Integer truncation in Shard API usage

Critical **mihaimaruseac** published **GHSA-h6fg-mjxg-hqq4** on Sep 24, 2020

Package

tensorflow, tensorflow-cpu, tensorflow-gpu (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

The `Shard` API in TensorFlow expects the last argument to be a function taking two `int64` (i.e., `long long`) arguments:

tensorflow/tensorflow/core/util/work_sharder.h
Lines 59 to 60 in `0e68f4d`

```
59    void Shard(int max_parallelism, thread::ThreadPool* workers, int64 total,
60               int64 cost_per_unit, std::function<void(int64, int64)> work);
```

However, there are several places in TensorFlow where a lambda taking `int` or `int32` arguments is being used:

tensorflow/tensorflow/core/kernels/random_op.cc
Lines 204 to 205 in `0e68f4d`

```
204    auto DoWork = [samples_per_alpha, num_alphas, &rng, samples_flat,
205                   alpha_flat](int start_output, int limit_output) {
```

tensorflow/tensorflow/core/kernels/random_op.cc
Lines 317 to 318 in `0e68f4d`

```
317    Shard(worker_threads.num_threads, worker_threads.workers,
318          num_alphas * samples_per_alpha, kElementCost, DoWork);
```

In these cases, if the amount of work to be parallelized is large enough, integer truncation occurs. Depending on how the two arguments of the lambda are used, this can result in segfaults, read/write outside of heap allocated arrays, stack overflows, or data corruption.

Patches

We have patched the issue in `27b4173` and `ca8c013`. 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.

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

Critical

CVE ID

CVE-2020-15202

Weaknesses

No CWEs