☆ Starred by 2 users

Owner:	gdeepti@chromium.org
CC:	da...@davidmanouchehri.com adamk@chromium.org vahl@chromium.org ecmziegler@chromium.org gdeepti@chromium.org dschuff@chromium.org ecmziegler@google.com
Status:	Verified *(Closed)*
Components:	Blink>JavaScript>WebAssembly
Modified:	Jan 8, 2021
Backlog-Rank:	----
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	Linux
Pri:	1
Type:	Bug-Security

reward-2000
Reproducible
Stability-Memory-AddressSanitizer
ClusterFuzz
Security_Impact-Stable
Security_Severity-High
ReleaseBlock-Stable
allpublic
reward-inprocess
ClusterFuzz-Verified
CVE_description-submitted
M-81
Target-81
CVE-2020-6419
reward_to-david_at_davidmanouchehri.com
Release-0-M81

Issue 1040325: CHECK failure: *old_buffer != memory_object->array_buffer() in wasm-objects.cc

Reported by ClusterFuzz on Wed, Jan 8, 2020, 9:52 PM EST Project Member

Code

Detailed Report: https://clusterfuzz.com/testcase?key=5945746400542720

Fuzzer: ochang_js_fuzzer
Job Type: linux_asan_d8_dbg
Platform Id: linux

Crash Type: CHECK failure
Crash Address:
Crash State:
  *old_buffer != memory_object->array_buffer() in wasm-objects.cc
  v8::internal::WasmMemoryObject::Grow
  v8::WebAssemblyMemoryGrow

Sanitizer: address (ASAN)

Regressed: https://clusterfuzz.com/revisions?job=linux_asan_d8_dbg&range=65645:65646

Reproducer Testcase: https://clusterfuzz.com/download?testcase_id=5945746400542720

Issue filed automatically.

The reproduce tool requires a ClusterFuzz source checkout. To prepare one, run:

git clone https://github.com/google/clusterfuzz && cd clusterfuzz && git checkout tags/reproduce-tool-stable

To reproduce this issue, run:

./reproduce.sh -t https://clusterfuzz.com/testcase-detail/5945746400542720 -b /path/to/build

Please use the GN arguments provided in this report when building the binary. If you have any feedback on reproducing test cases, let us know at https://forms.gle/Yh3qCYFveHj6E5jz5 so we can improve.

Comment 1 by clemensb@chromium.org on Thu, Jan 9, 2020, 2:49 AM EST Project Member

Status: Assigned (was: Untriaged)
Owner: gdeepti@chromium.org
Cc: adamk@chromium.org
Labels: Pri-1
Components: -Blink>JavaScript Blink>JavaScript>WebAssembly

First error after staging atomics (growing shared memory).
Deepti, can you take a look?

Comment 2 by sheriffbot@chromium.org on Thu, Jan 9, 2020, 9:36 AM EST Project Member

Labels: Target-81 M-81

Setting milestone and target because of Security_Impact=Head and high severity.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 3 by sheriffbot@chromium.org on Thu, Jan 9, 2020, 10:02 AM EST Project Member

Labels: ReleaseBlock-Stable

This is a serious security regression. If you are not able to fix this quickly, please revert the change that introduced it.

If this doesn't affect a release branch, or has not been properly classified for severity, please update the Security_Impact or Security_Severity labels, and remove the ReleaseBlock label. To disable this altogether, apply ReleaseBlock-NA.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 4 by bugdroid on Mon, Jan 13, 2020, 8:35 PM EST Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/8d511cbd209e90448f3f9197b2ac49757cd32ca5

commit 8d511cbd209e90448f3f9197b2ac49757cd32ca5
Author: Deepti Gandluri <gdeepti@chromium.org>
Date: Tue Jan 14 01:35:06 2020

[wasm] Growing memory should always allocate a new JS buffer

The UpdateSharedWasmMemoryObjects function only creates a new
JSArrayBuffer when the the legths of old/new ArrayBuffer objects
are unequal, but the CHECK in the Grow() funciton assumes that a new
object is always created. Fix so that a new ArrayBuffer is always
allocated.

