ThreatQuotient

Atom Feed Reader CDF

Version 1.0.0

April 30, 2024

ThreatQuotient
20130 Lakeview Center Plaza Suite 400
Ashburn, VA 20147

ThreatQ Supported

Support

Email:	support@threatq.com
Web:	support.threatq.com
Phone:	703.574.9893

Contents

Warning and Disclaimer

ThreatQuotient, Inc. provides this document “as is”, without representation or warranty of any kind, express or implied, including without limitation any warranty concerning the accuracy, adequacy, or completeness of such information contained herein. ThreatQuotient, Inc. does not assume responsibility for the use or inability to use the software product as a result of providing this information.

Copyright © 2024 ThreatQuotient, Inc.
All rights reserved. This document and the software product it describes are licensed for use under a software license agreement. Reproduction or printing of this document is permitted in accordance with the license agreement.

Support

This integration is designated as **ThreatQ Supported**.

Support Email: support@threatq.com
Support Web: https://support.threatq.com
Support Phone: 703.574.9893

Integrations/apps/add-ons designated as **ThreatQ Supported** are fully supported by ThreatQuotient's Customer Support team.

ThreatQuotient strives to ensure all ThreatQ Supported integrations will work with the current version of ThreatQuotient software at the time of initial publishing. This applies for both Hosted instance and Non-Hosted instance customers.

⚠ ThreatQuotient does not provide support or maintenance for integrations, apps, or add-ons published by any party other than ThreatQuotient, including third-party developers.

Integration Details

ThreatQuotient provides the following details for this integration:

Current Integration Version	1.0.0
Compatible with ThreatQ Versions	>= 5.6.0
Support Tier	ThreatQ Supported

Introduction

The Atom Feed Reader CDF enables analysts to automatically ingest Atom feeds from multiple sources, directly into ThreatQ.

The integration provides the following feed:

- **Atom Feed Reader -** ingests Atom feeds, from multiple sources, into the ThreatQ platform.

The integration ingests the following system objects:

- Reports
- Indicators

Installation

Perform the following steps to install the integration:

> The same steps can be used to upgrade the integration to a new version.

1. Log into https://marketplace.threatq.com/.
2. Locate and download the integration yaml file.
3. Navigate to the integrations management page on your ThreatQ instance.
4. Click on the **Add New Integration** button.
5. Upload the integration yaml file using one of the following methods:
    - Drag and drop the file into the dialog box
    - Select **Click to Browse** to locate the file on your local machine

    > ThreatQ will inform you if the feed already exists on the platform and will require user confirmation before proceeding. ThreatQ will also inform you if the new version of the feed contains changes to the user configuration. The new user configurations will overwrite the existing ones for the feed and will require user confirmation before proceeding.

6. The feed will be added to the integrations page. You will still need to configure and then enable the feed.

Configuration

ThreatQuotient does not issue API keys for third-party vendors. Contact the specific vendor to obtain API keys and other integration-related credentials.

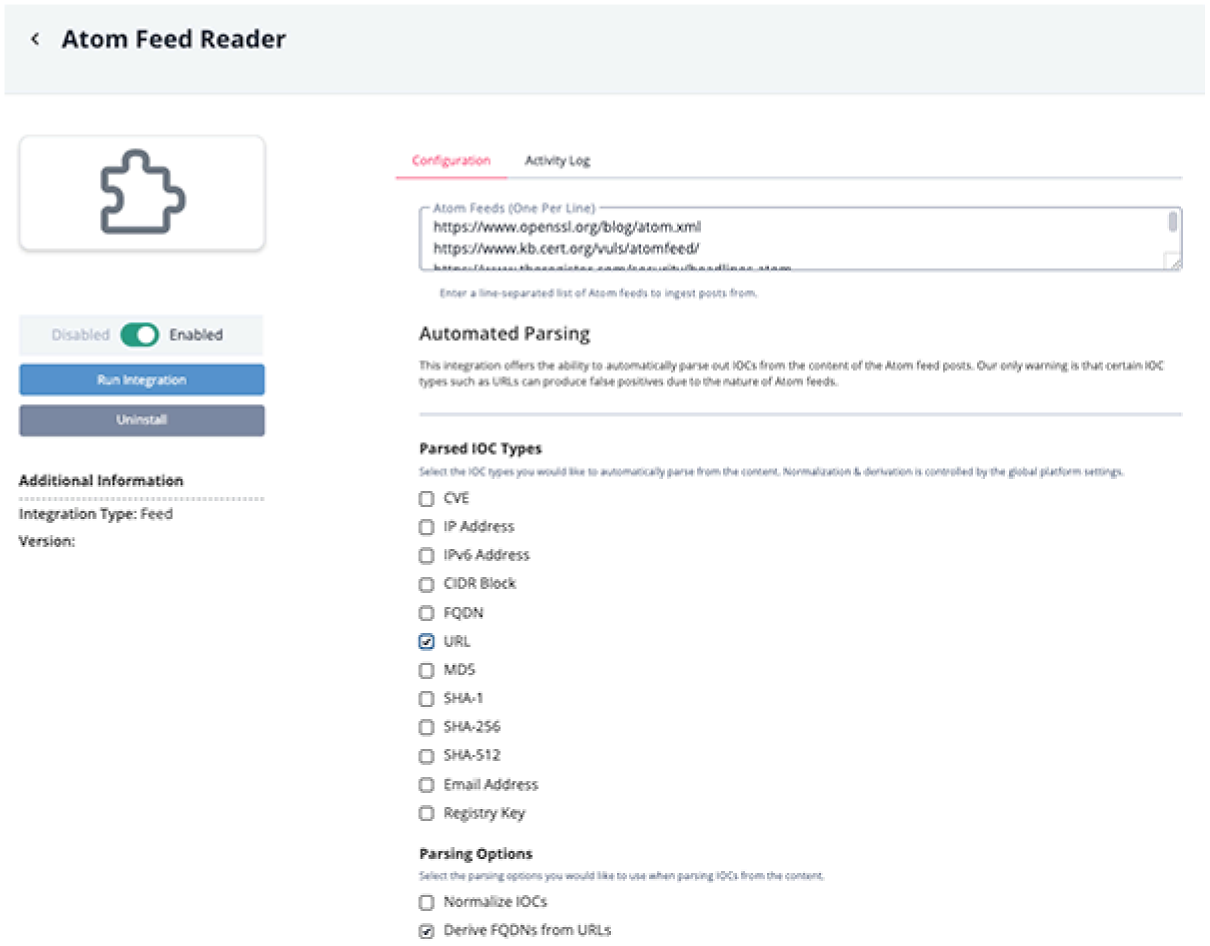
To configure the integration:

1. Navigate to your integrations management page in ThreatQ.
2. Select the **OSINT** option from the *Category* dropdown (optional).

   If you are installing the integration for the first time, it will be located under the **Disabled** tab.

3. Click on the integration entry to open its details page.
4. Enter the following parameters under the **Configuration** tab:

PARAMETER	DESCRIPTION
Atom Feeds	Enter a line-separated list of Atom feeds to ingest data from.
Parsed IOC Types	Select the IOC types to automatically parse from the content. Options include: ◦ CVE ◦ IP Address ◦ IPv6 Address ◦ CIDR Block ◦ FQDN ◦ URL ◦ MD5 ◦ SHA-1 ◦ SHA-256 ◦ SHA-512 ◦ Email Address ◦ Registry Key
Parsing Options	Select the parsing options to use when parsing IOCs from content. Options include: ◦ Normalize IOCs ◦ Derive FQDNs from URLs



Atom Feeds (One Per Line)
https://www.openssl.org/blog/atom.xml
https://www.kb.cert.org/vuls/atomfeed/

Enter a line-separated list of Atom feeds to ingest posts from.

Automated Parsing

This integration offers the ability to automatically parse out IOCs from the content of the Atom feed posts. Our only warning is that certain IOC types such as URLs can produce false positives due to the nature of Atom feeds.

Parsed IOC Types

Select the IOC types you would like to automatically parse from the content. Normalization & derivation is controlled by the global platform settings.

- ☐ CVE
- ☐ IP Address
- ☐ IPv6 Address
- ☐ CIDR Block
- ☐ FQDN
- ☑ URL
- ☐ MD5
- ☐ SHA-1
- ☐ SHA-256
- ☐ SHA-512
- ☐ Email Address
- ☐ Registry Key

Parsing Options

Select the parsing options you would like to use when parsing IOCs from the content.

- ☐ Normalize IOCs
- ☑ Derive FQDNs from URLs

Disabled / Enabled

Additional Information

Integration Type: Feed

Version:

5. Review any additional settings, make any changes if needed, and click on **Save**.
6. Click on the toggle switch, located above the *Additional Information* section, to enable it.

ThreatQ Mapping

Atom Feed Reader

The Atom Feed Reader feed pulls entries from one or more Atom feeds. Entries will be parsed and uploaded to ThreatQ as Reports.

`GET https://www.kb.cert.org/vuls/atomfeed/`

Sample Response:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xml:lang="en-us" xmlns="http://www.w3.org/2005/Atom"><title>CERT Recently
Published Vulnerability Notes</title><link href="https://kb.cert.org/vuls/"
rel="alternate"/><link href="https://kb.cert.org/vuls/atomfeed/" rel="self"/
><id>https://kb.cert.org/vuls/</id><updated>2024-04-18T18:47:30.440271+00:00</
updated><author><name>CERT</name><email>cert@cert.org</email><uri>https://
www.sei.cmu.edu</uri></author><subtitle>CERT publishes vulnerability advisories
called "Vulnerability Notes." Vulnerability Notes include summaries, technical
details, remediation information, and lists of affected vendors.  Many
vulnerability notes are the result of private coordination and disclosure
efforts.</subtitle><entry><title>VU#253266: Keras 2 Lambda Layers Allow
Arbitrary Code Injection in TensorFlow Models</title><link href="https://
kb.cert.org/vuls/id/253266" rel="alternate"/
><published>2024-04-16T20:16:27.223993+00:00</
published><updated>2024-04-18T18:47:30.440271+00:00</updated><id>https://
kb.cert.org/vuls/id/253266</id><summary type="html">
&lt;h3 id="overview"&gt;Overview&lt;/h3&gt;
&lt;p&gt;Lambda Layers in third party TensorFlow-based Keras models allow
attackers to inject arbitrary code into versions built prior to Keras 2.13 that
may then unsafely run with the same permissions as the running application. For
example, an attacker could use this feature to trojanize a popular model, save
it, and redistribute it, tainting the supply chain of dependent AI/ML
applications. &lt;/p&gt;
&lt;h3 id="description"&gt;Description&lt;/h3&gt;
&lt;p&gt;TensorFlow is a widely-used open-source software library for building
machine learning and artificial intelligence applications. The Keras framework,
implemented in Python, is a high-level interface to TensorFlow  that provides a
wide variety of features for the design, training, validation and packaging of
ML models. Keras provides an API for building neural networks from building
blocks called Layers. One such Layer type is a Lambda layer that allows a
developer to add arbitrary Python code to a model in the form of a lambda
function (an anonymous, unnamed function). Using the
&lt;code&gt;Model.save()&lt;/code&gt; or &lt;code&gt;save_model()&lt;/code&gt;
&lt;a href="https://keras.io/api/models/model_saving_apis/
model_saving_and_loading/#save_model-function"&gt;method&lt;/a&gt;, a developer
can then save a model that includes this code.&lt;/p&gt;
&lt;p&gt;The Keras 2 documentation for the &lt;code&gt;Model.load_model()&lt;/
```

