A Roadmap for Server-Side JavaScript Sandboxing

Daniel Hedin Mohammad M. Ahmadpanah Tamara Rezk Andrei Sabelfeld


Chalmers

KTH


Mälardalen


Inria

KTH TCS Seminar

June 9, 2025

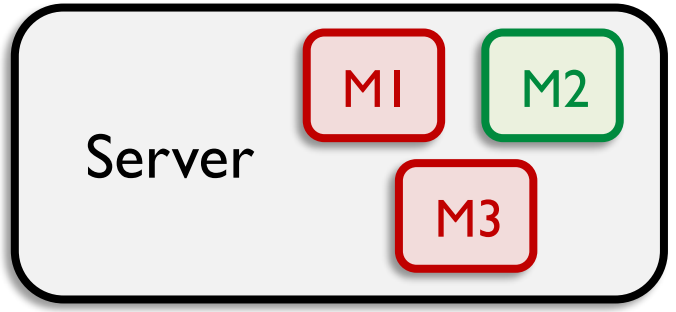
Sandboxing

- Securing *untrusted* code execution
- Server-side integration
  - Third-party modules
  - User code
- Use cases
  - User automation

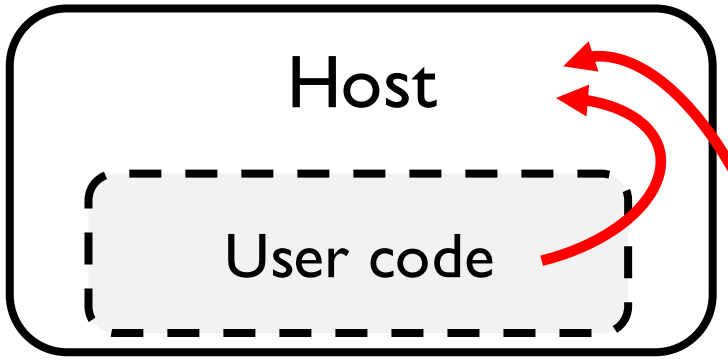
IFTTT zapier make Node-RED

  - Cloud-based code execution

CLOUDIFY nango JITSU



Sandbox breakout



- What if sandboxing *fails*?
  - Exposing sensitive data
  - Executing arbitrary code

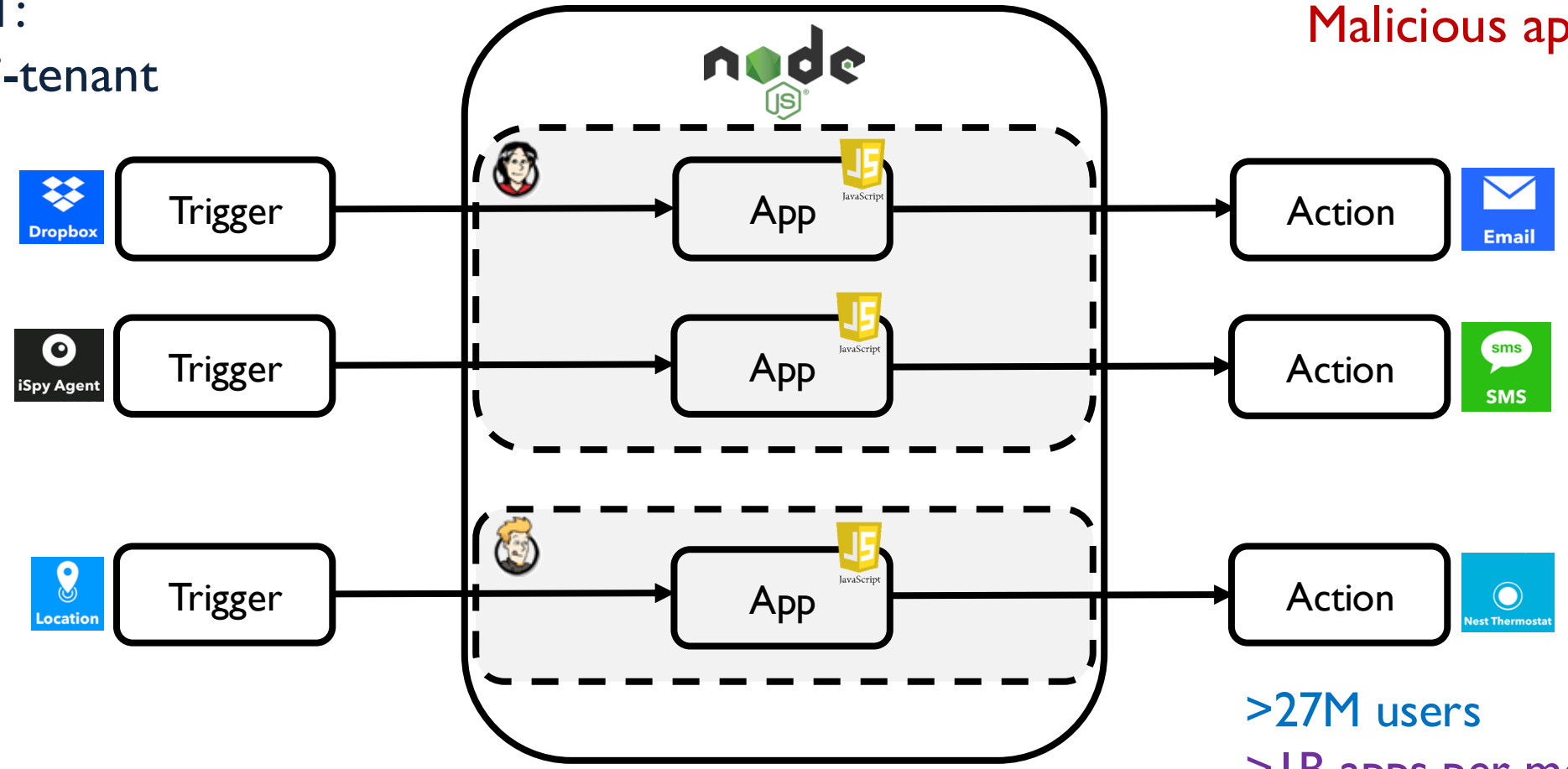
- Code exec via host object

```
function stack() {   new Error().stack;  stack();  }
try { stack(); }  catch (e) {
  e.constructor.constructor('return process')().mainModule
   .require('child_process').execSync('echo pwned!'); }
```

