Out of bounds access in TFLite operators

Moderate **mihaimaruseac** published **GHSA-cvpc-8phh-8f45** on Sep 24, 2020

Package

tensorflow-lite (tensorflow)

Affected versions	Patched versions
< 2.3.0	1.15.4, 2.0.3, 2.1.2, 2.2.1, 2.3.1

Description

Impact

In TensorFlow Lite, saved models in the flatbuffer format use a double indexing scheme: a model has a set of subgraphs, each subgraph has a set of operators and each operator has a set of input/output tensors. The flatbuffer format uses indices for the tensors, indexing into an array of tensors that is owned by the subgraph. This results in a pattern of double array indexing when trying to get the data of each tensor:

tensorflow/tensorflow/lite/kernels/kernel_util.cc
Line 36 in 0e68f4d

```
36      return &context->tensors[node->inputs->data[index]];
```

However, some operators can have some tensors be optional. To handle this scenario, the flatbuffer model uses a negative `-1` value as index for these tensors:

tensorflow/tensorflow/lite/c/common.h
Line 82 in 0e68f4d

```
82      #define kTfLiteOptionalTensor (-1)
```

This results in special casing during validation at model loading time:

tensorflow/tensorflow/lite/core/subgraph.cc
Lines 566 to 580 in 0e68f4d

```
566     for (int i = 0; i < length; i++) {
567       int index = indices[i];
568       // Continue if index == kTfLiteOptionalTensor before additional comparisons
569       // below, size_t(-1) is always >= context_tensors_size.
570       if (index == kTfLiteOptionalTensor) {
571         continue;
572       }
573       if (index < 0 || static_cast<size_t>(index) >= context_.tensors_size) {
574         ReportError(
575             "Invalid tensor index %d in %s. The subgraph has %d tensors\n", index,
576             label, context_.tensors_size);
577         consistent_ = false;
```

Unfortunately, this means that the `-1` index is a valid tensor index for any operator, including those that don't expect optional inputs and including for output tensors. Thus, this allows writing and reading from outside the bounds of heap allocated arrays, although only at a specific offset from the start of these arrays.

This results in both read and write gadgets, albeit very limited in scope.

Patches

We have patched the issue in several commits (`46d5b08` , `0030278` , `e11f555` , `cd31fd0` , `1970c21` , and `fff2c83`). We will release patch releases for all versions between 1.15 and 2.3.

We recommend users to upgrade to TensorFlow 1.15.4, 2.0.3, 2.1.2, 2.2.1, or 2.3.1.

Workarounds

A potential workaround would be to add a custom `Verifier` to the model loading code to ensure that only operators which accept optional inputs use the `-1` special value and only for the tensors that they expect to be optional. Since this allow-list type approach is erro-prone, we advise upgrading to the patched code.

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.

Severity

Moderate

CVE ID

CVE-2020-15211

Weaknesses

No CWEs