D5.6B TAIBOM Threats and Attacks Security Scenario Testing Against Selected Use Cases

Report

March 2025

COPPER HORSE

Contents

INTRODUCTION

The following tests represent emergent threats based on an exercise conducted by Copper Horse. In this report, the researchers test two use cases models which were specifically created for TAIBOM (Trustable AI Bill of Materials). The use cases are used to validate the implementation of TAIBOM by asking specific questions that allows the testing of hypotheses which satisfy TAIBOM goals. This works both in a practical environment and against the software implementations and can be targeted against the specific elements where the software is expected to fail.

The process via which these scenarios were selected originated in a very long list of threats which were generated as a part of a 'blue sky' thinking exercise, the list was then culled of the impractical/implausible to just a long list. A short list of threats that can be demonstrated as 'abuse cases' was then created from the long list.

In both Copper Horse's use cases, there are commonalities between the AI models in the 'class' of attack, given that both the targets are AI vision models. However, the context is different in how a successful attack manifests itself. For example, the changing of a street sign in a dataset from 20mph to 100mph or to 'No Entry' could have a direct effect on the user's or cars actions. The same method against the Short Writing model may be used to invert the semantic meaning of the text, causing the reader to misinterpret the intent of the text. One example shown later is to change 'Heaven' to 'Hell'. In this case, rather than presenting the result to the user for them to interpret themselves, the inference engine could be used automatically to interpret the results instead. In this instance the implications may not be so severe, however for the road sign recognition use case, this automated decision making may lead to very negative consequences with severe implications for human safety. In a machine-to-machine environment, simply being able to change the state of a system from '1' to '0', may in some circumstances kill someone. In the same way, the deliberate manipulation of street signs may cause a car to accelerate into a pedestrian area or on a dangerous corner[1].

[1] https://securing.ai/ai-security/semantic-adversarial-attacks/

USE CASE 1: SIGN DETECTION

Threat Table

Below is the testing table for the security threats for the Sign Detection model. Over 145 threat scenarios were identified, and 16 of the most plausible attacks have been implemented and carried out. TAIBOM is then deployed in various ways to help detect and mitigate these attacks as much as possible.

Threat		Description
1	Tampering with inference engine GUI elements	Modifications to the source code of the inference engine are altered to make a correctly working model look like it's performing incorrectly by displaying the wrong classification information in the GUI
2	Altering rescaling parameters	By altering parameters, the model fails to learn proper feature representations as it receives raw, unnormalised data
3	Manipulating model file configuration	The .keras file is a zip file, which contains two .json configuration files, and a .h5 file with the model weights. The former can be edited to interact with third-party, potentially malicious, software
4	Introducing bias in model weights	The model weights can be consciously altered to modify the model output, leading to a biased inferencing
5	Changing colour scheme in inferencing (demo)	Changing images colour schemes (RGB, BGR, HSV, grayscale) during inferencing leads to disruption due to incompatible expected input formats
6	Modifying (You Only Look Once) YOLO training dataset bounding boxes	Object detection training data coordinates of bounding boxes gets modified or deleted, leading to complete misdetections
7	Modify training images extensions	Changing images extensions to non-recognised ones causes the modified images to not be included in the training, leading to biases and poor performance
8	Modifying learning rate values	Changing the learning rate to extreme values causes either very slow training of the model or a failure to converge
9	Introduce gradient clipping	Introducing gradient clipping constrains the gradient magnitudes to disrupt optimisation and degrade model performance
10	Modifying requirements to introduce a vulnerability	The requirements.txt for a model is altered to introduce a previously patched CVE in a third-party library to the system
11	Validate a TAIBOM created in the future	Date/Time on local machine is changed to create a TAIBOM on a date in the future, then local time restored to before signature creation
12	Using Steganography to hide a malware file inside of a training data image	Steganography tool is used to hide a txt file within the pixel data of a jpg image inside a dataset
13	Change inference labels output	Speed limit classification labels in the inference are altered to produce extreme results to break the inference of a model
14	Sign TAIBOM-datapack with a TAIBOM-code	Try to overwrite the identity token of a user to try to create a false identity that could be used to run TAIBOM commands

Threat		Description
15	Sign code as data TAIBOM and sign a dataset as code	This test is designed to probe the ability of TAIBOM of distinguishing between data and code tags.
16	Sign ai-system placing data in the code field	Tests will be performed with varying dataset sizes to test TAIBOM performance against dataset size

1. Tampering with inference engine GUI elements

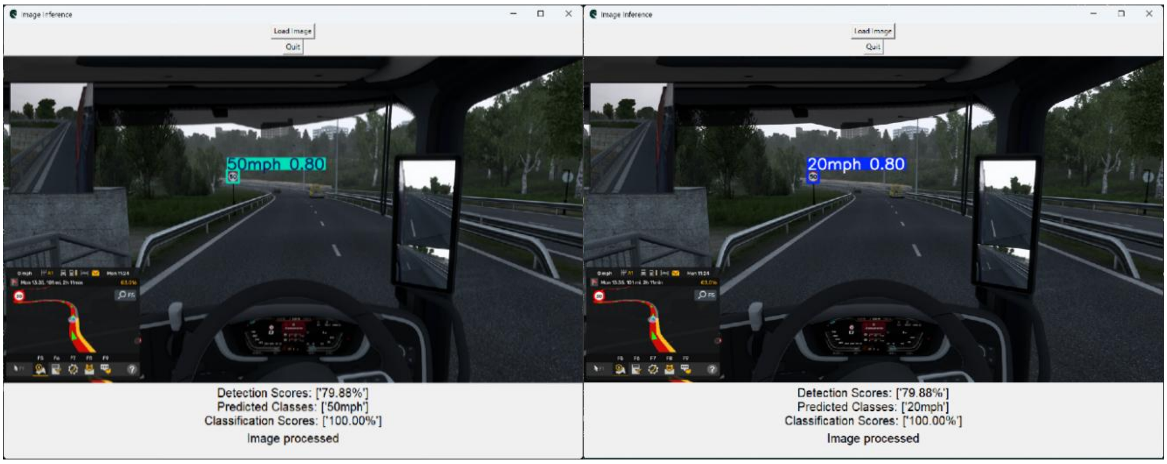
This test evaluates the capability of a TAIBOM to detect changes to the code in the inference script. The test is performed on **Release 8**.

The test is performed by modifying the confidence score output of the last layer of the second classification in the Convolutional Neural Network (CNN). The CNN is the model being used to classify speed limit signs.
The modification is made to the output predicted classes score:

```
classification = self.classification_model.predict(cropped_sign)[0]
```

is changed to:

```
classification = 1.0 - self.classification_model.predict(cropped_sign)[0]
```



```
Detection Scores: ['79.88%']
Predicted Classes: ['50mph']
Classification Scores: ['100.00%']
Image processed
```

```
Detection Scores: ['79.88%']
Predicted Classes: ['20mph']
Classification Scores: ['100.00%']
Image processed
```

The screenshots show the inference engine on a test image, before and after the tampering (test image: ets2_20241009_104702_00.png).

TEST fails as expected

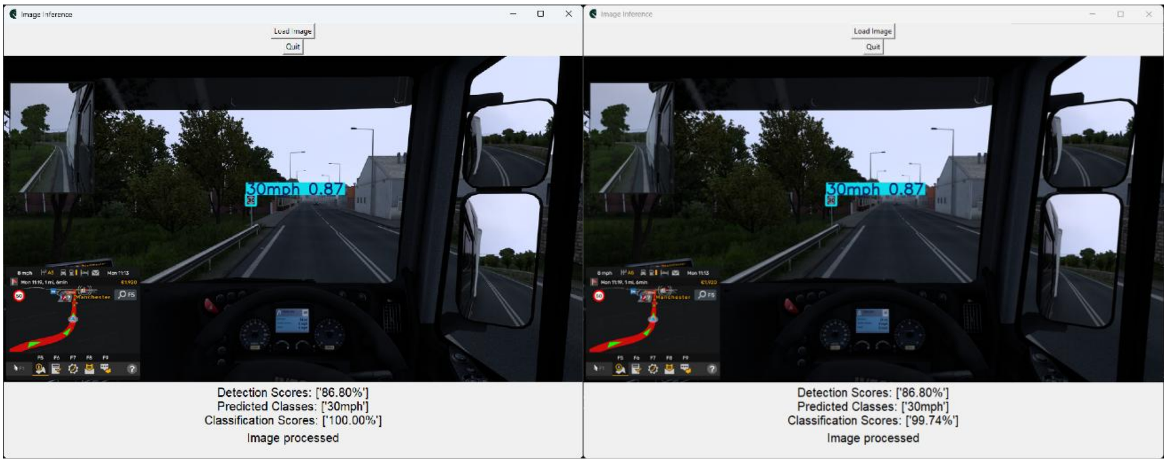
2. Altering rescaling parameters

This test evaluates the capability of a TAIBOM to detect changes in the model architecture. The test is performed on **Release 8**. This is done by modifying the scaling factor in the normalisation layer. The pixel values are rescaled internally to the model by a factor x4 smaller than the original:

```
rescale = tf.keras.layers.Rescaling(1./255)(resize)
```

is changed to:

```
rescale = tf.keras.layers.Rescaling(1./1020)(resize)
```



Detection Scores: ['86.80%']
Predicted Classes: ['30mph']
Classification Scores: ['100.00%']
Image processed

Detection Scores: ['86.80%']
Predicted Classes: ['30mph']
Classification Scores: ['99.74%']
Image processed

The screenshots show the inference engine on a test image, before and after the tampering (test image: ets2_20241009_102339_00.png).

While in both cases the results are correct, the classification scores are different, and slightly worse for the tampered model. The impact is not massive, as the effect of the change is mitigated by the following “Normalisation” layer in the model architecture.

TEST fails as expected

3. Manipulating model file configuration

This test evaluates the ability of a TAIBOM to detect changes made directly to a model's files

```
taibom data-taibom <identity_email> /home/ch/Documents/ETS2SignDetection-
main/TAIBOM/Release\ 10/inferencing/ --weights
```

This command has been applied to a folder containing (multiple) .keras model files. The TAIBOM hash created is able to detect any changes within the folder, including manual editing of the files within the .keras file (including .h5 saved weights file).

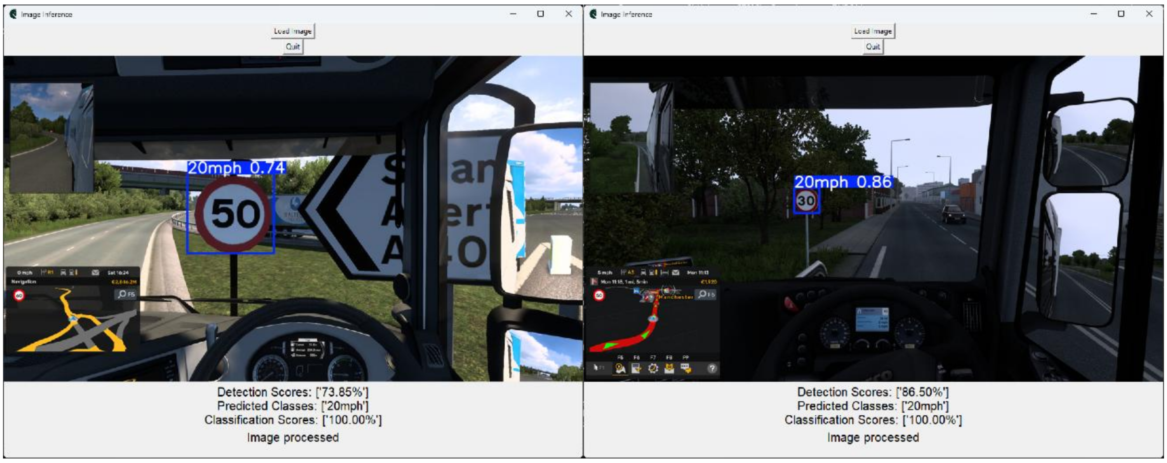
TEST fails as expected

4. Introducing bias in model weights

This test evaluates the capability of a TAIBOM to detect changes in the model weights. This test is performed on **Release 8**.

The model weights are consciously modified to introduce a bias in the inferencing engine. This is done with a python script which extracts the trained model weights, modifies them and overwriting the existing saved model weights.

Specifically for this case, the researchers add a large value to a specific element of the bias vector of the final output layer of the model. The class corresponding to the element of the bias vector that is modified with a large value is much more likely to be returned as output of the inferencing script. In this case, the bias vector element corresponding to the first class ("20mph") was increased.



```
Detection Scores: ['73.85%']
Predicted Classes: ['20mph']
Classification Scores: ['100.00%']
Image processed
```

```
Detection Scores: ['86.50%']
Predicted Classes: ['20mph']
Classification Scores: ['100.00%']
Image processed
```

According to the NIST paper ‘Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations’ (NIST AI 100-2 E2025)[2], this can be classified as a “white-box” attack as it assumes knowledge on the model architecture. On the other hand, for simple model architectures (i.e. when no customised layers or elements are included in the model), this attack can be performed also with little to no knowledge on the model, therefore classifying the attack as “black-box”.

```
taibom data-taibom <identity_email> /home/ch/Documents/ETS2SignDetection-main/TAIBOM/Release\ 8/inferencing/ --weights
```

TAIBOM is able to detect any change to the model weights contained in the .keras files.

TEST fails as expected

[2] https://csrc.nist.gov/pubs/ai/100/2/e2025/final

5. Changing colour scheme in inferencing

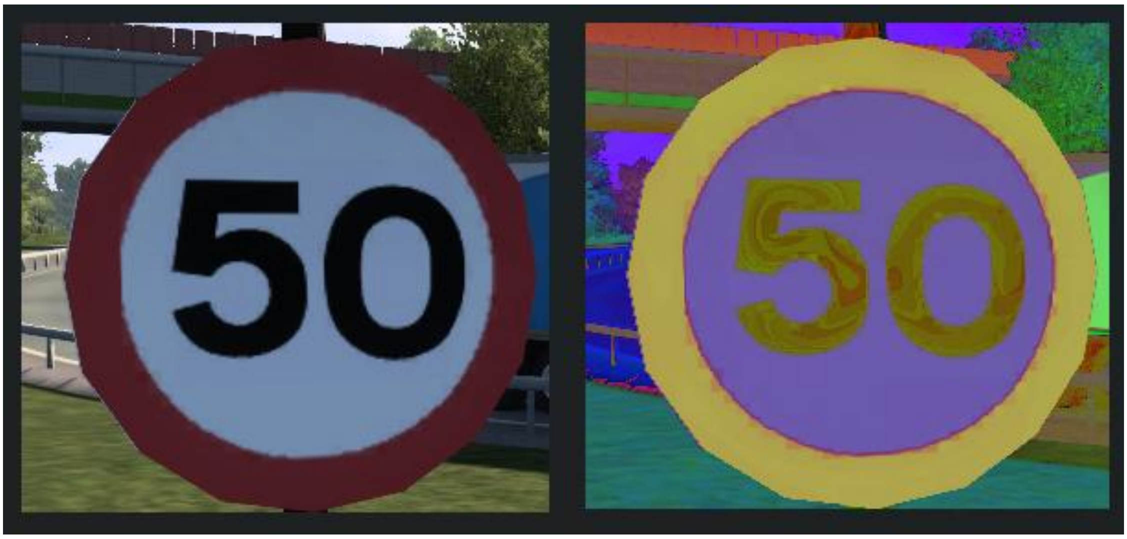
This test evaluates ability of a TAIBOM to detect changes in the inferencing engine. The test is performed on **Release 8**.

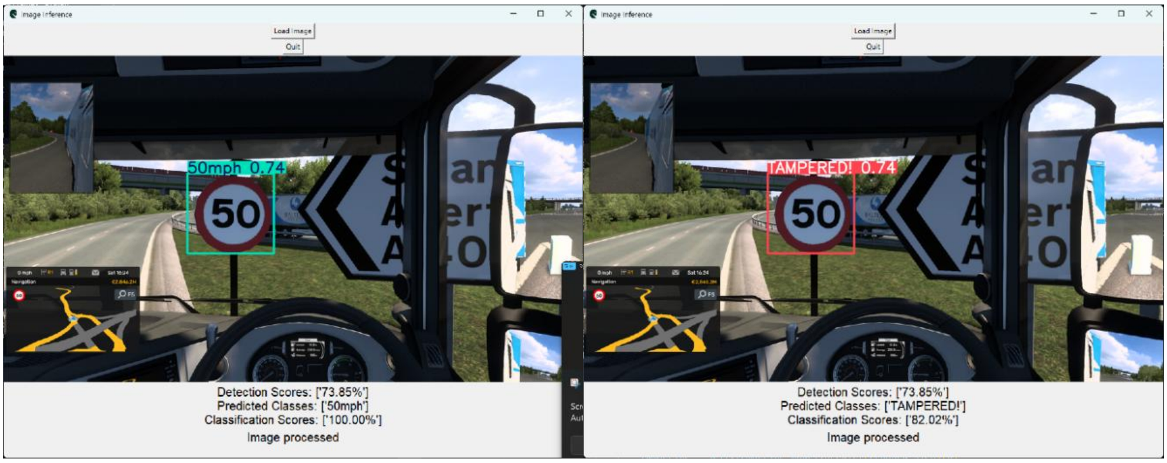
This is done by modifying the colour scheme of the (cropped) image of the speed limit sign passed to the classification networks after being detected by the object detection YOLO neural network:

```
cropped_sign = Image.fromarray(cv2.cvtColor(cropped_sign,
cv2.COLOR_BGR2RGB))
```

is changed to

```
cropped_sign = Image.fromarray(cv2.cvtColor(cropped_sign,
cv2.COLOR_BGR2HSV))
```





```
Detection Scores: ['73.85%']
Predicted Classes: ['50mph']
Classification Scores: ['100.00%']
Image processed
```

```
Detection Scores: ['73.85%']
Predicted Classes: ['TAMPERED!']
Classification Scores: ['82.02%']
Image processed
```

The screenshots show the inference engine on a test image, before and after the tampering (test image: /ETS2 in cab screen captures/ets2_20240809_144034_00.png), including the appearance of the cropped detected sign as it is passed to the CNNs.

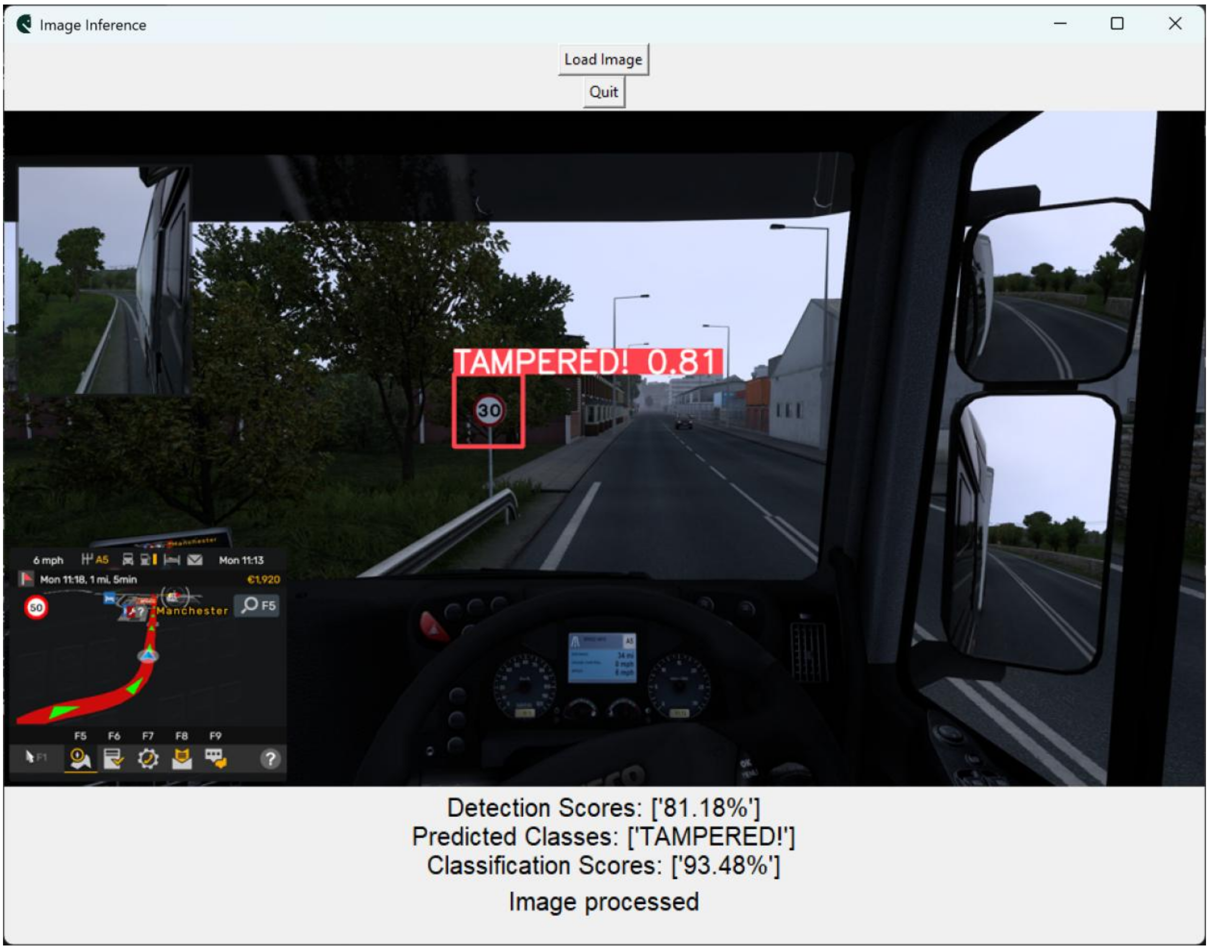
TEST fails as expected

6. Modifying YOLO training dataset bounding boxes

This test evaluates the capability of a TAIBOM to detect changes in the training dataset. The test is performed on **Release 8**.

This is done modifying the .txt files in the YOLO training dataset which contain the coordinates of the bounding boxes. Specifically, in this test the “height” and “width” values of each bounding box are multiplied by a factor 2. The YOLO object detection model is therefore trained to identify larger bounding boxes around the speed limit signs. The automatic modification of all the .txt files in the “YOLO_training” folder, is done through a python script.

This screenshot shows the inference engine on a test image, after the tampering (test image: /ETS2 in cab screen captures/ets2_20241009_102342_00.png). Note the larger bounding box around the detected sign. As the cropped image is passed to the CNNs, misclassifications occur for both CNNs.



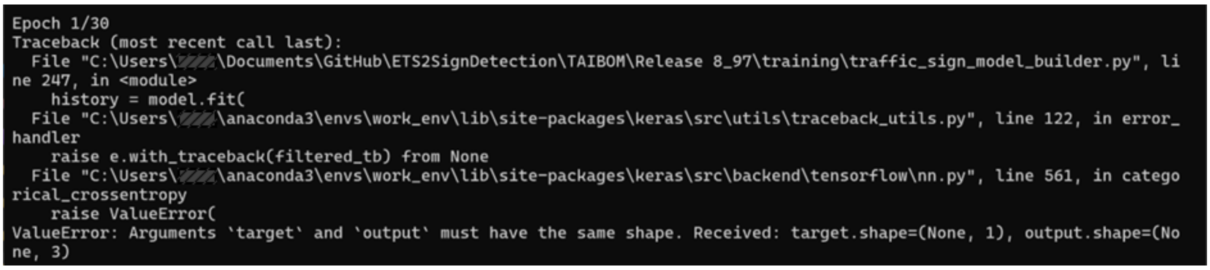
Detection Scores: ['81.18%']
Predicted Classes: ['TAMPERED!']
Classification Scores: ['93.48%']
Image processed

NOTE: trained model weights are also included in the modified “YOLO_training” folder.