IFTTT: architecture

IFTTT:
multi-tenant

Threat model:
Malicious app maker



>27M users
>1B apps per month
>900 partner services

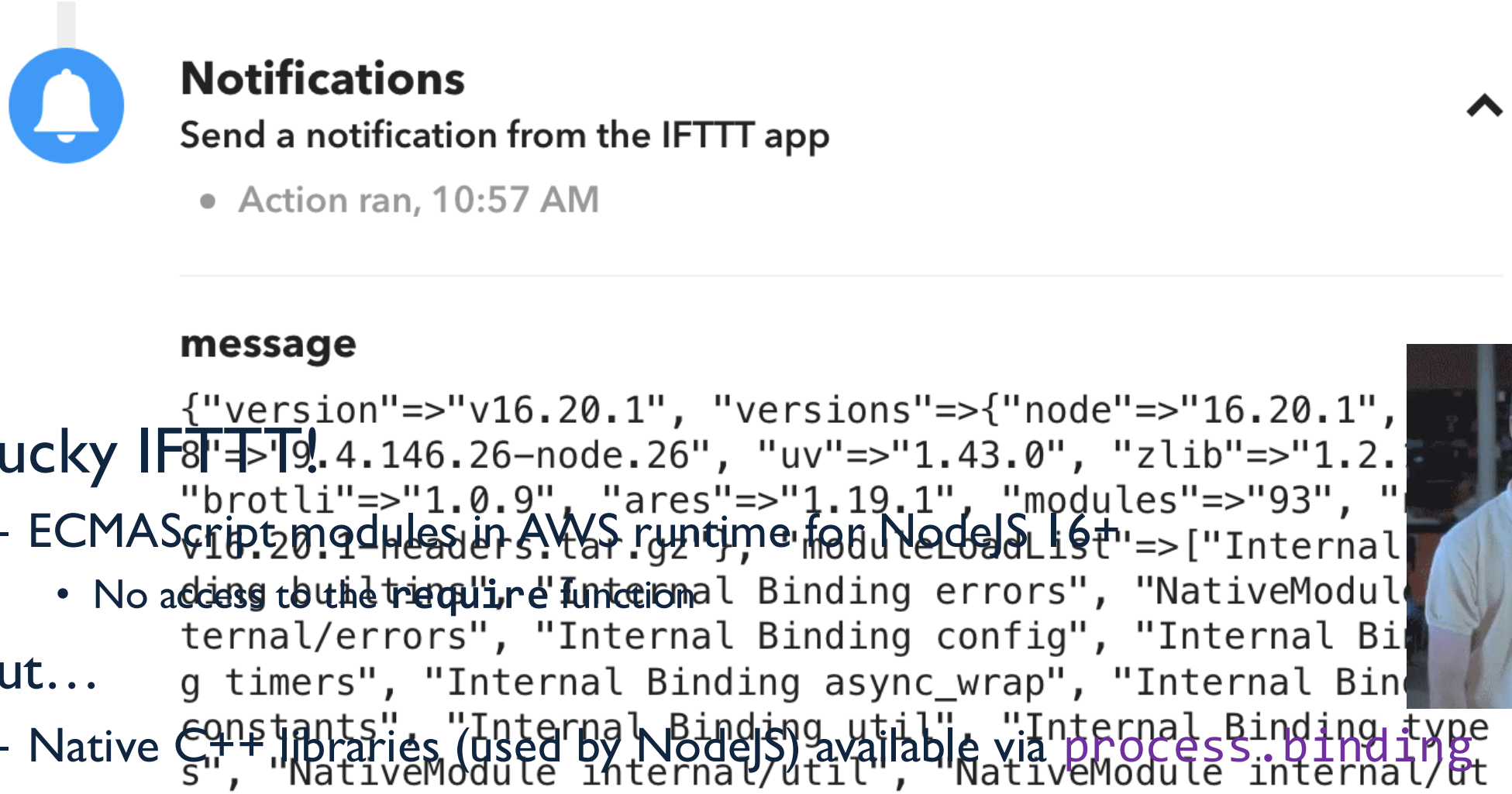
IFTTT: sandboxing

- JavaScript of the app runs inside AWS Lambda
- Node.js instances run in Amazon's version of Linux
- AWS Lambda's built-in sandbox at process level
- IFTTT: "App code is run in an **isolated** environment"

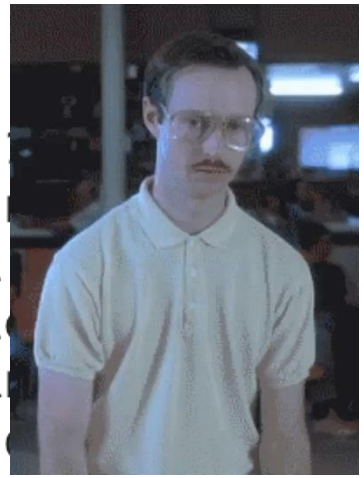

```
function runScriptCode(appCode, config) {
   … // set trigger and action parameters
   eval(appCode) }
```

  - Security checks on app code
    - TypeScript syntactic typing
    - Disallow `eval`, modules, sensitive APIs, and I/O
    - **vm2** isolation (received bounties; continuous interactions on fixing in 2020)

IFTTT: sandbox breakout



- Lucky IFTTT!
  - ECMAScript modules in AWS runtime for NodeJS 16+
    - No access to the `require` function
- But…
  - Native C++ libraries (used by NodeJS) available via `process.binding`



IFTTT: sandbox breakout (cont.)

