Concealed executable blocks in MATLAB/Simulink

SEP 1, 2026 · HIGH · CVSS 4.0: 8.4 · CVE-2026-105043 · CWE-451 · FIXED IN R2026B

High-severity vulnerability

A Simulink `.slx` model can carry concealed blocks that run arbitrary shell commands on your computer. The blocks are hidden so completely that the Simulink editor will not show them, no matter how you scroll or zoom.

The model can be made to look perfectly ordinary. Opening it and pressing *Run* executes attacker-controlled shell commands with your privileges. Verified on MATLAB/Simulink R2026a Update 2 and earlier.

MathWorks agrees that such blocks "reduce the visibility of executable model content during manual inspection" and that "additional safeguards are appropriate."

Update: R2026b, released 2026-09-24, fixes the issue: every block now has a minimum size of 5×5 pixels, so concealed blocks become visible and fit-to-view frames them. A backport to R2026a is targeted for January 2027. Details below.

SEVERITY

CVSS 4.0: 8.4 (High), `CVSS:4.0/AV:L/AC:L/AT:N/PR:N/UI:A/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N`

METRIC	VALUE	WHY
Attack Vector / User Interaction	Local / Active	The victim opens a crafted `.slx` file locally and deliberately runs or compiles the model.
Attack Complexity / Requirements	Low / None	The crafted model works the same way every time, needs no knowledge of the target, and works on every release before R2026b. The only precondition is the user's action, which User Interaction already covers.
Confidentiality / Integrity	High / High	The concealed blocks run arbitrary shell commands with the victim's privileges, giving full read and write access to everything the user can reach.
Availability	High	Shell commands can exhaust resources, fill disks, delete files recursively and kill processes.
Subsequent system	None	The effects stay within the user's own authority.

PICTURE THIS

A friend sends you `HelloWorld.py` and asks for help. You open it in your IDE, say VS Code or PyCharm, and see three harmless lines:

WHAT YOU SEE IN YOUR IDE

Pause

HelloWorld.py

```
1 def greet(name):
2     return f"Hello, {name}!"
3 print(greet("world"))

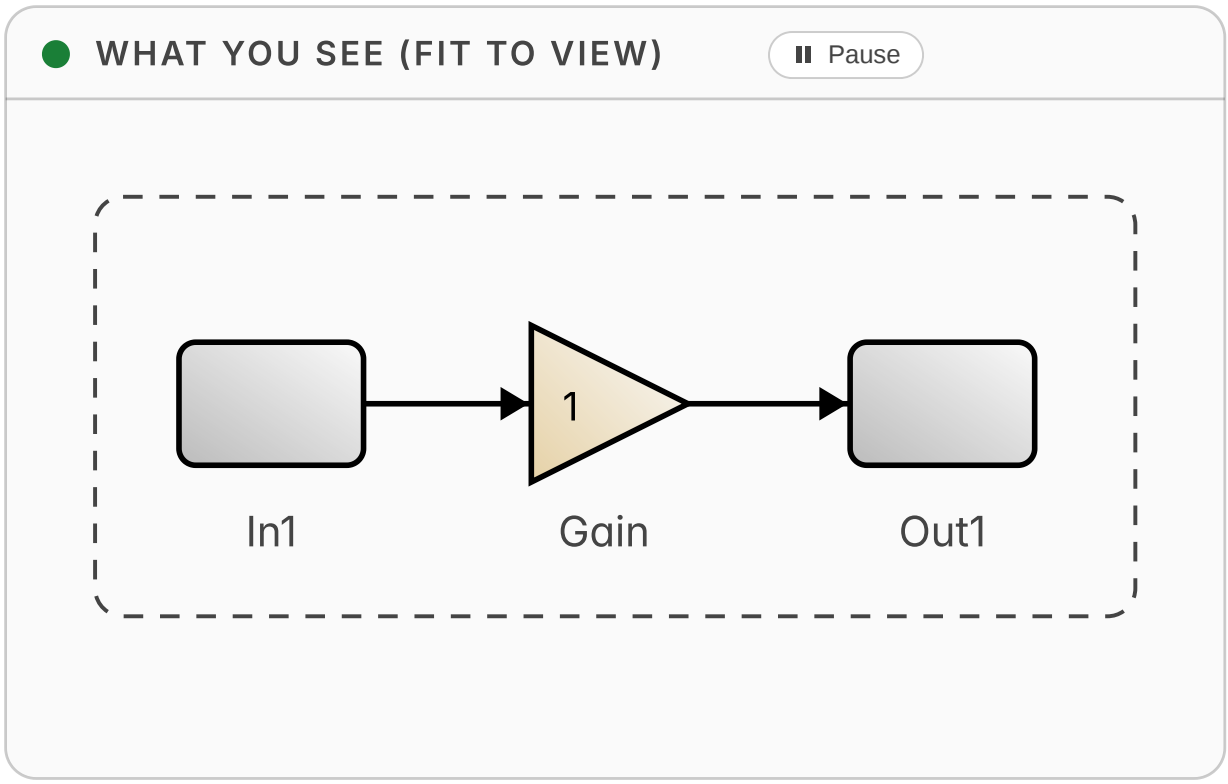

  end of file — nothing more to scroll to
```