TEST fails as expected

7. Modifying training images extensions

This test evaluates the capability of a TAIBOM to detect changes in the training dataset. The test is performed on **Release 8**.
This is done properly changing the extension of the training images from .jpg to .eps. The automatic conversion (conversion, not just renaming of the extension) of all the .jpg images in the “images” and “images_tampering” folders, is done through a python script.



```
Epoch 1/30
Traceback (most recent call last):
  File "C:\Users\[redacted]\Documents\GitHub\ETS2SignDetection\TAIBOM\Release 8_97\training\traffic_sign_model_builder.py", li
ne 247, in <module>
    history = model.fit(
  File "C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\utils\traceback_utils.py", line 122, in error_
handler
    raise e.with_traceback(filtered_tb) from None
  File "C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\backend\tensorflow\nn.py", line 561, in catego
rical_crossentropy
    raise ValueError(
ValueError: Arguments `target` and `output` must have the same shape. Received: target.shape=(None, 1), output.shape=(No
ne, 3)
```

The training is disrupted, as the shape of input images is not compatible with the expected shape of the first input layer of the model.

TEST fails as expected

8. Modifying learning rate values

This test evaluates the capability of a TAIBOM to detect changes in the model. The test is performed on **Release 8**.

This is done by enforcing a very large learning rate during the training of the CNNs.

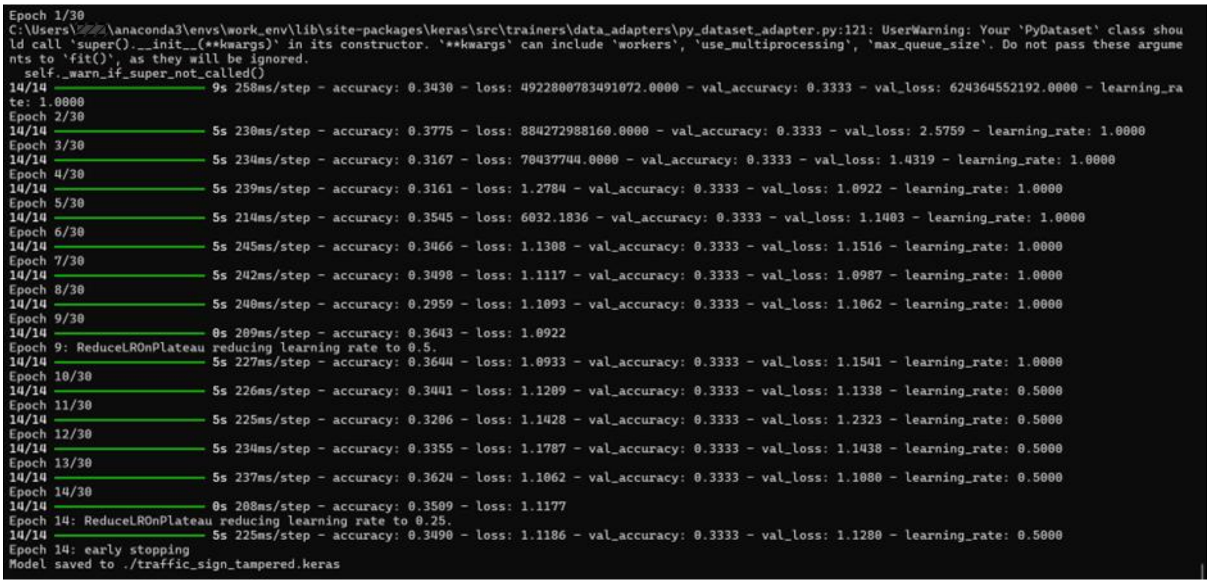
```
model.compile(optimizer=tf.keras.optimizers.Adam()…
```

is changed to

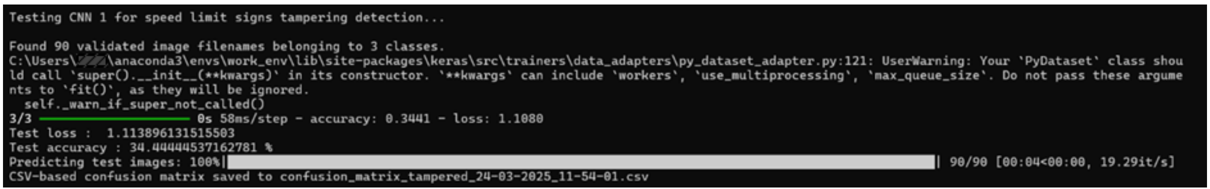
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1.0)…
```

The learning rate is therefore changed from the default value of 0.001, to a factor 1000 larger.

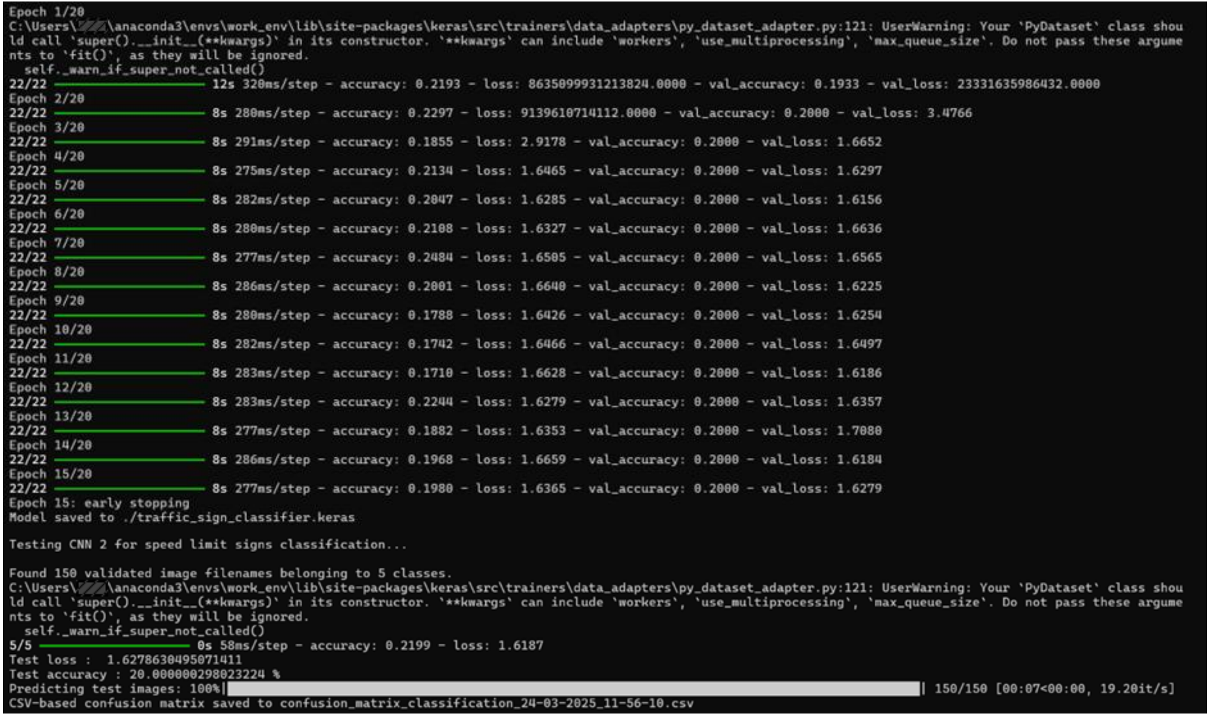
The training is seriously affected by the change, and the convergence is lost. This is visible monitoring the training progresses, as well as from the performances on the test data.



```
Epoch 1/30
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
14/14 ━━━━━━━━━━━━━━━━━━━━ 9s 258ms/step - accuracy: 0.3430 - loss: 4922800783491072.0000 - val_accuracy: 0.3333 - val_loss: 624364552192.0000 - learning_ra
te: 1.0000
Epoch 2/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 230ms/step - accuracy: 0.3775 - loss: 884272988160.0000 - val_accuracy: 0.3333 - val_loss: 2.5759 - learning_rate: 1.0000
Epoch 3/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 234ms/step - accuracy: 0.3167 - loss: 70437744.0000 - val_accuracy: 0.3333 - val_loss: 1.4319 - learning_rate: 1.0000
Epoch 4/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 239ms/step - accuracy: 0.3161 - loss: 1.2784 - val_accuracy: 0.3333 - val_loss: 1.0922 - learning_rate: 1.0000
Epoch 5/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 214ms/step - accuracy: 0.3545 - loss: 6032.1836 - val_accuracy: 0.3333 - val_loss: 1.1403 - learning_rate: 1.0000
Epoch 6/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 245ms/step - accuracy: 0.3466 - loss: 1.1308 - val_accuracy: 0.3333 - val_loss: 1.1516 - learning_rate: 1.0000
Epoch 7/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 242ms/step - accuracy: 0.3498 - loss: 1.1117 - val_accuracy: 0.3333 - val_loss: 1.0987 - learning_rate: 1.0000
Epoch 8/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 240ms/step - accuracy: 0.2959 - loss: 1.1093 - val_accuracy: 0.3333 - val_loss: 1.1062 - learning_rate: 1.0000
Epoch 9/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 209ms/step - accuracy: 0.3643 - loss: 1.0922
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.5.
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 227ms/step - accuracy: 0.3644 - loss: 1.0933 - val_accuracy: 0.3333 - val_loss: 1.1541 - learning_rate: 1.0000
Epoch 10/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 226ms/step - accuracy: 0.3441 - loss: 1.1209 - val_accuracy: 0.3333 - val_loss: 1.1338 - learning_rate: 0.5000
Epoch 11/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 225ms/step - accuracy: 0.3206 - loss: 1.1428 - val_accuracy: 0.3333 - val_loss: 1.2323 - learning_rate: 0.5000
Epoch 12/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 234ms/step - accuracy: 0.3355 - loss: 1.1787 - val_accuracy: 0.3333 - val_loss: 1.1438 - learning_rate: 0.5000
Epoch 13/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 237ms/step - accuracy: 0.3624 - loss: 1.1062 - val_accuracy: 0.3333 - val_loss: 1.1080 - learning_rate: 0.5000
Epoch 14/30
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step - accuracy: 0.3509 - loss: 1.1177
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.25.
14/14 ━━━━━━━━━━━━━━━━━━━━ 5s 225ms/step - accuracy: 0.3490 - loss: 1.1186 - val_accuracy: 0.3333 - val_loss: 1.1280 - learning_rate: 0.5000
Epoch 14: early stopping
Model saved to ./traffic_sign_tampered.keras
```



```
Testing CNN 1 for speed limit signs tampering detection...

Found 90 validated image filenames belonging to 3 classes.
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3441 - loss: 1.1080
Test loss :  1.113896131515503
Test accuracy : 34.44444537162781 %
Predicting test images: 100%|██████████████████████████████████████████████████████████████████████████████████████████████| 90/90 [00:04<00:00, 19.29it/s]
CSV-based confusion matrix saved to confusion_matrix_tampered_24-03-2025_11-54-01.csv
```



```
Epoch 1/20
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
22/22 ━━━━━━━━━━━━━━━━━━━━ 12s 320ms/step - accuracy: 0.2193 - loss: 8635099931213824.0000 - val_accuracy: 0.1933 - val_loss: 23331635986432.0000
Epoch 2/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 280ms/step - accuracy: 0.2297 - loss: 9139610714112.0000 - val_accuracy: 0.2000 - val_loss: 3.4766
Epoch 3/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 291ms/step - accuracy: 0.1855 - loss: 2.9178 - val_accuracy: 0.2000 - val_loss: 1.6652
Epoch 4/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 275ms/step - accuracy: 0.2134 - loss: 1.6465 - val_accuracy: 0.2000 - val_loss: 1.6297
Epoch 5/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 282ms/step - accuracy: 0.2047 - loss: 1.6285 - val_accuracy: 0.2000 - val_loss: 1.6156
Epoch 6/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 280ms/step - accuracy: 0.2108 - loss: 1.6327 - val_accuracy: 0.2000 - val_loss: 1.6636
Epoch 7/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 277ms/step - accuracy: 0.2484 - loss: 1.6505 - val_accuracy: 0.2000 - val_loss: 1.6565
Epoch 8/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 286ms/step - accuracy: 0.2001 - loss: 1.6640 - val_accuracy: 0.2000 - val_loss: 1.6225
Epoch 9/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 280ms/step - accuracy: 0.1788 - loss: 1.6426 - val_accuracy: 0.2000 - val_loss: 1.6254
Epoch 10/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 282ms/step - accuracy: 0.1742 - loss: 1.6466 - val_accuracy: 0.2000 - val_loss: 1.6497
Epoch 11/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 283ms/step - accuracy: 0.1710 - loss: 1.6628 - val_accuracy: 0.2000 - val_loss: 1.6186
Epoch 12/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 283ms/step - accuracy: 0.2244 - loss: 1.6279 - val_accuracy: 0.2000 - val_loss: 1.6357
Epoch 13/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 277ms/step - accuracy: 0.1882 - loss: 1.6353 - val_accuracy: 0.2000 - val_loss: 1.7080
Epoch 14/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 286ms/step - accuracy: 0.1968 - loss: 1.6659 - val_accuracy: 0.2000 - val_loss: 1.6184
Epoch 15/20
22/22 ━━━━━━━━━━━━━━━━━━━━ 8s 277ms/step - accuracy: 0.1980 - loss: 1.6365 - val_accuracy: 0.2000 - val_loss: 1.6279
Epoch 15: early stopping
Model saved to ./traffic_sign_classifier.keras

Testing CNN 2 for speed limit signs classification...

Found 150 validated image filenames belonging to 5 classes.
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.2199 - loss: 1.6187
Test loss :  1.6278630495071411
Test accuracy : 20.000000298023224 %
Predicting test images: 100%|████████████████████████████████████████████████████████████████████████████████████████████| 150/150 [00:07<00:00, 19.20it/s]
CSV-based confusion matrix saved to confusion_matrix_classification_24-03-2025_11-56-10.csv
```

The training of both CNNs is interrupted prematurely by the early stopping callback, since no improvement is detected.
The models' performances on the test data are completely random, resulting in an accuracy ~33% on 3 classes (CNN 1) and ~20% on 5 classes (CNN 2).

TEST fails as expected

9. Introduce gradient clipping

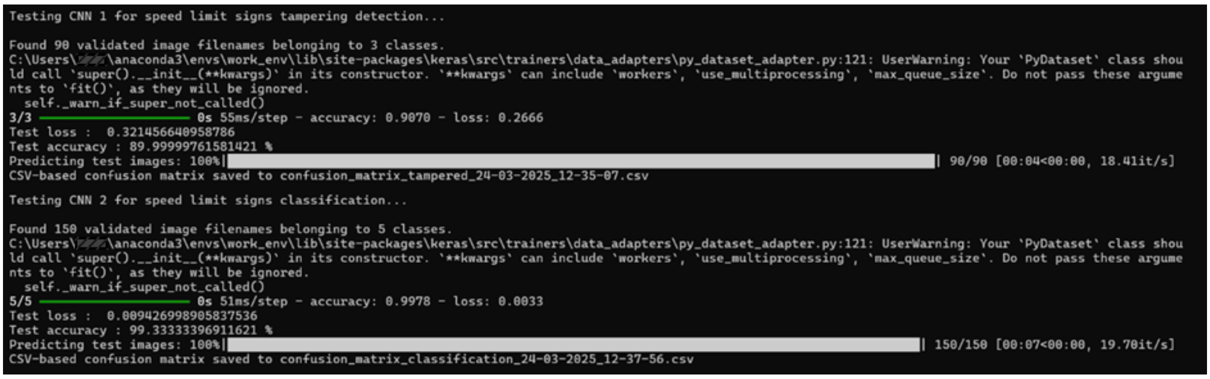
This test evaluates the capability of a TAIBOM to detect changes in the model. The test is performed on **Release 8**.

This is done introducing gradient clipping during the model optimisation step. An extremely small value was chosen for the gradient clipping:

```
model.compile(optimizer=tf.keras.optimizers.Adam()…
```

is changed to:

```
model.compile(optimizer=tf.keras.optimizers.Adam(clipnorm=0.00001)…
```



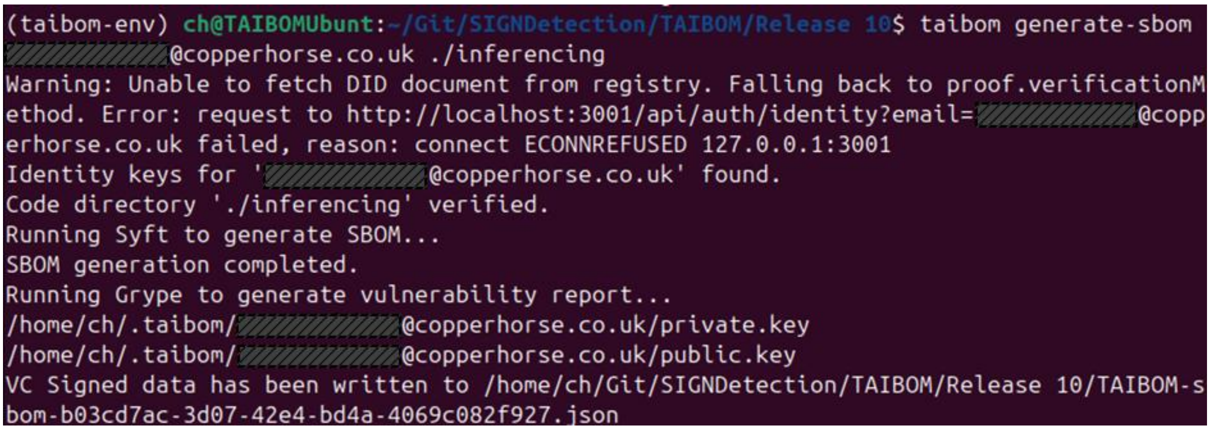
```
Testing CNN 1 for speed limit signs tampering detection...

Found 90 validated image filenames belonging to 3 classes.
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - accuracy: 0.9070 - loss: 0.2666
Test loss :  0.321456640958786
Test accuracy : 89.99999761581421 %
Predicting test images: 100%|██████████████████████████████████████████████████| 90/90 [00:04<00:00, 18.41it/s]
CSV-based confusion matrix saved to confusion_matrix_tampered_24-03-2025_12-35-07.csv

Testing CNN 2 for speed limit signs classification...