~~Bug: v8:10044~~, ~~chromium:1040325~~
Change-Id: I66912bdc091e65a57e5b50f4ed63b0da5492dcc4
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1999603
Reviewed-by: Ben Smith <binji@chromium.org>
Commit-Queue: Deepti Gandluri <gdeepti@chromium.org>
Cr-Commit-Position: refs/heads/master@{#65742}

[modify] https://crrev.com/8d511cbd209e90448f3f9197b2ac49757cd32ca5/src/objects/backing-store.cc
[modify] https://crrev.com/8d511cbd209e90448f3f9197b2ac49757cd32ca5/test/mjsunit/wasm/grow-shared-memory.js

Comment 5 by ClusterFuzz on Tue, Jan 14, 2020, 11:15 AM EST Project Member

Status: Verified (was: Assigned)
Labels: ClusterFuzz-Verified

ClusterFuzz testcase 5945746400542720 is verified as fixed in https://clusterfuzz.com/revisions?job=linux_asan_d8_dbg&range=65741:65742

If this is incorrect, please add the ClusterFuzz-Wrong label and re-open the issue.

Comment 6 by sheriffbot@chromium.org on Wed, Jan 15, 2020, 10:43 AM EST Project Member

Labels: -Restrict-View-SecurityTeam Restrict-View-SecurityNotify

Comment 7 by ellyj...@chromium.org on Wed, Jan 22, 2020, 4:45 PM EST Project Member

curious: Is this related to ~~issue 1010272~~? especially c35 on that bug

Comment 8 by gdeepti@chromium.org on Wed, Jan 22, 2020, 5:14 PM EST Project Member

Hi, yes it is - previously all the cases for growing by 0 were handled together so this behaved the same way for both shared/unshared memory. After a refactoring change, the shared memory case was split out, but we didn't test for grow(0), and shared memory separately. We now have a unit test for this specific case, and better fuzzer coverage so we catch cases like this earlier.

Comment 9 by adetaylor@chromium.org on Thu, Feb 13, 2020, 12:46 AM EST Project Member

Cc: da...@davidmanouchehri.com

Adding David Manouchehri who provided the test case in https://bugs.chromium.org/p/chromium/issues/detail?id=1010272#c35, as they've asked about this bug.

Comment 10 by adetaylor@chromium.org on Thu, Feb 13, 2020, 12:49 AM EST Project Member

Labels: reward-topanel

VRP panel, please see #c7 and #c8 which suggests that https://bugs.chromium.org/p/chromium/issues/detail?id=1010272#c35 was helpful in discovering this.

Comment 11 by da...@davidmanouchehri.com on Thu, Feb 13, 2020, 1:18 AM EST

Thanks for adding me to the ticket, cool to see that ClusterFuzz caught it. I was convinced this ticket was owned by glazunov. =P

I found this bug through variant analysis of https://bugs.chromium.org/p/chromium/issues/detail?id=776677 / CVE-2017-15399 if anyone is curious.

Exploitation of this one is more difficult than CVE-2017-15399 as you'd need to win the race between BroadcastSharedWasmMemoryGrow and the CHECK_NE.

```
int32_t WasmMemoryObject::Grow(Isolate* isolate,
                               Handle<WasmMemoryObject> memory_object,
                               uint32_t pages) {
...
  // Try to handle shared memory first.
  if (old_buffer->is_shared()) {
    if (FLAG_wasm_grow_shared_memory) {
      // Shared memories can only be grown in place; no copying.
      if (backing_store->GrowWasmMemoryInPlace(isolate, pages, maximum_pages)) {
        BackingStore::BroadcastSharedWasmMemoryGrow(isolate, backing_store,
                                                    new_pages);
        // <----------------------------------- Must win a race before this line
        CHECK_NE(*old_buffer, memory_object->array_buffer());
...
      }
    }
    return -1;
  }
...
}
```

I didn't submit a report as I wasn't able to provide a PoC that could reliably win such a race. In hindsight I should have committed my test case and sent it off to Gerrit, which would have helped spot and fix this much soon. Lesson learned!

