Division by 0 in `Conv3DBackprop*`

Low **mihaimaruseac** published **GHSA-c968-pq7h-7fxv** 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

The `tf.raw_ops.Conv3DBackprop*` operations fail to validate that the input tensors are not empty. In turn, this would result in a division by 0:

```
import tensorflow as tf

input_sizes = tf.constant([0, 0, 0, 0, 0], shape=[5], dtype=tf.int32)
filter_tensor = tf.constant([], shape=[0, 0, 0, 1, 0], dtype=tf.float32)
out_backprop = tf.constant([], shape=[0, 0, 0, 0, 0], dtype=tf.float32)

tf.raw_ops.Conv3DBackpropInputV2(input_sizes=input_sizes, filter=filter_tensor, out_backprop=out_backprop, strides=[1, 1, 1, 1, 1], padding='SAME', data_format='NDHWC', dilations
```

```
import tensorflow as tf

input_sizes = tf.constant([1], shape=[1, 1, 1, 1, 1], dtype=tf.float32)
filter_tensor = tf.constant([0, 0, 0, 1, 0], shape=[5], dtype=tf.int32)
out_backprop = tf.constant([], shape=[1, 1, 1, 1, 0], dtype=tf.float32)

tf.raw_ops.Conv3DBackpropFilterV2(input=input_sizes, filter_sizes=filter_tensor, out_backprop=out_backprop, strides=[1, 1, 1, 1, 1], padding='SAME', data_format='NDHWC', dilation
```

This is because the implementation does not check that the divisor used in computing the shard size is not zero:

```
const int64 size_A = output_image_size * dims.out_depth;
const int64 size_B = filter_total_size * dims.out_depth;
const int64 size_C = output_image_size * filter_total_size;
const int64 work_unit_size = size_A + size_B + size_C;
...
const size_t shard_size =
    use_parallel_contraction
      ? 1
      : (target_working_set_size + work_unit_size - 1) / work_unit_size;
```

Thus, if attacker controls the input sizes, they can trigger a denial of service via a division by zero error.

Patches

We have patched the issue in GitHub commit 311403edbc9816df80274bd1ea8b3c0c0f22c3fa.

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 Yakun Zhang and Ying Wang of Baidu X-Team.

Severity

Low

CVE ID

CVE-2021-29522

Weaknesses

No CWEs