Found 150 validated image filenames belonging to 5 classes.
C:\Users\[redacted]\anaconda3\envs\work_env\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class shou
ld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these argume
nts to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step - accuracy: 0.9978 - loss: 0.0033
Test loss :  0.009426998905837536
Test accuracy : 99.33333396911621 %
Predicting test images: 100%|████████████████████████████████████████████████| 150/150 [00:07<00:00, 19.70it/s]
CSV-based confusion matrix saved to confusion_matrix_classification_24-03-2025_12-37-56.csv
```

The training is marginally affected by the change. The convergence of the training is slower and the model performances on test data are also inferior to the baseline.

TEST fails as expected

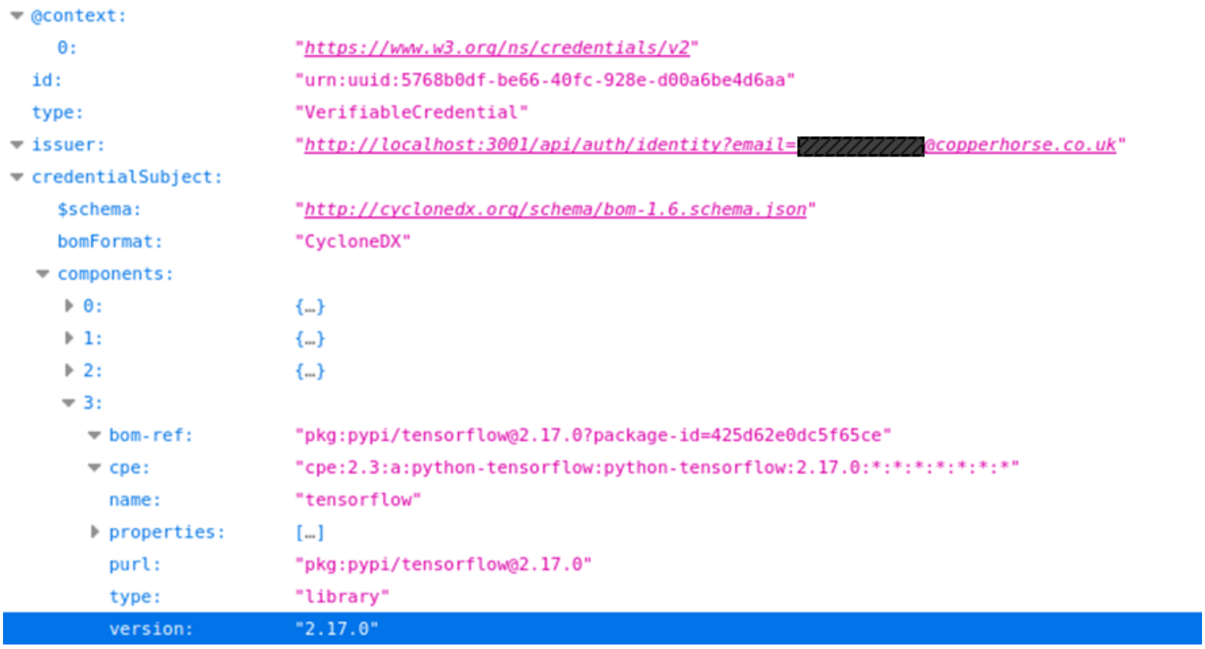
10. Modifying requirements to introduce a vulnerability

This test evaluates the ability for `generate-sbom` to correctly identify the dependencies and list the version of libraries used in the code. By downgrading TensorFlow (tf) from version 2.17 to 2.12 a known CVE (CVE-2024-3660) gets reintroduced, that could allow for arbitrary code execution.
Start by running `taibom generate-sbom` on the latest version. This generates the SBOM:



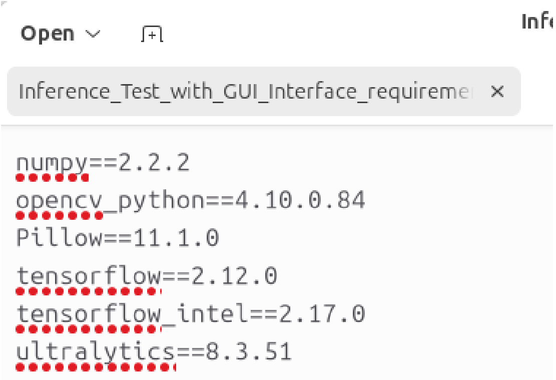
```
(taibom-env) ch@TAIBOMUbunt:~/Git/SIGNDetection/TAIBOM/Release 10$ taibom generate-sbom
[redacted]@copperhorse.co.uk ./inferencing
Warning: Unable to fetch DID document from registry. Falling back to proof.verificationM
ethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copp
erhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Code directory './inferencing' verified.
Running Syft to generate SBOM...
SBOM generation completed.
Running Grype to generate vulnerability report...
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/Git/SIGNDetection/TAIBOM/Release 10/TAIBOM-s
bom-b03cd7ac-3d07-42e4-bd4a-4069c082f927.json
```

This screenshot shows the Software Bill of Material (SBOM) that was created (stating latest TensorFlow version 2.17.0):



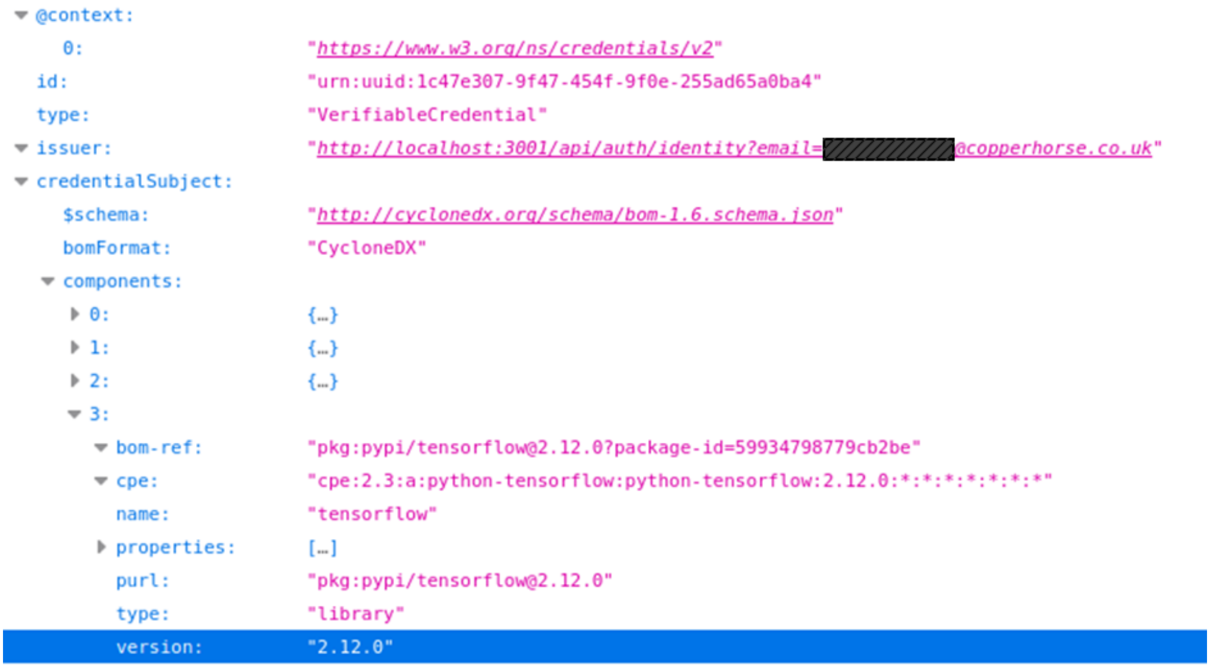
```
▼ @context:
    0:                     "https://www.w3.org/ns/credentials/v2"
  id:                      "urn:uuid:5768b0df-be66-40fc-928e-d00a6be4d6aa"
  type:                    "VerifiableCredential"
▼ issuer:                  "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
    $schema:               "http://cyclonedx.org/schema/bom-1.6.schema.json"
    bomFormat:             "CycloneDX"
  ▼ components:
    ▶ 0:                   {…}
    ▶ 1:                   {…}
    ▶ 2:                   {…}
    ▼ 3:
      ▼ bom-ref:           "pkg:pypi/tensorflow@2.17.0?package-id=425d62e0dc5f65ce"
      ▼ cpe:               "cpe:2.3:a:python-tensorflow:python-tensorflow:2.17.0:*:*:*:*:*:*:*"
        name:              "tensorflow"
      ▶ properties:        […]
        purl:              "pkg:pypi/tensorflow@2.17.0"
        type:              "library"
        version:           "2.17.0"
```

Next, by altering the TensorFlow version from 2.17 down to 2.12, the known CVE is re-introduced.



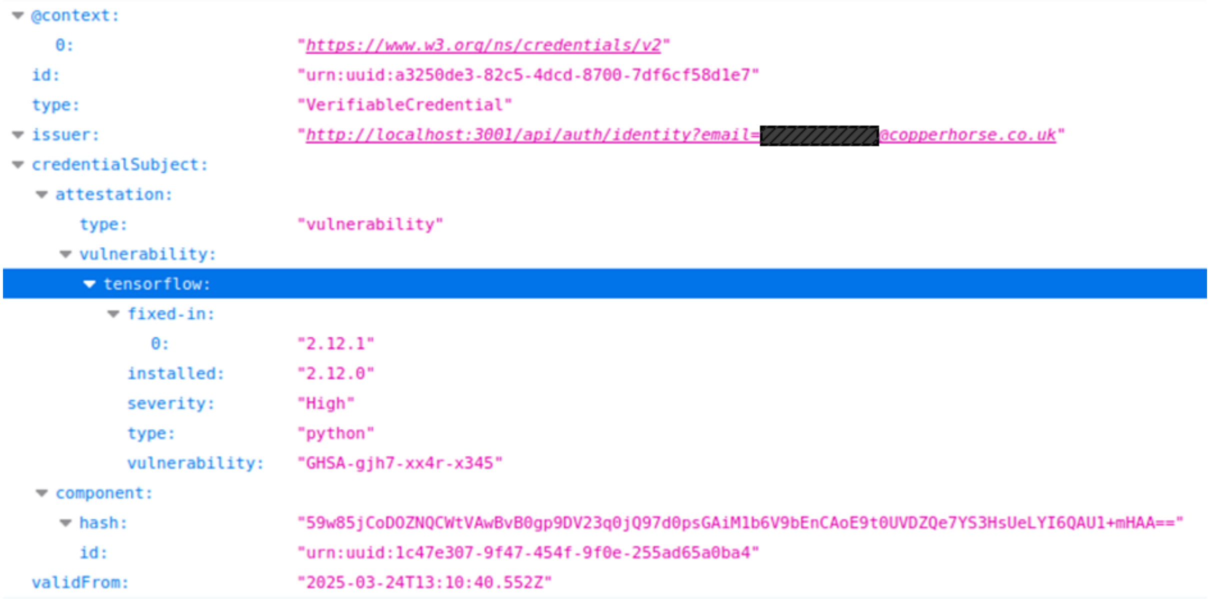
```
numpy==2.2.2
opencv_python==4.10.0.84
Pillow==11.1.0
tensorflow==2.12.0
tensorflow_intel==2.17.0
ultralytics==8.3.51
```

Generating a new SBOM states the TensorFlow version to be “2.12.0”:



```
▼ @context:
    0:                  "https://www.w3.org/ns/credentials/v2"
  id:                   "urn:uuid:1c47e307-9f47-454f-9f0e-255ad65a0ba4"
  type:                 "VerifiableCredential"
▼ issuer:               "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
    $schema:            "http://cyclonedx.org/schema/bom-1.6.schema.json"
    bomFormat:          "CycloneDX"
  ▼ components:
    ▶ 0:                {…}
    ▶ 1:                {…}
    ▶ 2:                {…}
    ▼ 3:
      ▼ bom-ref:        "pkg:pypi/tensorflow@2.12.0?package-id=59934798779cb2be"
      ▼ cpe:            "cpe:2.3:a:python-tensorflow:python-tensorflow:2.12.0:*:*:*:*:*:*:*"
        name:           "tensorflow"
      ▶ properties:     […]
        purl:           "pkg:pypi/tensorflow@2.12.0"
        type:           "library"
        version:        "2.12.0"
```

Alongside the TAIBOM-sbom a TAIBOM-attestation is created, which alerts of any known CVEs. In this case the vulnerability with TensorFlow 2.12.0 is picked up (noted as “High” severity), and says that it is patched in version 2.12.1:



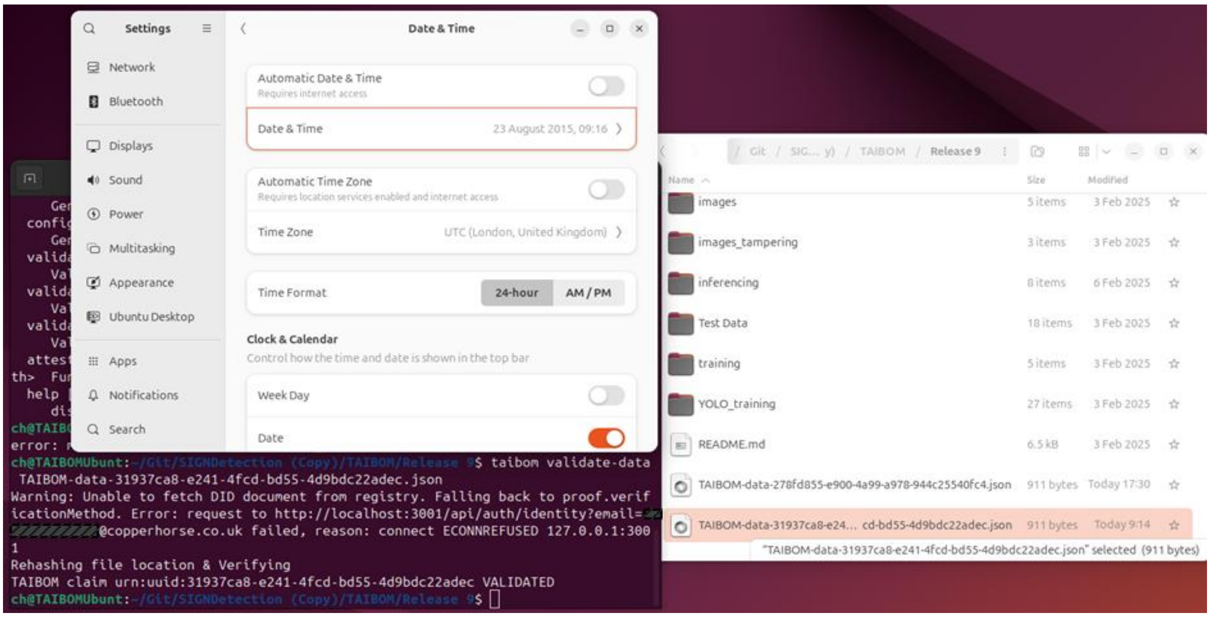
```
▼ @context:
    0:                        "https://www.w3.org/ns/credentials/v2"
  id:                         "urn:uuid:a3250de3-82c5-4dcd-8700-7df6cf58d1e7"
  type:                       "VerifiableCredential"
▼ issuer:                     "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
  ▼ attestation:
      type:                   "vulnerability"
    ▼ vulnerability:
      ▼ tensorflow:
        ▼ fixed-in:
            0:                "2.12.1"
          installed:          "2.12.0"
          severity:           "High"
          type:               "python"
          vulnerability:      "GHSA-gjh7-xx4r-x345"
  ▼ component:
    ▼ hash:                   "59w85jCoDOZNQCWtVAwBvB0gp9DV23q0jQ97d0psGAiM1b6V9bEnCAoE9t0UVDZQe7YS3HsUeLYI6QAU1+mHAA=="
      id:                     "urn:uuid:1c47e307-9f47-454f-9f0e-255ad65a0ba4"
  validFrom:                  "2025-03-24T13:10:40.552Z"
```

TEST generates SBOM and flags CVE as expected

11. Validate a TAIBOM created in the future

This test evaluates the ability of a TAIBOM to create and validate signatures with conflicting dates.

The date and time have been modified to a date in the past, in this case to 23rd August 2015 and using this date, a data-TAIBOM signature is created.



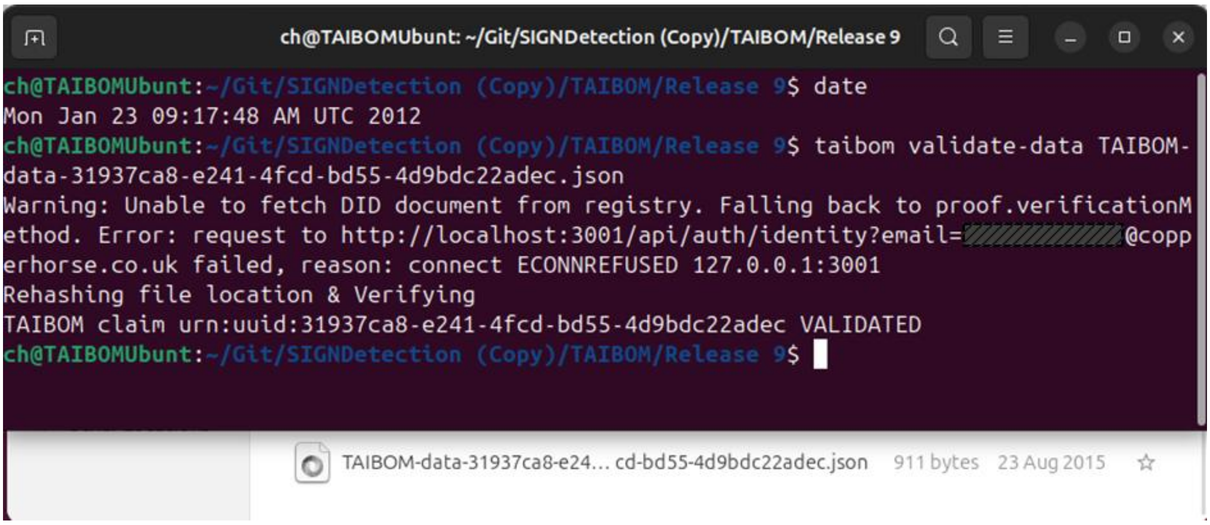
Automatic Date & Time — Requires internet access
Date & Time: 23 August 2015, 09:16
Automatic Time Zone — Requires location services enabled and internet access
Time Zone: UTC (London, United Kingdom)
Time Format: 24-hour | AM / PM
Clock & Calendar — Control how the time and date is shown in the top bar
Week Day
Date

Name	Size	Modified
images	5 items	3 Feb 2025
images_tampering	3 items	3 Feb 2025
inferencing	8 items	6 Feb 2025
Test Data	18 items	3 Feb 2025
training	5 items	3 Feb 2025
YOLO_training	27 items	3 Feb 2025
README.md	6.5 kB	3 Feb 2025
TAIBOM-data-278fd855-e900-4a99-a978-944c25540fc4.json	911 bytes	Today 17:30
TAIBOM-data-31937ca8-e24... cd-bd55-4d9bdc22adec.json	911 bytes	Today 9:14

"TAIBOM-data-31937ca8-e241-4fcd-bd55-4d9bdc22adec.json" selected (911 bytes)

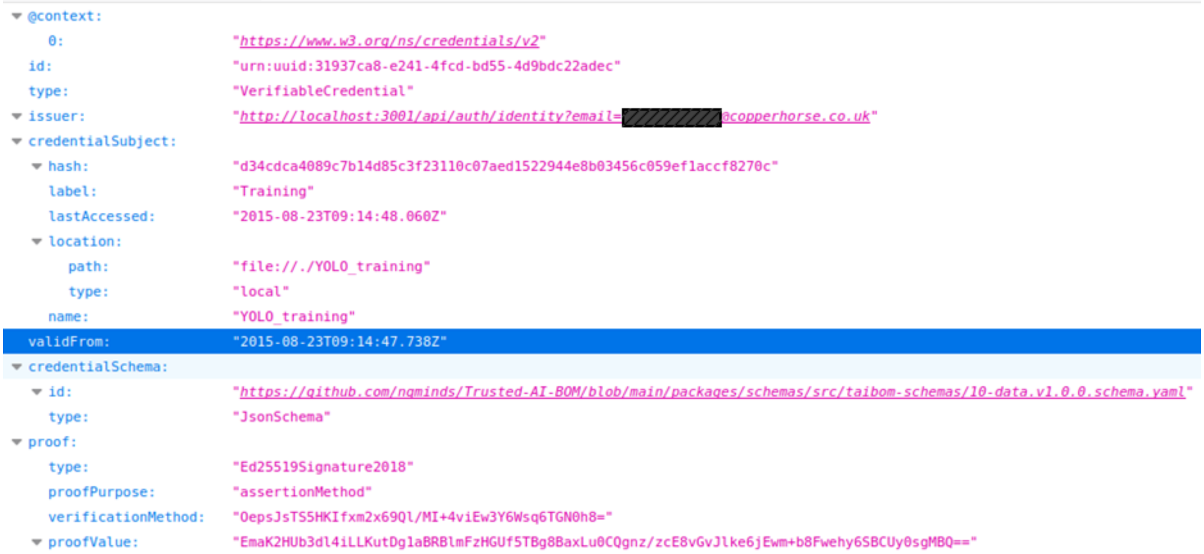
```
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$ taibom validate-data
 TAIBOM-data-31937ca8-e241-4fcd-bd55-4d9bdc22adec.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:300
1
Rehashing file location & Verifying
TAIBOM claim urn:uuid:31937ca8-e241-4fcd-bd55-4d9bdc22adec VALIDATED
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$
```

The time is then set back further to 23rd January 2012 and the signature is validated.



```
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$ date
Mon Jan 23 09:17:48 AM UTC 2012
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$ taibom validate-data TAIBOM-
data-31937ca8-e241-4fcd-bd55-4d9bdc22adec.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verificationM
ethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copp
erhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
TAIBOM claim urn:uuid:31937ca8-e241-4fcd-bd55-4d9bdc22adec VALIDATED
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$
```

TAIBOM-data-31937ca8-e24... cd-bd55-4d9bdc22adec.json 911 bytes 23 Aug 2015



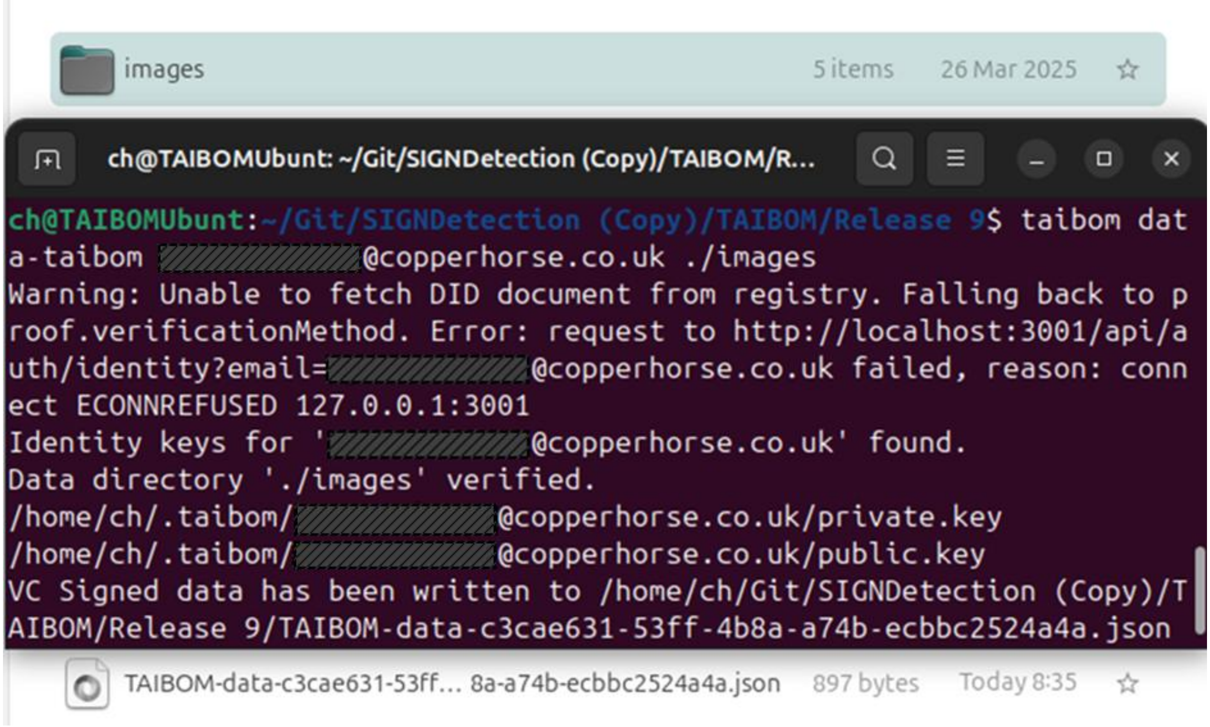
```
▼ @context:
    0:                  "https://www.w3.org/ns/credentials/v2"
  id:                   "urn:uuid:31937ca8-e241-4fcd-bd55-4d9bdc22adec"
  type:                 "VerifiableCredential"
▼ issuer:               "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
  ▼ hash:               "d34cdca4089c7b14d85c3f23110c07aed1522944e8b03456c059ef1accf8270c"
    label:              "Training"
    lastAccessed:       "2015-08-23T09:14:48.060Z"
  ▼ location:
      path:             "file://./YOLO_training"
      type:             "local"
    name:               "YOLO_training"
  validFrom:            "2015-08-23T09:14:47.738Z"
▼ credentialSchema:
  ▼ id:                 "https://github.com/ngminds/Trusted-AI-BOM/blob/main/packages/schemas/src/taibom-schemas/10-data.v1.0.0.schema.yaml"
    type:               "JsonSchema"
▼ proof:
    type:               "Ed25519Signature2018"
    proofPurpose:       "assertionMethod"
    verificationMethod: "OepsJsTS5HKIfxm2x69Ql/MI+4viEw3Y6Wsq6TGN0h8="
  ▼ proofValue:         "EmaK2HUb3dl4iLLKutDg1aBRBlmFzHGUf5TBg8BaxLu0CQgnz/zcE8vGvJlke6jEwm+b8Fwehy6SBCUy0sgMBQ=="
```

The TAIBOM has been successfully validated, despite the “validFrom” date being: "2015-08-23T09:14:47.738Z". The local machine’s time is taken verbatim when creating signatures, enabling date-spoofing and other forms of similar attack. It is also unintentional behaviour which allows a signature from the future to be validated using a date in the past.

TEST successfully validates a TAIBOM, but it should fail due to invalid date

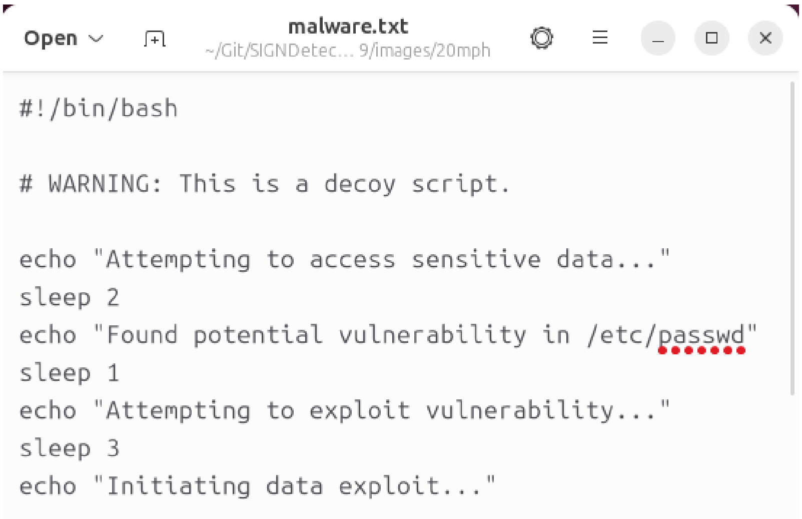
12. Using steganography to hide a malware file inside a training data image

This tests TAIBOM's ability to detect changes made to a dataset at a visually indistinguishable level. First a data TAIBOM is created of the untampered malware.


images
5 items
26 Mar 2025
ch@TAIBOMUbunt: ~/Git/SIGNDetection (Copy)/TAIBOM/R...
TAIBOM-data-c3cae631-53ff... 8a-a74b-ecbbc2524a4a.json
897 bytes
Today 8:35

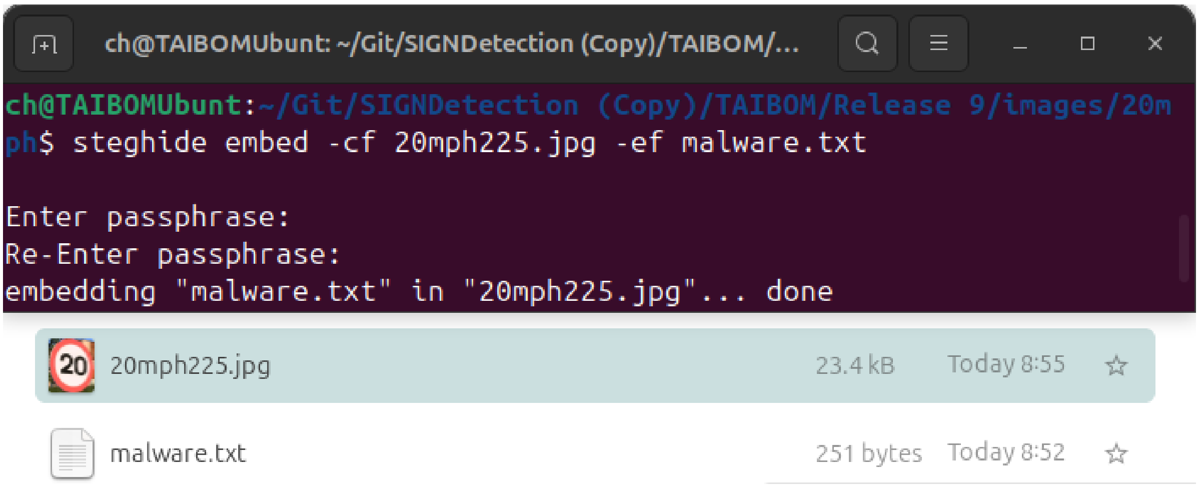
```
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$ taibom dat
a-taibom [redacted]@copperhorse.co.uk ./images
Warning: Unable to fetch DID document from registry. Falling back to p
roof.verificationMethod. Error: request to http://localhost:3001/api/a
uth/identity?email=[redacted]@copperhorse.co.uk failed, reason: conn
ect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory './images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/Git/SIGNDetection (Copy)/T
AIBOM/Release 9/TAIBOM-data-c3cae631-53ff-4b8a-a74b-ecbbc2524a4a.json
```

Next, a fake malware script is prepared as a plain txt file:



```
#!/bin/bash

# WARNING: This is a decoy script.

echo "Attempting to access sensitive data..."
sleep 2
echo "Found potential vulnerability in /etc/passwd"
sleep 1
echo "Attempting to exploit vulnerability..."
sleep 3
echo "Initiating data exploit..."
```

The txt file is then embedded into the “20mph225.jpg” image in the 20mph training dataset:

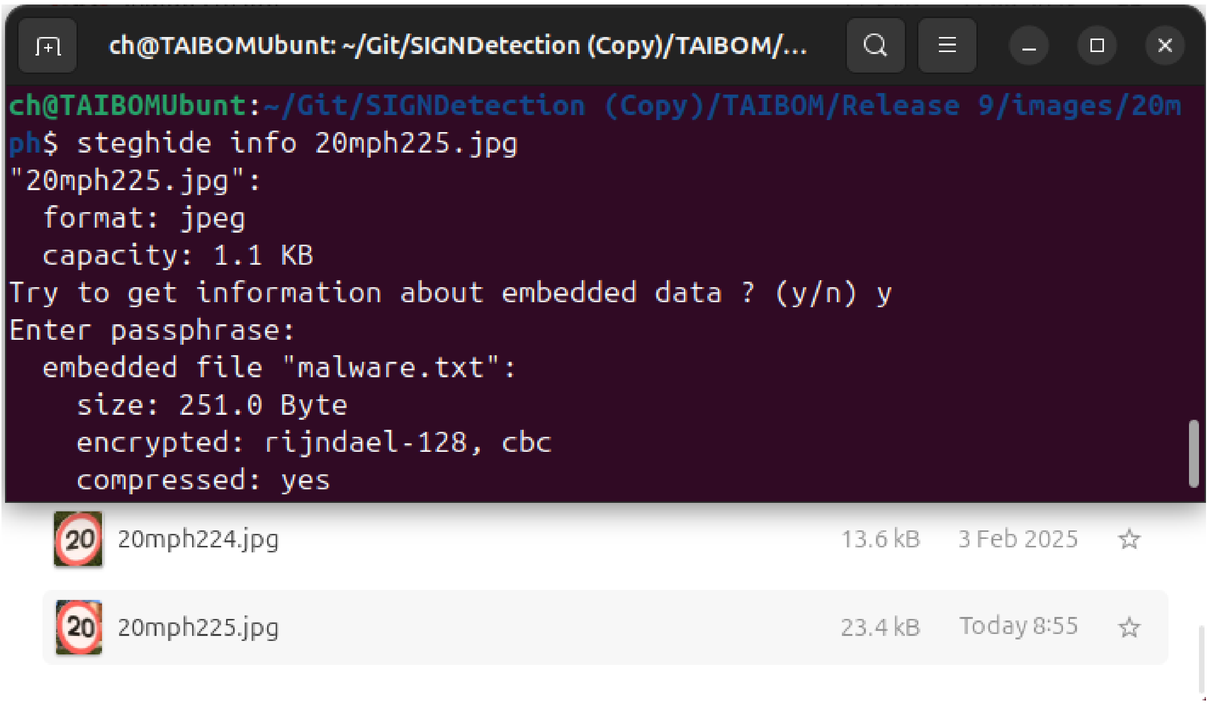


ch@TAIBOMUbunt: ~/Git/SIGNDetection (Copy)/TAIBOM/...

```
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9/images/20m
ph$ steghide embed -cf 20mph225.jpg -ef malware.txt

Enter passphrase:
Re-Enter passphrase:
embedding "malware.txt" in "20mph225.jpg"... done
```

20mph225.jpg 23.4 kB Today 8:55
malware.txt 251 bytes Today 8:52

All traces of the malware file are then removed, and it is validated that the file has been successfully embedded into the jpg. Note this isn’t the same as adding a malware file to the dataset, as this altered imaged is indistinguishable from all other images in the dataset:

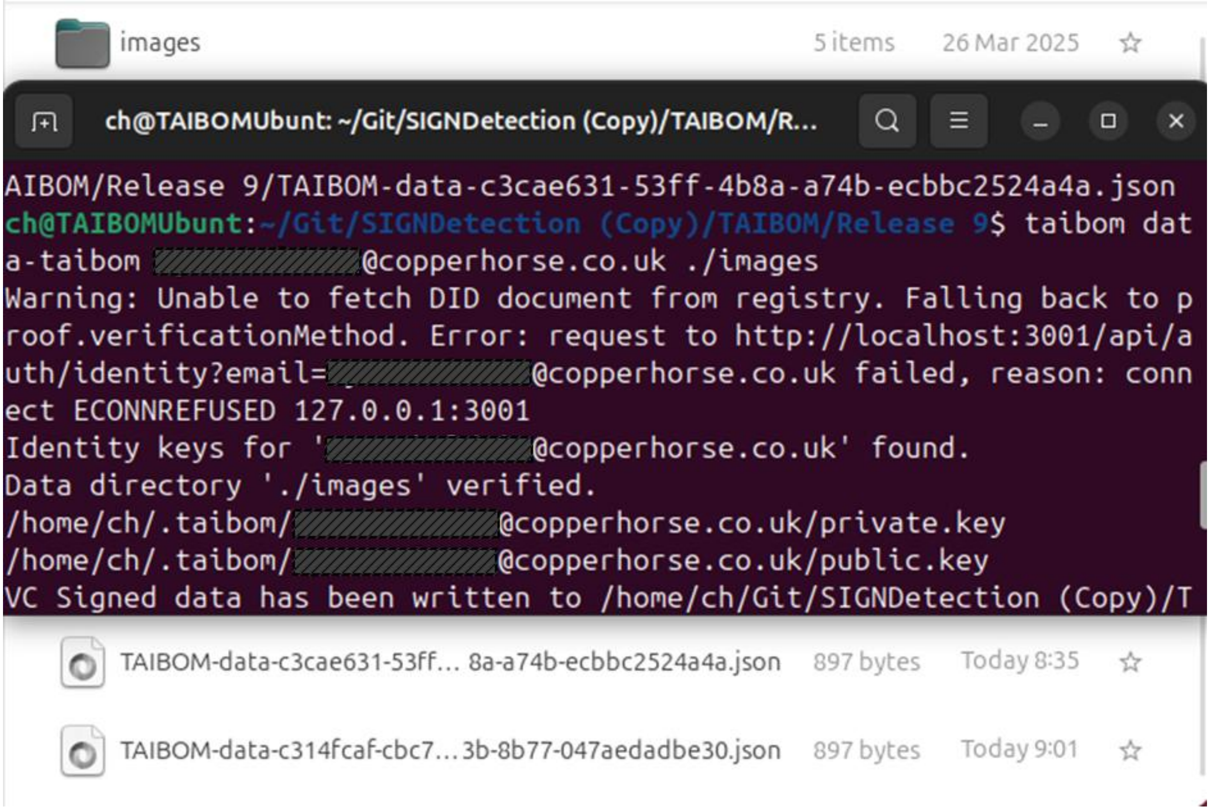


ch@TAIBOMUbunt: ~/Git/SIGNDetection (Copy)/TAIBOM/...

```
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9/images/20m
ph$ steghide info 20mph225.jpg
"20mph225.jpg":
  format: jpeg
  capacity: 1.1 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "malware.txt":
    size: 251.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

20mph224.jpg 13.6 kB 3 Feb 2025
20mph225.jpg 23.4 kB Today 8:55

Finally, a new data TAIBOM is created on the "./images", and the hashes are compared:



images 5 items 26 Mar 2025
ch@TAIBOMUbunt: ~/Git/SIGNDetection (Copy)/TAIBOM/R...

```
AIBOM/Release 9/TAIBOM-data-c3cae631-53ff-4b8a-a74b-ecbbc2524a4a.json
ch@TAIBOMUbunt:~/Git/SIGNDetection (Copy)/TAIBOM/Release 9$ taibom dat
a-taibom [redacted]@copperhorse.co.uk ./images
Warning: Unable to fetch DID document from registry. Falling back to p
roof.verificationMethod. Error: request to http://localhost:3001/api/a
uth/identity?email=[redacted]@copperhorse.co.uk failed, reason: conn
ect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory './images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/Git/SIGNDetection (Copy)/T
```

TAIBOM-data-c3cae631-53ff... 8a-a74b-ecbbc2524a4a.json 897 bytes Today 8:35
TAIBOM-data-c314fcaf-cbc7...3b-8b77-047aedadbe30.json 897 bytes Today 9:01

TAIBOM was successfully able to detect the differences between the two version of the image dataset and generated different hashes to prove the difference.

TEST generates different hashes correctly

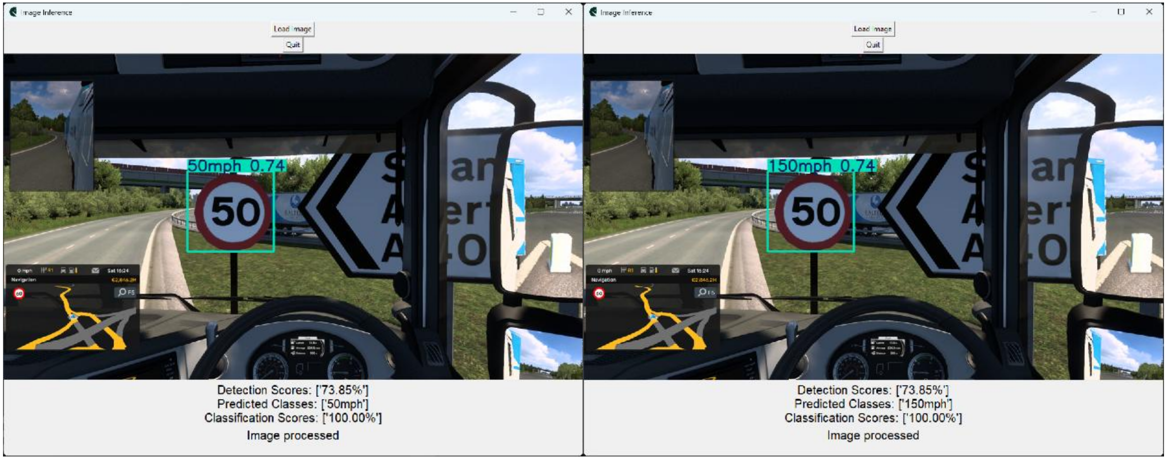
13. Change inference labels output

This test is designed to probe the ability of a TAIBOM to detect modifications on the inference code. This is done modifying the hardcoded names of the classes in the inference engine.

```
self.class_labels = ["20mph", "30mph", "40mph", "50mph", "National-Speed-Limit", "rear", "TAMPERED!"]
```

is changed to:

```
self.class_labels = ["120mph", "130mph", "140mph", "150mph", "International-Speed-Limit", "front", "ALTERED!"]
```



```
Detection Scores: ['73.85%']
Predicted Classes: ['50mph']
Classification Scores: ['100.00%']
Image processed
```

```
Detection Scores: ['73.85%']
Predicted Classes: ['150mph']
Classification Scores: ['100.00%']
Image processed
```

TEST fails as expected

14. Sign TAIBOM-datapack with a TAIBOM-code

This test is designed to probe the ability of a TAIBOM-datapack by checking the consistency of its list of inputs.

TAIBOM-datapack tags are designed to merge multiple TAIBOM-data tags into a single hash. They are used to create a unique tag for a composite training dataset. The command to create a TAIBOM-datapack tag requires a list of TAIBOM-data tags.

The command for the creation of a TAIBOM-datapack is tested by including a TAIBOM-code tag in the list of inputs for generating a TAIBOM-datapack:

```
taibom datapack-taibom  <identity_email> "SignDetection dataset with code" TAIBOM-data-13048d7c-4ac8-48da-b24f-ceb4b2889568.json TAIBOM-code-41bd6983-5238-400f-af81-9d398f754254.json
```

The `datapack-taibom` command accepts a TAIBOM-code tag where a TAIBOM-data is instead expected, confirming that no check is performed on the command input data.

The taibom-datapack is generated with no error

15. Sign code as data TAIBOIM and sign a dataset as code

This test is designed to probe the ability of a TAIBOM to distinguish between data and code tags.

TAIBOM-data and TAIBOM-code tags are the fundamental items for generating unique hashes for an AI model. The hashes are generated according to the content of a folder or for a specific file. TAIBOM-data tags should be created for the training and testing datasets, and for the trained model weights and model configuration file. TAIBOM-code tags should be created for the executable scripts which are responsible for the model training and for the inferencing engine.
When generating TAIBOM-code and TAIBOM-data hashes for the same folder, in the output TAIBOM-code and TAIBOM-data .json files, the created hashes are identical. TAIBOM-data and TAIBOM-code schemes, however, include different fields, and the hashes contained in the “proofValue” field are different in the two cases.

The taibom-data hash is generated with no error even when

16. Sign ai-system with TAIBOM-data where TAIBOM-code is expected

This test is designed to probe the ability of a TAIBOM to consistently merge TAIBOM tags when creating a TAIBOM ai-system.
TAIBOM-ai-system tags are designed to merge a TAIBOM-data (or TAIBOM-config) tag with a TAIBOM-code tag into a single hash. They are used to create a unique tag for an AI model, either for building and training a model, or for an inferencing engine. The command to create a TAIBOM-ai-system tag requires one TAIBOM-data (or TAIBOM-config) tag and one TAIBOM-code tag.

The command for the creation of a TAIBOM-ai-system tag is tested by including a TAIBOM-data tag where a TAIBOM-code tag is expected:

```
taibom system-taibom <identity_email> TAIBOM-data-13048d7c-4ac8-48da-b24f-ceb4b2889568.json TAIBOM-datapack-7e2c1185-9527-471a-b343-d482f118f852.json --name "SignDetection test"
```

The `system-taibom` command accepts a TAIBOM-data tag where a TAIBOM-code is instead expected, confirming that no check is performed on the command input data.

The TAIBOM hash is generated with no error

Use Case 2: Short Writing

Threat Table

Below is the testing table of threats for the second use case, the short writing model. Over 170 theoretical threats scenarios were created, and 15 of those attacks were implemented to test the efficacy of a TAIBOM in providing protection against these attacks. All aspects of the model were attacked, from datasets and training to inference and the models weights themselves. All aspects were tampered with to see how TAIBOM could be used to detect these threats.

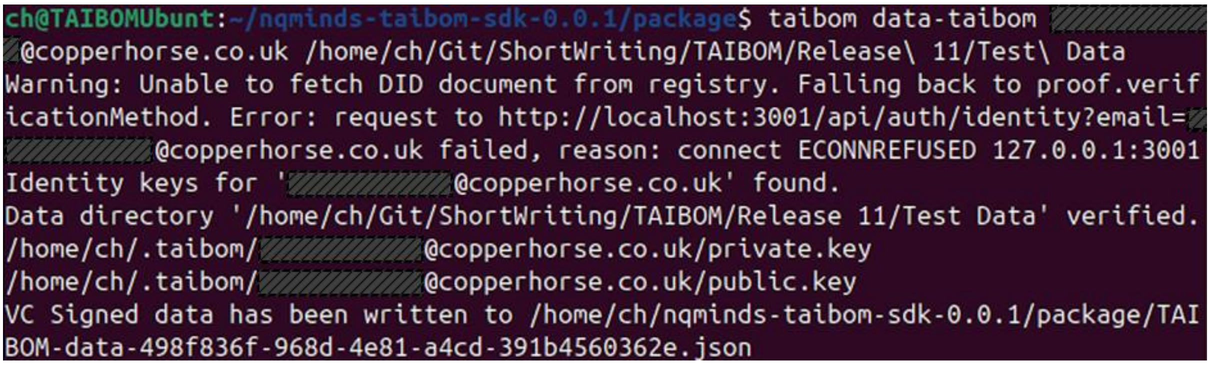
Threat		Description
1	Executable files replace existing png/jpg files in the training dataset	Malware is executed within the system, leading to data breaches, ransomware attacks, or a complete system compromise
2	Manipulation of TAIBOM signature	An attempt to deceive the authenticity of files in a project creates mistrust in the system
3	Changing Labels in the dataset	The final model is correct according to the accuracy metrics, but the actual translations is incorrect
4	Data augmentation	The algorithm for data augmentation can be modified creating distorted training images, leading to poorer model performance
5	Modifying access permissions to the dataset	"read" permissions are removed from training images, disrupting the model training
6	Introducing bias in the training data	Would lead to loss in accuracy as the model would favour certain spatial features more than others
7	Pollution of the Dataset	Would lead to loss in accuracy as the model would not be able to determine correct patterns in the spatial features it finds
8	Merging code and data with different identities into an AI-system	This test is aimed to find out how TAIBOM handles multiple signatures when creating a system-taibom.
9	Merging multiple data-taibom with different identities into a datapack	This test finds out how TAIBOM handles multiple signatures when creating a datapack
10	Introducing bias in the inferencing engine	This test evaluates TAIBOM's capability of detecting changes in the inference script by modifying the weights by a large factor
11	Pollution during Inferencing	Leads to the model not being able to identify the current features present in the image and lead to misclassification of images.
12	Stealing TAIBOM identity tokens	Identity tokens are passed from one person to another to modify and generate signatures under one alias
13	Overwriting TAIBOM identity tokens	Overwrites the identity token of a user to create a false identity that could be used to run TAIBOM commands
14	Modifying training data metadata	If training data metadata is relevant for model training, any modification would lead to a disruption of the model training
15	How does TAIBOM scale against larger datasets, is	Tests will be performed with varying dataset sizes to measure TAIBOM performance. This approach is used to assess how

Threat		Description
	performance affected by dataset size?	well TAIBOM adapts to changes in data volume to test the scalability of the software.

1. Executable files replace existing png/jpg files in the training dataset

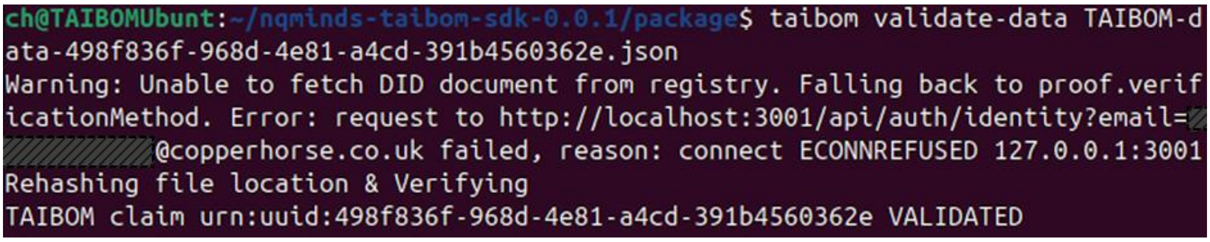
This test will evaluate the vulnerability of our dataset to injection of malicious code. The test will attempt to hide a harmful payload inside the dataset by replacing one of the image files with a disguised Python script. The script is designed to simulate malware or malicious code, designed to compromise the system.

To do this, the researchers use a TAIBOM to generate a cryptographic hash of the original, unmodified dataset as shown in the image below.



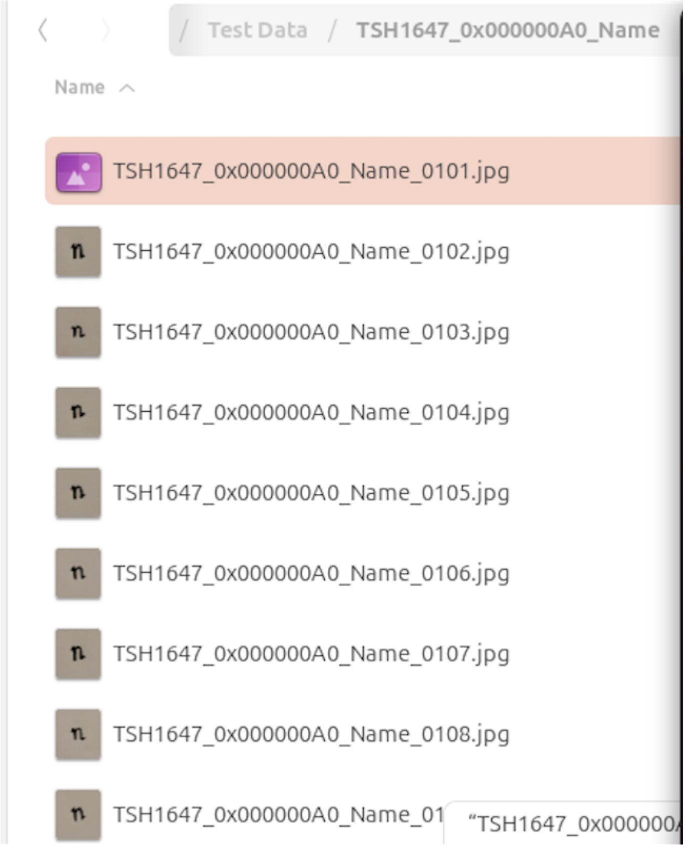
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted]
@copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Test\ Data
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Test Data' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-498f836f-968d-4e81-a4cd-391b4560362e.json
```

The hash serves as a reliable baseline, enabling any future TAIBOM run to detect any modifications or tampering and can be checked at any time as shown below.



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-498f836f-968d-4e81-a4cd-391b4560362e.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
TAIBOM claim urn:uuid:498f836f-968d-4e81-a4cd-391b4560362e VALIDATED
```

To do this test, a python script containing just one line, `print(‘Hello World!’)` has been created. The extension has been changed from .py to .jpg to hide the malicious code in the dataset as shown below.

