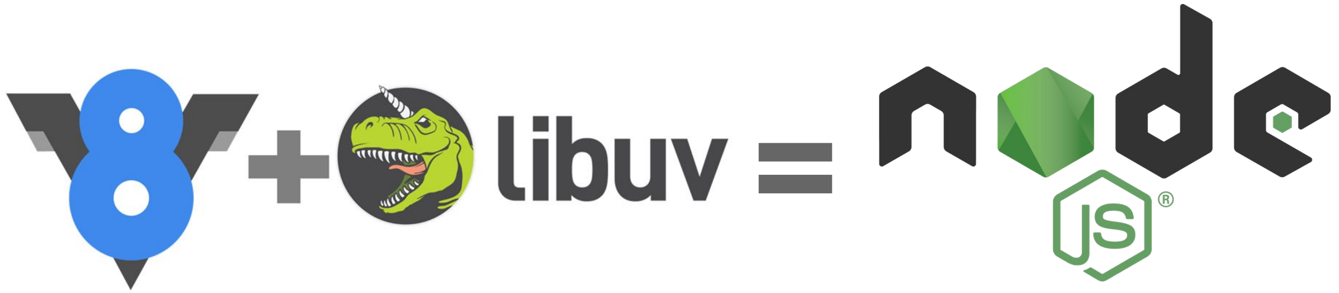
MININODE: Reducing the Attack Surface of Node.js Applications

Igibek Koishybayev
North Carolina State University
ikoishy@ncsu.edu

Alexandros Kapravelos
North Carolina State University
akaprav@ncsu.edu

What is Node.js?



npm

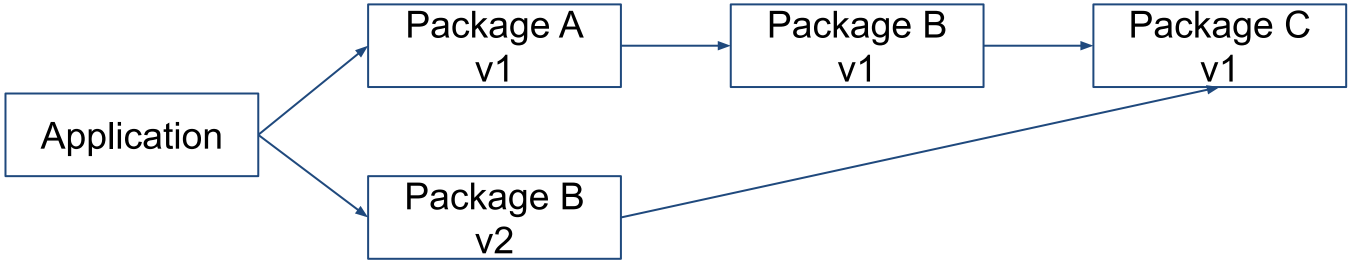
yarn

CommonJS

Node Package Manager (NPM)

NPM is the largest package manager by number of hosted packages

- 1.3M packages as of 7/19/2020
- Majority of the packages are simple packages consisting only one function
- Installs dependencies transitively

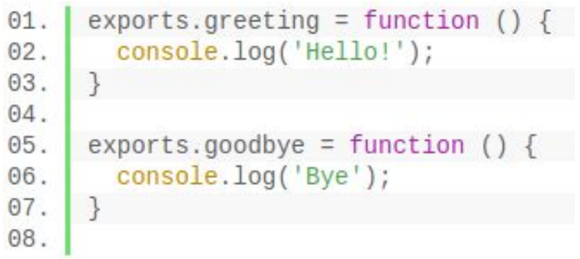


CommonJS module system

- CommonJS is **not** standard module system, but a workaround

- Modules use *exports* object to export and *require()* function to import

- Underhood *require()* wraps module's code to isolate its scope

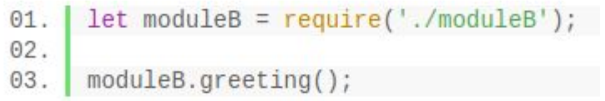


```
exports.greeting = function () {
  console.log('Hello!');
}

exports.goodbye = function () {
  console.log('Bye');
}

```

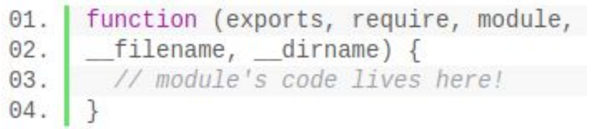
ModuleB. Example of exporting functionality



```
let moduleB = require('./moduleB');

moduleB.greeting();
```

