MININODE: Reducing the Attack Surface of Node.js Applications

Igibek Koishybayev
North Carolina State University
ikoishy@ncsu.edu

Alexandros Kapravelos
North Carolina State University
akapra@ncsu.edu
What is Node.js?
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.

ModuleB. Example of exporting functionality:
```javascript
01. exports.greeting = function () {
02.     console.log('Hello!');
03. }
04.
05. exports.goodbye = function () {
06.     console.log('Bye');
07. }
08.
```

ModuleA. Example of importing functionality:
```javascript
01. let moduleB = require('./moduleB');
02.
03. moduleB.greeting();
```

Require function wrapper:
```javascript
01. function (exports, require, module, __filename, __dirname) {
02.     // module's code lives here!
03. }
04. 
```
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- Some popular packages may depend on 200 other packages, some of which outdated and/or vulnerable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

Example of directly manipulating the `require()` function

```javascript
01. const express = require('express');
02. const app = express();
03. app.get('/xyz', function (req, res) {
04.     let encoder = require(req.header.encoder);
05.     // rest of the code....
06. });
```
Mininode’s Architecture

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

PART I
Mininode

1. Installation
2. Running tests

PART II
Demo App

1. Demo app
2. Coarse/fine grain reduction
3. “Seeded” fine-grain reduction
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

```javascript
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'))
```