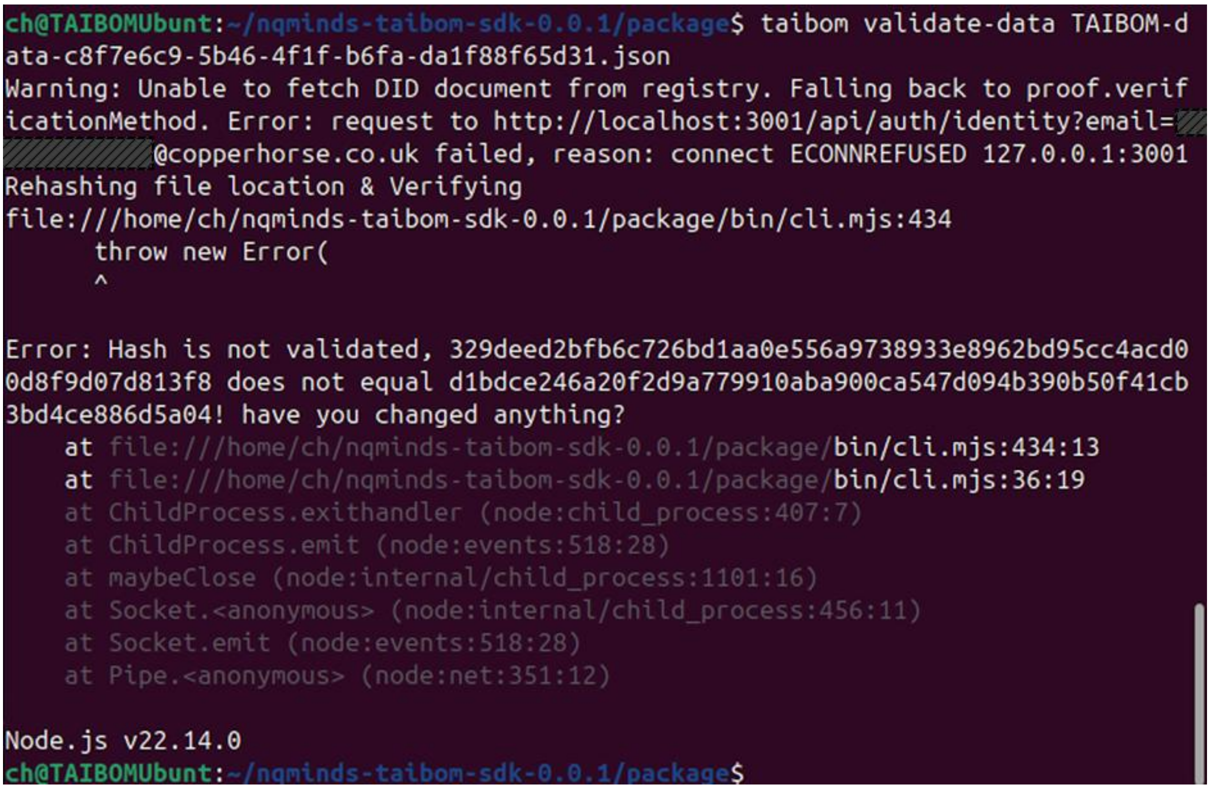


/ Test Data / TSH1647_0x000000A0_Name

Name

TSH1647_0x000000A0_Name_0101.jpg
TSH1647_0x000000A0_Name_0102.jpg
TSH1647_0x000000A0_Name_0103.jpg
TSH1647_0x000000A0_Name_0104.jpg
TSH1647_0x000000A0_Name_0105.jpg
TSH1647_0x000000A0_Name_0106.jpg
TSH1647_0x000000A0_Name_0107.jpg
TSH1647_0x000000A0_Name_0108.jpg
TSH1647_0x000000A0_Name_01
“TSH1647_0x000000A

The test dataset is then validated using TAIBOM and it can detect a change has been made to the dataset. This image demonstrates how TAIBOM can detect the change to the dataset by comparing the hashes between the original and the tampered dataset.



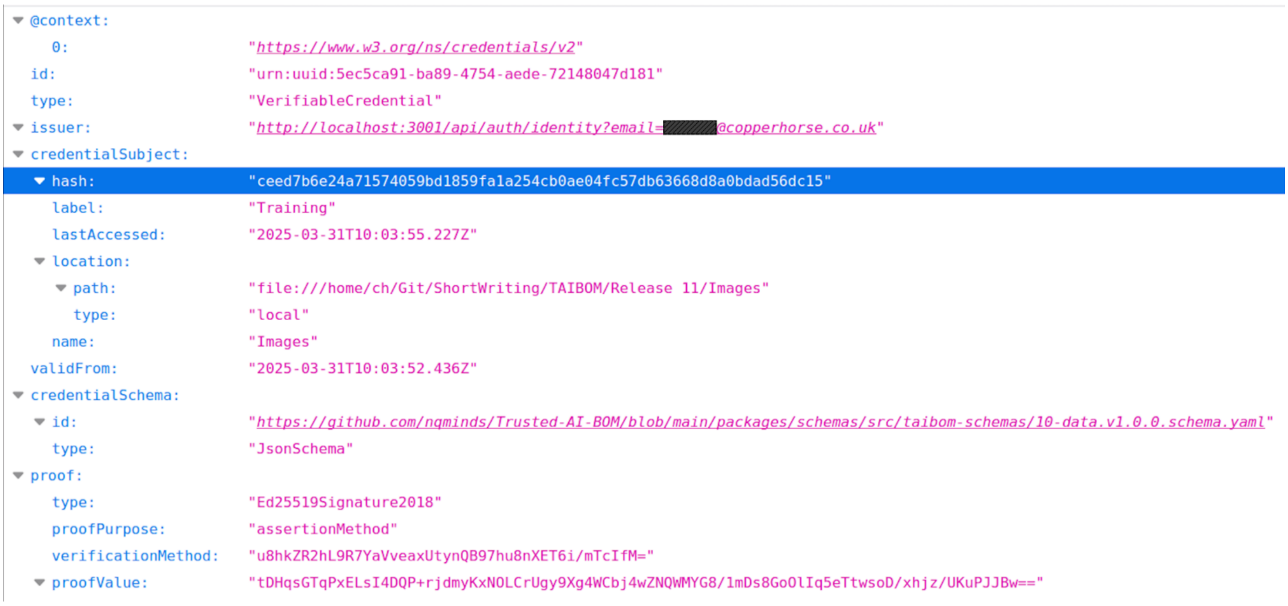
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-c8f7e6c9-5b46-4f1f-b6fa-da1f88f65d31.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, 329deed2bfb6c726bd1aa0e556a9738933e8962bd95cc4acd0
0d8f9d07d813f8 does not equal d1bdce246a20f2d9a779910aba900ca547d094b390b50f41cb
3bd4ce886d5a04! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$
```

TEST fails as expected

2. Manipulation of TAIBOM signature

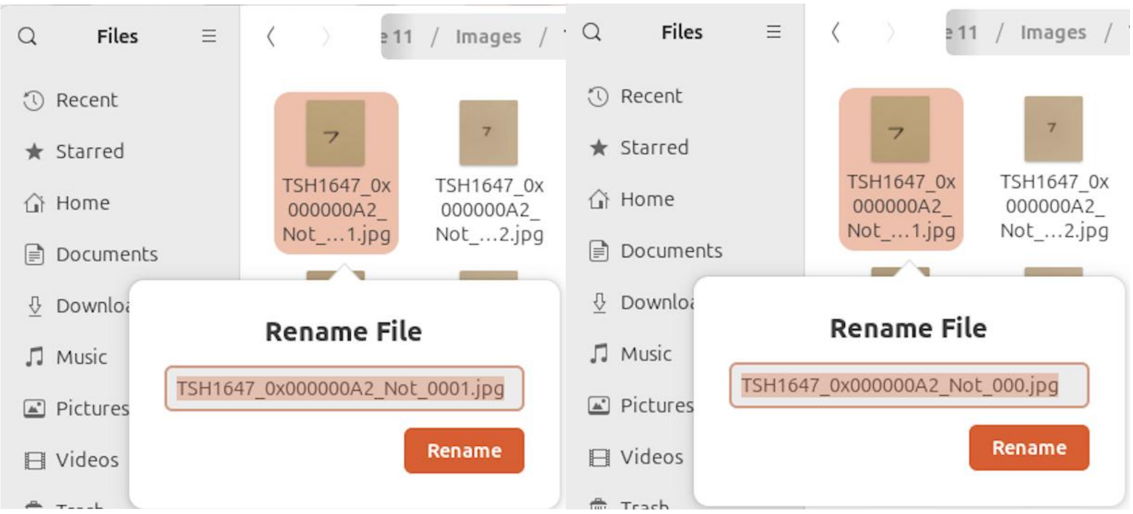
This test involves an attempt to bypass the TAIBOM validation function by altering the cryptographic hash in the dataset's TAIBOM signature. The primary objective is to evaluate the system's ability to identify and detect modified hashes.

Firstly, a signature is created using TAIBOM against the Images folder in Release 11. The signature which is generated has a cryptographic hash (starting ceed7b…') as highlighted below:



```
▼ @context:
    0:                    "https://www.w3.org/ns/credentials/v2"
  id:                     "urn:uuid:5ec5ca91-ba89-4754-aede-72148047d181"
  type:                   "VerifiableCredential"
▼ issuer:                 "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
  ▼ hash:                 "ceed7b6e24a71574059bd1859fa1a254cb0ae04fc57db63668d8a0bdad56dc15"
    label:                "Training"
    lastAccessed:         "2025-03-31T10:03:55.227Z"
  ▼ location:
    ▼ path:               "file:///home/ch/Git/ShortWriting/TAIBOM/Release 11/Images"
      type:               "local"
    name:                 "Images"
  validFrom:              "2025-03-31T10:03:52.436Z"
▼ credentialSchema:
  ▼ id:                   "https://github.com/nqminds/Trusted-AI-BOM/blob/main/packages/schemas/src/taibom-schemas/10-data.v1.0.0.schema.yaml"
    type:                 "JsonSchema"
▼ proof:
    type:                 "Ed25519Signature2018"
    proofPurpose:         "assertionMethod"
    verificationMethod:   "u8hkZR2hL9R7YaVveaxUtynQB97hu8nXET6i/mTcIfM="
  ▼ proofValue:           "tDHqsGTqPxELsI4DQP+rjdmyKxNOLCrUgy9Xg4WCbj4wZNQWMYG8/1mDs8GoOlIq5eTtwsoD/xhjz/UKuPJJBw=="
```

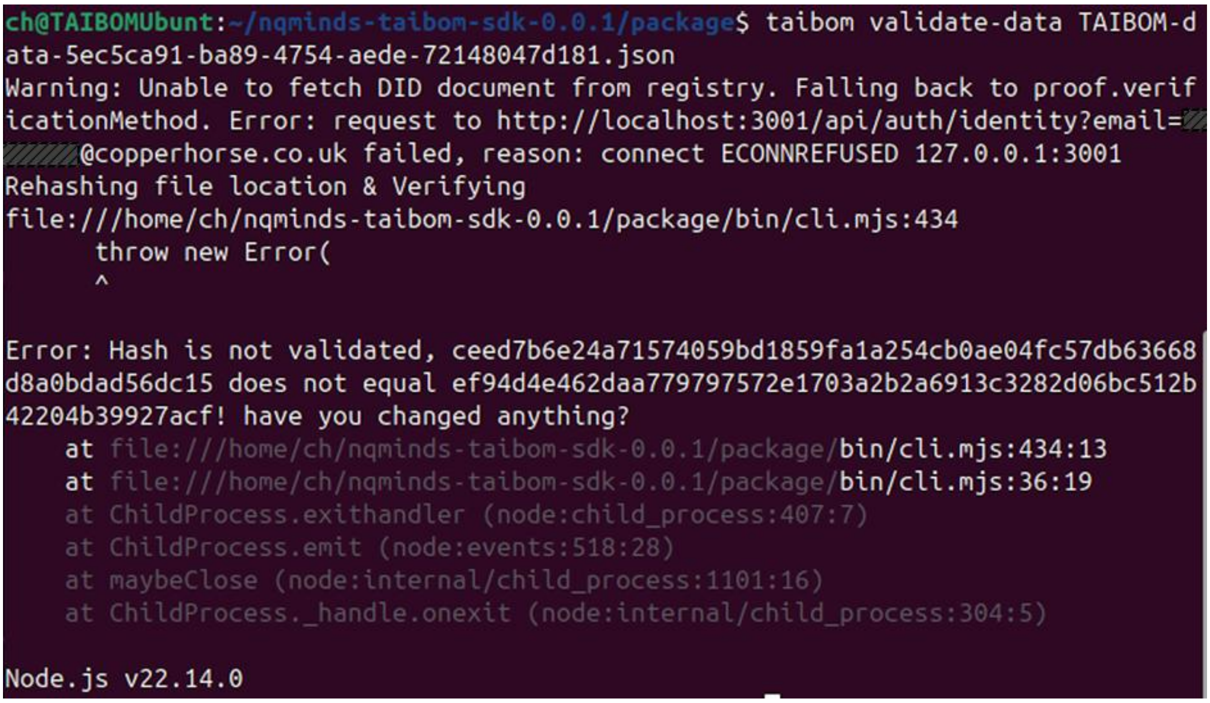
To simulate the role of an attacker, the dataset is modified by changing the name of one file in the dataset as shown in the picture.



TSH1647_0x000000A2_Not_0001.jpg

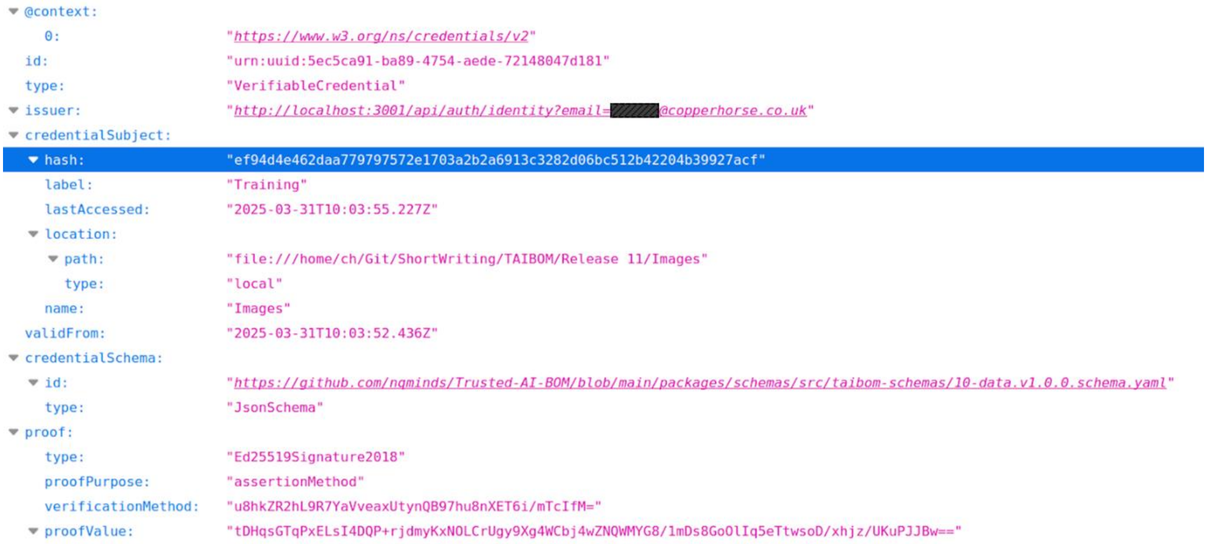
TSH1647_0x000000A2_Not_000.jpg

The dataset is then validated against the original signature from earlier, and the validation fails TAIBOM correctly determined that the hashes do not match:



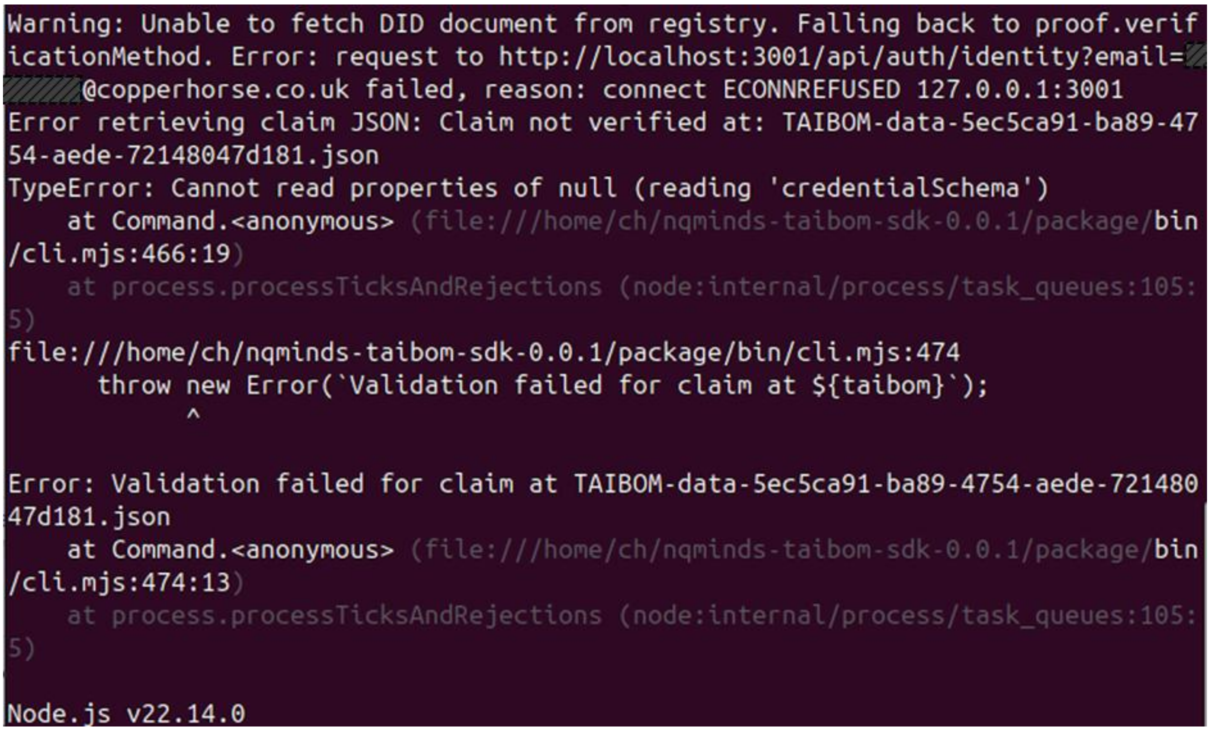
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-5ec5ca91-ba89-4754-aede-72148047d181.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, ceed7b6e24a71574059bd1859fa1a254cb0ae04fc57db63668
d8a0bdad56dc15 does not equal ef94d4e462daa779797572e1703a2b2a6913c3282d06bc512b
42204b39927acf! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at ChildProcess._handle.onexit (node:internal/child_process:304:5)

Node.js v22.14.0
```

The second Hash string (‘ef94d...’) as shown in the picture above is then manually copied into the data-TAIBOM signature from earlier (replacing the ‘ceed7b…’ hash), to try and trick the system by stating it was the original one all along:



```
▼ @context:
    0:                  "https://www.w3.org/ns/credentials/v2"
  id:                   "urn:uuid:5ec5ca91-ba89-4754-aede-72148047d181"
  type:                 "VerifiableCredential"
▼ issuer:               "http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk"
▼ credentialSubject:
  ▼ hash:               "ef94d4e462daa779797572e1703a2b2a6913c3282d06bc512b42204b39927acf"
    label:              "Training"
    lastAccessed:       "2025-03-31T10:03:55.227Z"
  ▼ location:
    ▼ path:             "file:///home/ch/Git/ShortWriting/TAIBOM/Release 11/Images"
      type:             "local"
    name:               "Images"
  validFrom:            "2025-03-31T10:03:52.436Z"
▼ credentialSchema:
  ▼ id:                 "https://github.com/nqminds/Trusted-AI-BOM/blob/main/packages/schemas/src/taibom-schemas/10-data.v1.0.0.schema.yaml"
    type:               "JsonSchema"
▼ proof:
    type:               "Ed25519Signature2018"
    proofPurpose:       "assertionMethod"
    verificationMethod: "u8hkZR2hL9R7YaVveaxUtynQB97hu8nXET6i/mTcIfM="
  ▼ proofValue:         "tDHqsGTqPxELsI4DQP+rjdmyKxNOLCrUgy9Xg4WCbj4wZNQWMYG8/1mDs8GoOlIq5eTtwsoD/xhjz/UKuPJJBw=="
```

The modified data-TAIBOM signature is then validated again after the tampered hash has been inserted. The TAIBOM system successfully identifies that changes have been made to the signature, producing an error as illustrated in the following image:



```
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Error retrieving claim JSON: Claim not verified at: TAIBOM-data-5ec5ca91-ba89-47
54-aede-72148047d181.json
TypeError: Cannot read properties of null (reading 'credentialSchema')
    at Command.<anonymous> (file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin
/cli.mjs:466:19)
    at process.processTicksAndRejections (node:internal/process/task_queues:105:
5)
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:474
      throw new Error(`Validation failed for claim at ${taibom}`);
            ^

Error: Validation failed for claim at TAIBOM-data-5ec5ca91-ba89-4754-aede-721480
47d181.json
    at Command.<anonymous> (file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin
/cli.mjs:474:13)
    at process.processTicksAndRejections (node:internal/process/task_queues:105:
5)

Node.js v22.14.0
```

Error Shown: Validation failed for claim at TAIBOM-data-5ec5ca910ba89-4753-aede-72148047d181.json

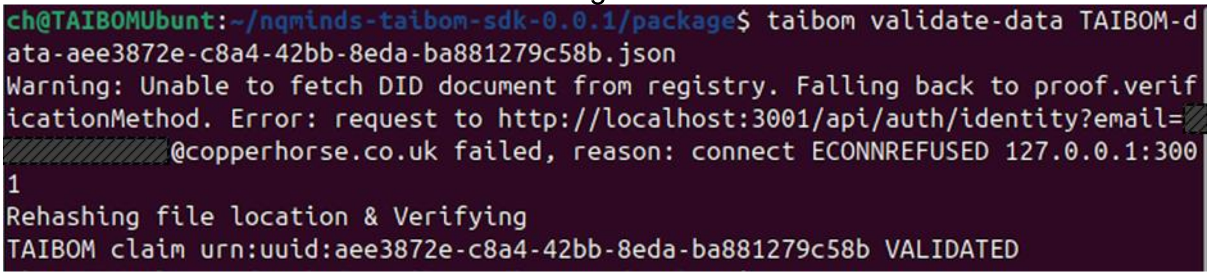
TEST fails as expected

3. Changing Labels in the dataset

The primary objective of this test is to evaluate the dataset's vulnerability to file name manipulation. Changing the labels could lead to many different problems, the one that is shown in this demonstration leads to a change in translation of a handwritten shorthand example.