A source-code analogy: what you see versus what actually runs. The harmful code lives far below the visible file, concealed and unselectable.

NOW IN SIMULINK

A Simulink model is a diagram of connected blocks. You read it the way you would read any diagram: you look at the canvas, scroll, and zoom. What you see is what the model does ... or so Simulink leads you to believe.

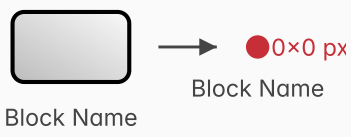
The figure starts with everything Simulink shows you: an input, a gain, an output. It then zooms out to reveal the same model's concealed passenger.



The model, fit to view, is just In1 → Gain → Out1. Zoom out and a concealed 0×0 block sits far off-canvas. It executes shell commands when you run the model.

CONCEALING A BLOCK

The concealed block is an ordinary MATLAB Function block, hidden by combining three fully supported properties. Each is harmless on its own. Together, they make the block impossible to draw.



Zero area

`Position [20 20 20 20]` collapses the block to zero pixels wide and zero tall: there is nothing to draw on the canvas.



Name suppressed

`ShowName off` removes the block's name label, which would otherwise be drawn on the canvas.



Off-canvas

`Position [10000 10000 10000 10000]` places the block far from the rest of the model, outside the area the editor can ever pan or zoom to.

IMPACT

A MATLAB Function block can call any MATLAB function, which already gives it access to files and the network. Through `coder.extrinsic('system')`, it can also run arbitrary shell commands.

The attack requires no more than opening a third-party Simulink model and running or compiling it. The shell commands then execute with the victim's privileges. Severity: **CVSS 4.0 8.4 (High)**, `CVSS:4.0/AV:L/AC:L/AT:N/PR:N/UI:A/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N`. The full range of impact:

- **Code execution** — downloading and running a further payload.
- **Confidentiality** — directory listing and data exfiltration.
- **Integrity** — overwriting or deleting files.
- **Availability** — resource exhaustion and infinite loops.

Simulink models are routinely shared: shipped by suppliers, exchanged at work, downloaded from public repositories, attached to publications. Every one of those is a delivery channel.

WHY INSPECTION FAILS

The checks engineers normally rely on all fail silently. Detection is possible, but only through methods most people never use.

- ✗ **Fit-to-view** — should frame every block but frames only what has area; a 0×0 block gives it nothing to include, so you believe the canvas shows every block.
- ✗ **Zoom and pan** — the concealed block is too far off-canvas to reach, and even at the canvas limits nothing appears.
- ✗ **Rubber-band drag** — the block lies beyond any region you can drag a selection over, and even a successful selection would give no visible feedback.
- ✗ **Select-All (Ctrl+A)** — selects the hidden block, but produces no visible feedback anywhere.
- ✓ **find_system()** — enumerates every block programmatically, hidden or not.

