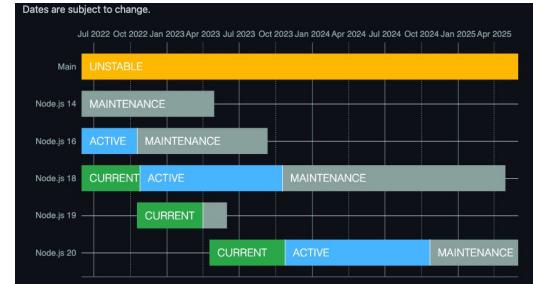
The life cycle stages of version acceptance



# Be Intentional on Node.js Version Usage

## Defining the “Maturity Criteria” to be used

- Only consider LTS versions
  - Only consider Active and Maintenance, prefer Active whenever possible
- Accept Security Releases on day of release
  - These supersede all previous releases in a particular LTS line
- Lag Minor releases by 2 weeks
  - Allow for discovery of critical issues immediately after release
- Move to Major releases (new LTS) when LTS is declared
  - Lag is built into the Current -> Active release cycle by Node.js itself but sometimes libraries can still lag
- Deprecate on Node.js schedule
  - Non-negotiable - plan ahead to move to new version **before** EOL dates



Tip: Node.js publishes their release schedule [here](#)

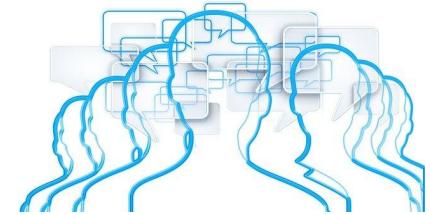
# Use “Golden Images”

- Golden images (usually Docker containers) are centrally owned & built
- Golden images have no known vulnerabilities at the time of building
- Centrally hosted and the basis for downstream container images
- Continuously updated for base OS specific vulnerabilities
- Updated with Node.js based on maturity criteria
- Configured for Enterprise use and compliance



Tip: Consider Distroless images to decrease vulnerability surface and associated maintenance of containers

# Node.js/JavaScript Center of Excellence



- The group meets on a regular cadence
- The main goal of the CoE is to govern and recommend best practices and policies for using Node.js, and by extension, JavaScript within the enterprise
- The CoE is made up of subject matter experts (SMEs) from across the Enterprise. For fair representation, the center has at least one member from each major platform or application utilizing Node.js and/or JavaScript

# Main Responsibilities of the Center of Excellence

- Approves Node.js versions as they are released as part of the Open Source Version Life Cycle Management process, following the maturity criteria
- Generates policy in conjunction with the Open Source Program Office (OSPO) for responsible Node.js and JavaScript usage across the Enterprise
- Drives any Node.js or JavaScript specific Enterprise Tech Backlog (ETB) items relating to policy adherence as needed
- Promotes the safe and responsible usage of Node.js, JavaScript, and associated open source modules within Capital One
- Provides SMEs for code reviews for any public-facing open source projects written using JavaScript
- Promotes contributions to Node.js itself, or Node.js and JavaScript open source modules external to Capital One
- Drives the automation of language version approvals and management of Node.js or JavaScript based Community Images
- Works with the appropriate groups within the enterprise to drive for standard ways of consuming and utilizing Node.js that fit within enterprise systems (and attempt to adhere to industry standard best practices)
- Owns the npm public registry presence of Capital One at [https://www.npmjs.com/~cof\\_official](https://www.npmjs.com/~cof_official)



# Track Package Usage

A Software Bill of Materials (SBOM) helps audit usage

- SBOMs should be tracked for each release
  - Required artifact for a release
- Provides a “usage” lineage that can assist in vulnerability detection
- Allows for auditability of software releases
- Helps provide visibility to what is in use and for determining frequency of use across an Enterprise when aggregated reporting is in place
- Assists in the detection of unsanctioned use and/or remediation activities related to vulnerabilities, licensing, or deprecations



# Developers should be educated

Code safely and responsibly

- Continuously invest in your development (and your developers) around security best practices in the code you write
- Beware of the top ten OWASP (<https://owasp.org/www-project-top-ten/>) and the common remediations
- Be aware that the security landscape changes constantly
- Keep dependencies up-to-date



# Evaluate packages before use

Tools and processes can assist in detecting issues early

- Train developers on look-up services before adding dependencies into their projects (I.e. basic due diligence)
  - [npm public registry](#), [Socket](#), [Snyk](#), [CVE Details](#), etc.
- Evaluate packages through release cadence, GitHub issues, maintainability, etc. before bringing them into a project
- Use industry standard tools to scan dependencies



Tip: Block releases when standards are not met or level of CVE exceeds thresholds!



# Use tools whenever possible

To reduce developer toil, push detection “left” as much as possible

- Besides scanning for vulnerable packages, scanning for code through static code analysis is just as important
- Use linting to capture before code is even submitted to source control
- Use tools integrated into GitHub to block PRs based on static code scan results
- Use other scanning tools that help auto-remediate or discover issues early in the coding lifecycle
- Use dependency tools to automatically handle dependency updates
  - [Dependabot](#), [Snyk.io](#), etc.



Tip: You must have **reliable** automated testing in place to trust automation!



# Npm Package Developer Best Practices

Developers can protect themselves to a degree

- Use only trusted packages (although these have been taken over the past)
- Lock versions using the correct syntax in the package.json and use lock-files appropriately
- Monitor network and process activity on Node.js processes
- Don't publish private code on external GitHub public repositories
- For produced packages, apply owned scope to all internal/externally published packages



Tip: The OpenSSF [npm Best Practices Guide](#) contains additional and more detailed guidance.



# Npm Package Publishing Best Practices

The public registry is doing a lot but you can do more!

- Setup 2FA on the public npm registry for publishing
- Create and own your organization on the registry (there are ways to “reclaim” if it is already taken)
- Use scoped packages (if possible)



Tip: The OpenSSF [npm Best Practices Guide](#) contains additional and more detailed guidance.



# In Summary

- The vulnerability landscape changes constantly
- Be well-managed to reduce risk
- Keep your developers up-to-date on security practices and responsible usage
- Have reliable and automated testing to trust the tools that help reduce developer toil
- Use dashboarding and monitoring to get a holistic view of your own usage
- Follow the recommendations by npm/OpenSSF on publishing of both internal and external packages

# Want to learn more?

- Want to learn more about our Tech? Check out [Capital One Tech](#) to find out more about enterprise software solutions, ideas and stories.
- At Capital One, we celebrate and honor the differences that makes us all unique- inside and outside of work. Help us create a more equitable future for all! Join us! Visit [Capital One Careers](#) to view our open roles.
- Follow us on Twitter at [CapitalOneTech](#)



# Thank you!

Contact Me:

LinkedIn: <https://www.linkedin.com/in/stevehusak/>

Twitter: @shusak

Email: [stephen.husak@capitalone.com](mailto:stephen.husak@capitalone.com)

Slides: <https://github.com/shusak/finos2023>



All trademarks, logos, and brand names are the property of their respective owners. All company, product and service names used in this presentation are for identification purposes only. Use of these names, trademarks, and brands does not imply endorsement.





# OPEN SOURCE IN FINANCE FORUM New York

PRESENTED BY



Thank you!