Starred by 7 users

Owner:	neis@chromium.org
CC:	adetaylor@chromium.org nicohartmann@chromium.org pbomm...@chromium.org mslekova@chromium.org vahl@chromium.org ecmziegler@google.com
Status:	Fixed *(Closed)*
Components:	Blink>JavaScript>Compiler
Modified:	Nov 9, 2021
Backlog-Rank:	----
Editors:	----
EstimatedDays:	----
NextAction:	----
OS:	Linux, Android, Windows, Chrome, Mac, Fuchsia, Lacros
Pri:	1
Type:	Bug-Security

Hotlist-Merge-Review
Security_Impact-Stable
Security_Severity-High
allpublic
reward-inprocess
CVE_description-submitted
M-92
Target-92
external_security_report
merge-merged-9.0
FoundIn-92
LTS-Merged-90
LTS-Security-90
merge-merged-9.2
LTS-Size-Small
LTS-Complexity-Trivial
merge-merged-9.3
reward-21000
Release-2-M92
CVE-2021-30598

Issue 1234764: v8/Turbofan: Invalid rotate-right optimization + Typer hardening bypass

Reported by manfp...@gmail.com on Fri, Jul 30, 2021, 9:43 AM EDT

Code

Report description

v8/Turbofan: Invalid rotate-right optimization + Typer hardening bypass

---

Bug location

Which product or website have you found a vulnerability in?

Google Chrome

The problem

Please describe the technical details of the vulnerability

Tested on:
==========

- Chromium 92.0.4515.107 Arch Linux (other platforms affected too)
- v8/d8 on current main branch on Linux

Test case:
==========

Execute the following snippet in Chrome's Javascript console or d8:

```
function foo(a,b) {
    a = a|0;
    b = b|0;
    return (a >>> b) ^ (a << (32-b));
}

console.log(foo(1, 0));
```

```
for (var i = 0; i < 3e5; i++) foo(1, 0);
console.log(foo(1, 0));
```

The expected behavior would be to print "0" twice; instead "0" and "1" are printed.
I will show below how this can be used to construct mis-typed values, and achieve arbitrary code execution in the context of v8/Chrome's Renderer process using a typer hardening bypass.

Root Cause:
===========

The relevant optimization is `MachineOperatorReducer::TryMatchWord32Ror()`, around line 1838 of v8's `src/compiler/machine-operator-reducer.cc`:

```
Reduction MachineOperatorReducer::TryMatchWord32Ror(Node* node) {
  DCHECK(IrOpcode::kWord32Or == node->opcode() ||
         IrOpcode::kWord32Xor == node->opcode());
  Int32BinopMatcher m(node);
  Node* shl = nullptr;
  Node* shr = nullptr;
  // Recognize rotation, we are matching:
  //  * x << y | x >>> (32 - y) => x ror (32 - y), i.e  x rol y
  //  * x << (32 - y) | x >>> y => x ror y
  //  * x << y ^ x >>> (32 - y) => x ror (32 - y), i.e. x rol y
  //  * x << (32 - y) ^ x >>> y => x ror y
  // as well as their commuted form.
  ...
}
```

Consider the last two reductions with the case `y = 0` (or more generally, `y` divisible by 32).
In this case, both left and right shift are by an amount divisible by 32; but as shift amounts are truncated to their last 5 bits this is equivalent to no shift occuring at all, so the left-hand side actually reduces to `x ^ x = 0`.
However, the right-hand side reduces to `x ror 0 = x` (again, rotation amounts divisible by 32 are equivalent to 0), which is wrong for non-zero values of `x`.

Fix:
====

The easiest fix would be to disallow the reduction for the Word32Xor operator.
However, the optimization can still be performed for the case where `y` and `32 - y` are constants and known to be non-zero (modulo 32).
It's of course enough to check this for `y`.
A suggested patch is attached as `machine-operator-reducer.cc.patch`.
Optimizing the case of y being constant 0 to `0` does not seem to make sense, as this should already be handled by the existing reductions for shifts and xor.

Note also that the Word32Or-case is unaffected by the bug, as `x << 0 | x >>> 32` is equal to `x | x = x`, which matches `ror(x, 0)`.

Exploiting the bug:
===================

The rest of this write-up will describe how exactly the bug was triggered in a way that causes a type confusion, and how this type confusion can be leveraged into arbitrary code exeuction using a new typer-hardening bypass.
A full exploit that executes a customizable shellcode (in the example, for Linux x64) is attached as `ror_rce.js`.
It can either be run with a current version of `d8` or embedded into a html file openened with `chrome --no-sandbox` (however the example shellcode has no observable effect when run in Chrome's renderer; there doesn't seem to be an easy portable way to pop a calc on Linux :( ).