Comment 12 by natashapabrai@google.com on Wed, Feb 19, 2020, 7:00 PM EST Project Member

Labels: -reward-topanel reward-0

Unfortunately the Panel declined to reward this report as it was found by another fuzzer.

Comment 13 by da...@davidmanouchehri.com on Wed, Feb 19, 2020, 7:10 PM EST

No worries. To clarify/confirm, when was it found by another fuzzer? My test case was provided on Nov 11, 2019, which was much earlier than ochang_js_fuzzer (Jan 8, 2020 according to this ticket).

Comment 14 by adetaylor@chromium.org on Tue, Feb 25, 2020, 12:27 AM EST Project Member

Labels: CVE-2020-6419 CVE_description-missing

Allocating CVE because the first mention of this was external in https://bugs.chromium.org/p/chromium/issues/detail?id=1010272#c35, AIUI.

Comment 15 by adetaylor@google.com on Wed, Feb 26, 2020, 6:45 PM EST Project Member

Labels: reward_to-david_at_davidmanouchehri.com

Comment 16 by natashapabrai@google.com on Wed, Feb 26, 2020, 7:23 PM EST Project Member

Labels: -reward-0 reward-2000 reward-unpaid

Congrats! The Panel re-visited this report and decided to award $2,000! Nice one!

Comment 17 by da...@davidmanouchehri.com on Wed, Feb 26, 2020, 9:24 PM EST

Thanks, that was quite a genuine gesture; you folks are all awesome!

I promise this will be my last and only poorly reported security bug. =P

Comment 18 by natashapabrai@google.com on Tue, Mar 3, 2020, 11:42 AM EST Project Member

Labels: -reward-unpaid reward-inprocess

Comment 19 by sheriffbot on Tue, Apr 21, 2020, 2:54 PM EDT Project Member

Labels: -Restrict-View-SecurityNotify allpublic

This bug has been closed for more than 14 weeks. Removing security view restrictions.

For more details visit https://www.chromium.org/issue-tracking/autotriage - Your friendly Sheriffbot

Comment 20 by adetaylor@google.com on Mon, Jun 1, 2020, 5:26 PM EDT Project Member

Labels: -Security_Impact-Head relnotes_update_needed Release-0-M81 Security_Impact-Stable

Hmmm. I'm trying to work out if I should have allocated a CVE here (I'll need to submit details to MITRE, but I can only allocate one if it affected a shipping product i.e. stable).

As far as I can tell, here's the timeline.
1. this bug was introduced prior to November but was only triggered using the --experimental-wasm-threads flag
2. that flag was sometimes enabled on desktop Chrome (according to https://chromium.googlesource.com/v8/v8/+log/7d420621887c9ceaef827db99ef2e627bc023d22..6e2e31e5fb21085e4f041d952e023b308a61e90a?pretty=fuller&n=10000, both the commit comment and code comments)
3. that commit (which is the regression range for this bug) enabled the flag by default, which is what caused the fuzzer to find it
4. the fix was 8d511cbd209e90448f3f9197b2ac49757cd32ca5 which went into M81 initial release.

As such I believe that Security_Impact-Head is effectively wrong, and this did impact some production stable configurations. Therefore it does deserve a CVE, as well as a mention in the M81 release notes, which I will edit in due course. Adjusting labels to that effect.

Comment 21 by adetaylor@chromium.org on Wed, Jun 3, 2020, 7:11 PM EDT Project Member

Labels: -CVE_description-missing CVE_description-submitted

Comment 22 by adetaylor@google.com on Fri, Jan 8, 2021, 5:43 PM EST Project Member

Labels: -relnotes_update_needed