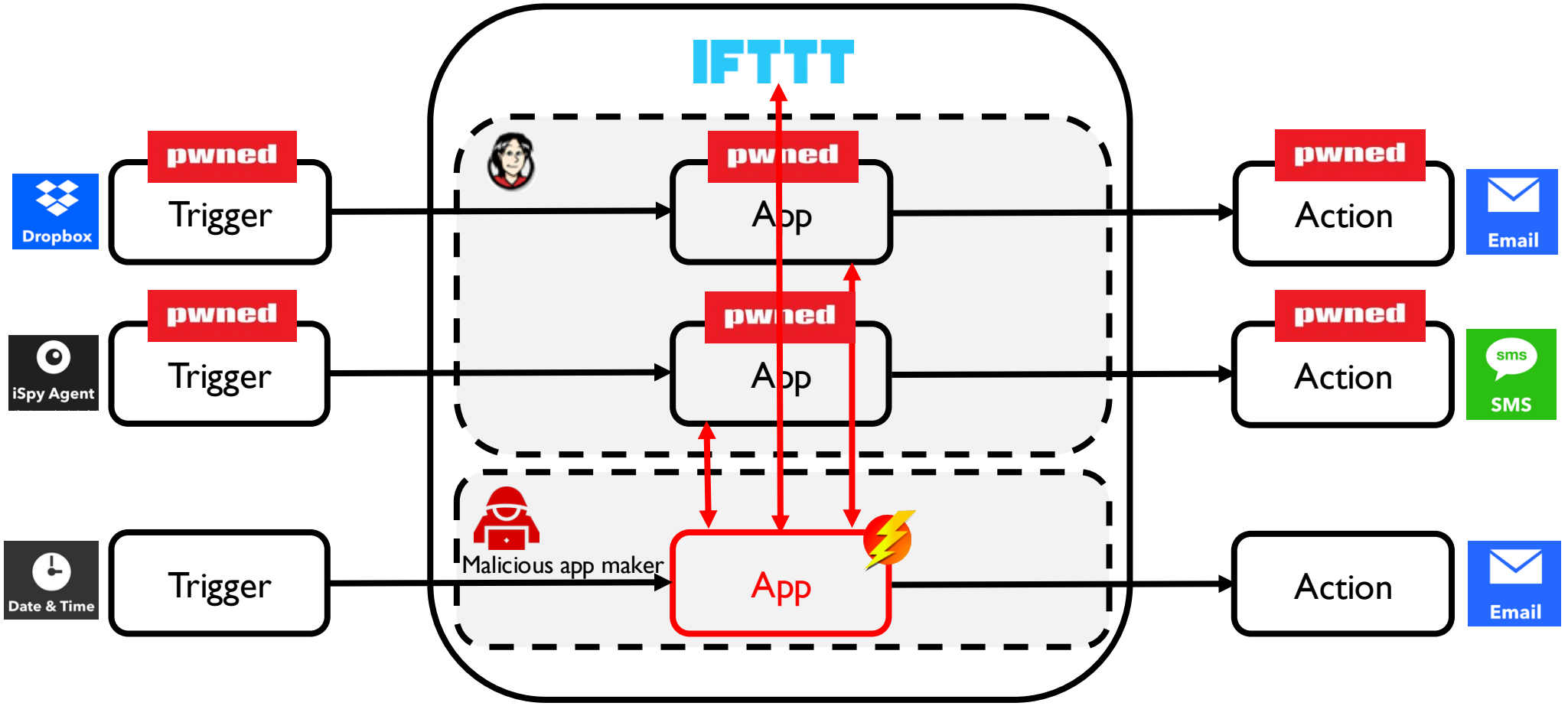
```
function stack() {   new Error().stack;  stack();  }
try { stack(); }  catch (e) {
 let process = e.constructor.constructor('return process')();
 let spawn_sync = process.binding('spawn_sync');
 …
 IfNotifications.sendNotification.setMessage(spawn_sync.spawn(trigger.cmd))
}
```

Notifications
Send a notification from the IFTTT app

message
$ {whoami } output: sbx_user1051 | error:

- Remote code execution (shell access)
- DLL injection (dynamically loading C++ modules)
  - Writing to `/tmp` via `process.binding('fs')`
  - `process.dlopen(module)` to load the injected binary module

IFTTT: sandbox breakout (cont.)



User installs *benign* apps from the app store

Compromised: Trigger and action data of the benign apps of the ***other*** users

IFTTT: sandboxing now

- AWS Lambda's built-in sandbox at process level
- Security checks on script code of the app
  - TypeScript syntactic typing
  - Disallow `eval`, modules, sensitive APIs, and I/O
  - Finally, **isolated-vm** (received bounties; continuous interactions on fixing in 2023)

AWS Lambda

Notifications

Send a notification from the IFTTT app

- Action ran, 2:58 PM

message

```
CATCH process is not defined | ReferenceError: process is n
ot defined at eval (eval at <anonymous> (<isolated-vm>:13:3
9), <anonymous>:3:1) at <isolated-vm>:13:68
```

CloudJS*: sandboxing

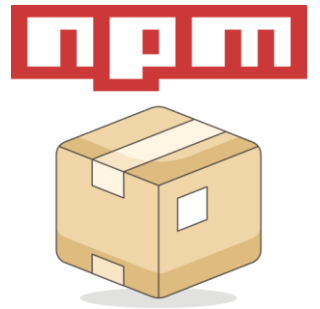
- Cloud-based JS code execution service
  - Integrated into several well-known automation platforms
- **Multi-tenant** AWS Lambda
- **Rich execution environment**
  - Including *storage* and *networking* capabilities
- **No sandboxing mechanism!**
  - Restricting access `child_process` by overriding `global.require`
  - Multiple ways to access `require` remained
  - **vm2** isolation
  - Breaking via the error stack technique
  - Vulnerabilities confirmed with PoCs in 2025

- Powerful modules (e.g., `axios`, `crypto`) shared between users of different platforms
  - **Platform-level cross-user attacks**
  - Significantly challenging to adapt `isolated-vm`
  - In contact on fixing with integrating our solution

* Name redacted for the ethical reasons

vm2: popular JS sandbox

- Sandboxing solution used in
  - Cloud platforms
  - User automation apps
  - AI agent frameworks
  - Development SDKs
- Over **585M** npm downloads since 2014
- Popularity reasons
  - Easy-to-use
  - Support for `require` (CJS modules)
  - Module mocking and API-level JS injection
  - Language-based (affordable overhead)
  - vm-based + proxy membranes



vm2: deprecated JS sandbox

XmiliaH on Jul 10, 2023 Collaborator Author •••

Xion (SeungHyun Lee) of KAIST Hacking Lab found the vulnerabilities I am not able to fix without changing the whole sandboxing strategy.