To trigger the type confusion in the first place, some further primitives are needed that are now described.

Typer-opaque constants

It is useful to have values that are unknown to the Typer-phase, but that can be fully reduced to constants during later phases like EarlyOptimization, where `MachineOperatorReducer` is first run.
This can be used to introduce uncertainty into types where needed to prevent e.g. constant folding.
This primitive can be obtained using the LoadElimination-phase: We use a local object `o` to hold a constant `c0` equal to zero; whenever we load this constant the typer can speculate that this is a number, but cannot reason about its exact value.

During LoadElimination, this is replaced by a constant `0` node, enabling further optimizations.
Note: For unknown reasons, the LoadElimination only seems to work properly when causing the function's optimization with a loop instead of the "%PrepareFunctionForOptimization"/"%OptimizeFunctionOnNextCall" intrinsics.

This primitive can also be used to introduce partially-known types: For example, `(o.c0 & 1)` can be typed to `Range(0, 1)` (and is later constant-folded into `0`).

Typer-transparent variables

Sometimes, the opposite construct is useful: a value whose type is fully known to the typer, but that isn't constant-folded away.
This can be used to prevent unwanted optimizations, while maintaining type information that's as precise as possible.
To do this, speculative conversions can be abused to make the typer assume a certain branch being taken, which results in a constant type (this is not a bug in itself, as the speculative assumptions are appropriately guarded by checks).
The following construct seems to work well:

```
    let c0a = arg_true ? 0 : "x";
    let c0 = Math.max(c0a, 0) + c0a;
```

where `arg_true` is a function argument that is always set to `true` (giving `false` correctly leads to deoptimization).
It somehow results in `c0` having Type `Range(0, 0)` without being constant-folded away.
I did not investigate the exact reason for this construct having the desired effect.

Breaking the Typer

There are still some obstacles in constructing a mis-typed value.
First, let's look at `TryMatchWord32Ror` in some more detail:

```
  // Recognize rotation, we are matching:
  //  * x << y | x >>> (32 - y) => x ror (32 - y), i.e  x rol y
  //  * x << (32 - y) | x >>> y => x ror y
  //  * x << y ^ x >>> (32 - y) => x ror (32 - y), i.e. x rol y
  //  * x << (32 - y) ^ x >>> y => x ror y
```

```
  // as well as their commuted form.

  ...

  if (mshl.left().node() != mshr.left().node()) return NoChange();

  if (mshl.right().HasResolvedValue() && mshr.right().HasResolvedValue()) {
    // Case where y is a constant.
    if (mshl.right().ResolvedValue() + mshr.right().ResolvedValue() != 32)
      return NoChange();
  } else {
    Node* sub = nullptr;
    Node* y = nullptr;
    if (mshl.right().IsInt32Sub()) {
      sub = mshl.right().node();
      y = mshr.right().node();
    } else if (mshr.right().IsInt32Sub()) {
      sub = mshr.right().node();
      y = mshl.right().node();
    } else {
      return NoChange();
    }

    Int32BinopMatcher msub(sub);
    if (!msub.left().Is(32) || msub.right().node() != y) return NoChange();
  }

  node->ReplaceInput(0, mshl.left().node());
  node->ReplaceInput(1, mshr.right().node());
  NodeProperties::ChangeOp(node, machine()->Word32Ror());
  return Changed(node);
```

It at first seems easiest to use the first case where both shift amounts are known constants, but there is a problem:
If either of the shift amonuts is a zero constant, the shift will be completely optimized away.
We could instead use shift amounts like -32 and 64; however having both shift amounts outside the interval `[0, 31]` will unfortunately in the end lead to an unconstrained output type due to internals of the typer described below.
Instead, we will use the described technique of "typer-transparent variables" and have one shift amount be equal to the constructed `c0` (which has type `Range(0,0)`), and the other one equal to `32-c0` (which has type `Range(32,32)`).
Those expressions precisely match the second case, while also giving fully transparent information about their values to the typer, but also not enabling optimization of `x >> c0` to `x`.

Now, let's look at the relevant pieces of `operation-typer.cc` that are responsible for giving our initial expression its type:

```
Type OperationTyper::NumberBitwiseXor(Type lhs, Type rhs) {
  DCHECK(lhs.Is(Type::Number()));
  DCHECK(rhs.Is(Type::Number()));

  lhs = NumberToInt32(lhs);
  rhs = NumberToInt32(rhs);

  if (lhs.IsNone() || rhs.IsNone()) return Type::None();

  double lmin = lhs.Min();
  double rmin = rhs.Min();
  double lmax = lhs.Max();
  double rmax = rhs.Max();
  if ((lmin >= 0 && rmin >= 0) || (lmax < 0 && rmax < 0)) {
    // Xor-ing negative or non-negative values results in a non-negative value.
    return Type::Unsigned31();
  }
  if ((lmax < 0 && rmin >= 0) || (lmin >= 0 && rmax < 0)) {
    // Xor-ing a negative and a non-negative value results in a negative value.
    // TODO(jarin) Use a range here.
    return Type::Negative32();
  }
  return Type::Signed32();
}
```

Unfortunately, this only computes sign information, so we need inputs where the optimization results in a value with the wrong sign bit.
Here is the typer for left shifts (the one for logical right-shifts is quite similar, except for the overflow-check):

```
Type OperationTyper::NumberShiftLeft(Type lhs, Type rhs) {
  ...

  lhs = NumberToInt32(lhs);
  rhs = NumberToUint32(rhs);

  ...

  if (max_rhs > 31) {
    // rhs can be larger than the bitmask
    max_rhs = 31;
    min_rhs = 0;
  }

  if (max_lhs > (kMaxInt >> max_rhs) || min_lhs < (kMinInt >> max_rhs)) {
    // overflow possible
    return Type::Signed32();
  }

  double min =
      std::min(static_cast<int32_t>(static_cast<uint32_t>(min_lhs) << min_rhs),
               static_cast<int32_t>(static_cast<uint32_t>(min_lhs) << max_rhs));
  double max =
      std::max(static_cast<int32_t>(static_cast<uint32_t>(max_lhs) << min_rhs),
               static_cast<int32_t>(static_cast<uint32_t>(max_lhs) << max_rhs));

  if (max == kMaxInt && min == kMinInt) return Type::Signed32();
  return Type::Range(min, max, zone());
```

}
```

Problematic is the check `max_rhs > 31`: If the shift amounts have to sum to 32, one of them has to lie outside the range `[0, 31]`, which makes it completely unknown to the typer.
But there is one case that still works: consider `x=-1` and the expression `(x>>>0) ^ (x<<32)`.
The typer can easily reason the left side to be equal to `-1` and thus negative; and for the right side it considers all possible shifts from `x<<31` to `x<<0`, which are still all negative.
Thus, both sides get typed to a range of negative 32-bit integers; which results in the xor-Result having type `Unsigned31`.
However, after the faulty optimization the result is actually `ror(-1, 0) = -1`, which is negative, thus breaking the typer's range-tracking.

There are two additional details:
First, because the logical right-shift works on unsigned 32-bit values, we have to supply the value `2**32-1` instead of `-1`, or else the typer's `NumberToUint32` operation couldn't reason about the result of the Uint32-truncation.
Conversely, we have to subtract `2**32` from the result (which has type `Range(2**32-1, 2**32-1)` at this point) to actually bring it in the negative range (in the view of the typer).
This is not actually a problem: After the SimplifiedLowering-stage decides that all relevant values can be truncated to 32-bit words the constants `2**32-1` and `-1` get merged (thus passing the check of `TryMatchWord32Ror` that the left-hand sides of the shifts are the same node), and the subtraction gets truncated to a `Word32Sub(..., 0)`, which gets optimized away.

Second, this new subtraction has the problem of getting constant-folded away; to prevent this, we can simply use the technique of "typer-opaque constants"; instead of subtracting `2**32` we subtract `2**32 + (o.c0&1)` which has type `Range(2**32, 2**32+1)`, but is known to be `2**32` after SimplifiedLowering.
This just widens the left-hand side's type to `Range(-2, -1)`, still maintaining the information that it is negative.

To make that wrongly-typed value a bit nicer we finally (arithmetically) right-shift by `31`, resulting in a type of `Range(0,0)` but a real value of `-1`.

The full PoC for breaking the typer is:

```
function foo(arg_true) {
    let o = {c0: 0};
    let c0a = arg_true ? 0 : "x";
    let c0 = (Math.max(c0a, 0) + c0a);
    let v01 = 2**32 + (o.c0 & 1);
    let ra = ((2**32-1) >>> c0) - v01;
    let rb = ((-1) << (32-c0));
    return (ra^rb) >> 31;
}

for (var i = 0; i < 3e4; i++) foo(true);
console.log(foo(true));
```

The resulting output is `-1`, while the the type of the final `SpeculativeNumberShiftRight` node is `Range(0, 0)`.

Typer hardening bypass

I additionally found a new typer hardening bypass to escalate this bug into arbitrary code execution.
Array accesses should all be guarded by appropriate `CheckBounds` nodes to prevent out-of-bound-accesses in case of typer bugs.
However, this is not the case when accesssing an array via an iterator.
Consider `JSCallReducer::ReduceArrayIteratorPrototypeNext` in `src/compiler/js-call-reducer.cc`; the only two index-checks performed are the following (around line 6250):
```

  Node* check = graph()->NewNode(simplified()->NumberLessThan(), index, length);

  ...

    index = etrue = graph()->NewNode(
        common()->TypeGuard(
            Type::Range(0.0, length_access.type.Max() - 1.0, graph()->zone())),
        index, etrue, if_true);
```

However, this is not enough: Due to the LoadElimination stage, the length of the array can actually be propagated directly from the value we give into the array constructor, which means that type information is also propagated.
This can result in the elimination of the `NumberLessThan()`-check, leading to an out-of-bounds-access.

Thus, we can simply construct an array with a length shorter than its type implies, construct an iterator and incrementally push it out-of-bounds with repeated `next()`-calls (for this we can't use a loop, as the type-constraints for the current index position have to propagate from each call to the next).
From there, read-write-access can be obtained into an adjacent array of a different type, which enables getting object addresses and constructing fake objects.
Then, well-known techniques can be used to obtain arbtirary read+write, get a rwx page using web assembly, and execute shellcode (for details see `ror_rce.js`).

Fixing the bypass
-----------------

To fix the bypass, the `TypeGuard()` could simply be replaced by a proper `CheckBounds()`-node.
A suggested patch is attached as `js-call-reducer.cc.patch`.


#### Please briefly explain who can exploit the vulnerability, and what they gain when doing so

An attacker running a malicious website can get arbitrary code-execution in the context of Chrome's renderer process.


---

### The cause


#### What version of Chrome have you found the security issue in?

Chromium 92.0.4515.107 Arch Linux; current main branch of v8


#### Is the security issue related to a crash?

No


#### Choose the type of vulnerability
```

Remote Code Execution (RCE)

Please provide your credit information

Manfred Paul (@_manfp)

Comment 1 by manfp...@gmail.com on Fri, Jul 30, 2021, 9:43 AM EDT

ror_testcase.js
176 bytes View Download

ror_rce.js
7.7 KB View Download

js-call-reducer.cc.patch
337 bytes View Download

machine-operator-reducer.cc.patch
682 bytes View Download

Comment 2 by chrom...@appspot.gserviceaccount.com on Fri, Jul 30, 2021, 9:44 AM EDT Project Member

Labels: external_security_report

Comment 3 by meacer@google.com on Mon, Aug 2, 2021, 6:25 AM EDT Project Member

Status: Assigned (was: Unconfirmed)
Owner: neis@chromium.org
Labels: Security_Severity-High Security_Impact-Stable FoundIn-92 OS-Android OS-Chrome OS-Fuchsia OS-Linux OS-Mac OS-Windows OS-Lacros Pri-1
Components: Blink>JavaScript>Compiler

Comment 4 by neis@chromium.org on Mon, Aug 2, 2021, 6:48 AM EDT Project Member

Thanks for the detailed report.

Comment 5 by neis@chromium.org on Mon, Aug 2, 2021, 9:31 AM EDT Project Member

Status: Started (was: Assigned)

Comment 6 by sheriffbot on Mon, Aug 2, 2021, 12:47 PM EDT Project Member

Labels: M-92 Target-92

Setting milestone and target because of high severity.

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

Comment 7 by neis@chromium.org on Tue, Aug 3, 2021, 2:42 AM EDT Project Member

For the record, this bug has been around since late 2014.

Comment 8 by neis@chromium.org on Tue, Aug 3, 2021, 3:56 AM EDT Project Member

Cc: nicohartmann@chromium.org

Comment 9 by Git Watcher on Tue, Aug 3, 2021, 4:15 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/ca386a4b383165ccaed628c19a1366a273fa371e

commit ca386a4b383165ccaed628c19a1366a273fa371e
Author: Georg Neis <neis@chromium.org>
Date: Tue Aug 03 07:04:09 2021

[compiler] Fix bug in MachineOperatorReducer::TryMatchWord32Ror

~~Bug: chromium:1234764~~
Change-Id: Ie899f00e9247bdf67b59aa3ebb7def2948ccdb6a
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3067332
Reviewed-by: Nico Hartmann <nicohartmann@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/heads/master@{#76050}

[modify] https://crrev.com/ca386a4b383165ccaed628c19a1366a273fa371e/src/compiler/machine-operator-reducer.cc
[modify] https://crrev.com/ca386a4b383165ccaed628c19a1366a273fa371e/test/unittests/compiler/machine-operator-reducer-unittest.cc

Comment 10 by neis@chromium.org on Tue, Aug 3, 2021, 4:27 AM EDT Project Member

Labels: tests_pending

Comment 11 by Git Watcher on Tue, Aug 3, 2021, 4:28 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/65b20a0e65e1078f5dd230a5203e231bec790ab4

commit 65b20a0e65e1078f5dd230a5203e231bec790ab4
Author: Georg Neis <neis@chromium.org>
Date: Mon Aug 02 20:14:20 2021

[compiler] Harden JSCallReducer::ReduceArrayIteratorPrototypeNext

~~Bug: chromium:1234764~~
Change-Id: I5b1053accf77331687939c789b7ed94df1219287
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3067327
Reviewed-by: Nico Hartmann <nicohartmann@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/heads/master@{#76052}

[modify] https://crrev.com/65b20a0e65e1078f5dd230a5203e231bec790ab4/src/compiler/js-call-reducer.cc

Comment 12 by neis@chromium.org on Tue, Aug 3, 2021, 4:31 AM EDT Project Member

Status: Fixed (was: Started)

Manfred, thanks again for the truly excellent reports. I have followed your suggestions for the fix and hardening.

Comment 13 by sheriffbot on Tue, Aug 3, 2021, 12:42 PM EDT Project Member

Labels: reward-topanel

Comment 14 by sheriffbot on Tue, Aug 3, 2021, 1:42 PM EDT Project Member

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

Comment 15 by sheriffbot on Tue, Aug 3, 2021, 2:07 PM EDT Project Member

Labels: Merge-Request-92 Merge-Request-93

This is sufficiently serious that it should be merged to stable. But I can't see a Chromium repo commit here, so you will need to investigate what - if anything - needs to be merged to M92. Is there a fix in some other repo which should be merged? Or, perhaps this ticket is a duplicate of some other ticket which has the real fix: please track that down and ensure it is merged appropriately.

This is sufficiently serious that it should be merged to beta. But I can't see a Chromium repo commit here, so you will need to investigate what - if anything - needs to be merged to M93. Is there a fix in some other repo which should be merged? Or, perhaps this ticket is a duplicate of some other ticket which has the real fix: please track that down and ensure it is merged appropriately.

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

Comment 16 by sheriffbot on Tue, Aug 3, 2021, 2:13 PM EDT Project Member

Labels: -Merge-Request-93 Hotlist-Merge-Review Merge-Review-93

This bug requires manual review: M93'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://chromium.googlesource.com/chromium/src.git/+/main/docs/process/merge_request.md#when-to-request-a-merge
- 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 ToT?
4. Does this change need to be merged into other active release branches (M-1, M+1)?
5. Why are these changes required in this milestone after branch?
6. Is this a new feature?
7. If it is a new feature, is it behind a flag using finch?

Chrome OS Only:
8. Was the change reviewed and approved by the Eng Prod Representative? See Eng Prod ownership by component: http://go/cros-engprodcomponents

Please contact the milestone owner if you have questions.
Owners: benmason@(Android), govind@(iOS), geohsu@(ChromeOS), pbommana@(Desktop)

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

Comment 17 by neis@chromium.org on Wed, Aug 4, 2021, 3:51 AM EDT Project Member

Rec #c16:
1) Yes.
2) At least the one in #c9 but ideally also the one in #c11.
3) Yes (verified with unittest in v8, not with the full exploit in Chrome).
4) Yes.
5) Security bug fix.
6) No.

Comment 18 by neis@google.com on Mon, Aug 9, 2021, 3:10 AM EDT Project Member

Cc: adetaylor@chromium.org pbomm...@chromium.org

Hi, I'd like to merge this ASAP. Is there anything missing?

Comment 19 by vahl@chromium.org on Mon, Aug 9, 2021, 3:28 AM EDT Project Member

Labels: -Merge-Request-92 -Merge-Review-93 Merge-Approved-92 Merge-Approved-93

Please go head and merge #c9 and #c11 to V8 9.2 and 9.3 branches

Comment 20 by Git Watcher on Mon, Aug 9, 2021, 4:42 AM EDT Project Member

Labels: merge-merged-9.3

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/e298903b7b05cb111b6ac2c02af1668097190af9

commit e298903b7b05cb111b6ac2c02af1668097190af9
Author: Georg Neis <neis@chromium.org>
Date: Mon Aug 09 07:46:40 2021

Merged: [compiler] Fix bug in MachineOperatorReducer::TryMatchWord32Ror

Revision: ca386a4b383165ccaed628c19a1366a273fa371e

~~BUG=chromium:1234764~~
NOTRY=true
NOPRESUBMIT=true
NOTREECHECKS=true
R=nicohartmann@chromium.org

Change-Id: I4effe6275f8f956b2345714479954feab9a237ab
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3080562
Reviewed-by: Lutz Vahl <vahl@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/branch-heads/9.3@{#20}
Cr-Branched-From: 7744dce208a555494e4a33e24fadc71ea20b3895-refs/heads/9.3.345@{#1}
Cr-Branched-From: 4b6b4cabf3b6a20cdfda72b369df49f3311c4344-refs/heads/master@{#75728}

[modify] https://crrev.com/e298903b7b05cb111b6ac2c02af1668097190af9/src/compiler/machine-operator-reducer.cc
[modify] https://crrev.com/e298903b7b05cb111b6ac2c02af1668097190af9/test/unittests/compiler/machine-operator-reducer-unittest.cc

Comment 21 by Git Watcher on Mon, Aug 9, 2021, 4:44 AM EDT Project Member

Labels: merge-merged-9.2

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/452d41efdf7107bd17e9b9887657a6e1018bccd8

commit 452d41efdf7107bd17e9b9887657a6e1018bccd8
Author: Georg Neis <neis@chromium.org>
Date: Mon Aug 09 07:52:11 2021

Merged: [compiler] Fix bug in MachineOperatorReducer::TryMatchWord32Ror

Revision: ca386a4b383165ccaed628c19a1366a273fa371e

~~BUG=chromium:1234764~~
NOTRY=true
NOPRESUBMIT=true
NOTREECHECKS=true
R=nicohartmann@chromium.org

Change-Id: I9cbe8646daa7137a33b13e19105b1679b0218f1c
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3080563
Reviewed-by: Lutz Vahl <vahl@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/branch-heads/9.2@{#51}
Cr-Branched-From: 51238348f95a1f5e0acc321efac7942d18a687a2-refs/heads/9.2.230@{#1}
Cr-Branched-From: 587a04f02ab0487d194b55a7137dc2045e071597-refs/heads/master@{#74656}

[modify] https://crrev.com/452d41efdf7107bd17e9b9887657a6e1018bccd8/src/compiler/machine-operator-reducer.cc
[modify] https://crrev.com/452d41efdf7107bd17e9b9887657a6e1018bccd8/test/unittests/compiler/machine-operator-reducer-unittest.cc

Comment 22 by gov...@chromium.org on Mon, Aug 9, 2021, 1:49 PM EDT Project Member

Please merge your change to M93 branch 4577 ASAP so we can take it in for this week Beta Release. Thank you.

Comment 23 by vahl@chromium.org on Tue, Aug 10, 2021, 2:32 AM EDT Project Member

This V8 only changes are already merged into V8 9.3 and is included in the V8 9.3 lkgr: https://chromium.googlesource.com/v8/v8/+log/refs/heads/9.3-lkgr which sould be used for the Beta release.
govind@ please let us know in case there is a need to back merge the change anywhere else.

Comment 24 by neis@chromium.org on Tue, Aug 10, 2021, 3:13 AM EDT Project Member

Merge of the CL in #c11 is still missing. Coming today.

Comment 25 by vahl@chromium.org on Tue, Aug 10, 2021, 3:32 AM EDT Project Member

Right #c11 needs to land, but will do so today. This one is as well a V8 change, so will be part of https://chromium.googlesource.com/v8/v8/+log/refs/heads/9.3-lkgr afterwards.

Comment 26 by Git Watcher on Tue, Aug 10, 2021, 4:31 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/85fcc01d0bfbf4de666cbb3a6ea3147b479a5a20

commit 85fcc01d0bfbf4de666cbb3a6ea3147b479a5a20
Author: Georg Neis <neis@chromium.org>
Date: Tue Aug 10 07:18:10 2021

Merged: [compiler] Harden JSCallReducer::ReduceArrayIteratorPrototypeNext

Revision: 65b20a0e65e1078f5dd230a5203e231bec790ab4

~~BUG=chromium:1234764~~
NOTRY=true
NOPRESUBMIT=true
NOTREECHECKS=true
R=vahl@chromium.org

Change-Id: I6c13c005efd3714b02e13db35fc1eb3311782620
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3084362
Reviewed-by: Lutz Vahl <vahl@chromium.org>
Reviewed-by: Georg Neis <neis@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/branch-heads/9.3@{#23}
Cr-Branched-From: 7744dce208a555494e4a33e24fadc71ea20b3895-refs/heads/9.3.345@{#1}
Cr-Branched-From: 4b6b4cabf3b6a20cdfda72b369df49f3311c4344-refs/heads/master@{#75728}

[modify] https://crrev.com/85fcc01d0bfbf4de666cbb3a6ea3147b479a5a20/src/compiler/js-call-reducer.cc

Comment 27 by Git Watcher on Tue, Aug 10, 2021, 4:32 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/fc6818160b390034c249b7329f6c0be8e4f5b11b

commit fc6818160b390034c249b7329f6c0be8e4f5b11b
Author: Georg Neis <neis@chromium.org>
Date: Tue Aug 10 07:29:33 2021

Merged: [compiler] Harden JSCallReducer::ReduceArrayIteratorPrototypeNext

Revision: 65b20a0e65e1078f5dd230a5203e231bec790ab4

~~BUG=chromium:1234764~~
NOTRY=true
NOPRESUBMIT=true
NOTREECHECKS=true
R=vahl@chromium.org

Change-Id: I45faf253695011092de144c8e29bafac5337adec
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3084363
Reviewed-by: Lutz Vahl <vahl@chromium.org>
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Commit-Position: refs/branch-heads/9.2@{#53}
Cr-Branched-From: 51238348f95a1f5e0acc321efac7942d18a687a2-refs/heads/9.2.230@{#1}
Cr-Branched-From: 587a04f02ab0487d194b55a7137dc2045e071597-refs/heads/master@{#74656}

[modify] https://crrev.com/fc6818160b390034c249b7329f6c0be8e4f5b11b/src/compiler/js-call-reducer.cc

Comment 28 by neis@chromium.org on Tue, Aug 10, 2021, 4:33 AM EDT Project Member

Labels: -Merge-Approved-92 -Merge-Approved-93

Comment 29 by amyressler@google.com on Wed, Aug 11, 2021, 2:25 PM EDT Project Member

Labels: -reward-topanel reward-unpaid reward-21000

*** 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 30 by amyressler@chromium.org on Wed, Aug 11, 2021, 2:39 PM EDT Project Member

Another congratulations, Manfred! The VRP Panel has awarded you $21,000 for this report (V8 exploit + patch bonuses). Thank you for the detailed and excellent report for this issue!

Comment 31 by amyressler@google.com on Fri, Aug 13, 2021, 11:39 AM EDT Project Member

Labels: -reward-unpaid reward-inprocess

Comment 32 by amyressler@google.com on Mon, Aug 16, 2021, 10:10 AM EDT Project Member

Labels: Release-2-M92

Comment 33 by amyressler@google.com on Mon, Aug 16, 2021, 10:20 AM EDT Project Member

Labels: CVE-2021-30598 CVE_description-missing

Comment 34 by rzanoni@google.com on Tue, Aug 17, 2021, 7:45 AM EDT Project Member

Labels: LTS-Security-90 LTS-Merge-Request-90

Comment 35 by rzanoni@google.com on Thu, Aug 19, 2021, 11:28 AM EDT Project Member

Labels: LTS-Size-Small LTS-Complexity-Trivial

Comment 36 by gianluca@google.com on Fri, Aug 20, 2021, 3:28 AM EDT Project Member

Labels: -LTS-Merge-Request-90 LTS-Merge-Approved-90

Comment 37 by Git Watcher on Fri, Aug 20, 2021, 11:49 AM EDT Project Member

Labels: merge-merged-9.0

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/fe05610f4425d1d79c79c16c68cfd308a115e56c

commit fe05610f4425d1d79c79c16c68cfd308a115e56c
Author: Georg Neis <neis@chromium.org>
Date: Tue Aug 03 07:04:09 2021

[M90-LTS][compiler] Fix bug in MachineOperatorReducer::TryMatchWord32Ror

(cherry picked from commit ca386a4b383165ccaed628c19a1366a273fa371e)

~~Bug: chromium:1234764~~
No-Try: true
No-Presubmit: true
No-Tree-Checks: true
Change-Id: Ie899f00e9247bdf67b59aa3ebb7def2948ccdb6a
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3067332
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Original-Commit-Position: refs/heads/master@{#76050}
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3099688
Reviewed-by: Georg Neis <neis@chromium.org>
Reviewed-by: Artem Sumaneev <asumaneev@google.com>
Commit-Queue: Roger Felipe Zanoni da Silva <rzanoni@google.com>
Cr-Commit-Position: refs/branch-heads/9.0@{#71}
Cr-Branched-From: bd0108b4c88e0d6f2350cb79b5f363fbd02f3eb7-refs/heads/9.0.257@{#1}
Cr-Branched-From: 349bcc6a075411f1a7ce2d866c3dfeefc2efa39d-refs/heads/master@{#73001}

