CVE-2021-32629 Detail

MODIFIED

This CVE record has been updated after NVD enrichment efforts were completed. Enrichment data supplied by the NVD may require amendment due to these changes.

Current Description

Cranelift is an open-source code generator maintained by Bytecode Alliance. It translates a target-independent intermediate representation into executable machine code. There is a bug in 0.73 of the Cranelift x64 backend that can create a scenario that could result in a potential sandbox escape in a Wasm program. This bug was introduced in the new backend on 2020-09-08 and first included in a release on 2020-09-30, but the new backend was not the default prior to 0.73. The recently-released version 0.73 with default settings, and prior versions with an explicit build flag to select the new backend, are vulnerable. The bug in question performs a sign-extend instead of a zero-extend on a value loaded from the stack, under a specific set of circumstances. If those circumstances occur, the bug could allow access to memory addresses upto 2GiB before the start of the Wasm program heap. If the heap bound is larger than 2GiB, then it would be possible to read memory from a computable range dependent on the size of the heaps bound. The impact of this bug is highly dependent on heap implementation, specifically: * if the heap has bounds checks, and * does not rely exclusively on guard pages, and * the heap bound is 2GiB or smaller * then this bug cannot be used to reach memory from another Wasm program heap. The impact of the vulnerability is mitigated if there is no memory mapped in the range accessible using this bug, for example, if there is a 2 GiB guard region before the Wasm program heap. The bug in question performs a sign-extend instead of a zero-extend on a value loaded from the stack, when the register allocator reloads a spilled integer value narrower than 64 bits. This interacts poorly with another optimization: the instruction selector elides a 32-to-64-bit zero-extend operator when we know that an instruction producing a 32-bit value actually zeros the upper 32 bits of its destination register. Hence, we rely on these zeroed bits, but the type of the value is still i32, and the spill/reload reconstitutes those bits as the sign extension of the i32's MSB. The issue would thus occur when: * An i32 value in a Wasm program is greater than or equal to 0x8000_0000; * The value is spilled and reloaded by the register allocator due to high register pressure in the program between the value's definition and its use; * The value is

QUICK INFO

CVE Dictionary Entry:
CVE-2021-32629
NVD Published Date:
05/24/2021
NVD Last Modified:
11/21/2024
Source:
GitHub, Inc.

Cranelift (via its AOT compiler) had a bug where it performed a sign-extend instead of a zero-extend when loading a value from the stack. This allowed access to memory addresses up to 2 GiB before the start of the Wasm heap.

4 specific preconditions that had to be met:
The i32 value was greater than or equal to 0x8000_0000.
The value was spilled and reloaded due to high register pressure.
The value was produced by an instruction that zeroes the upper 32 bits (like add, sub, mul, and, or).
The value was then zero-extended to 64 bits in the Wasm program.

It allowed the program to access memory up to 2 GiB before the Wasm heap if the zero-extend turned into a sign-extend so we have a sandbox escape. Even though it affected Cranelift, the logic is similar to WAMR because both perform AOT compilation and implement sandboxing.

Cranelift is An open-source code generator maintained by the Bytecode Alliance.
It compiles intermediate representations into executable machine code.
It's commonly used in WebAssembly runtimes (like Wasmtime, Lucet) to turn WebAssembly code into native CPU instructions.

produced by an instruction that we know to be “special” in that it zeroes the upper 32 bits of its destination: add, sub, mul, and, or; * The value is then zero-extended to 64 bits in the Wasm program; * The resulting 64-bit value is used. Under these circumstances there is a potential sandbox escape when the i32 value is a pointer. The usual code emitted for heap accesses zero-extends the Wasm heap address, adds it to a 64-bit heap base, and accesses the resulting address. If the zero-extend becomes a sign-extend, the program could reach backward and access memory up to 2GiB before the start of its heap. In addition to assessing the nature of the code generation bug in Cranelift, we have also determined that under specific circumstances, both Lucet and Wasmtime using this version of Cranelift may be exploitable. See referenced GitHub Advisory for more details.

+View Analysis Description

Metrics

CVSS Version 4.0	CVSS Version 3.x	CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NIST: NVD **Base Score:** 8.8 HIGH

Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

CNA: GitHub, Inc. **Base Score:** 7.2 HIGH

Vector: CVSS:3.1/AV:L/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:N

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource
https://crates.io/crates/cranelift-codegen	**Product** **Third Party Advisory**
https://crates.io/crates/cranelift-codegen	**Product** **Third Party Advisory**
https://github.com/bytecodealliance/wasmtime/commit/95559c01aaa7c061088a433040f31e8291fb09d0	**Patch** **Third Party Advisory**
https://github.com/bytecodealliance/wasmtime/commit/95559c01aaa7c061088a433040f31e8291fb09d0	**Patch**