```
code&gt; &lt;a href="https://keras.io/2.16/api/models/model_saving_apis/
model_saving_and_loading/#load_model-function"&gt;method&lt;/a&gt; describes a
mechanism for disallowing the loading of a native version 3 Keras model
(&lt;code&gt;.keras&lt;/code&gt; file) that includes a Lambda layer when
setting &lt;code&gt;safe_mode&lt;/code&gt; (&lt;a href="https://keras.io/api/
models/model_saving_apis/model_saving_and_loading/#loadmodel-
function"&gt;documentation&lt;/a&gt;):&lt;/p&gt;
&lt;blockquote&gt;
&lt;p&gt;safe_mode: Boolean, whether to disallow unsafe lambda deserialization.
When safe_mode=False, loading an object has the potential to trigger arbitrary
code execution. This argument is only applicable to the TF-Keras v3 model
format. Defaults to True.&lt;/p&gt;
&lt;/blockquote&gt;
&lt;p&gt;This is the behavior of version 2.13 and later of the Keras API: an
exception will be raised in a program that attempts to load a model with Lambda
layers stored in version 3 of the format. This check, however, does not exist
in the prior versions of the API. Nor is the check performed on models that
have been stored using earlier versions of the Keras serialization format
(i.e., v2 SavedModel, legacy H5).&lt;/p&gt;
&lt;p&gt;This means systems incorporating older versions of the Keras code base
prior to versions 2.13 may be susceptible to running arbitrary code when
loading older versions of Tensorflow-based models.&lt;/p&gt;
&lt;h4 id="similarity-to-other-frameworks-with-code-injection-
vulnerabilities"&gt;Similarity to other frameworks with code injection
vulnerabilities&lt;/h4&gt;
&lt;p&gt;The code injection vulnerability in the Keras 2 API is an example of a
common security weakness in systems that provide a mechanism for packaging data
together with code. For example, the security issues associated with the Pickle
mechanism in the standard Python library are well documented, and arise because
the Pickle format includes a mechanism for serializing code inline with its
data. &lt;/p&gt;
&lt;h4 id="explicit-versus-implicit-security-policy"&gt;Explicit versus
implicit security policy&lt;/h4&gt;
&lt;p&gt;The TensorFlow security documentation at &lt;a href="https://
github.com/tensorflow/tensorflow/blob/master/SECURITY.md)"&gt;https://
github.com/tensorflow/tensorflow/blob/master/SECURITY.md)&lt;/a&gt; includes a
specific warning about the fact that models are not just data, and makes a
statement about the expectations of developers in the TensorFlow development
community:&lt;/p&gt;
&lt;blockquote&gt;
&lt;p&gt;&lt;strong&gt;Since models are practically programs that TensorFlow
executes, using untrusted models or graphs is equivalent to running untrusted
code.&lt;/strong&gt; (emphasis in &lt;a href="https://github.com/tensorflow/
tensorflow/blob/2fe6b745ea90276b17b28f30d076bc9447918fd7/
SECURITY.md"&gt;earlier version&lt;/a&gt;) &lt;/p&gt;
&lt;/blockquote&gt;
&lt;p&gt;The implications of that statement are not necessarily widely
understood by all developers of TensorFlow-based systems.The last few years has
seen rapid growth in the community of developers building AI/ML-based systems,
and publishing pretrained models through community hubs like huggingface (&lt;a
href="https://huggingface.co/"&gt;https://huggingface.co/&lt;/a&gt;) and kaggle
```