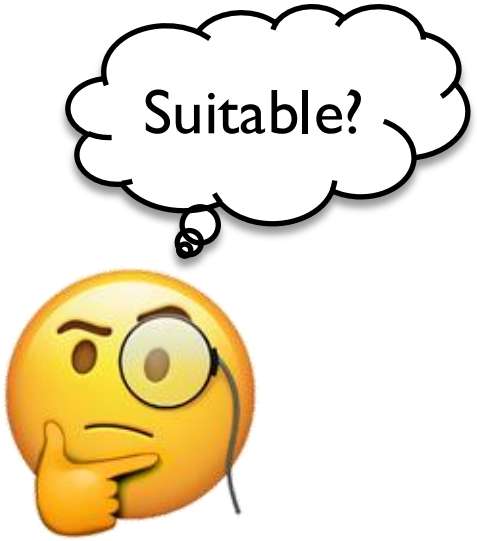
- Fatal flaws in vm2
  - Language-level: Unmodeled reflected APIs
  - Breaking language constructs: `import`, `eval`, and `async`
  - Also in other vm-based solutions
    - "vm is not a security mechanism. Do **not** use it to run untrusted code."
    - SandTrap: fine-grained policies + allowing for complex host/sandbox interaction
    - SandTrap's response: significantly locked down; some detrimental effects on use cases

vm2: deprecated JS sandbox (cont.)

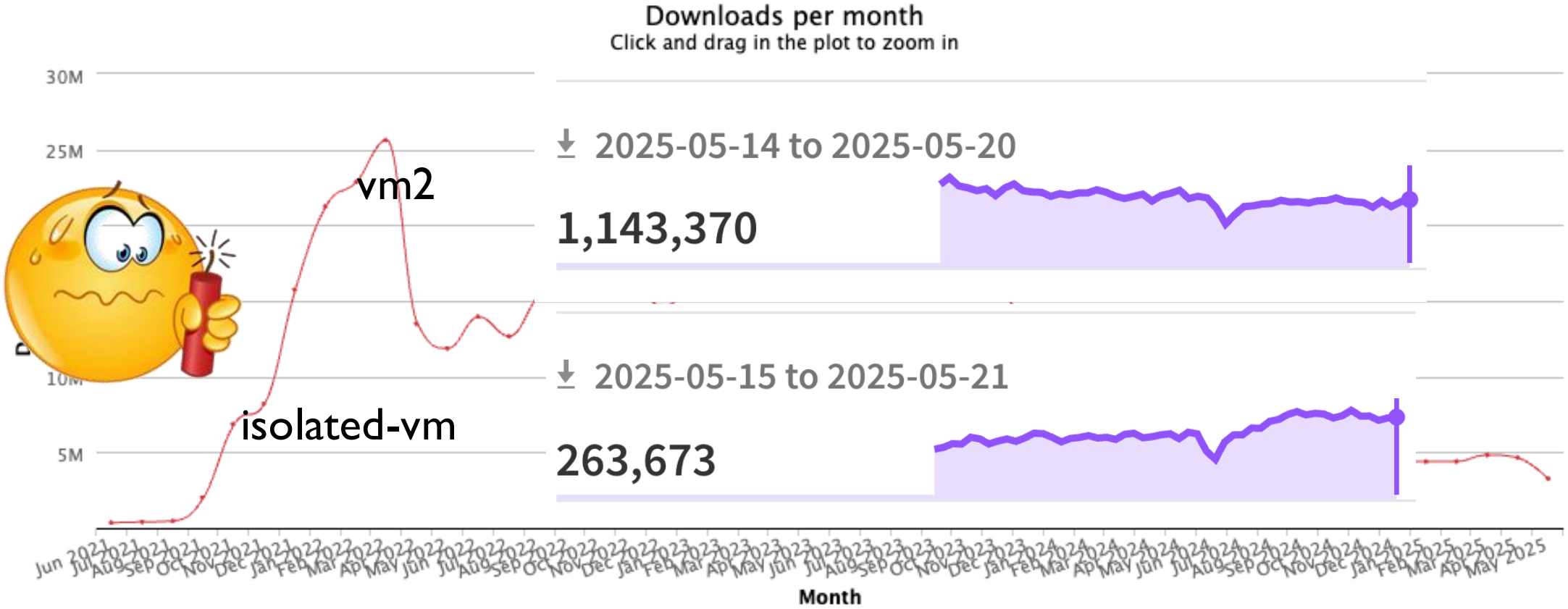
‼ Project Discontinued ‼

TL;DR The library contains critical security issues and should not be used for production! The maintenance of the project has been discontinued. Consider migrating your code to isolated-vm.

- Suggested alternative: **`isolated-vm`**
  - Pros: secure, using v8's interface (runtime-based)
  - Cons:
    - No CJS/ES support
    - Limited support for policies
    - Sharing between isolates are risky/far from transparent