ModuleA. Example of importing functionality



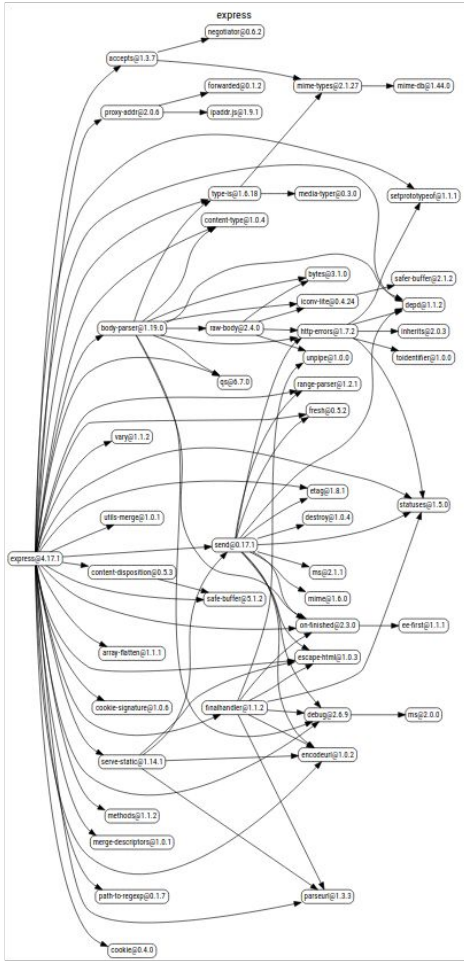
```
function (exports, require, module,
__filename, __dirname) {
  // module's code lives here!
}
```

Require function wrapper

NC STATE UNIVERSITY

Problem statement

- Node.js applications suffer from dependency explosion
- Some popular packages may depend on 200 other packages, some of which outdated and/or vulnerable
- All applications by default have access to built-in modules (fs, net, and *etc*)

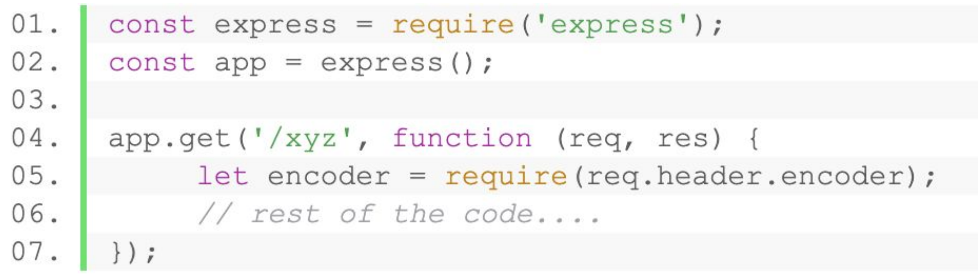


Threat Model

The attacker can execute arbitrary code due to vulnerability inside Node.js application and, thus, load unused modules.

There are two ways to load the unused modules:

1. Directly manipulating the `require` function
2. Indirectly manipulating the `require` function

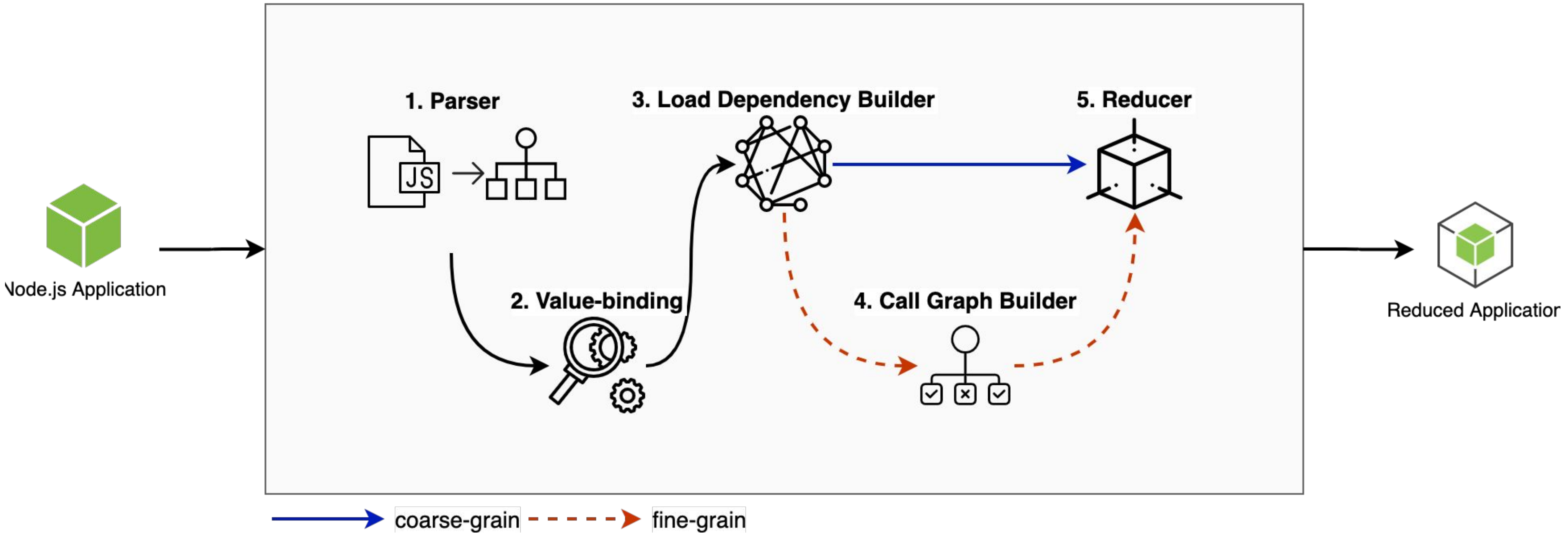


```
const express = require('express');
const app = express();

app.get('/xyz', function (req, res) {
        let encoder = require(req.header.encoder);
        // rest of the code....
});
```