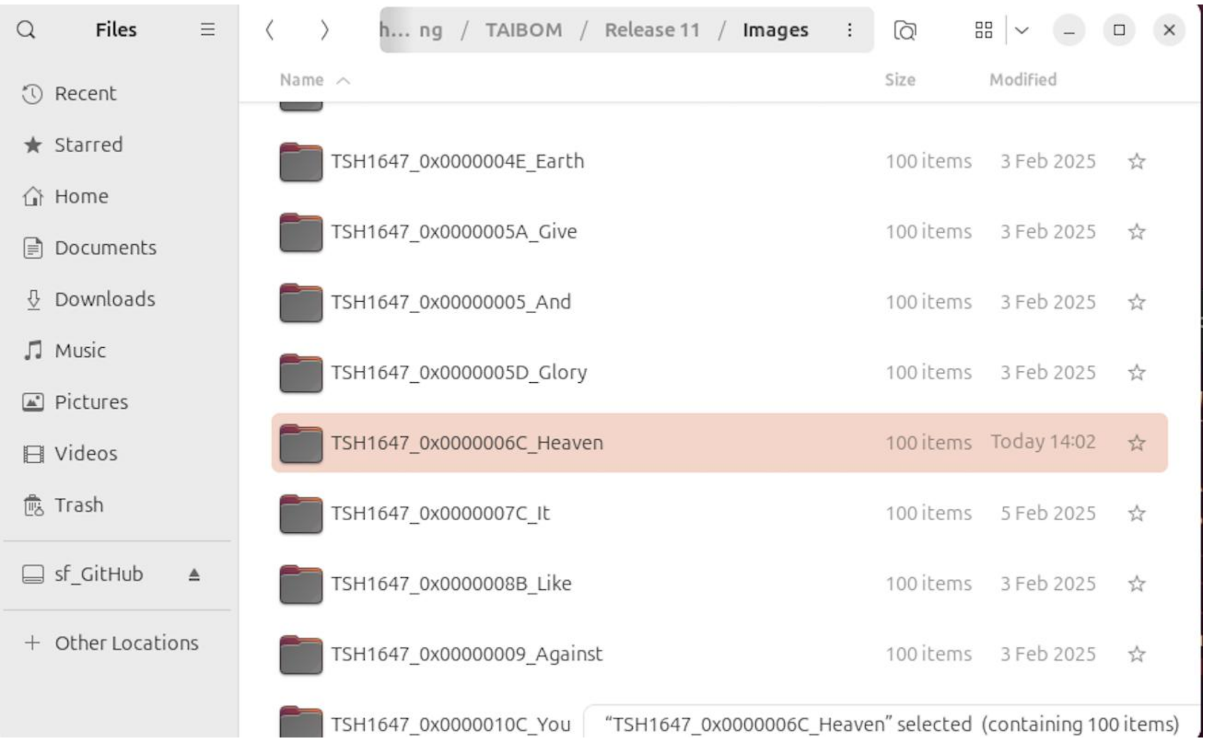
An attempt will be made to change the label 'TSH1647_0x0000006C_Heaven' to 'TSH1647_0x0000006C_Hell'. Successful execution of this change would result in the alteration of the symbol's name and translation from 'Heaven' to 'Hell’.

First a TAIBOM hash need to be generated against the Images folder as shown in the image below:



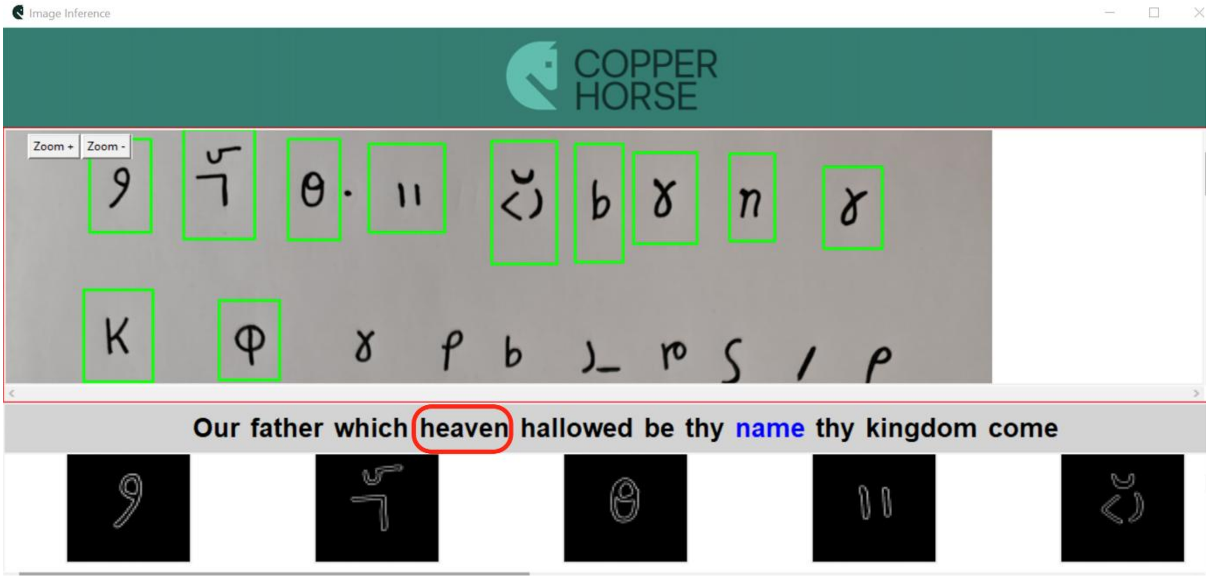
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-aee3872e-c8a4-42bb-8eda-ba881279c58b.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:300
1
Rehashing file location & Verifying
TAIBOM claim urn:uuid:aee3872e-c8a4-42bb-8eda-ba881279c58b VALIDATED
```

Then a folder is selected to rename, in this test “TSH1647_0x0000006C_Heaven” has been chosen as shown in the image below:



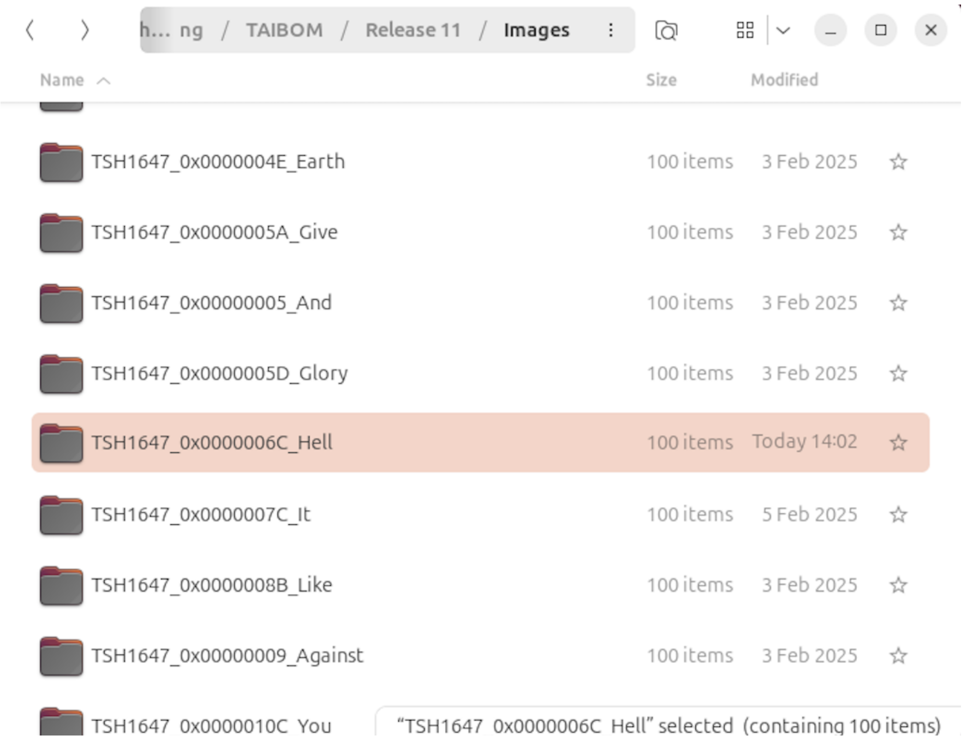
```
TSH1647_0x0000004E_Earth        100 items   3 Feb 2025
TSH1647_0x0000005A_Give         100 items   3 Feb 2025
TSH1647_0x00000005_And          100 items   3 Feb 2025
TSH1647_0x0000005D_Glory        100 items   3 Feb 2025
TSH1647_0x0000006C_Heaven       100 items   Today 14:02
TSH1647_0x0000007C_It           100 items   5 Feb 2025
TSH1647_0x0000008B_Like         100 items   3 Feb 2025
TSH1647_0x00000009_Against      100 items   3 Feb 2025
TSH1647_0x0000010C_You
“TSH1647_0x0000006C_Heaven” selected (containing 100 items)
```

In addition, the image attached below shows the inference of an example image of handwritten shorthand based on the original model with the labels unchanged:



Our father which heaven hallowed be thy name thy kingdom come

The folder “TSH1647_0x0000006C_Heaven” is then renamed to “TSH1647_0x0000006C_Hell” as shown below:

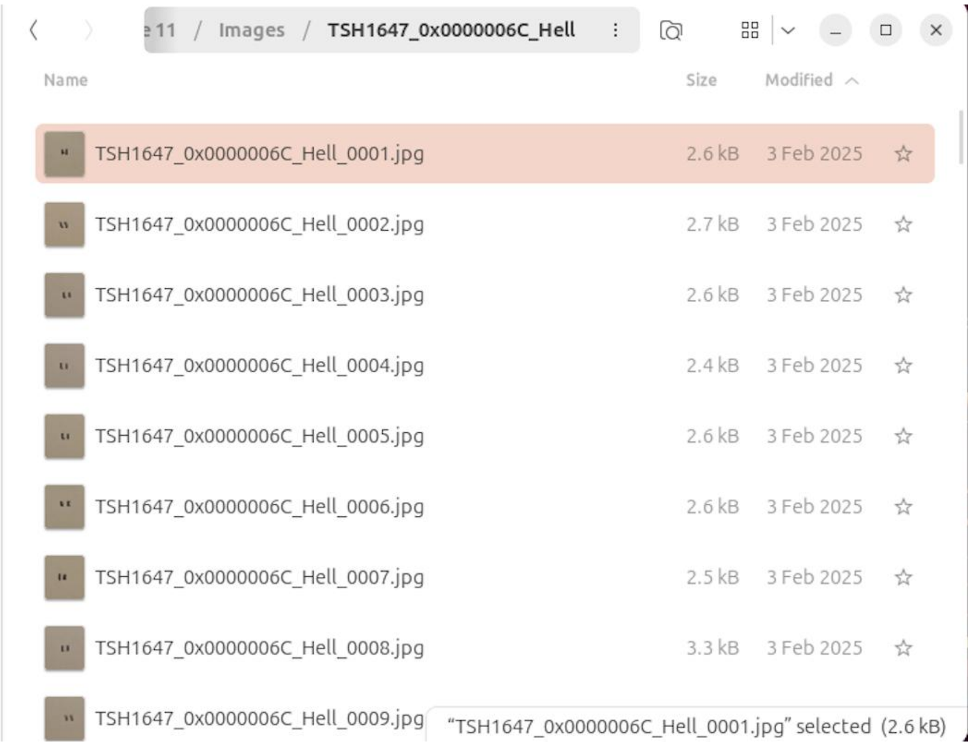


h... ng / TAIBOM / Release 11 / Images

Name	Size	Modified
TSH1647_0x0000004E_Earth	100 items	3 Feb 2025
TSH1647_0x0000005A_Give	100 items	3 Feb 2025
TSH1647_0x00000005_And	100 items	3 Feb 2025
TSH1647_0x0000005D_Glory	100 items	3 Feb 2025
TSH1647_0x0000006C_Hell	100 items	Today 14:02
TSH1647_0x0000007C_It	100 items	5 Feb 2025
TSH1647_0x0000008B_Like	100 items	3 Feb 2025
TSH1647_0x00000009_Against	100 items	3 Feb 2025
TSH1647_0x0000010C_You		

“TSH1647_0x0000006C_Hell” selected (containing 100 items)

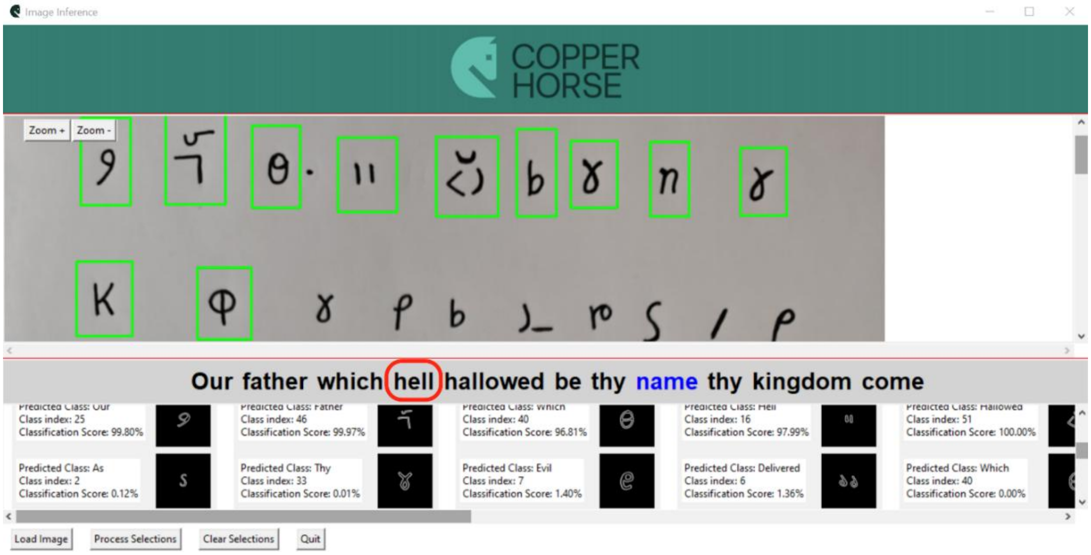
Using a Python script, all the files within the folder are renamed. The script modifies file names from the pattern 'TSH1647_0x0000006C_Heaven_0001', 'TSH1647_0x0000006C_Heaven_0002' etc., to 'TSH1647_0x0000006C_Hell_0001', 'TSH1647_0x0000006C_Hell_0002' etc.



Name	Size	Modified
TSH1647_0x0000006C_Hell_0001.jpg	2.6 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0002.jpg	2.7 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0003.jpg	2.6 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0004.jpg	2.4 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0005.jpg	2.6 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0006.jpg	2.6 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0007.jpg	2.5 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0008.jpg	3.3 kB	3 Feb 2025
TSH1647_0x0000006C_Hell_0009.jpg		

“TSH1647_0x0000006C_Hell_0001.jpg” selected (2.6 kB)

This change causes the inference test on the example image of handwritten shorthand to change slightly, the result marked by the red round box show that the translation of the symbol has changed from Heaven in the earlier picture to Hell:



Our father which hell hallowed be thy name thy kingdom come

Predicted Class: Our
Class index: 25
Classification Score: 99.80%

Predicted Class: Father
Class index: 46
Classification Score: 99.97%

Predicted Class: Which
Class index: 40
Classification Score: 96.81%

Predicted Class: Hell
Class index: 16
Classification Score: 97.99%

Predicted Class: Hallowed
Class index: 51
Classification Score: 100.00%

Predicted Class: As
Class index: 2
Classification Score: 0.12%

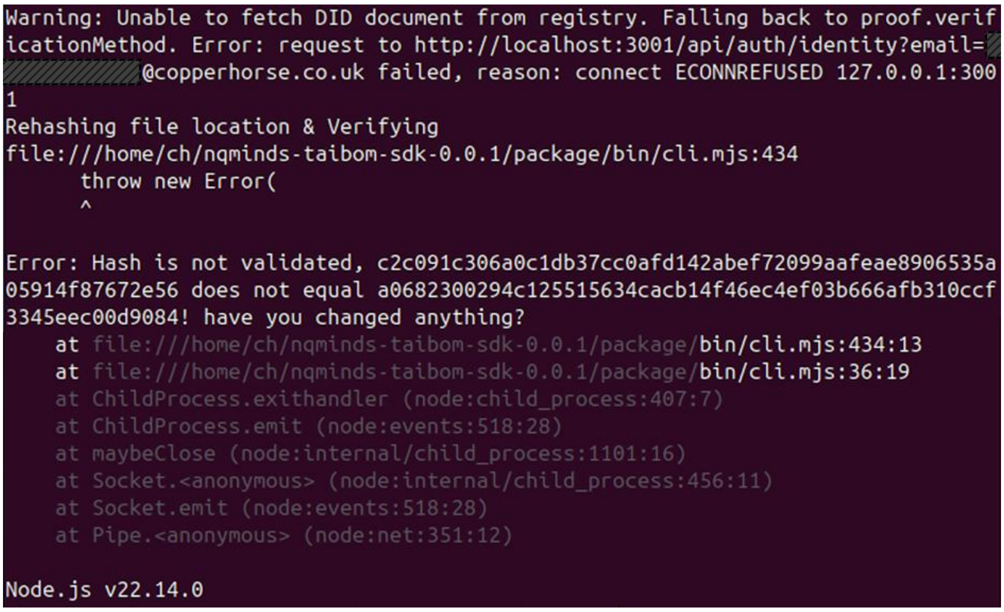
Predicted Class: Thy
Class index: 33
Classification Score: 0.01%

Predicted Class: Evil
Class index: 7
Classification Score: 1.40%

Predicted Class: Delivered
Class index: 6
Classification Score: 1.36%

Predicted Class: Which
Class index: 40
Classification Score: 0.00%

The next step is to validate the TAIBOM to see if it can detect any of the changes made. The image below demonstrates that TAIBOM can identify the change. The newly generated hash differs from the baseline hash stored in the dataset signature.



```
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:300
1
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, c2c091c306a0c1db37cc0afd142abef72099aafeae8906535a
05914f87672e56 does not equal a0682300294c125515634cacb14f46ec4ef03b666afb310ccf
3345eec00d9084! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
```

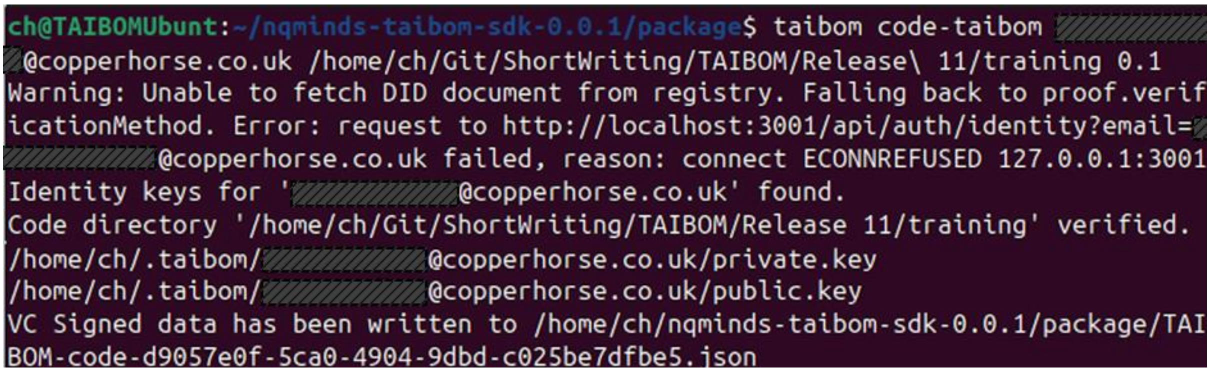
TEST fails as expected

4. Data Augmentation tampering

This test evaluates the capability of a TAIBOM to detect small changes in the training code of our model. There are many parameters and hyperparameters that can be changed to adversely affect the quality of an AI model. These variables control how a model is trained and how the data is processed to be fed into the machine learning architecture for it to learn patterns.

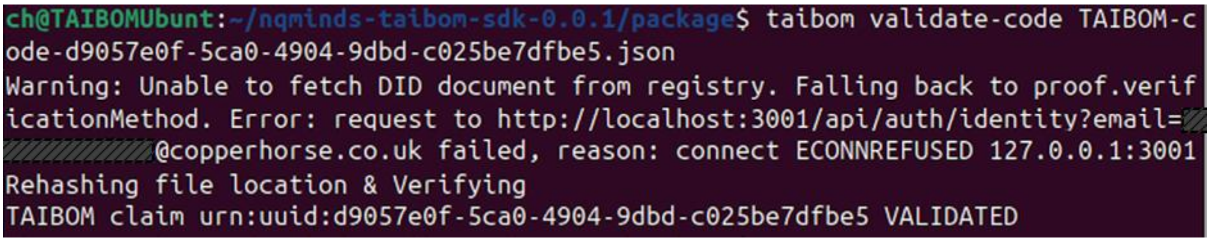
In this version of the test, the values for the variable “zoom_range” are changed from [0.5, 1.0] to [1.5 to 2.0]. This will ensure that all the images going from the dataset to the model are zoomed out for a random value between 50% to 100% thereby reducing the accuracy of the model.

To start with a TAIBOM signature is created on the folder containing the code as shown below:



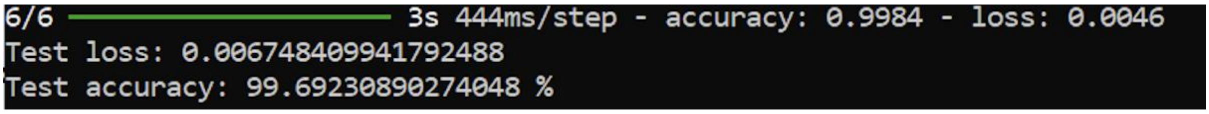
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom code-taibom [redacted]
@copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/training 0.1
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Code directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/training' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-code-d9057e0f-5ca0-4904-9dbd-c025be7dfbe5.json
```

As seen in the image, the Verifiable Credential (VC) signed data has been written to the TAIBOM package directory. Following this, the code is validated by running the command `taibom validate-code TAIBOM-code-d9057###.json`