```
(&lt;a href="https://www.kaggle.com"&gt;https://www.kaggle.com&lt;/a&gt;). It
is not clear that all members of this new community understand the potential
risk posed by a third-party model, and may (incorrectly) trust that a model
loaded using a trusted library should only execute code that is included in
that library. Moreover, a user may also assume that a pretrained model, once
loaded, will only execute included code whose purpose is to compute a
prediction and not exhibit any side effects outside of those required for those
calculations (e.g., that a model will not include code to communicate with a
network). &lt;/p&gt;
&lt;p&gt;To the degree possible, AI/ML framework developers and model
distributors should strive to align the explicit security policy and the
corresponding implementation to be consistent with the implicit security policy
implied by these assumptions.&lt;/p&gt;
&lt;h3 id="impact"&gt;Impact&lt;/h3&gt;
&lt;p&gt;Loading third-party models built using Keras could result in arbitrary
untrusted code running at the privilege level of the ML application
environment.&lt;/p&gt;
&lt;h3 id="solution"&gt;Solution&lt;/h3&gt;
&lt;p&gt;Upgrade to Keras 2.13 or later. When loading models, ensure the
&lt;code&gt;safe_mode&lt;/code&gt; parameter is not set to
&lt;code&gt;False&lt;/code&gt; (per &lt;a href="https://keras.io/api/models/
model_saving_apis/model_saving_and_loading"&gt;https://keras.io/api/models/
model_saving_apis/model_saving_and_loading&lt;/a&gt;, it is
&lt;code&gt;True&lt;/code&gt; by default). Note: An upgrade of Keras may
require dependencies upgrade, learn more at https://keras.io/getting_started/
&lt;/p&gt;
&lt;p&gt;If running pre-2.13 applications in a &lt;a href="https://github.com/
tensorflow/tensorflow/blob/master/SECURITY.md"&gt;sandbox&lt;/a&gt;, ensure no
assets of value are in scope of the running application to minimize potential
for data exfiltration.&lt;/p&gt;
&lt;h4 id="advice-for-model-users"&gt;Advice for Model Users&lt;/h4&gt;
&lt;p&gt;Model users should only use models developed and distributed by
trusted sources, and should always verify the behavior of models before
deployment. They should  follow the same development and deployment best
practices to applications that integrate ML models as they would to any
application incorporating any third party component. Developers should upgrade
to the latest versions of the Keras package practical (v2.13+ or v3.0+), and
use version 3 of the Keras serialization format to both load third-party models
and save any subsequent modifications.&lt;/p&gt;
&lt;h4 id="advice-for-model-aggregators"&gt;Advice for Model Aggregators&lt;/
h4&gt;
&lt;p&gt;Model aggregators should distribute models based on the latest, safe
model formats when possible, and should incorporate scanning and introspection
features to identify models that include unsafe-to-deserialize features and
either to prevent them from being uploaded, or flag them so that model users
can perform additional due diligence. &lt;/p&gt;
&lt;h4 id="advice-for-model-creators"&gt;Advice for Model Creators&lt;/h4&gt;
&lt;p&gt;Model creators should upgrade to the latest versions of the Keras
package (v2.13+ or v3.0+). They should avoid the use of unsafe-to-deserialize
features in order to avoid the inadvertent introduction of security
vulnerabilities, and to encourage the adoption of standards that are less
```

```
susceptible to exploitation by malicious actors. Model creators should save
models using the latest version of formats (Keras v3 in the case of the Keras
package), and, when possible, give preference to formats that disallow the
serialization of models that include arbitrary code (i.e., code that the user
has not explicitly imported into the environment). Model developers should re-
use third-party base models with care, only building on models from trusted
sources.&lt;/p&gt;
&lt;h4 id="general-advice-for-framework-developers"&gt;General Advice for
Framework Developers&lt;/h4&gt;
&lt;p&gt;AI/ML-framework developers should avoid the use of naÃ¯ve language-
native serialization facilities (e.g., the Python &lt;code&gt;pickle&lt;/
code&gt; package has well-established security weaknesses, and should not be
used in sensitive applications).&lt;/p&gt;
&lt;p&gt;In cases where it's desirable to include a mechanism for embedding
code, restrict the code that can be executed by, for example: &lt;/p&gt;
&lt;ul&gt;
&lt;li&gt;disallow certain language features (e.g., &lt;code&gt;exec&lt;/
code&gt;)&lt;/li&gt;
&lt;li&gt;explicitly allow only a "safe" language subset&lt;/li&gt;
&lt;li&gt;provide a sandboxing mechanism (e.g., to prevent network access) to
minimize potential threats.&lt;/li&gt;
&lt;/ul&gt;
&lt;h3 id="acknowledgements"&gt;Acknowledgements&lt;/h3&gt;
&lt;p&gt;This document was written by Jeffrey Havrilla, Allen Householder,
Andrew Kompanek, and Ben Koo.&lt;/p&gt;
</summary><content type="html">
&lt;div class="row" id="content"&gt;
  &lt;div class="large-9 medium-9 columns"&gt;
    &lt;div class="blog-post"&gt;
      &lt;div class="row"&gt;
        &lt;div class="large-12 columns"&gt;

          &lt;h3 id="overview"&gt;Overview&lt;/h3&gt;
&lt;p&gt;Lambda Layers in third party TensorFlow-based Keras models allow
attackers to inject arbitrary code into versions built prior to Keras 2.13 that
may then unsafely run with the same permissions as the running application. For
example, an attacker could use this feature to trojanize a popular model, save
it, and redistribute it, tainting the supply chain of dependent AI/ML
applications. &lt;/p&gt;
&lt;h3 id="description"&gt;Description&lt;/h3&gt;
&lt;p&gt;TensorFlow is a widely-used open-source software library for building
machine learning and artificial intelligence applications. The Keras framework,
implemented in Python, is a high-level interface to TensorFlow  that provides a
wide variety of features for the design, training, validation and packaging of
ML models. Keras provides an API for building neural networks from building
blocks called Layers. One such Layer type is a Lambda layer that allows a
developer to add arbitrary Python code to a model in the form of a lambda
function (an anonymous, unnamed function). Using the
&lt;code&gt;Model.save()&lt;/code&gt; or &lt;code&gt;save_model()&lt;/code&gt;
&lt;a href="https://keras.io/api/models/model_saving_apis/
model_saving_and_loading/#save_model-function"&gt;method&lt;/a&gt;, a developer
```