DETECTION AND MITIGATION

Until a fix ships, treat every third-party model as untrusted code. Before running or compiling a model:

```
% Enumerate every block, including hidden ones, without trusting the canvas
load_system('suspect.slx');
find_system('suspect', 'LookUnderMasks', 'all', 'FollowLinks', 'on', 'IncludeCo
```

- **Flag dimensionless blocks** whose four `Position` coordinates are equal, or that sit far from other blocks.
- **Scrutinize blocks that call `coder.extrinsic('system')`.** Such a function can reach the shell and should not be trusted in a third-party model.
- **Prefer a sandbox.** Open models on a throwaway machine or container with no network and no sensitive files.

SUGGESTED FIXES

For MathWorks, the fixes are straightforward and layered:

- **Enforce a minimum block geometry** (at least 1×1 pixel) so zero-area blocks cannot slip past rendering and selection.
- **Add edit-time / Model Advisor checks** for zero- or near-zero-area blocks, blocks placed far outside the canvas bounds, blocks fully covered by others, and MATLAB Function blocks that declare extrinsic shell functionality.
- **Provide a hidden-block inventory** so that “show me everything in this model” through the GUI is actually complete.

‘BUT MODELS ARE CODE’ — WHY THAT MISSES THE POINT

The obvious objection: “executing an untrusted model runs untrusted code.” Of course it does. Executable blocks are a known, documented capability, and a concealed block on its own is harmless. This report does not claim that models can execute code.

The broken promise is narrower: what Simulink’s inspection tools show must match what the model runs. Only then can users see what they are about to run and choose knowingly. A hidden payload takes that informed choice away.

Two separate properties are at stake here. “Untrusted code requires trust in the author” is one. “The tool must not misreport what will run” is the other. MathWorks’ documentation covers only the first. A user only accepts the risk of what a model visibly contains.

The objection also implies that a careful user would have found the block. However, Simulink presents the canvas as the complete rendering of a model, and every affordance it offers for answering “what is in this model” returns the same answer for a model with a concealed block as for one without. The user is not failing to look carefully. The product answers their question incorrectly.

This is what CWE-451 calls “visual truncation”:

> Visual truncation: important information could be truncated from the display [...] or place the potentially-dangerous indicator outside of the user’s field of view
>
> — CWE-451, MITRE

MathWorks’ own response concedes exactly this. In its words, zero-area blocks “can reduce the visibility of executable model content during manual inspection,” and “additional safeguards are appropriate.”

TIMELINE

DATE	EVENT
2026-06-03	Reported to MathWorks with written advisory and defanged proof-of-concept.
2026-07-29	MITRE CVE identifier requested (CAN-2026-2034772).
2026-09-01	Public disclosure, following the pre-stated coordinated-disclosure deadline. MathWorks asked for an extension, but no agreement was reached.
2026-09-24	MathWorks releases R2026b, which enforces a 5×5-pixel minimum block size and fixes the issue.
2026-10-02	MITRE publishes CVE-2026-105043.

UPDATE: FIXED IN R2026B

MathWorks' R2026b release fixes the issue completely. The Simulink release notes list it as a behavior change:

> Blocks have a minimum size of 5-by-5 pixels
>
> All blocks in a model must have a size of at least 5-by-5 pixels. Simulink adjusts block coordinates so that the width and height are at least 5 pixels in these situations:
>
> - The coordinates you specify for the Position parameter result in a width or height smaller than 5 pixels.
> - You open or load a model created in a previous release that contains one or more blocks with smaller dimensions.
>
> — MathWorks, Simulink release notes, R2026b

This closes both halves of the problem. A block specified as 0×0 is now drawn at 5×5 pixels, so it is no longer invisible. And because every block now has area, fit-to-view frames all of them, so the canvas once again shows the complete model.

Users of R2026a and earlier remain affected until they upgrade or the announced backport ships, and should keep following the detection and mitigation steps above.

CREDITS

Found and reported by Alexander Boll.