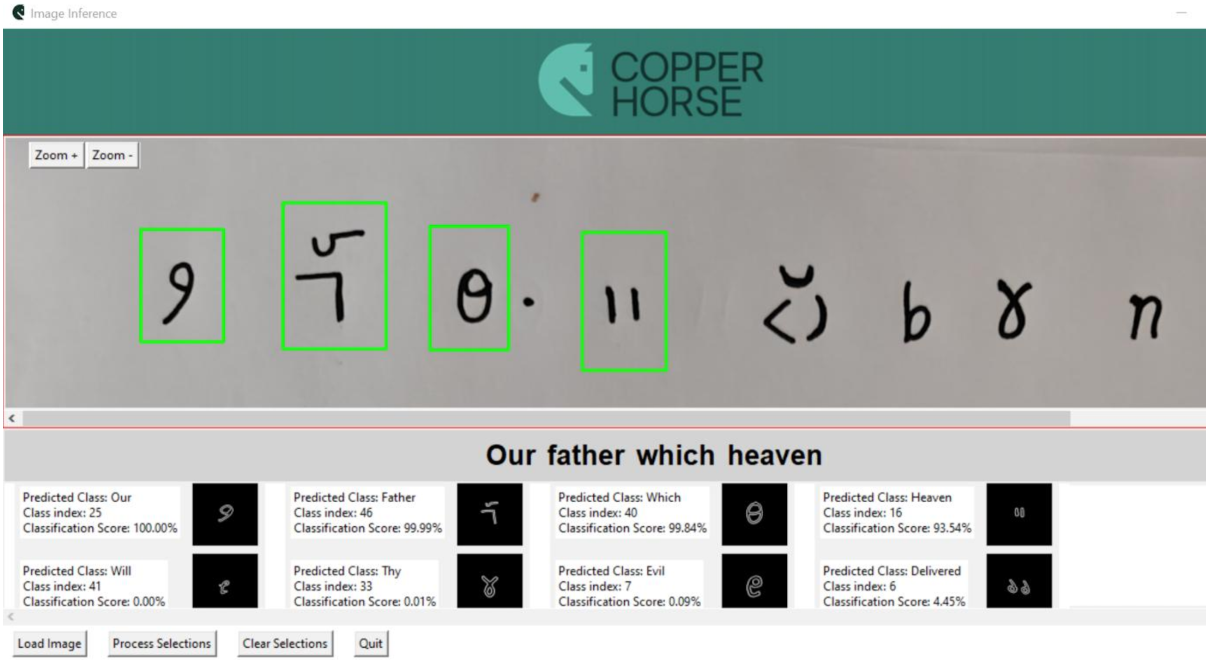
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-code TAIBOM-c
ode-d9057e0f-5ca0-4904-9dbd-c025be7dfbe5.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
TAIBOM claim urn:uuid:d9057e0f-5ca0-4904-9dbd-c025be7dfbe5 VALIDATED
```

Before any changes are made, the model is compiled again without the Short Writing Model Builder.py file being modified. While the model is being built and tested, the image below shows the actual accuracy of the model that has not been tampered with:



```
6/6 ━━━━━━━━━━━━━━━━━━━━ 3s 444ms/step - accuracy: 0.9984 - loss: 0.0046
Test loss: 0.006748409941792488
Test accuracy: 99.69230890274048 %
```

The earlier example of the handwritten shorthand prayer is also run through the inference engine to showcase the actual performance of the model without any tampering:



Our father which heaven

Predicted Class: Our
Class index: 25
Classification Score: 100.00%

Predicted Class: Father
Class index: 46
Classification Score: 99.99%

Predicted Class: Which
Class index: 40
Classification Score: 99.84%

Predicted Class: Heaven
Class index: 16
Classification Score: 93.54%

Predicted Class: Will
Class index: 41
Classification Score: 0.00%

Predicted Class: Thy
Class index: 33
Classification Score: 0.01%

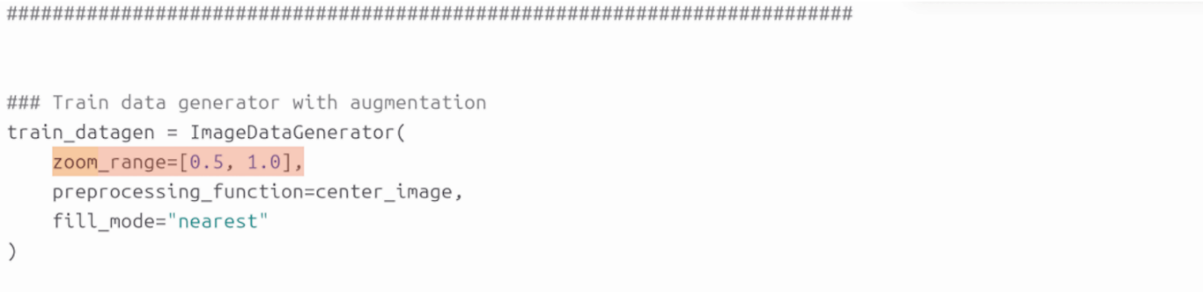
Predicted Class: Evil
Class index: 7
Classification Score: 0.09%

Predicted Class: Delivered
Class index: 6
Classification Score: 4.45%

In the image above, the model correctly identifies the first four symbols of the Lord's Prayer: “Our Father which heaven” (note for reference: the ‘in’ is missing which is substituted in the original text by the dot after the symbol for ‘which’).

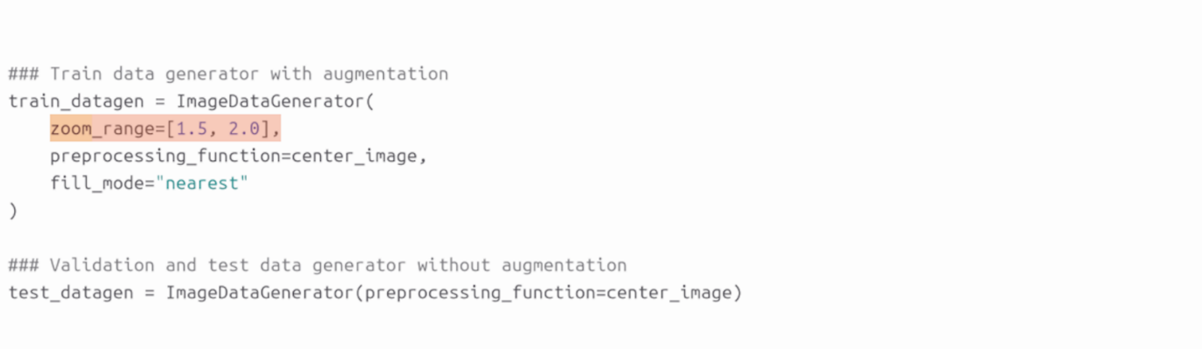
After confirming the verification and doublechecking that both the model builder and the inference engine is working okay, next the code written inside the Short Writing Model Builder.py is modified and the `zoom_range` value is tampered with.
In this test case, the original code is shown in the image:

```
zoom_range=[0.5,1.0]
```



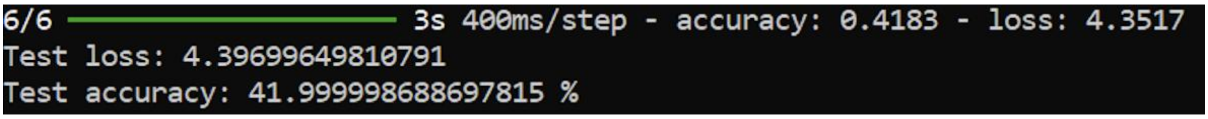
```
##########################################################################


### Train data generator with augmentation
train_datagen = ImageDataGenerator(
    zoom_range=[0.5, 1.0],
    preprocessing_function=center_image,
    fill_mode="nearest"
)
```

The value of the target variable is then changed to [1.5, 2.0] in the code as shown below:



```
### Train data generator with augmentation
train_datagen = ImageDataGenerator(
    zoom_range=[1.5, 2.0],
    preprocessing_function=center_image,
    fill_mode="nearest"
)

### Validation and test data generator without augmentation
test_datagen = ImageDataGenerator(preprocessing_function=center_image)
```

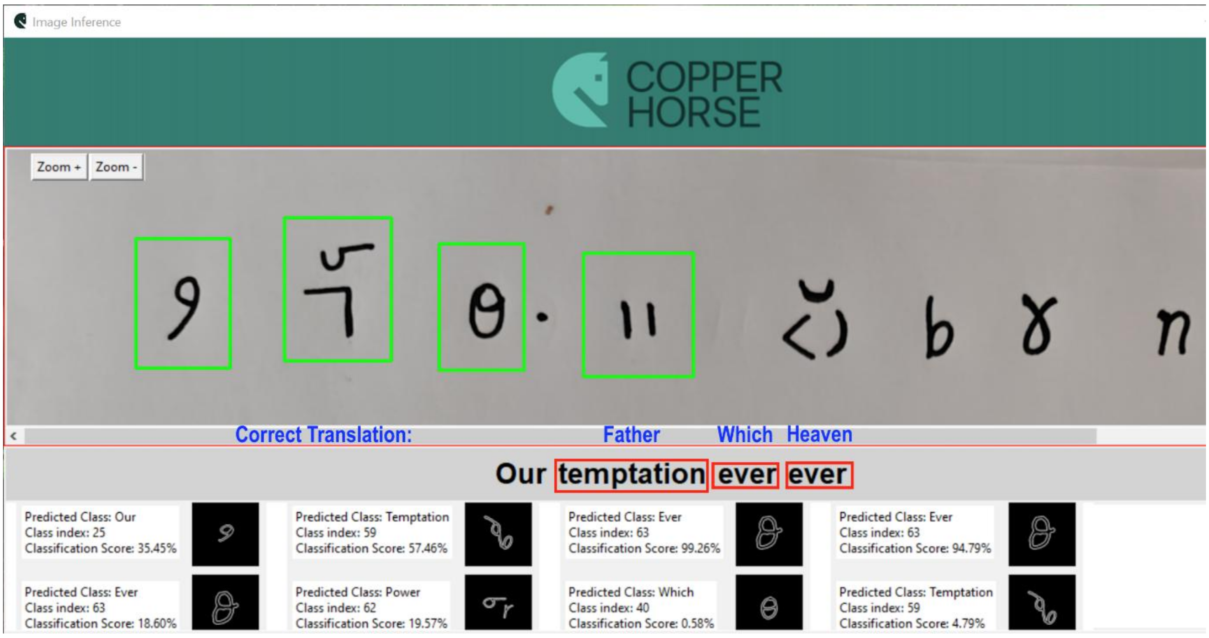
Before validating the changes, the Short Writing Model Builder.py is run again to compile another model to simulate what the effect the tampered changes could have on the model. The image shown below shows the new accuracy of the model after the changes.



```
6/6 ━━━━━━━━━━━━━━━━━━━━ 3s 400ms/step - accuracy: 0.4183 - loss: 4.3517
Test loss: 4.39699649810791
Test accuracy: 41.999998688697815 %
```

As shown in the image, the accuracy of the model drops down from 99.6 % to 41.9%.

Just to make sure the earlier example of the handwritten shorthand prayer is once again passed through the inference engine to showcase the actual performance of the model after tampering:



Correct Translation: Father Which Heaven

Our temptation ever ever

Predicted Class: Our
Class index: 25
Classification Score: 35.45%

Predicted Class: Temptation
Class index: 59
Classification Score: 57.46%

Predicted Class: Ever
Class index: 63
Classification Score: 99.26%

Predicted Class: Ever
Class index: 63
Classification Score: 94.79%

Predicted Class: Ever
Class index: 63
Classification Score: 18.60%

Predicted Class: Power
Class index: 62
Classification Score: 19.57%

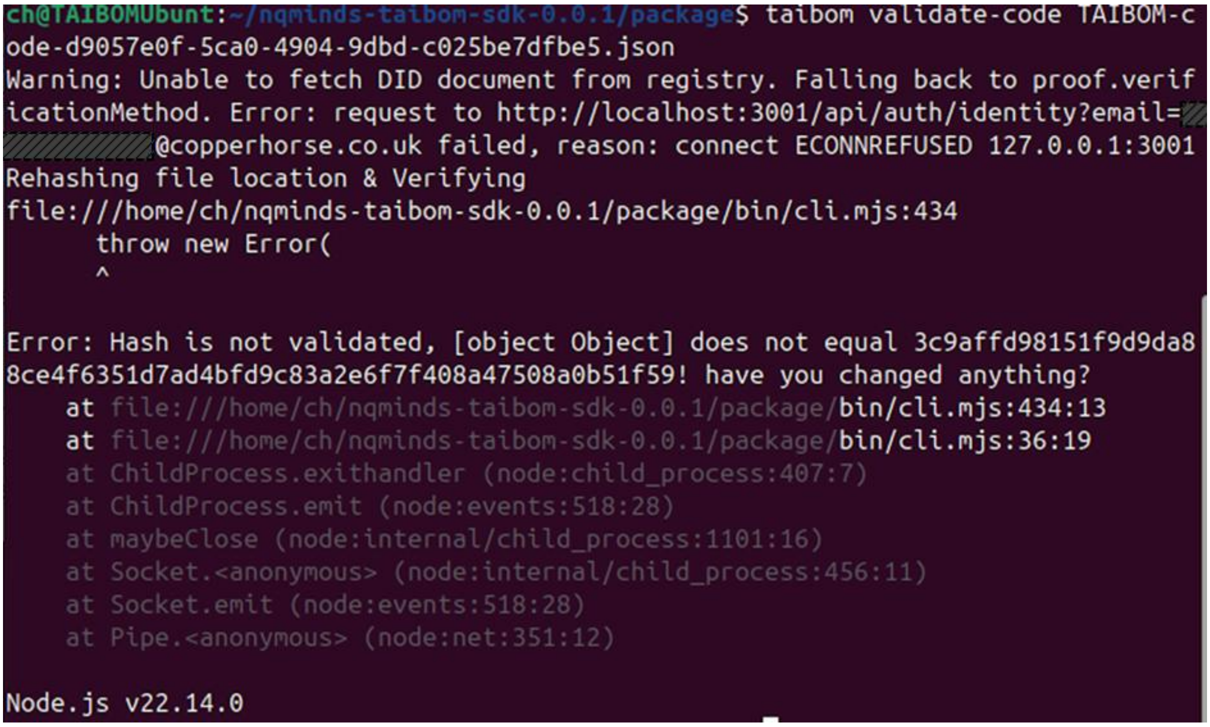
Predicted Class: Which
Class index: 40
Classification Score: 0.58%

Predicted Class: Temptation
Class index: 59
Classification Score: 4.79%

In the image above, the model is not able to correctly classify three of the first four symbols (the ones with the red boxes drawn around them) of the Lord's Prayer where the correct translation is: "Our Father which heaven".

Even the correct classification score of the "Our" went down from 100% to 35.45%, and similarly, it went down from 99.84 to 0.58% for "Which" thus incorrectly classifying it as "Ever".

Then TAIBOM is used to detect the changes, by validating the signatures again. When this test is performed, it's confirmed that TAIBOM can detect the change to `zoom_range` as the cryptographic hash differs from the original, causing validation to fail as shown below:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-code TAIBOM-c
ode-d9057e0f-5ca0-4904-9dbd-c025be7dfbe5.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, [object Object] does not equal 3c9affd98151f9d9da8
8ce4f6351d7ad4bfd9c83a2e6f7f408a47508a0b51f59! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
```

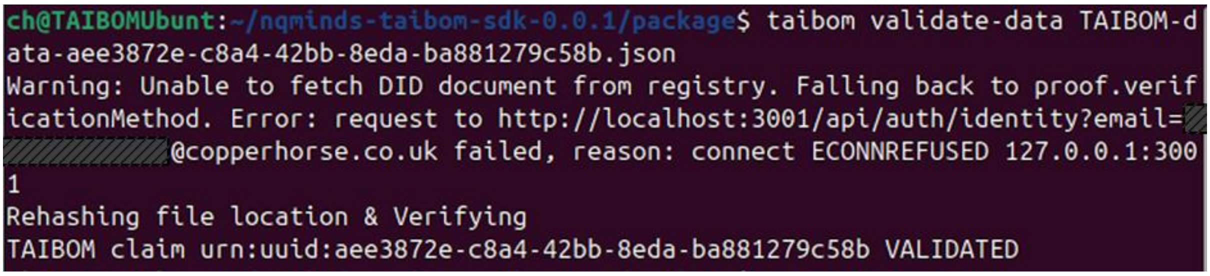
TEST fails as expected

5. Modifying access permissions to the dataset

This test evaluates the vulnerability of a TAIBOM to detect changes in the access permissions of a file. Changing the access permissions could lead to the model not gaining access to the all the labels during training which would result in either a wrong model or the code might break because of the unexpected loss of access permissions.

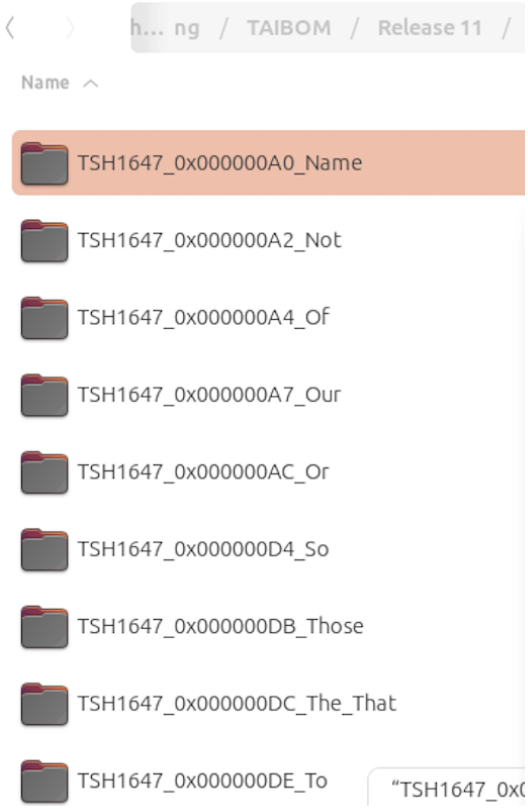
The Linux command `chmod -r` is used to change the permissions of a file in the dataset to remove the read permissions on the file. This denies TAIBOM read access to the file.

First TAIBOM is ran on the existing dataset to create a baseline hash as shown in the image below:



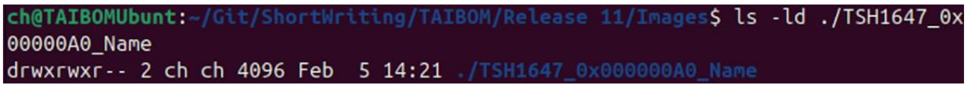
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-aee3872e-c8a4-42bb-8eda-ba881279c58b.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:300
1
Rehashing file location & Verifying
TAIBOM claim urn:uuid:aee3872e-c8a4-42bb-8eda-ba881279c58b VALIDATED
```

Then the directory “SH147_0x000000A0_Name” is selected as the folder which will have its permissions changed.



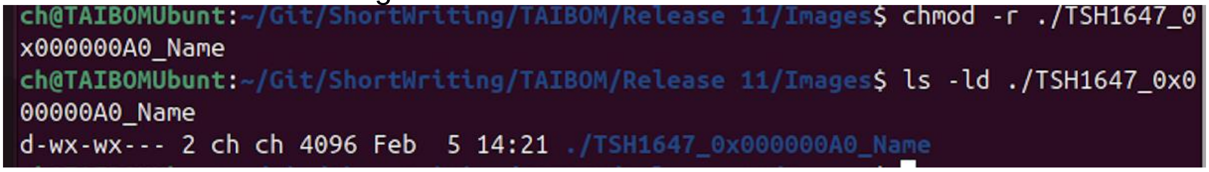
TSH1647_0x000000A0_Name
TSH1647_0x000000A2_Not
TSH1647_0x000000A4_Of
TSH1647_0x000000A7_Our
TSH1647_0x000000AC_Or
TSH1647_0x000000D4_So
TSH1647_0x000000DB_Those
TSH1647_0x000000DC_The_That
TSH1647_0x000000DE_To

Next the command `ls -ld` is executed, which displays the current permissions of the folder, the image below shows the current folder permissions d = directory, r = read, x = executable and w = write:



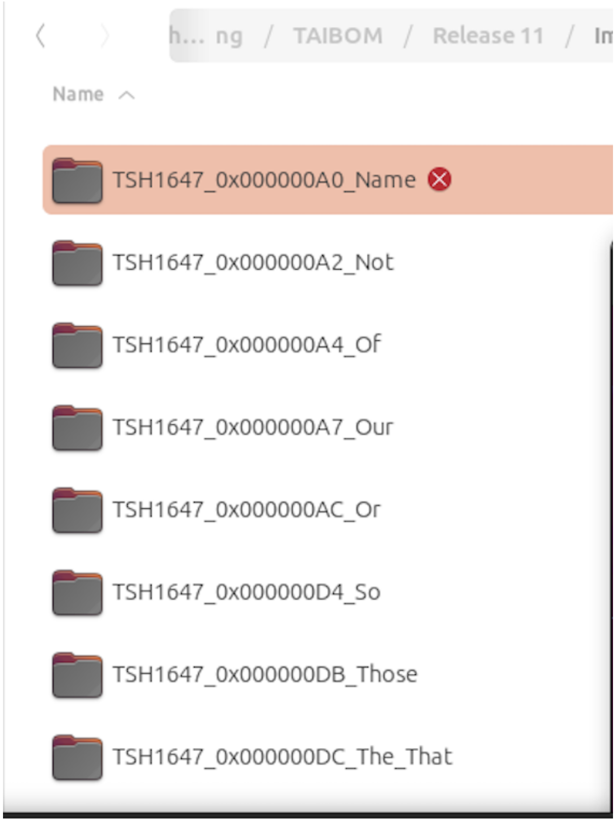
```
ch@TAIBOMUbunt:~/Git/ShortWriting/TAIBOM/Release 11/Images$ ls -ld ./TSH1647_0x
00000A0_Name
drwxrwxr-- 2 ch ch 4096 Feb  5 14:21 ./TSH1647_0x000000A0_Name
```

Now the command `chmod -r` is executed, this removes the read permission from the folder as shown in the image below:



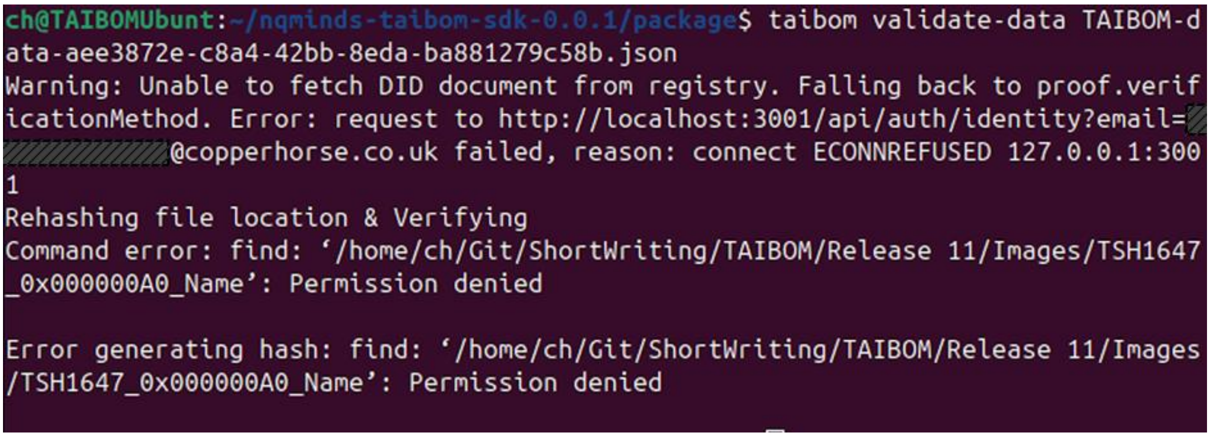
```
ch@TAIBOMUbunt:~/Git/ShortWriting/TAIBOM/Release 11/Images$ chmod -r ./TSH1647_0
x000000A0_Name
ch@TAIBOMUbunt:~/Git/ShortWriting/TAIBOM/Release 11/Images$ ls -ld ./TSH1647_0x0
00000A0_Name
d-wx-wx--- 2 ch ch 4096 Feb  5 14:21 ./TSH1647_0x000000A0_Name
```

The change in permission can be seen as a red x appears after the name of the folder that now cannot be accessed as shown below:



TSH1647_0x000000A0_Name ☒
TSH1647_0x000000A2_Not
TSH1647_0x000000A4_Of
TSH1647_0x000000A7_Our
TSH1647_0x000000AC_Or
TSH1647_0x000000D4_So
TSH1647_0x000000DB_Those
TSH1647_0x000000DC_The_That

The next step is to run a TAIBOM validation on the modified dataset to test if TAIBOM can detect this change.



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-aee3872e-c8a4-42bb-8eda-ba881279c58b.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:300
1
Rehashing file location & Verifying
Command error: find: ‘/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images/TSH1647
_0x000000A0_Name’: Permission denied

Error generating hash: find: ‘/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images
/TSH1647_0x000000A0_Name’: Permission denied
```

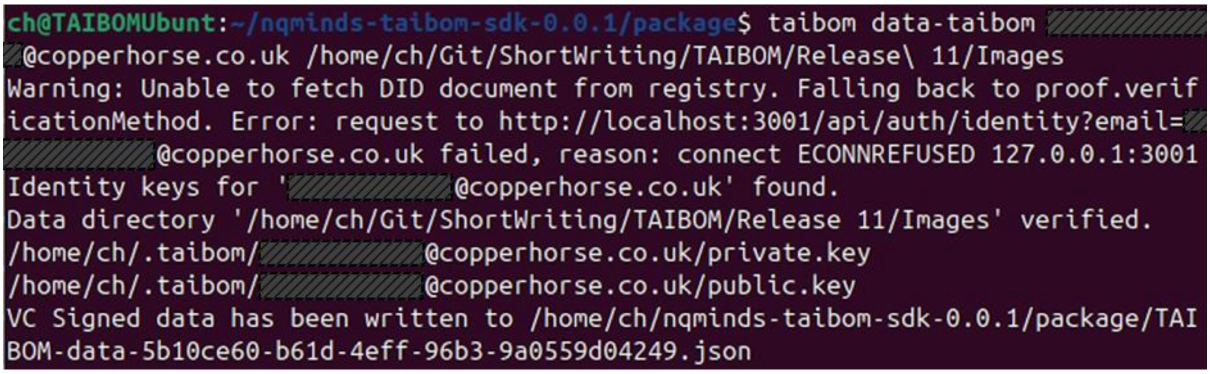
As can be seen in the image above, TAIBOM is not able to generate the hash as it doesn’t have permission to access the modified folder.