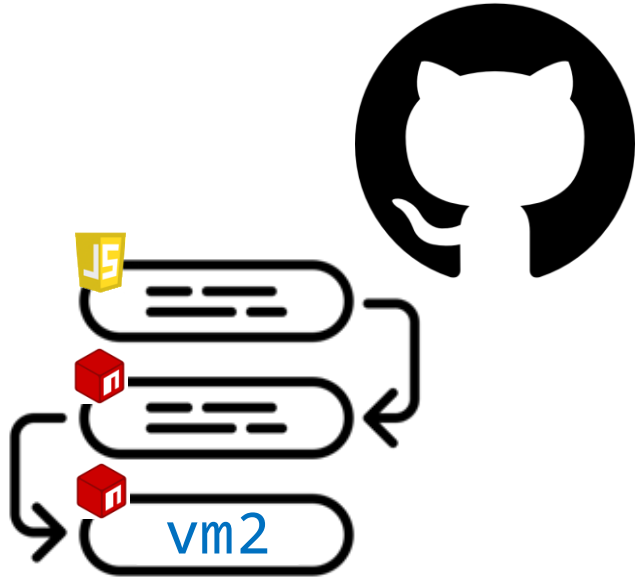


vm2: deprecated yet popular (!) JS sandbox



Post-vm2 migration analysis

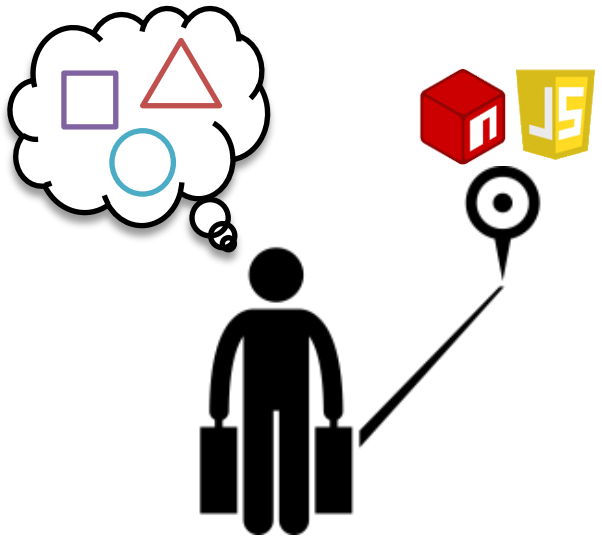
- Crawling the ***GitHub dependents*** pages
  - Using `github-dependents-info`
  - 183,149 unique repos collected
- Dealing with GitHub *false positives*
  - All repositories cloned
  - `git log` to find ***actual use*** of `vm2`
  - Careful regex patterns for `require`/`import` `vm2`
  - 3,127 repos with actual use identified
  - Excluding hits from less interesting packages (e.g., `degenerator`, `yarn`, and `eslint`)
  - **1,159 repos** identified to have migrations away from vm2



Post-vm2 migration analysis (cont.)

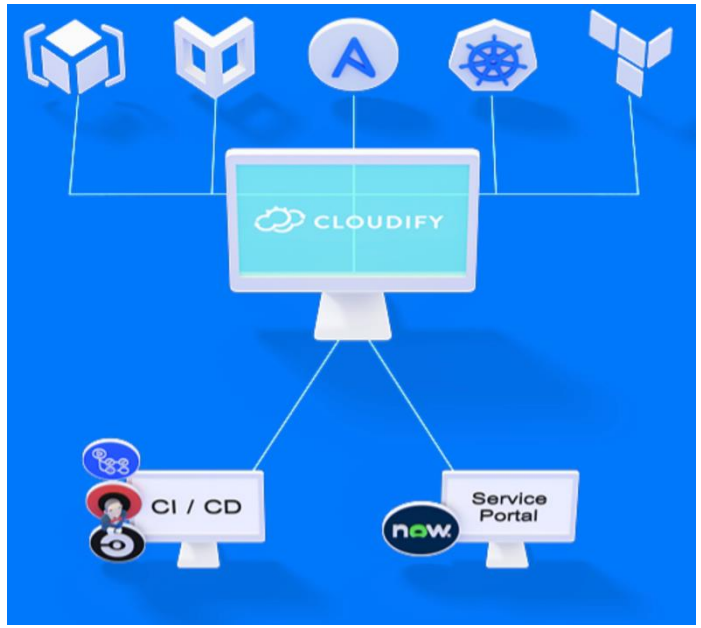
- **70** repos analyzed (+5 stars)
  - 19 high security needs in core features
  - Only 8 with isolated-vm-based solutions: sometimes ***overly complex ad-hoc injections***
  - 15 known-as-insecure alternatives (eval, require, vm): *no sandboxing needed!*
  - 10 dropped feature/discontinued
  - Some with **expensive workarounds**: *process-level* isolation and standalone *runtimes*

- **Use cases**
  - Pure isolation
  - Isolation with access to CommonJS
  - Isolation with injection capabilities
  - Isolation with more advanced policies (e.g., controlling fs)
  - Some combination of the above and more



Cloudify

- Cloud integration service
  - A common API to other cloud/web services
- User-defined *widgets* to extend the platform
- Security need: critical when multi-tenant
- Migration:
  - **From:** NodeVM full `require` access + injected interaction APIs
  - **To:** isolated-vm, with complex workarounds and feature loss



Cloudify (cont.)

Merged **[NE-5433-6171]Replace vm2 package with isolated-vm for backendwidget** #2587

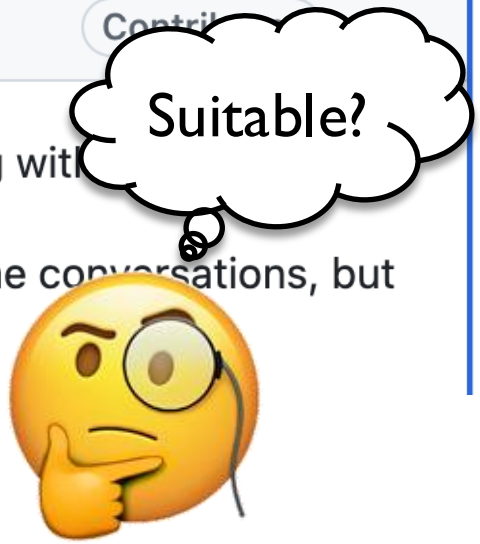
jyoti-siddareddi merged 18 commits into `master` from `NE-5433-6171-replace-vm2` on Oct 27, 2023

