Using TensorFlow Securely

This document discusses how to safely deal with untrusted programs (models or model parameters), and input data. Below, we also provide guidelines on how to report vulnerabilities in TensorFlow.

TensorFlow models are programs

TensorFlow's runtime system interprets and executes programs. What machine learning practitioners term **models** are expressed as programs that TensorFlow executes. TensorFlow programs are encoded as computation **graphs**. The model's parameters are often stored separately in **checkpoints**.

At runtime, TensorFlow executes the computation graph using the parameters provided. Note that the behavior of the computation graph may change depending on the parameters provided. TensorFlow itself is not a sandbox. When executing the computation graph, TensorFlow may read and write files, send and receive data over the network, and even spawn additional processes. All these tasks are performed with the permissions of the TensorFlow process. Allowing for this flexibility makes for a powerful machine learning platform, but it has implications for security.

The computation graph may also accept **inputs**. Those inputs are the data you supply to TensorFlow to train a model, or to use a model to run inference on the data.

TensorFlow models are programs, and need to be treated as such from a security perspective.

Running untrusted models

As a general rule: **Always** execute untrusted models inside a sandbox (e.g., nsjail).

There are several ways in which a model could become untrusted. Obviously, if an untrusted party supplies TensorFlow kernels, arbitrary code may be executed. The same is true if the untrusted party provides Python code, such as the Python code that generates TensorFlow graphs.

Even if the untrusted party only supplies the serialized computation graph (in form of a `GraphDef`, `SavedModel`, or equivalent on-disk format), the set of computation primitives available to TensorFlow is powerful enough that you should assume that the TensorFlow process effectively executes arbitrary code. One common solution is to allow only a few safe Ops. While this is possible in theory, we still recommend you sandbox the execution.

It depends on the computation graph whether a user provided checkpoint is safe. It is easily possible to create computation graphs in which malicious checkpoints can trigger unsafe behavior. For example, consider a graph that contains a `tf.cond` depending on the value of a `tf.Variable`. One branch of the `tf.cond` is harmless, but the other is unsafe. Since the `tf.Variable` is stored in the checkpoint, whoever provides the checkpoint now has the ability to trigger unsafe behavior, even though the graph is not under their control.

In other words, graphs can contain vulnerabilities of their own. To allow users to provide checkpoints to a model you run on their behalf (e.g., in order to compare model quality for a fixed model architecture), you must carefully audit your model, and we recommend you run the TensorFlow process in a sandbox.

Accepting untrusted Inputs

It is possible to write models that are secure in a sense that they can safely process untrusted inputs assuming there are no bugs. There are two main reasons to not rely on this: First, it is easy to write models which must not be exposed to untrusted inputs, and second, there are bugs in any software system of sufficient complexity. Letting users control inputs could allow them to trigger bugs either in TensorFlow or in dependent libraries.

In general, it is good practice to isolate parts of any system which is exposed to untrusted (e.g., user-provided) inputs in a sandbox.

A useful analogy to how any TensorFlow graph is executed is any interpreted programming language, such as Python. While it is possible to write secure Python code which can be exposed to user supplied inputs (by, e.g., carefully quoting and sanitizing input strings, size-checking input blobs, etc.), it is very easy to write Python programs which are insecure. Even secure Python code could be rendered insecure by a bug in the Python interpreter, or in a bug in a Python library used (e.g., this one).

Running a TensorFlow server

TensorFlow is a platform for distributed computing, and as such there is a TensorFlow server (`tf.train.Server`). **The TensorFlow server is meant for internal communication only. It is not built for use in an untrusted network.**

For performance reasons, the default TensorFlow server does not include any authorization protocol and sends messages unencrypted. It accepts connections from anywhere, and executes the graphs it is sent without performing any checks. Therefore, if you run a `tf.train.Server` in your network, anybody with access to the network can execute what you should consider arbitrary code with the privileges of the process running the `tf.train.Server`.

When running distributed TensorFlow, you must isolate the network in which the cluster lives. Cloud providers provide instructions for setting up isolated networks, which are sometimes branded as “virtual private cloud.” Refer to the instructions for GCP and AWS) for details.

