`CHECK`-fail due to integer overflow

Low **mihaimaruseac** published **GHSA-xvjm-fvxx-q3hv** 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 denial of service via a `CHECK`-fail in caused by an integer overflow in constructing a new tensor shape:

```
import tensorflow as tf

input_layer = 2**60-1
sparse_data = tf.raw_ops.SparseSplit(
    split_dim=1,
    indices=[(0, 0), (0, 1), (0, 2),
    (4, 3), (5, 0), (5, 1)],
    values=[1.0, 1.0, 1.0, 1.0, 1.0, 1.0],
    shape=(input_layer, input_layer),
    num_split=2,
    name=None
    )
```

This is because the implementation builds a dense shape without checking that the dimensions would not result in overflow:

```
sparse::SparseTensor sparse_tensor;
OP_REQUIRES_OK(context,
               sparse::SparseTensor::Create(
                 input_indices, input_values,
                 TensorShape(input_shape.vec<int64>()), &sparse_tensor));
```

The `TensorShape` constructor uses a `CHECK` operation which triggers when `InitDims` returns a non-OK status.

```
template <class Shape>
TensorShapeBase<Shape>::TensorShapeBase(gtl::ArraySlice<int64> dim_sizes) {
  set_tag(REP16);
  set_data_type(DT_INVALID);
  TF_CHECK_OK(InitDims(dim_sizes));
}
```

In our scenario, this occurs when adding a dimension from the argument results in overflow:

```
template <class Shape>
Status TensorShapeBase<Shape>::InitDims(gtl::ArraySlice<int64> dim_sizes) {
  ...
  Status status = Status::OK();
  for (int64 s : dim_sizes) {
    status.Update(AddDimWithStatus(internal::SubtleMustCopy(s)));
    if (!status.ok()) {
      return status;
    }
  }
}

template <class Shape>
Status TensorShapeBase<Shape>::AddDimWithStatus(int64 size) {
  ...
  int64 new_num_elements;
  if (kIsPartial && (num_elements() < 0 || size < 0)) {
    new_num_elements = -1;
  } else {
    new_num_elements = MultiplyWithoutOverflow(num_elements(), size);
    if (TF_PREDICT_FALSE(new_num_elements < 0)) {
        return errors::Internal("Encountered overflow when multiplying ",
                                num_elements(), " with ", size,
                                ", result: ", new_num_elements);
      }
  }
  ...
}
```

This is a legacy implementation of the constructor and operations should use `BuildTensorShapeBase` or `AddDimWithStatus` to prevent `CHECK`-failures in the presence of overflows.

Patches

We have patched the issue in GitHub commit 4c0ee937c0f61c4fc5f5d32d9bb4c67428012a60.

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 researchers from University of Virginia and University of California, Santa Barbara.

Severity

Low

CVE ID

CVE-2021-29584

Weaknesses

No CWEs