0-DAY NO CVE HIGH PUBLIC

n8n, hidden Function node in-process vm2

n8n 2.42.0 Function / FunctionItem / LangChain Code still run JS via leftover vm2 NodeVM in the main process. Palette hidden is not an ACL. Code v2 already uses the task-runner child. No CVE yet.

NAME n8n, hidden Function node in-process vm2	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR n8n
AFFECTED n8n through 2.42.0 (86c23326); unpublished leftover of GHSA-j4p8. Default member with workflow create/import/execute. Image n8nio/n8n:2.42.0.	PUBLISHED 29 Sept 2026
UPDATED 29 Sept 2026	TAGS 0-day, n8n, rce, vm2, cwe-94, cwe-269, authenticated

TL;DR

Lab files and proof-of-concept here.

01 hidden is not an ACL

The proof of concept is on GitHub: **abraxas/n8n-function-vm2** (loopback client; @abraxas_null). The lab stack is `lab/` : Dockerfile, docker-compose.yml, run.sh. Authorized lab only. It talks to loopback.

This is **n8n 2.42.0** (`86c23326`), n8n GmbH. No CVE yet. CWE-94 / CWE-269. 8.8 High (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`). Authenticated `POST /rest/workflows` . Default member with create/import/execute on a personal project. Not OS LPE. Not a public vm2 gadget.

Same product, sibling: Databricks path join.

02 How I found it

I read the 16 September 2026 GHSA wave first, then the **n8n@2.42.0** tree: expression sandbox, Code/Function/vm2, credentials, webhooks, HTTP/SSRF, Git node, community packages, auth/SSO, node injection, public API. Most of the Highs in that wave are closed on this pin. Code was not finished.

GHSA-j4p8-h8mh-rh8q moved **`n8n-nodes-base.code`** onto the JS task-runner child. Production spawn adds `--disallow-code-generation-from-strings` and `--disable-proto=delete` . FS helpers are stubbed as unsupported. That is a real move.

The leftover is the nodes they hid instead of moving. `Function.node.ts` is `hidden: true` . So is FunctionItem. So is LangChain Code. Palette hide is a UI flag. Workflow JSON save/import resolves `nodeTypes.getByNameAndVersion` and continues. The catalog will still hand you a hidden node if you ask by name.

Default `NODES_EXCLUDE` is only Execute Command and Local File Trigger. Function still loads. v3 breaking-change rules already mark Function / FunctionItem **removed**.

On 2.42.0 they are leftover, not an intended long-term sandbox.

The execute path is in-process **vm2** `NodeVM`. Catalog `vm2` is 3.12.2. Abandoned. Not a security boundary. The main Node process does not get the task-runner hardening flags. Function also injects `this.helpers`, including helpers the JS runner lists as unsupported. I did not need a public vm2 gadget to prove the process boundary.

I stood up stock `n8nio/n8n:2.42.0`, signed up a **member** (not instance owner), imported `n8n-nodes-base.function`, executed it. Function `pid=7` is the n8n main PID. Same JS in Code v2: `hasProcess: false`. Task-runner child is pid 26. Witness `N8N-FUNCTION-VM2-WITNESS`.

I am not printing Function `functionCode` you can paste at someone else's `/rest/workflows`. The leftover `NodeVM` next to `JsTaskRunnerSandbox` is the useful part.

Wrong turns already recorded: Function type rejected as hidden/unknown (then the node is actually excluded); Function pid equal to the task-runner child (then it already moved); Code v2 seeing `process` (then the runner is not the control); a vm2 exploit gadget, a reverse shell, OS LPE. Theatre. The witness is main PID vs `hasProcess: false`.

03 Palette hide, then NodeVM

JAVASCRIPT Copy

```
export class Function implements INodeType {
    description: INodeTypeDescription = {
        displayName: 'Function',
        name: 'function',
        hidden: true,
```

Default exclude:

```
JAVASCRIPT
@Env('NODES_EXCLUDE')
exclude: JsonStringArray = ['n8n-nodes-base.executeCommand', 'n8n-nodes-base.loca
```

Then the leftover sandbox:

```
JAVASCRIPT
const vm = new NodeVM(options);
// ...
const functionCode = this.getNodeParameter('functionCode', 0) as string;
items = await vm.run(`module.exports = async function() {${functionCode}\n}()`, _
```

Sibling `Code.node.ts` already does this:

```
JAVASCRIPT
const sandbox = new JsTaskRunnerSandbox(workflowMode, this);
```

Task-runner spawn (`task-runner-process-js.ts`):

```
JAVASCRIPT
const flags = this.runnerConfig.insecureMode
    ? []
    : ['--disallow-code-generation-from-strings', '--disable-proto=delete'];
```

Those flags are on the child. They are not on the process that still runs Function.