```
can then save a model that includes this code.&lt;/p&gt;
&lt;p&gt;The Keras 2 documentation for the &lt;code&gt;Model.load_model()&lt;/
code&gt; &lt;a href="https://keras.io/2.16/api/models/model_saving_apis/
model_saving_and_loading/#load_model-function"&gt;method&lt;/a&gt; describes a
mechanism for disallowing the loading of a native version 3 Keras model
(&lt;code&gt;.keras&lt;/code&gt; file) that includes a Lambda layer when
setting &lt;code&gt;safe_mode&lt;/code&gt; (&lt;a href="https://keras.io/api/
models/model_saving_apis/model_saving_and_loading/#loadmodel-
function"&gt;documentation&lt;/a&gt;):&lt;/p&gt;
&lt;blockquote&gt;
&lt;p&gt;safe_mode: Boolean, whether to disallow unsafe lambda deserialization.
When safe_mode=False, loading an object has the potential to trigger arbitrary
code execution. This argument is only applicable to the TF-Keras v3 model
format. Defaults to True.&lt;/p&gt;
&lt;/blockquote&gt;
&lt;p&gt;This is the behavior of version 2.13 and later of the Keras API: an
exception will be raised in a program that attempts to load a model with Lambda
layers stored in version 3 of the format. This check, however, does not exist
in the prior versions of the API. Nor is the check performed on models that
have been stored using earlier versions of the Keras serialization format
(i.e., v2 SavedModel, legacy H5).&lt;/p&gt;
&lt;p&gt;This means systems incorporating older versions of the Keras code base
prior to versions 2.13 may be susceptible to running arbitrary code when
loading older versions of Tensorflow-based models.&lt;/p&gt;
&lt;h4 id="similarity-to-other-frameworks-with-code-injection-
vulnerabilities"&gt;Similarity to other frameworks with code injection
vulnerabilities&lt;/h4&gt;
&lt;p&gt;The code injection vulnerability in the Keras 2 API is an example of a
common security weakness in systems that provide a mechanism for packaging data
together with code. For example, the security issues associated with the Pickle
mechanism in the standard Python library are well documented, and arise because
the Pickle format includes a mechanism for serializing code inline with its
data. &lt;/p&gt;
&lt;h4 id="explicit-versus-implicit-security-policy"&gt;Explicit versus
implicit security policy&lt;/h4&gt;
&lt;p&gt;The TensorFlow security documentation at &lt;a href="https://
github.com/tensorflow/tensorflow/blob/master/SECURITY.md)"&gt;https://
github.com/tensorflow/tensorflow/blob/master/SECURITY.md)&lt;/a&gt; includes a
specific warning about the fact that models are not just data, and makes a
statement about the expectations of developers in the TensorFlow development
community:&lt;/p&gt;
&lt;blockquote&gt;
&lt;p&gt;&lt;strong&gt;Since models are practically programs that TensorFlow
executes, using untrusted models or graphs is equivalent to running untrusted
code.&lt;/strong&gt; (emphasis in &lt;a href="https://github.com/tensorflow/
tensorflow/blob/2fe6b745ea90276b17b28f30d076bc9447918fd7/
SECURITY.md"&gt;earlier version&lt;/a&gt;) &lt;/p&gt;
&lt;/blockquote&gt;
&lt;p&gt;The implications of that statement are not necessarily widely
understood by all developers of TensorFlow-based systems.The last few years has
seen rapid growth in the community of developers building AI/ML-based systems,
and publishing pretrained models through community hubs like huggingface (&lt;a
href="https://huggingface.co/"&gt;https://huggingface.co/&lt;/a&gt;) and kaggle
(&lt;a href="https://www.kaggle.com"&gt;https://www.kaggle.com&lt;/a&gt;). It
is not clear that all members of this new community understand the potential
risk posed by a third-party model, and may (incorrectly) trust that a model
loaded using a trusted library should only execute code that is included in
that library. Moreover, a user may also assume that a pretrained model, once
loaded, will only execute included code whose purpose is to compute a
prediction and not exhibit any side effects outside of those required for those
calculations (e.g., that a model will not include code to communicate with a
network). &lt;/p&gt;
&lt;p&gt;To the degree possible, AI/ML framework developers and model
distributors should strive to align the explicit security policy and the
corresponding implementation to be consistent with the implicit security policy
implied by these assumptions.&lt;/p&gt;
&lt;h3 id="impact"&gt;Impact&lt;/h3&gt;
&lt;p&gt;Loading third-party models built using Keras could result in arbitrary
untrusted code running at the privilege level of the ML application
environment.&lt;/p&gt;
&lt;h3 id="solution"&gt;Solution&lt;/h3&gt;
&lt;p&gt;Upgrade to Keras 2.13 or later. When loading models, ensure the
&lt;code&gt;safe_mode&lt;/code&gt; parameter is not set to
&lt;code&gt;False&lt;/code&gt; (per &lt;a href="https://keras.io/api/models/
model_saving_apis/model_saving_and_loading"&gt;https://keras.io/api/models/
model_saving_apis/model_saving_and_loading&lt;/a&gt;, it is
&lt;code&gt;True&lt;/code&gt; by default). Note: An upgrade of Keras may
require dependencies upgrade, learn more at https://keras.io/getting_started/
&lt;/p&gt;
&lt;p&gt;If running pre-2.13 applications in a &lt;a href="https://github.com/
tensorflow/tensorflow/blob/master/SECURITY.md"&gt;sandbox&lt;/a&gt;, ensure no
assets of value are in scope of the running application to minimize potential
for data exfiltration.&lt;/p&gt;
&lt;h4 id="advice-for-model-users"&gt;Advice for Model Users&lt;/h4&gt;
&lt;p&gt;Model users should only use models developed and distributed by
trusted sources, and should always verify the behavior of models before
deployment. They should  follow the same development and deployment best
practices to applications that integrate ML models as they would to any
application incorporating any third party component. Developers should upgrade
to the latest versions of the Keras package practical (v2.13+ or v3.0+), and
use version 3 of the Keras serialization format to both load third-party models
and save any subsequent modifications.&lt;/p&gt;
&lt;h4 id="advice-for-model-aggregators"&gt;Advice for Model Aggregators&lt;/
h4&gt;
&lt;p&gt;Model aggregators should distribute models based on the latest, safe
model formats when possible, and should incorporate scanning and introspection
features to identify models that include unsafe-to-deserialize features and
either to prevent them from being uploaded, or flag them so that model users
can perform additional due diligence. &lt;/p&gt;
&lt;h4 id="advice-for-model-creators"&gt;Advice for Model Creators&lt;/h4&gt;
&lt;p&gt;Model creators should upgrade to the latest versions of the Keras
package (v2.13+ or v3.0+). They should avoid the use of unsafe-to-deserialize
```