Note that `tf.train.Server` is different from the server created by `tensorflow/serving` (the default binary for which is called `ModelServer`). By default, `ModelServer` also has no built-in mechanism for authentication. Connecting it to an untrusted network allows anyone on this network to run the graphs known to the `ModelServer`. This means that an attacker may run graphs using untrusted inputs as described above, but they would not be able to execute arbitrary graphs. It is possible to safely expose a `ModelServer` directly to an untrusted network, **but only if the graphs it is configured to use have been carefully audited to be safe**.

Similar to best practices for other servers, we recommend running any `ModelServer` with appropriate privileges (i.e., using a separate user with reduced permissions). In the spirit of defense in depth, we recommend authenticating requests to any TensorFlow server connected to an untrusted network, as well as sandboxing the server to minimize the adverse effects of any breach.

Vulnerabilities in TensorFlow

TensorFlow is a large and complex system. It also depends on a large set of third party libraries (e.g., `numpy`, `libjpeg-turbo`, PNG parsers, `protobuf`). It is possible that TensorFlow or its dependent libraries contain vulnerabilities that would allow triggering unexpected or dangerous behavior with specially crafted inputs.

What is a vulnerability?

Given TensorFlow's flexibility, it is possible to specify computation graphs which exhibit unexpected or unwanted behavior. The fact that TensorFlow models can perform arbitrary computations means that they may read and write files, communicate via the network, produce deadlocks and infinite loops, or run out of memory. It is only when these behaviors are outside the specifications of the operations involved that such behavior is a vulnerability.

A `FileWriter` writing a file is not unexpected behavior and therefore is not a vulnerability in TensorFlow. A `MatMul` allowing arbitrary binary code execution **is** a vulnerability.

This is more subtle from a system perspective. For example, it is easy to cause a TensorFlow process to try to allocate more memory than available by specifying a computation graph containing an ill-considered `tf.tile` operation. TensorFlow should exit cleanly in this case (it would raise an exception in Python, or return an error `Status` in C++). However, if the surrounding system is not expecting the possibility, such behavior could be used in a denial of service attack (or worse). Because TensorFlow behaves correctly, this is not a vulnerability in TensorFlow (although it would be a vulnerability of this hypothetical system).

As a general rule, it is incorrect behavior for TensorFlow to access memory it does not own, or to terminate in an unclean way. Bugs in TensorFlow that lead to such behaviors constitute a vulnerability.

One of the most critical parts of any system is input handling. If malicious input can trigger side effects or incorrect behavior, this is a bug, and likely a vulnerability.

Reporting vulnerabilities

Please email reports about any security related issues you find to `security@tensorflow.org`. This mail is delivered to a small security team. Your email will be acknowledged within one business day, and you'll receive a more detailed response to your email within 7 days indicating the next steps in handling your report. For critical problems, you may encrypt your report (see below).

Please use a descriptive subject line for your report email. After the initial reply to your report, the security team will endeavor to keep you informed of the progress being made towards a fix and announcement.

In addition, please include the following information along with your report:

- Your name and affiliation (if any).
- A description of the technical details of the vulnerabilities. It is very important to let us know how we can reproduce your findings.
- An explanation who can exploit this vulnerability, and what they gain when doing so – write an attack scenario. This will help us evaluate your report quickly, especially if the issue is complex.
- Whether this vulnerability public or known to third parties. If it is, please provide details.

If you believe that an existing (public) issue is security-related, please send an email to `security@tensorflow.org`. The email should include the issue ID and a short description of why it should be handled according to this security policy.

Once an issue is reported, TensorFlow uses the following disclosure process:

- When a report is received, we confirm the issue and determine its severity.
- If we know of specific third-party services or software based on TensorFlow that require mitigation before publication, those projects will be notified.
- An advisory is prepared (but not published) which details the problem and steps for mitigation.
- The vulnerability is fixed and potential workarounds are identified.
- Wherever possible, the fix is also prepared for the branches corresponding to all releases of TensorFlow at most one year old. We will attempt to commit these fixes as soon as possible, and as close together as possible.
- Patch releases are published for all fixed released versions, a notification is sent to discuss@tensorflow.org, and the advisory is published.

Note that we mostly do patch releases for security reasons and each version of TensorFlow is supported for only 1 year after the release.

Past security advisories are listed below. We credit reporters for identifying security issues, although we keep your name confidential if you request it.

Encryption key for `security@tensorflow.org` If your disclosure is extremely sensitive, you may choose to encrypt your report using the key below. Please only use this for critical security reports.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQENBFpqdzwBCADTeAHLNEe9Vm77AxhmGP+CdjlY84O6DouOCDSq00zFYdIU/7aI
LjYwhEmDEvLnRCYeFGdIHVtW9YrVktqYE9HXVQC7nULU6U6cvkQbwHCdrjaDaylP
aJUXkNrrxibhx9YYdy465CfusAaZ0aM+T9DpcZg98SmsSml/HAiiY4mbg/yNVdPs
SEp/Ui4zdIBNNs6at2gGZrd4qWhdM0MqGJlehqdeUKRICE/mdedXwsWLM8AfEA0e
OeTVhZ+EtYCypiF4fVl/NsqJ/zhBJpCx/1FBI1Uf/lu2TE4eOS1FgmIqb2j4T+jY
e+4C8kGB405PAC0n50YpOrOs6k7fiQDjYmbNABEBAAG0LVRlbnNvckZsb3cgU2Vj
dXJpdHkgPHNlY3VyaXR5QHRlbnNvcmZsb3cub3JnPokBTgQTAQgAOBYhBEkvXzHm
gOJBnwP4Wxnef3wVoM2yBQJaanc8AhsDBQsJCAcCBhUKCQgLAgQWAgMBAh4BAheA
AAoJEBnef3wVoM2yNlkIAICqetv33MD9W6mPAXH3eon+KJoeHQHYOuwWfYkUF6CC
o+X2dlPqBSqMG3bFuTrrcwjr9w1V8HkNuzzOJvCm1CJVKaxMzPuXhBq5+DeT67+a
T/wK1L2R1bF0gs7Pp40W3np8iAFEh8sgqtxXvLGJLGDZ1Lnfdprg3HciqaVAiTum
HBFwszszZZ1wAnKJs5KVteFN7GSSng3qBcj0E0ql2nPGEqCVh+6RG/TU5C8gEsEf
3DX768M4okmFDKTzLNBm+l08kkBFt+P43rNK8dyC4PXk7yJa93SmS/dlK6DZ16Yw
2FS1StiZSVqygTW59rM5XNwdhKVXy2mf/RtNSr84gSi5AQ0EWmp3PAEIALInfBLR
N6fAUGPFj+K3za3PeD0fWDijlC9f4Ety/icwWPkOBdYVBn0atzI21thPRbfuUxfe
zr76xNNrtRRlbDSAChA1J5T86EflowcQor8dNC6fS+oHFCGeUjfEAm16P6mGTo0p
osdG2XnnTHOOEFbEUeWOwR/zT0QRaGGknoy2pc4doWcJptqJIdTl1K8xyBieik/b
nSoClqQdZJa4XA3H9G+F4NmoZGEguC5GGb2P9NHYAJ3MLHBHywZip8g9oojIwda+
OCLL4UPEZ89cl0EyhXM0nIAmGn3Chdjfu3ebF0SeuToGN8E1goUs3qSE77ZdzIsR
BzZSDFrgmZH+uP0AEQEAAYkBNgQYAQgAIBYhBEkvXzHmgOJBnwP4Wxnef3wVoM2y
BQJaanc8AhsMAAoJEBnef3wVoM2yX4wIALcYZbQhSEzCsTl56UHofze6C3QuFQIH
J4MIKrkTfwiHlCujv7GASGU2Vtis5YEyOoMidUVLlwnebE388MmaJYRm0fhYq6lP
A3vnOCcczy1tbo846bRdv012zdUA+wY+mOITdOoUjAhYulUR0kiA2UdLSfYzbWwy
7Obq96Jb/cPRxk8jKUu2rqC/KDrkFDtAtjdIHh6nbbQhFuaRuWntISZgpIJxd8Bt
Gwi0imUVd9m9wZGuTbDGi6YTNk0GPpX5OMF5hjtM/objzTihSw9UN+65Y/oSQM81
v//Fw6ZeY+HmRDFdirjD7wXtIuER4vqCryIqR6Xe9X8oJXz9L/Jhslc=
=CDME
-----END PGP PUBLIC KEY BLOCK-----
```

Known Vulnerabilities

At this time there are no known vulnerability with TensorFlow-models. For a list of known vulnerabilities and security advisories for TensorFlow, click here.