04 What an attacker can do

As a default member, import or create a workflow whose node `type` is `n8n-nodes-base.function` (or `functionItem`, or `@n8n/n8n-nodes-langchain.code`) and

execute it. Palette hide does not stop that. The JS runs **in the n8n main process**.

That is host RCE as the n8n UID: instance secrets, other users' credential ciphertext on disk, the DB. Multi-tenant member to instance takeover. Not OS LPE. Not unauthenticated. Code v2 on the same instance does not see `process` that way.

05 The lab (run this at home)

Source of truth is `lab/`. Image `n8nio/n8n:2.42.0`. Port **18201**. Bind it to loopback. Default `NODES_EXCLUDE`. Attacker is a member.

DOCKERFILE Copy

```dockerfile
# Loopback lab image pin for n8n-function-vm2. Full stack: docker-compose.yml
# Fallback registry: docker.n8n.io/n8nio/n8n:2.42.0
FROM n8nio/n8n:2.42.0
```

`docker-compose.yml`:

YAML Copy

```yaml
name: n8n-function-vm2

services:
  n8n:
    build:
      context: .
      dockerfile: Dockerfile
    image: n8n-function-vm2:2.42.0
    ports:
      - "127.0.0.1:18201:5678"
    environment:
      N8N_HOST: "127.0.0.1"
      N8N_PORT: "5678"
      N8N_PROTOCOL: "http"
      N8N_LISTEN_ADDRESS: "0.0.0.0"
      N8N_SECURE_COOKIE: "false"
      WEBHOOK_URL: "http://127.0.0.1:18201/"
```

```
      N8N_ENCRYPTION_KEY: "lab-n8n-function-vm2-key"
      N8N_DIAGNOSTICS_ENABLED: "false"
      N8N_PERSONALIZATION_ENABLED: "false"
      N8N_HIRING_BANNER_ENABLED: "false"
      N8N_VERSION_NOTIFICATIONS_ENABLED: "false"
      N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS: "false"
      GENERIC_TIMEZONE: "UTC"
      TZ: "UTC"
```

PLAIN TEXT Copy

```
git clone https://github.com/abraxas/n8n-function-vm2
cd n8n-function-vm2/lab
./run.sh
```

The proof of concept is written for `127.0.0.1:18201`. Do not publish the port off loopback.

06 What the tree actually consumes

Member session. Create Function workflow. Run it. Create Code v2 workflow with the same idea. Compare PIDs.

A few other ways to lose without learning anything:

- Function type rejected as hidden/unknown. Then the node is actually excluded. Lab requires it to load.
- Function pid equal to the task-runner child. Then it already moved. Lab requires Function pid = n8n main.
- Code v2 seeing `process`. Then the runner is not the control.
- A vm2 exploit gadget, a reverse shell, OS LPE. Theatre. The witness is main

PID vs `hasProcess: false`.

Last lab run, trimmed:

PLAIN TEXT Copy

```
authenticated-as=member
function-fields pid=7 witness=N8N-FUNCTION-VM2-WITNESS
code-fields hasProcess=False pid=None
compare function_pid=7 code_pid=None main_pid=7 runner_pid=26 who=member
SUCCESS N8N-FUNCTION-VM2 who=member function_pid=7 code_pid=None main_pid=7 runne
```

The client that produced it is on GitHub. I am not reprinting session tokens.

07 What this is not

It is not "GHSA-j4p8 never shipped." Code v2 is on the runner. Function / FunctionItem / LangChain Code are not. It is not unauthenticated. It is not OS LPE. It is not a published vm2 CVE of its own; it is leftover in-process `NodeVM` after the Code-node move.

08 The fix

Add `n8n-nodes-base.function`, `n8n-nodes-base.functionItem`, and `@n8n/n8n-nodes-langchain.code` to default `NODES_EXCLUDE`, or route them through `JsTaskRunnerSandbox`. Palette `hidden` is not enough. Re-run the loopback client against a patched build: Function must fail the same class of way an excluded node does, or run in the task-runner child the way Code v2 already does.

I am not going to print a workflow JSON you can paste at someone else's `/rest/workflows`. The leftover `NodeVM` is the useful part. If you own the box, run the proof of concept against loopback.

Same product: Databricks path join.

09 References

- Proof of concept: abraxas/n8n-function-vm2 · n8n-function-vm2-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-94 · CWE-269
- Tree: n8n 2.42.0 · `Function.node.ts` · `Code.node.ts` · `nodes.config.ts` · `task-runner-process-js.ts`
- Nearby patched: GHSA-j4p8-h8mh-rh8q (Code node helpers FS; this leftover is Function)
- SECURITY.md
- Product: n8n

Abraxas Labs

Independent security research by abraxas.

Analysis of 0-days, n-days, malware, and the

leftovers that still matter.

GitHub **@abraxas_null** **abraxas.null@proton.me** **abraxaslabs.tech**

Home Research Open Source Reading Room About GitHub @abraxas_null Console

---

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs