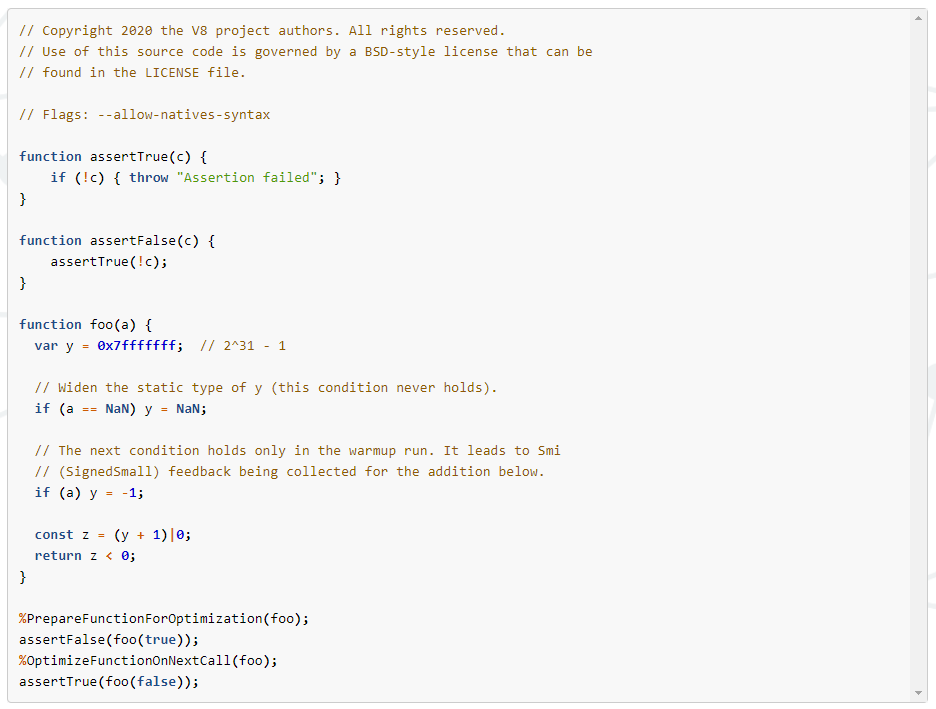
# POC CVE 2020-16040



# Simplified lowering has **4 steps with 3 phrases**:

The core of the Simplified Lowering Phase’s code can be found in src/compiler/simplified-lowering.cc

## Generate Traversal

## Propagate Phase

* Traversal\_nodes\_ vector is tranversed in reverse(???)
  + Why is the End node the first node that is visited?
  + Though the turbolizer graph is read top down...

### 2A Example output

--{Propagate phase}--

**Visit #45: SpeculativeNumberBitwiseOr (trunc: truncate-to-word32)**

initial #43: truncate-to-word32

initial #44: truncate-to-word32

initial #43: truncate-to-word32

initial #36: no-value-use

**Visit #43: SpeculativeSafeIntegerAdd (trunc: truncate-to-word32)**

initial #39: no-truncation (but identify zeros)

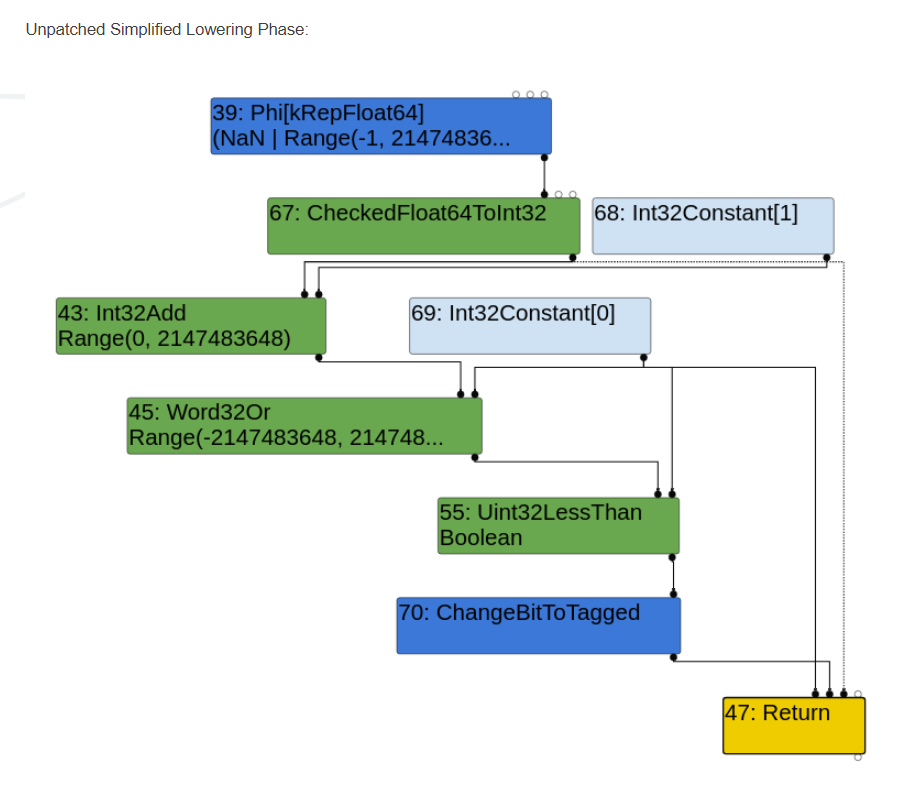
initial #42: no-truncation (but identify zeros)

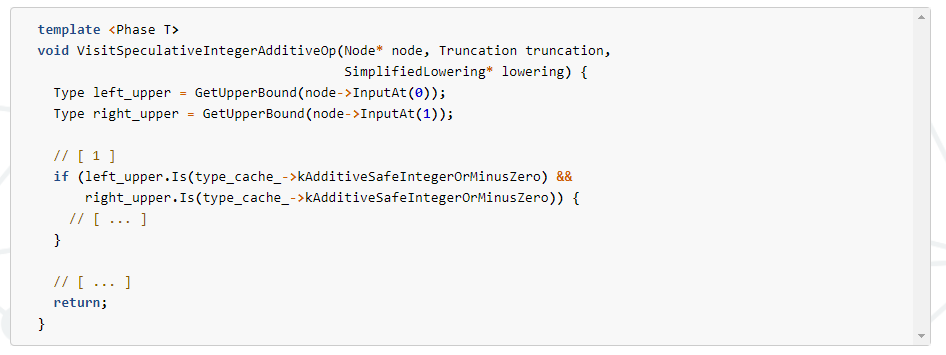
initial #22: no-value-use

initial #36: no-value-use

### 2B Analysis

* **SpeculativeNumberBitwiseOr** node will propagate a **Word32** truncation to its first input **(#43, the SpeculativeSafeIntegerAdd node**)
* Iterate over the traversal\_nodes **backwards** and call **PropagateTruncation**on each node and call **VisitNode**



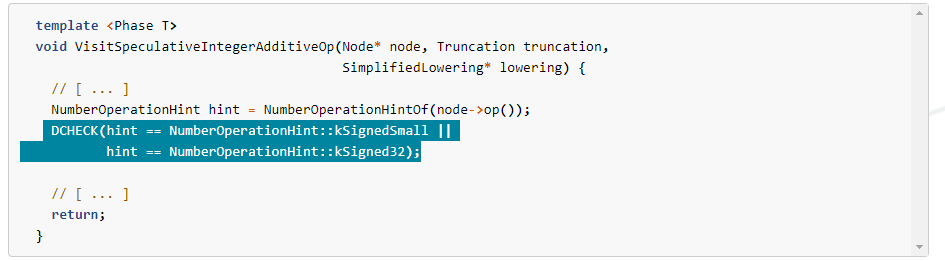


**type\_cache\_->kAdditiveSafeIntegerOrMinusZero** is a UnionType between a **Type::MinusZero()** and a **Range(-4503599627370496, 4503599627370496)**

This is the reason why the type of y has been widened to NaN, in order to get past the if statement. Entering this if statement will just return and not proceed further.

What is the implication here??? What does return and not proceed further mean? Proceed to where?

Within the function, this is the part we want to reach and beyond:



To pass this check, we need to pass the **DCHECK**. Hence we need SignedSmall or Signed32 feedback, now this explains the comments in the POC about the warmup round and collection of feedback.

What happens if you remove the SignedSmall feedback? Instead of SpeculativeSafeIntegerAdd node, it is now a SpeculativeNumberAdd node , this will make it impossible to trigger since the latter does not call VisitSpeculativeIntegerAdditiveOp  during **the Simplified Lowering Phase.**

Why is the call to VisitBinop important??

VisitBinop will process the input nodes with the left\_use and right\_use arguments.

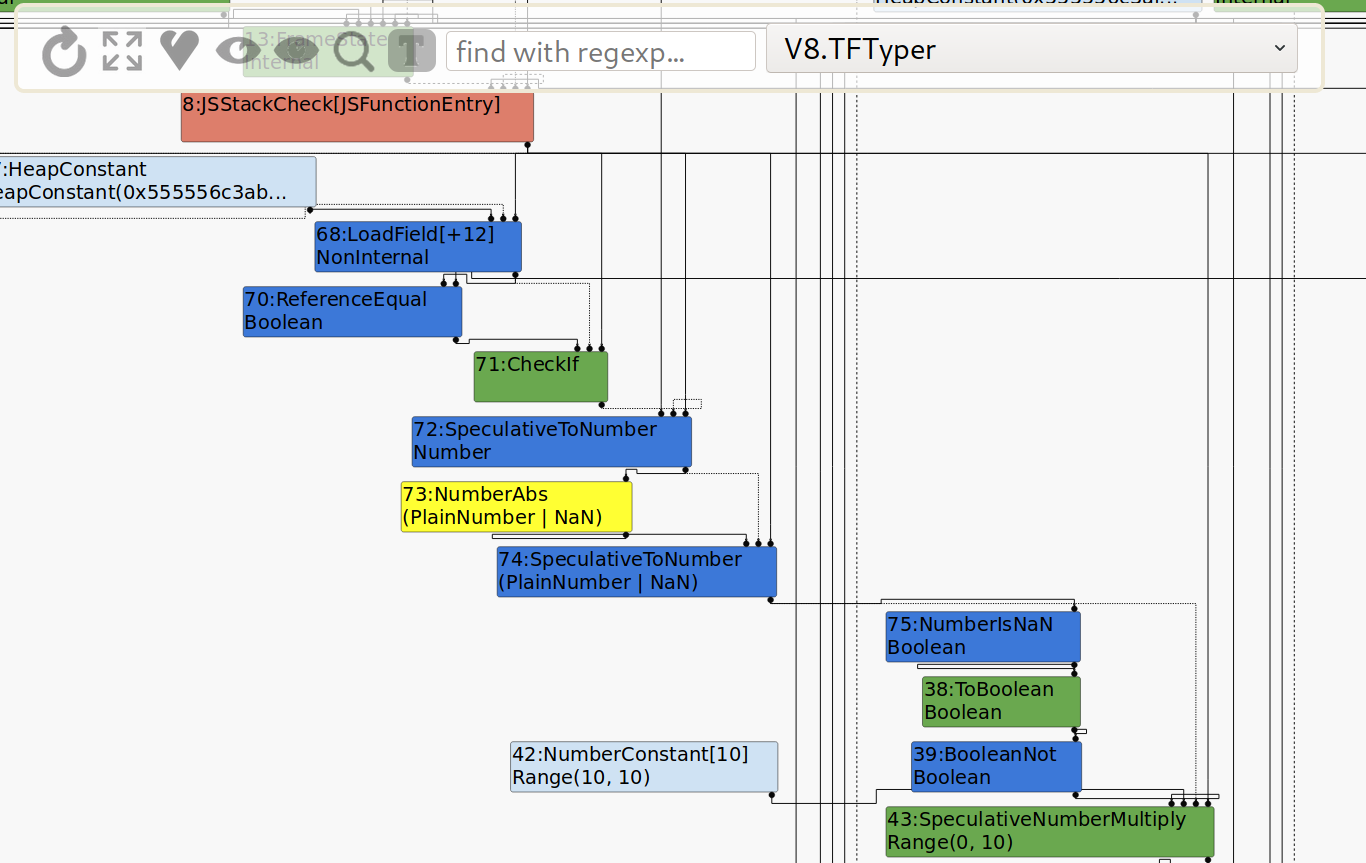
What does this phase do…?

## Retype Phase

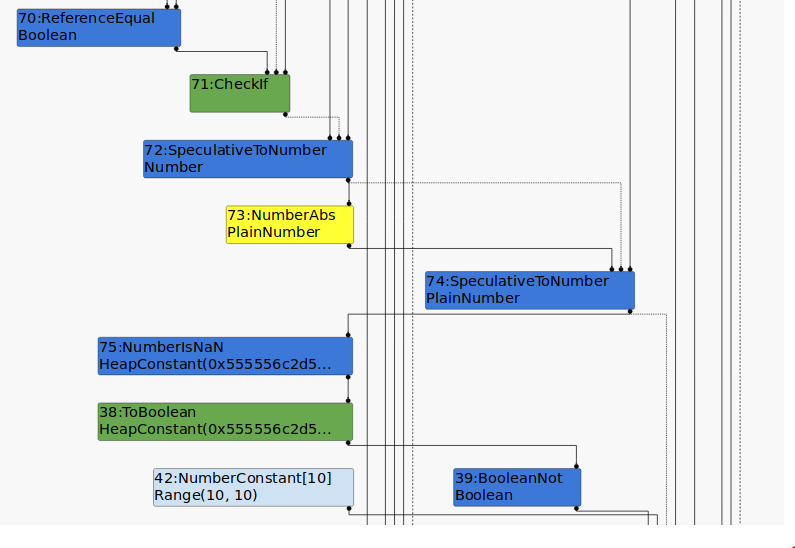
Traversal nodes now traversed from start to finish.

## Lower Phase

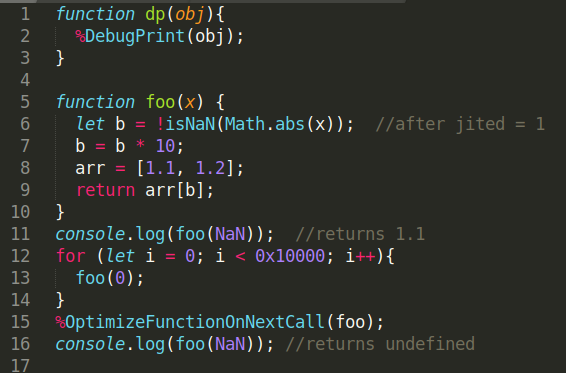
# Correct Graph:



# Patched:



# POC



NaN type has been removed after optimized in the typer phase. Therefore, isNaN checks that it is not NaN.

* PlainNumber type represents any **floating-point number with exception of -0**

# <https://googleprojectzero.blogspot.com/2021/01/in-wild-series-chrome-infinity-bug.html>

# [1] for (var i = -Infinity; i < 0; i += Infinity) { }

Bug: Adding 2 integers doesn’t always result in an integer. -Infinity + +Infinity = NaN

After i = NaN, continue to do more stuff

Initially after [1], range of i: Actual = NaN, inferred in (-Infinity, +Infinity)

# [2] i = Math.max(i, 0x100000800);

Now range of i: Actual = NaN, inferred = [0x100000800, +Infinity]

# [3] i = Math.min(0x100000801, i);

Now range of i: Actual = ???, inferred = [0x100000800, 0x0x100000801]