```
features in order to avoid the inadvertent introduction of security
vulnerabilities, and to encourage the adoption of standards that are less
susceptible to exploitation by malicious actors. Model creators should save
models using the latest version of formats (Keras v3 in the case of the Keras
package), and, when possible, give preference to formats that disallow the
serialization of models that include arbitrary code (i.e., code that the user
has not explicitly imported into the environment). Model developers should re-
use third-party base models with care, only building on models from trusted
sources.&lt;/p&gt;
&lt;h4 id="general-advice-for-framework-developers"&gt;General Advice for
Framework Developers&lt;/h4&gt;
&lt;p&gt;AI/ML-framework developers should avoid the use of naÃ¯ve language-
native serialization facilities (e.g., the Python &lt;code&gt;pickle&lt;/
code&gt; package has well-established security weaknesses, and should not be
used in sensitive applications).&lt;/p&gt;
&lt;p&gt;In cases where it's desirable to include a mechanism for embedding
code, restrict the code that can be executed by, for example: &lt;/p&gt;
&lt;ul&gt;
&lt;li&gt;disallow certain language features (e.g., &lt;code&gt;exec&lt;/
code&gt;)&lt;/li&gt;
&lt;li&gt;explicitly allow only a "safe" language subset&lt;/li&gt;
&lt;li&gt;provide a sandboxing mechanism (e.g., to prevent network access) to
minimize potential threats.&lt;/li&gt;
&lt;/ul&gt;
&lt;h3 id="acknowledgements"&gt;Acknowledgements&lt;/h3&gt;
&lt;p&gt;This document was written by Jeffrey Havrilla, Allen Householder,
Andrew Kompanek, and Ben Koo.&lt;/p&gt;

        &lt;/div&gt;
      &lt;/div&gt;
      &lt;div class="row"&gt;
        &lt;div class="large-12 columns"&gt;
          &lt;h3&gt; Vendor Information &lt;/h3&gt;

          &lt;div id="vendorinfo"&gt;
            One or more vendors are listed for this advisory. Please reference
the full report for more information.
          &lt;/div&gt;

        &lt;/div&gt;
      &lt;/div&gt;
      &lt;br/&gt;


      &lt;div class="row"&gt;
        &lt;div class="large-12 columns"&gt;
          &lt;h3&gt; References &lt;/h3&gt;
          &lt;ul&gt;
```

```
            &lt;li&gt;&lt;a href="https://keras.io/api/models/
model_saving_apis/model_saving_and_loading/#loadmodel-function"
class="vulreflink safereflink" target="_blank" rel="noopener"&gt;https://
keras.io/api/models/model_saving_apis/model_saving_and_loading/#loadmodel-
function&lt;/a&gt;&lt;/li&gt;



            &lt;li&gt;&lt;a href="https://github.com/tensorflow/tensorflow/
blob/master/SECURITY.md" class="vulreflink safereflink" target="_blank"
rel="noopener"&gt;https://github.com/tensorflow/tensorflow/blob/master/
SECURITY.md&lt;/a&gt;&lt;/li&gt;



            &lt;li&gt;&lt;a href="https://github.com/Azure/counterfit/wiki/
Abusing-ML-model-file-formats-to-create-malware-on-AI-systems:-A-proof-of-
concept" class="vulreflink safereflink" target="_blank"
rel="noopener"&gt;https://github.com/Azure/counterfit/wiki/Abusing-ML-model-
file-formats-to-create-malware-on-AI-systems:-A-proof-of-concept&lt;/a&gt;&lt;/
li&gt;



            &lt;li&gt;&lt;a href="https://splint.gitbook.io/cyberblog/security-
research/tensorflow-remote-code-execution-with-malicious-model"
class="vulreflink safereflink" target="_blank" rel="noopener"&gt;https://
splint.gitbook.io/cyberblog/security-research/tensorflow-remote-code-execution-
with-malicious-model&lt;/a&gt;&lt;/li&gt;



            &lt;li&gt;&lt;a href="https://5stars217.github.io/2023-03-30-on-
malicious-models/" class="vulreflink safereflink" target="_blank"
rel="noopener"&gt;https://5stars217.github.io/2023-03-30-on-malicious-models/
&lt;/a&gt;&lt;/li&gt;



            &lt;li&gt;&lt;a href="https://hiddenlayer.com/research/models-are-
code/" class="vulreflink safereflink" target="_blank"
rel="noopener"&gt;https://hiddenlayer.com/research/models-are-code/&lt;/
a&gt;&lt;/li&gt;



          &lt;/ul&gt;
        &lt;/div&gt;
      &lt;/div&gt;
      &lt;h3&gt;Other Information&lt;/h3&gt;
        &lt;div class="vulcontent"&gt;
          &lt;table class="unstriped"&gt;
            &lt;tbody&gt;

              &lt;tr&gt;
                &lt;td width="200"&gt;&lt;b&gt;CVE IDs:&lt;/b&gt;&lt;/td&gt;
                &lt;td&gt;


                &lt;a href="http://web.nvd.nist.gov/view/vuln/detail?
vulnId=2024-3660"&gt;CVE-2024-3660  &lt;/a&gt;


                &lt;/td&gt;
              &lt;/tr&gt;

              &lt;tr&gt;
                &lt;td&gt;
                  &lt;b&gt;Date Public:&lt;/b&gt;
                &lt;/td&gt;
                &lt;td&gt;2024-02-23&lt;/td&gt;
              &lt;/tr&gt;
              &lt;tr&gt;
                &lt;td&gt;&lt;b&gt;Date First Published:&lt;/b&gt;&lt;/td&gt;
                &lt;td id="datefirstpublished"&gt;2024-04-16&lt;/td&gt;
              &lt;/tr&gt;
              &lt;tr&gt;
                &lt;td&gt;&lt;b&gt;Date Last Updated: &lt;/b&gt;&lt;/td&gt;
                &lt;td&gt;2024-04-18 18:47 UTC&lt;/td&gt;
              &lt;/tr&gt;
              &lt;tr&gt;
                &lt;td&gt;&lt;b&gt;Document Revision: &lt;/b&gt;&lt;/td&gt;
                &lt;td&gt;4 &lt;/td&gt;
              &lt;/tr&gt;
            &lt;/tbody&gt;
          &lt;/table&gt;
        &lt;/div&gt;
    &lt;/div&gt;
  &lt;/div&gt;
  &lt;div class="large-3 medium-3 columns" data-sticky-container&gt;
  &lt;div class="sticky" data-sticky data-anchor="content"&gt;
    &lt;div class="sidebar-links"&gt;
      &lt;ul class="menu vertical"&gt;
        &lt;li&gt;&lt;a href="https://vuls.cert.org/confluence/display/VIN/
Vulnerability+Note+Help" target="_blank" rel="noopener"&gt;About vulnerability
notes&lt;/a&gt;&lt;/li&gt;
        &lt;li&gt;&lt;a href="mailto:cert@cert.org?Subject=VU%23253266
Feedback"&gt;Contact us about this vulnerability&lt;/a&gt;&lt;/li&gt;
        &lt;li&gt;&lt;a href="https://vuls.cert.org/confluence/display/VIN/
```

```
Case+Handling#CaseHandling-Givingavendorstatusandstatement" target="_blank"
&gt;Provide a vendor statement&lt;/a&gt;&lt;/li&gt;
      &lt;/ul&gt;
    &lt;/div&gt;
  &lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
</content></entry><entry><title>VU#123335: Multiple programming languages fail
to escape arguments properly in Microsoft Windows</title><link href="https://
kb.cert.org/vuls/id/123335" rel="alternate"/
><published>2024-04-10T15:13:48.856313+00:00</
published><updated>2024-04-18T13:33:28.597485+00:00</updated><id>https://
kb.cert.org/vuls/id/123335</id><summary type="html">
&lt;h2 id="overview"&gt;Overview&lt;/h2&gt;
&lt;p&gt;Various programming languages lack proper validation mechanisms for
commands and in some cases also fail to escape arguments correctly when
invoking commands within a Microsoft Windows environment. The command injection
vulnerability in these programming languages, when running on Windows, allows
attackers to execute arbitrary code disguised as arguments to the command. This
vulnerability may also affect the application that executes commands without
specifying the file extension.&lt;/p&gt;
&lt;h2 id="description"&gt;Description&lt;/h2&gt;
&lt;p&gt;Programming languages typically provide a way to execute commands (for
e.g., os/exec in Golang) on the operating system to facilitate interaction with
the OS. Typically, the programming languages also allow for passing
&lt;code&gt;arguments&lt;/code&gt; which are considered data (or variables) for
the command to be executed. The arguments themselves are expected to be not
executable and the command is expected to be executed along with properly
escaped arguments, as inputs to the command. Microsoft Windows typically
processes these commands using a &lt;code&gt;CreateProcess&lt;/code&gt;
function that spawns a &lt;code&gt;cmd.exe&lt;/code&gt; for execution of the
command. Microsoft Windows has documented some of the concerns related to how
these should be properly escaped before execution as early as 2011. See &lt;a
href="https://learn.microsoft.com/en-us/archive/blogs/
twistylittlepassagesallalike/everyone-quotes-command-line-arguments-the-wrong-
way"&gt;https://learn.microsoft.com/en-us/archive/blogs/
twistylittlepassagesallalike/everyone-quotes-command-line-arguments-the-wrong-
way&lt;/a&gt;. &lt;/p&gt;
&lt;p&gt;A vulnerability was discovered in the way multiple programming
languages fail to properly escape the arguments in a Microsoft Windows command
execution environment.  This can lead confusion at execution time where an
expected argument for a command could be executed as another command itself. An
attacker with knowledge of the programming language can carefully craft inputs
that will be processed by the compiled program as commands. This unexpected
behavior is due to lack of neutralization of arguments by the programming
language (or its command execution module) that initiates a Windows execution
environment.  The researcher has found multiple programming languages, and
their command execution modules fail to perform such sanitization and/or
validation before processing these in their runtime environment. &lt;/p&gt;
&lt;h2 id="impact"&gt;Impact&lt;/h2&gt;
&lt;p&gt;Successful exploitation of this vulnerability permits an attacker to
```

```
execute arbitrary commands. The complete impact of this vulnerability depends
on the implementation that uses a vulnerable programming language or such a
vulnerable module.&lt;/p&gt;
&lt;h2 id="solution"&gt;Solution&lt;/h2&gt;
&lt;h4 id="updating-the-runtime-environment"&gt;Updating the runtime
environment&lt;/h4&gt;
&lt;p&gt;Please visit the Vendor Information section so see if your programming
language Vendor has released the patch for this vulnerability and update the
runtime environment that can prevent abuse of this vulnerability. &lt;/p&gt;
&lt;h4 id="update-the-programs-and-escape-manually"&gt;Update the programs and
escape manually&lt;/h4&gt;
&lt;p&gt;If the runtime of your application doesn't provide a patch for this
vulnerability and you want to execute batch files with user-controlled
arguments, you will need to perform the escaping and neutralization of the data
to prevent any intended command execution. &lt;/p&gt;
&lt;p&gt;Security researcher has more detailed information in the &lt;a
href="https://flatt.tech/research/posts/batbadbut-you-cant-securely-execute-
commands-on-windows/"&gt;blog post&lt;/a&gt; which provides details on specific
languages that were identified and their Status. &lt;/p&gt;
&lt;h2 id="acknowledgements"&gt;Acknowledgements&lt;/h2&gt;
&lt;p&gt;Thanks to the reporter, &lt;a href="https://flatt.tech/research/posts/
batbadbut-you-cant-securely-execute-commands-on-windows/"&gt;RyotaK&lt;/
a&gt;.This document was written by Timur Snoke.&lt;/p&gt;
</summary><content type="html">
```

ThreatQuotient provides the following default mapping for this feed:

FEED DATA PATH	THREATQ ENTITY	THREATQ OBJECT TYPE OR ATTRIBUTE KEY	PUBLISHED DATE	EXAMPLES	NOTES	
`.entry[].title`, `.entry[].title['#text']`	Report.Value	N/A	N/A	`.updated`	`[Bug 15103] CVE-2022-1615 [SECURITY] GnuTLS gnutls_rnd() can fail and give predictable random values`	N/A
`.entry[].content`, `.entry[].content['#text']`	Report.Description	N/A	N/A	N/A	`<HTML>`	N/A
`.entry[].summary`, `.entry[].summary['#text']`	Report.Description	N/A	N/A	N/A	`<HTML>`	N/A
`.entry[].author.name`	Report.Attribute	Author	N/A	`.updated`	`OpenSSL Foundation, Inc.`	N/A
`.entry[].author.name`	Report.Source	N/A	N/A	N/A	`OpenSSL Foundation, Inc.`	N/A
N/A	Report.Source	N/A	N/A	N/A	`Atom Feed Reader`	Source applied by default
`.entry[].link['#href']`	Report.Attribute	External Reference	N/A	`.updated`	N/A	N/A
`.entry[].id`	Report.Attribute	ID	N/A	`.updated`	N/A	Only ingested if different from the `link['#href']` field.
`<Entry Title/Content>`	Report.Indicator	CVE	N/A	`.updated`	N/A	Entries are parsed for CVE IDs.

Average Feed Run

Object counts and Feed runtime are supplied as generalities only - objects returned by a provider can differ based on credential configurations and Feed runtime may vary based on system resources and load.

METRIC	RESULT
Run Time	1 minute
Reports	90
Report Attributes	200
Indicators	69

Known Issues / Limitations

- The feed utilizes **since** and **until** dates to make sure entries are not re-ingested if they haven't been updated. If you need to ingest historical posts, run the feed manually by setting the **since** date back.

Change Log

- **Version 1.0.0**
    - Initial release