Vorbert-Kruk reviewed on Oct 17, 2023 View reviewed changes

Vorbert-Kruk left a comment

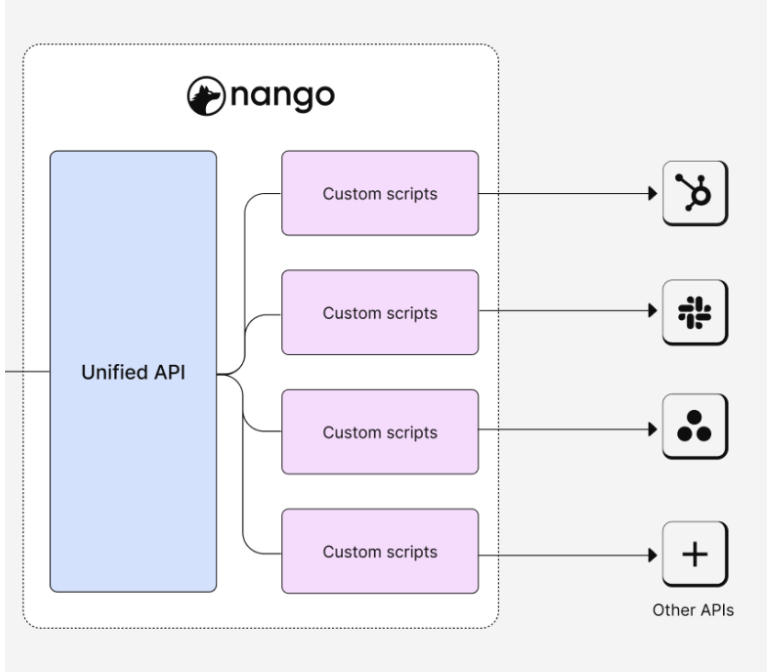
Overall, I'm not certain about the amount of breaking changes that we are introducing with

Maybe it's an outcome of not reviewing finalized PR or from not being involved in some conversations, but it seems that we are breaking more stuff that was initially planned



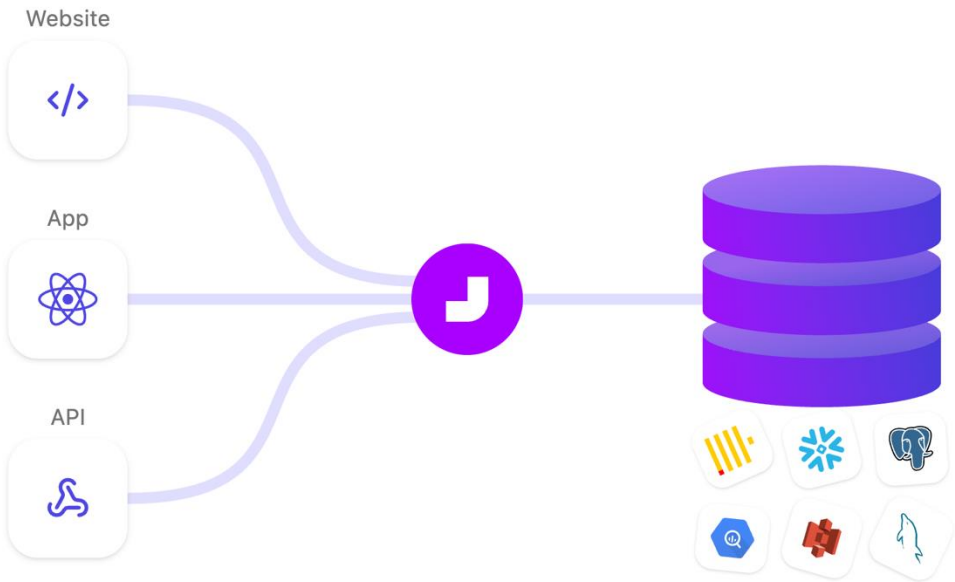
Nango

- API integration platform
  - A unified API to interact with multiple external APIs
- User-contributed APIs and integrations can be added to the project via pull requests
- Security need: critical when multi-tenant
- Migration:
  - **From:** NodeVM with full require access + injected interaction APIs
  - **To:** first *quickjs-emscripten* (overhaul + major limits), then *redesigned!*



Jitsu

- Event collection and processing, streaming to data warehouses
  - For site analytics and data collection
- User-defined functions in its cloud environment
- Security need: isolation with CJS, critical when multi-tenant
- Migration:
  - **From:** NodeVM with limited require
    - *Opt for vm2 due to ease of use*
  - **To:** handmade isolated-vm-based



Sandboxing alternatives

- JS-Interpreter
- quickjs-emscripten
- SES (npm package)
- isolated-vm
- Non-alternatives:
  - jailed, safe-eval, near-membrane: known breakouts
  - NodeSentry, BreakApp, deno-vm: unavailable
  - Worker thread, tiny worker: unrestricted access to env

	Sandbox
Runtime-based	TreeHouse [32]
	BreakApp [65]
	jailed
	deno-vm
	isolated-vm
Language-based	vm2
	realms-shim
	ses
	safe-eval
	notevil
	SandTrap [14]
	MIR [66]
	near-membrane
	AdSafe
	Caja

[SandDriller]

JS-interpreter

- Sandboxed JavaScript **interpreter** in JavaScript
  - Line-by-line execution of *ES5*
- **Setup phase:** injecting *functions* and *objects*
- **Execution phase:** only *primitive values*
- **No module support**

quickjs-emscripten

- **Interpreter-based** sandbox supporting most of *ES3*
  - A Wasm binding of the QuickJS interpreter
- Sharing only **primitive values** + injecting **host functions** into the sandbox using *a special API*
  - Unmodified host functions cannot be injected
- Deployable on the client side too

SES

- An implementation of *Hardened JS*
  - A language subset to implement isolation via *compartments*
- **Locking down** the shared execution environment
- Shared values are **hardened** or **frozen**
  - Preventing attempts to tampering with the global objects
- **Only SES-compatible source modules**
- Requiring `strict` mode
- **Forbidden**: dynamic imports, direct calls to `eval`
- **Heavy impacts on the host**
  - No load of many modules after lockdown

isolated-vm

- Exposing lightweight **V8's Isolate API**
  - Same isolation mechanism used by Chromium to separate tabs, iframes, web workers, and service workers
- Sharing only **by references**
- **Transparent sharing** only for
  - *Cloneable values*: objects containing methods are *not* cloneable
  - Functions receiving/returning *only cloneable values*
- **Limitations**
  - No prototype chain traversing of shared objects
  - Function references *cannot* be interacted with as regular objects
  - Complex sharing is ***not*** supported
    - The code of both the host and the sandbox must be changed (adapters)

isolated-vm (cont.)