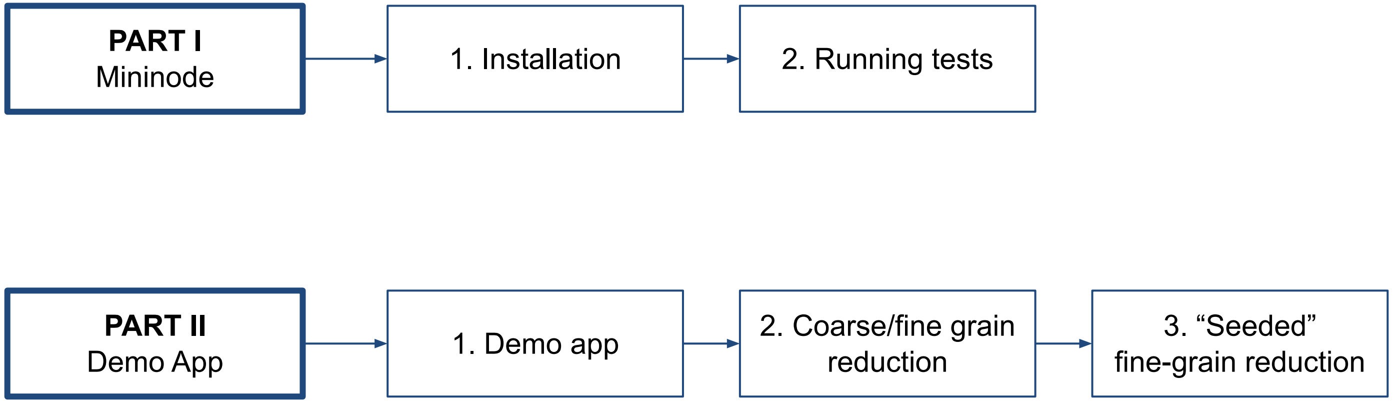
Example of directly manipulating the require() function

Mininode's Architecture



Mininode consists of five parts and supports two modes of reduction: (1) coarse-grain; (2) fine-grain

Tutorial Outline



Demo

https://github.com/wspr-ncsu/mininode/wiki/TCPC'21---Tutorial-Session
(short version: https://go.ncsu.edu/tpcp-mininode)

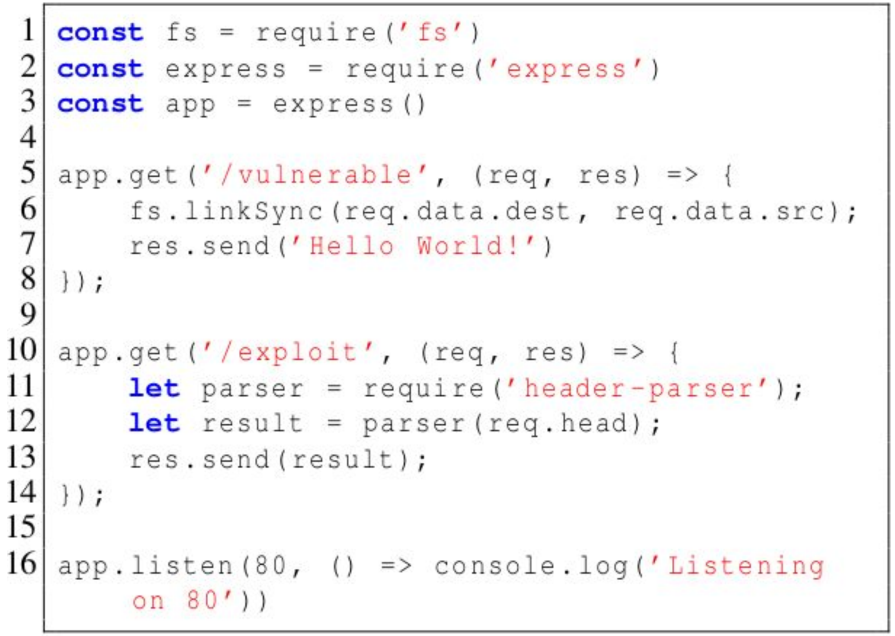
Thank you

Questions?

Repository: https://github.com/wspr-ncsu/mininode
Tutorial: https://go.ncsu.edu/tpcp-mininode

Threat Model: Complex Example

Attacker can “indirectly” manipulate the require() function



```
const fs = require('fs')
const express = require('express')
const app = express()

app.get('/vulnerable', (req, res) => {
    fs.linkSync(req.data.dest, req.data.src);
    res.send('Hello World!')
});

app.get('/exploit', (req, res) => {
    let parser = require('header-parser');
    let result = parser(req.head);
    res.send(result);
});

app.listen(80, () => console.log('Listening on 80'))
```