[modify] https://crrev.com/fe05610f4425d1d79c79c16c68cfd308a115e56c/src/compiler/machine-operator-reducer.cc
[modify] https://crrev.com/fe05610f4425d1d79c79c16c68cfd308a115e56c/test/unittests/compiler/machine-operator-reducer-unittest.cc

Comment 38 by Git Watcher on Mon, Aug 23, 2021, 7:18 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/de534edddbe8e06ba060746375d296956cb40d5d

commit de534edddbe8e06ba060746375d296956cb40d5d
Author: Georg Neis <neis@chromium.org>
Date: Mon Aug 02 20:14:20 2021

[M90-LTS][compiler] Harden JSCallReducer::ReduceArrayIteratorPrototypeNext

(cherry picked from commit 65b20a0e65e1078f5dd230a5203e231bec790ab4)

~~Bug: chromium:1234764~~
No-Try: true
No-Presubmit: true
No-Tree-Checks: true
Change-Id: I5b1053accf77331687939c789b7ed94df1219287
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3067327
Commit-Queue: Georg Neis <neis@chromium.org>
Cr-Original-Commit-Position: refs/heads/master@{#76052}
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3099689
Reviewed-by: Georg Neis <neis@chromium.org>
Reviewed-by: Artem Sumaneev <asumaneev@google.com>
Commit-Queue: Roger Felipe Zanoni da Silva <rzanoni@google.com>
Cr-Commit-Position: refs/branch-heads/9.0@{#73}
Cr-Branched-From: bd0108b4c88e0d6f2350cb79b5f363fbd02f3eb7-refs/heads/9.0.257@{#1}
Cr-Branched-From: 349bcc6a075411f1a7ce2d866c3dfeefc2efa39d-refs/heads/master@{#73001}

[modify] https://crrev.com/de534edddbe8e06ba060746375d296956cb40d5d/src/compiler/js-call-reducer.cc

Comment 39 by rzanoni@google.com on Mon, Aug 23, 2021, 7:20 AM EDT Project Member

Labels: -LTS-Merge-Approved-90 LTS-Merged-90

Comment 40 by amyressler@google.com on Thu, Aug 26, 2021, 1:44 PM EDT Project Member

Labels: -CVE_description-missing CVE_description-submitted

Comment 41 by neis@chromium.org on Tue, Sep 21, 2021, 4:56 AM EDT Project Member

Cc: mslekova@chromium.org

Comment 42 by Git Watcher on Tue, Sep 21, 2021, 5:18 AM EDT Project Member

The following revision refers to this bug:
https://chromium.googlesource.com/v8/v8/+/db9c2e058b46594ffa101b059dad6d3d53a8aa9f

commit db9c2e058b46594ffa101b059dad6d3d53a8aa9f
Author: Georg Neis <neis@chromium.org>
Date: Tue Sep 21 08:43:44 2021

[compiler] Add some regression tests

Bug: chromium:1228407, ~~chromium:1234764~~, ~~chromium:1234770~~, chromium:1247763
Change-Id: I1e8ffaa04eeda22b71ece2f59038e5c92861fde0
Reviewed-on: https://chromium-review.googlesource.com/c/v8/v8/+/3172751
Commit-Queue: Georg Neis <neis@chromium.org>
Commit-Queue: Maya Lekova <mslekova@chromium.org>
Auto-Submit: Georg Neis <neis@chromium.org>
Reviewed-by: Maya Lekova <mslekova@chromium.org>
Cr-Commit-Position: refs/heads/main@{#76955}

[add] https://crrev.com/db9c2e058b46594ffa101b059dad6d3d53a8aa9f/test/mjsunit/compiler/regress-crbug-1234764.js
[add] https://crrev.com/db9c2e058b46594ffa101b059dad6d3d53a8aa9f/test/mjsunit/compiler/regress-crbug-1247763.js
[add] https://crrev.com/db9c2e058b46594ffa101b059dad6d3d53a8aa9f/test/mjsunit/compiler/regress-crbug-1234770.js
[add] https://crrev.com/db9c2e058b46594ffa101b059dad6d3d53a8aa9f/test/mjsunit/compiler/regress-crbug-1228407.js

Comment 43 by neis@chromium.org on Tue, Sep 21, 2021, 5:20 AM EDT Project Member

Labels: -tests_pending

Comment 44 by sheriffbot on Tue, Nov 9, 2021, 1:30 PM EST 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