SECURITY

Running untrusted code is an extraordinarily difficult problem which must be approached with great care. Use of `isolated-vm` to run untrusted code does not automatically make your application safe. Through carelessness or misuse of the library it can be possible to leak sensitive data or grant undesired privileges to an isolate.

PROJECT STATUS

`isolated-vm` is currently in *maintenance mode*. New features are not actively being added but existing features and new versions of nodejs are supported as possible. There are some major architectural changes which need to be added to improve the stability and security of the project. I don't have as much spare time as I did when I started this project, so there is not currently any plan for these improvements.

Lesson learned: suitability of sandboxes

- **Functionality**
  - *Method of sharing:* by cloning or by reference
  - *Type of sharing:* data only, with callables, full object sharing
  - *Module support:* CommonJS or ES modules
- **Security**
  - *Privilege escalation:* reaching unintended functionality
  - *Cross-boundary poisoning:* insecure modification of shared values
- **Performance**
  - Setup + cross-boundary interactions + execution

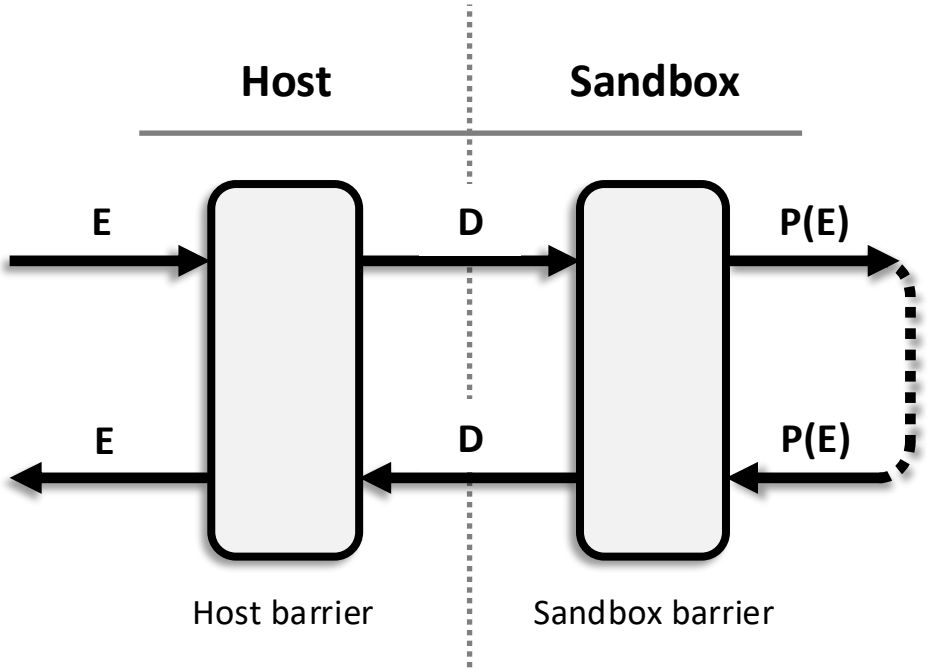
Fiberglass

Secure sharing of ***objects*** and ***modules***

- A proxy-based sandbox on top of **isolated-vm**
  - **Robust isolation**: no breakouts please!
  - Supports **full sharing** and controlled injection of **CommonJS** modules
  - Secure, mediated **bidirectional reference transfer** between host and sandbox
- Against *cross-boundary poisoning*: **modules are shared read-only**
- Against *breakouts*: **enforces controlled referencing**
  - Intrinsics are mapped to their local equivalent

Fiberglass (cont.)

- **One-sided barriers**, isolate-to-isolate sharing
  - **Sending** via *isolated-vm references* to pass values freely
  - **Receiving** via *proxies* to represent the received value locally
  - Unlike vm2 and SandTrap with *mutually recursive proxies*
  - Allows for *one-sided policy enforcement*
- Passing objects/functions using references to ***capabilities***
  - Precise *control on interactions* with sharing with other actors
- **Module allow-listing**



Suitability of Fiberglass

- Instantiated for Cloudify
- Ideal for the rich execution environment in CloudJS
  - With complex modules (`axios`, `crypto`, and `jsonwebtoken`)
- **Affordable overhead** on primitive operations
  - Synthetic benchmarks for calls from/to sandbox and property read/write
  - Subsumed by other factors in real-world scenarios (e.g., Cloudify)

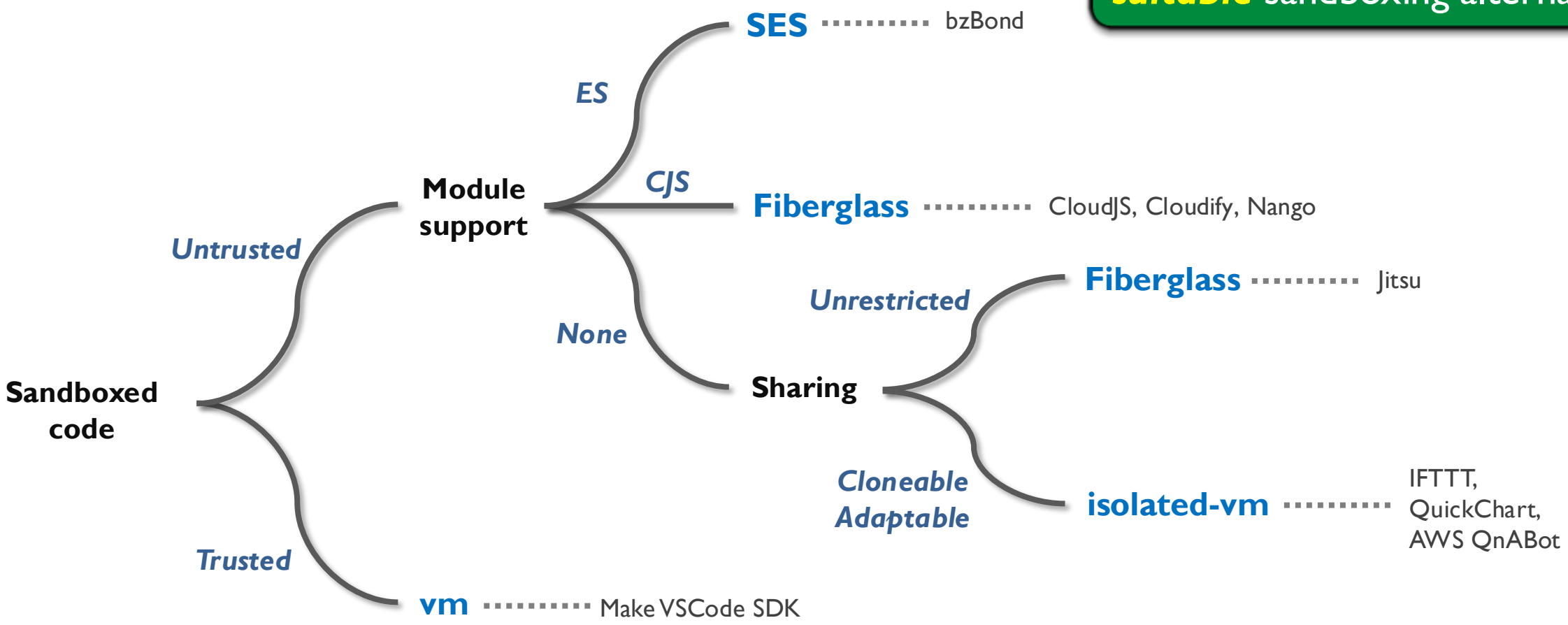
```
let fiberglass = require('fiberglass');
let sandbox = new fiberglass.FiberGlass(
    { },
    { modules : [ 'lodash', 'body/json', ... ] }
);
```

Considerable manual effort (isolated-vm)

Automatic way of sharing complex objects between the host and isolated-vm (Fiberglass)

Decision tree

A **systematic guide** to ***suitable*** sandboxing alternatives



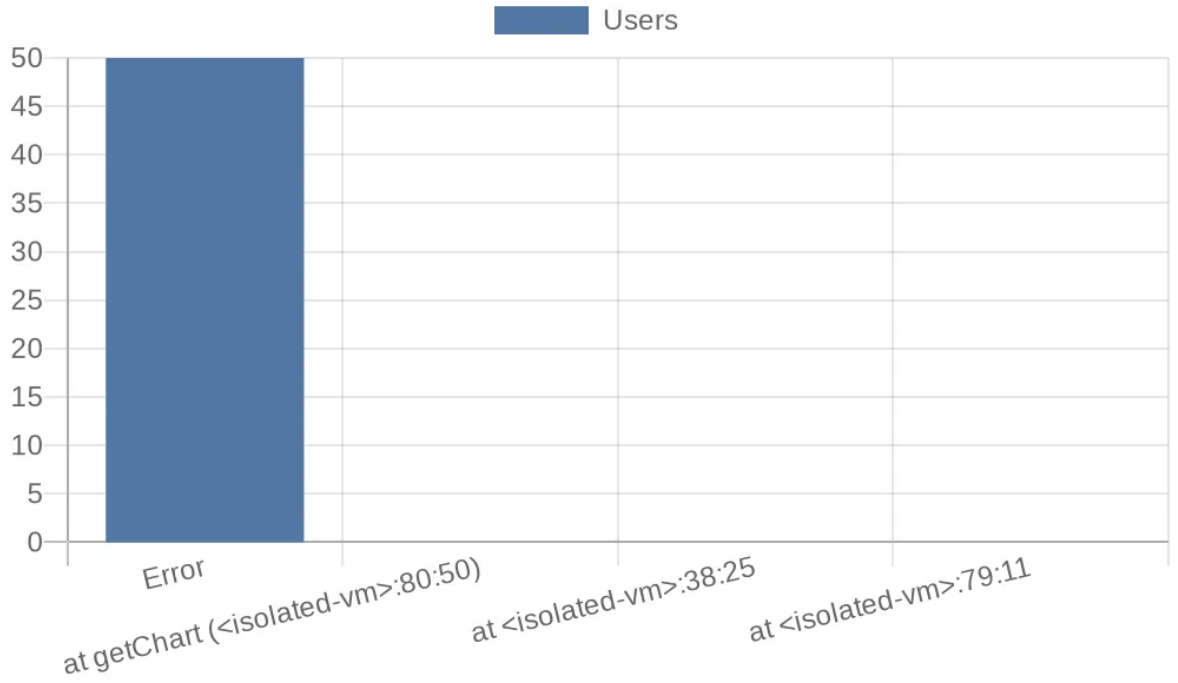
Takeaways

- Secure integration of untrusted code is ***still*** a serious challenge!
  - Bounties from IFTTT (2020, 2023) and breakouts confirmed by CloudJS (2025)
  - A clear lack of go-to substitutes for the *popular*, *deprecated* **vm2**
- Trade-off between ***functionality***, ***ease-of-use***, and ***security***
  - Migration analysis: *major efforts to find* ***suitable*** *alternatives*, sometimes sacrificing functionality, security, or both!
- **Fiberglass**
  - Robust isolated-vm + supporting full sharing/CJS
  - In contact with CloudJS on integrating Fiberglass
- **Decision tree** for developers to navigate the wild

Backup slides

QuickChart

- Generating chart images from a URL, might contain user code
- Security need: pure isolation, critical
- Migration:
  - From: basic use of NodeVM
  - To: isolated-vm
  - In the repo: none!



Metlo

- Open-source API security tool
- Security need: isolation with advanced policies, critical when multi-tenant
- Migration:
  - From: NodeVM plus mocking
  - To: Seems to be using VM without any protection!