TEST fails as expected

6. Introducing Bias in the training data

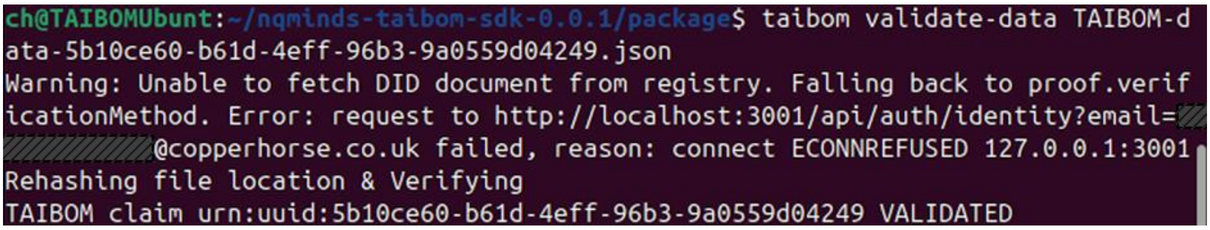
This test evaluates if a TAIBOM can detect if there is a bias introduced in the training dataset. A biased dataset would lead to a model that would tend to have most of the classifications leaning towards biased labels, the degree of this would depend on the ratio of the biased label compared to the rest of the label.

To do this, a TAIBOM signature is created against the folder containing all the labels:



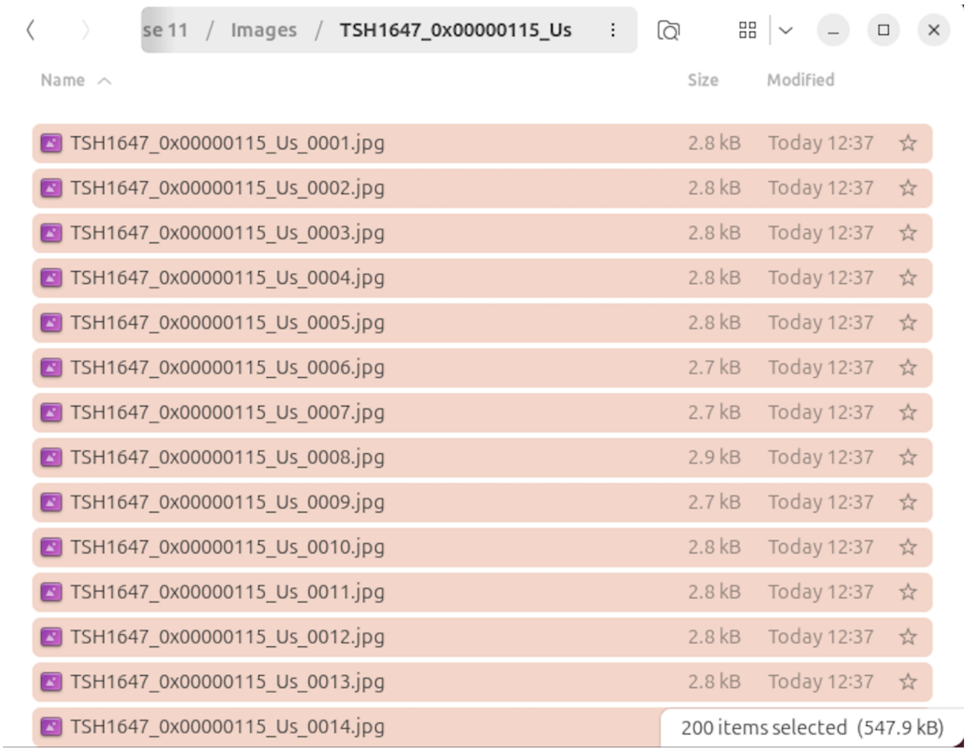
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted]
@copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Images
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-5b10ce60-b61d-4eff-96b3-9a0559d04249.json
```

After creation of the signature, it is validated it to make sure that the TAIBOM is working properly before making any changes:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-5b10ce60-b61d-4eff-96b3-9a0559d04249.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
TAIBOM claim urn:uuid:5b10ce60-b61d-4eff-96b3-9a0559d04249 VALIDATED
```

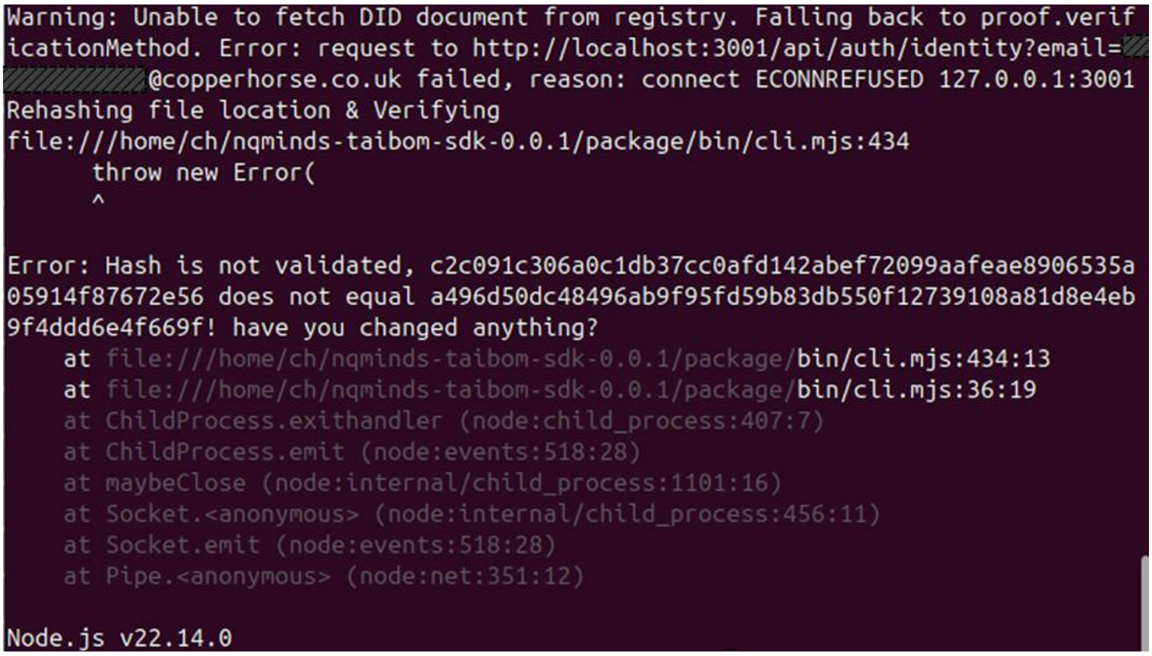
After validating it, extra datapoints are needed to create bias in the document. For this test, the folder “TSH1647_0x00000115_US” is selected as the target label. The number of images in this label will be increased to from 100 to 200 thus creating a biased dataset.



Name	Size	Modified
TSH1647_0x00000115_Us_0001.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0002.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0003.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0004.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0005.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0006.jpg	2.7 kB	Today 12:37
TSH1647_0x00000115_Us_0007.jpg	2.7 kB	Today 12:37
TSH1647_0x00000115_Us_0008.jpg	2.9 kB	Today 12:37
TSH1647_0x00000115_Us_0009.jpg	2.7 kB	Today 12:37
TSH1647_0x00000115_Us_0010.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0011.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0012.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0013.jpg	2.8 kB	Today 12:37
TSH1647_0x00000115_Us_0014.jpg		

200 items selected (547.9 kB)

With the dataset now biased towards label “TSH1647_0x00000115_US”, the TAIBOM signature associated with the dataset is validated again:



```
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, c2c091c306a0c1db37cc0afd142abef72099aafeae8906535a
05914f87672e56 does not equal a496d50dc48496ab9f95fd59b83db550f12739108a81d8e4eb
9f4ddd6e4f669f! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
```

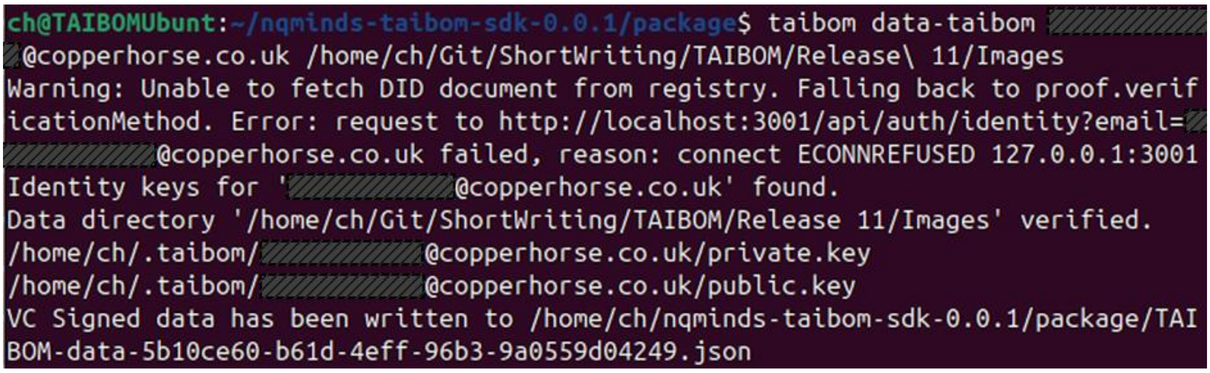
As shown in the picture, TAIBOM can detect that some changes have been made to the dataset as the cryptographic has generated before the changes (c2c09...) is different from the one generated after the changes (a496d...).

TEST fails as expected

7. Pollution of the Dataset

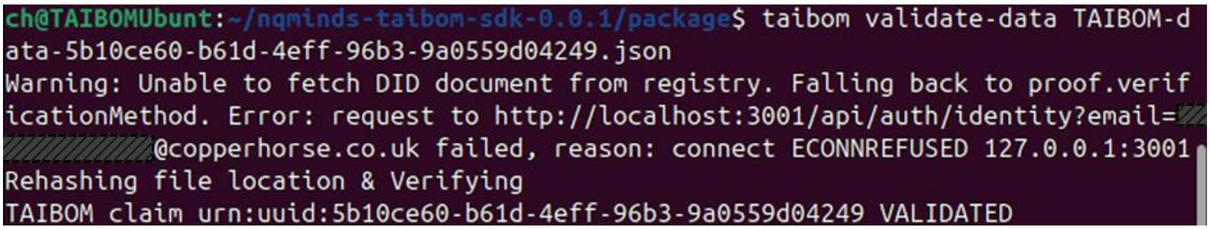
This test evaluates the vulnerability of a TAIBOM to data pollution due to mismanagement of the dataset or an external attack. A polluted dataset could lead to misclassification as the model generated through it is not correctly able to identify the relationship between features extracted from the dataset.

To perform this test, first a TAIBOM signature needs to be generated against the dataset containing all the labels:



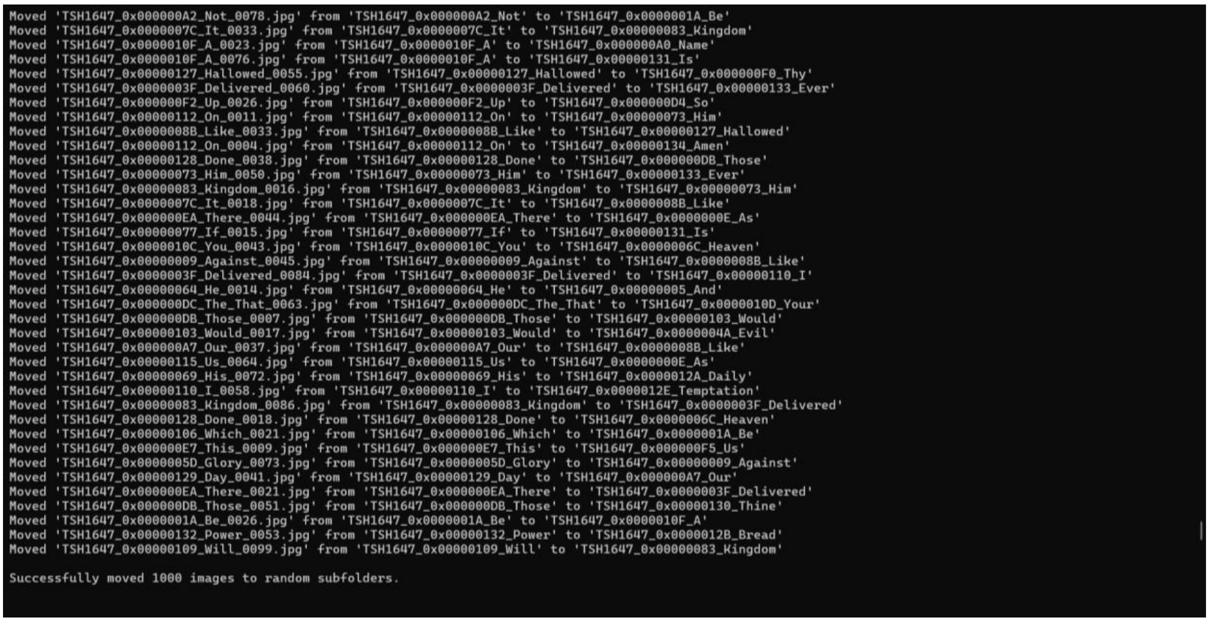
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted]
@copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Images
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-5b10ce60-b61d-4eff-96b3-9a0559d04249.json
```

After this, the signature is validated before any changes are made to the dataset:

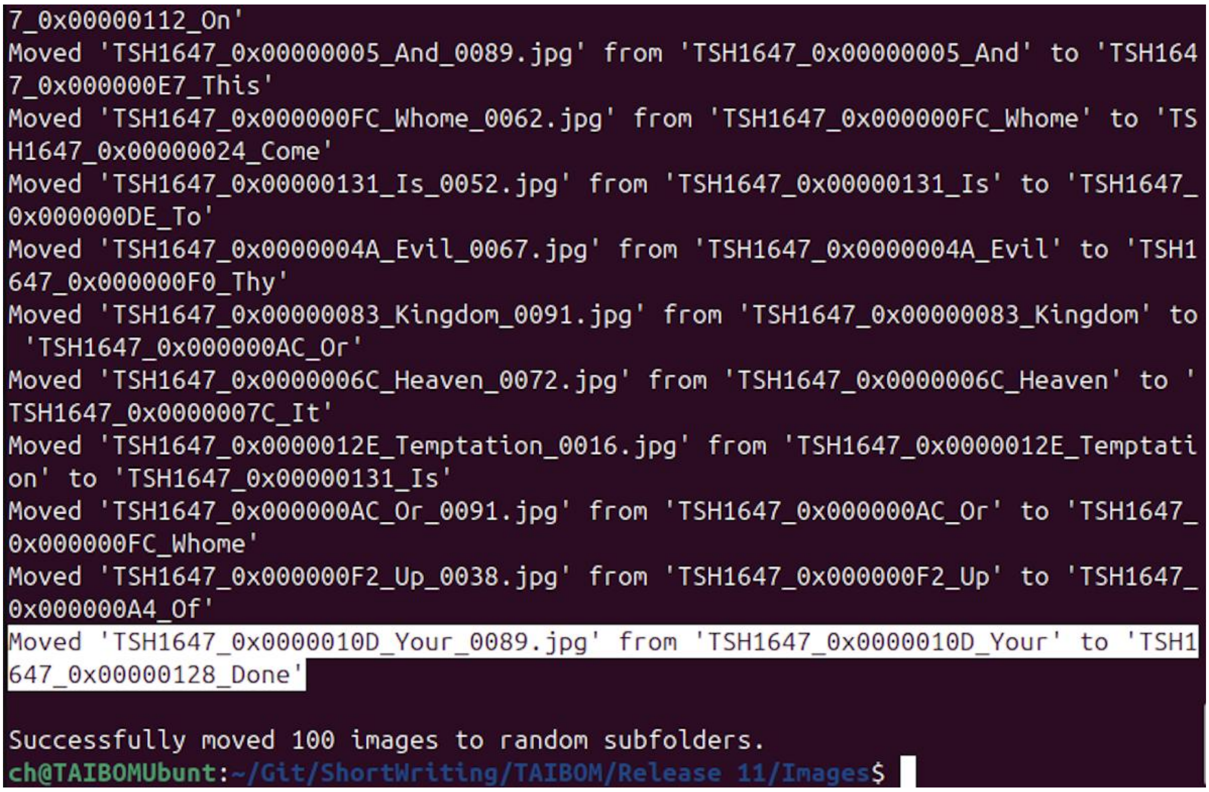


```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-5b10ce60-b61d-4eff-96b3-9a0559d04249.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
TAIBOM claim urn:uuid:5b10ce60-b61d-4eff-96b3-9a0559d04249 VALIDATED
```

Now to simulate the process of polluting a dataset, a python script was written that did the following steps one by one. In the first step, it collected all the images from the dataset, next it randomly selected 1000 images from this collection, and then assigned them back randomly into the labels as shown below:



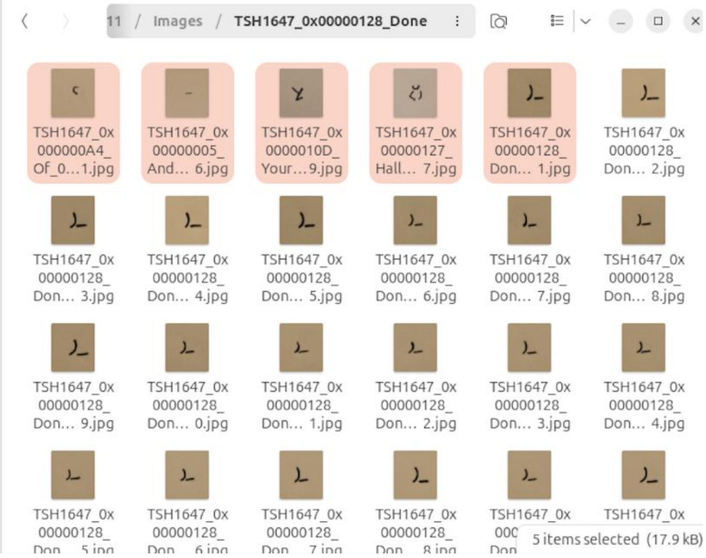
```
Moved 'TSH1647_0x000000A2_Not_0078.jpg' from 'TSH1647_0x000000A2_Not' to 'TSH1647_0x0000001A_Be'
Moved 'TSH1647_0x0000007C_It_0033.jpg' from 'TSH1647_0x0000007C_It' to 'TSH1647_0x00000083_Kingdom'
Moved 'TSH1647_0x0000010F_A_0023.jpg' from 'TSH1647_0x0000010F_A' to 'TSH1647_0x000000A0_Name'
Moved 'TSH1647_0x0000010F_A_0076.jpg' from 'TSH1647_0x0000010F_A' to 'TSH1647_0x00000131_Is'
Moved 'TSH1647_0x00000127_Hallowed_0055.jpg' from 'TSH1647_0x00000127_Hallowed' to 'TSH1647_0x000000F0_Thy'
Moved 'TSH1647_0x0000003F_Delivered_0060.jpg' from 'TSH1647_0x0000003F_Delivered' to 'TSH1647_0x00000133_Ever'
Moved 'TSH1647_0x000000F2_Up_0026.jpg' from 'TSH1647_0x000000F2_Up' to 'TSH1647_0x000000D4_So'
Moved 'TSH1647_0x00000112_On_0011.jpg' from 'TSH1647_0x00000112_On' to 'TSH1647_0x00000073_Him'
Moved 'TSH1647_0x0000008B_Like_0033.jpg' from 'TSH1647_0x0000008B_Like' to 'TSH1647_0x00000127_Hallowed'
Moved 'TSH1647_0x00000112_On_0004.jpg' from 'TSH1647_0x00000112_On' to 'TSH1647_0x00000134_Amen'
Moved 'TSH1647_0x00000128_Done_0038.jpg' from 'TSH1647_0x00000128_Done' to 'TSH1647_0x000000DB_Those'
Moved 'TSH1647_0x00000073_Him_0050.jpg' from 'TSH1647_0x00000073_Him' to 'TSH1647_0x00000133_Ever'
Moved 'TSH1647_0x00000083_Kingdom_0016.jpg' from 'TSH1647_0x00000083_Kingdom' to 'TSH1647_0x00000073_Him'
Moved 'TSH1647_0x0000007C_It_0018.jpg' from 'TSH1647_0x0000007C_It' to 'TSH1647_0x0000008B_Like'
Moved 'TSH1647_0x000000EA_There_0044.jpg' from 'TSH1647_0x000000EA_There' to 'TSH1647_0x0000000E_As'
Moved 'TSH1647_0x00000077_If_0015.jpg' from 'TSH1647_0x00000077_If' to 'TSH1647_0x00000131_Is'
Moved 'TSH1647_0x0000010C_You_0043.jpg' from 'TSH1647_0x0000010C_You' to 'TSH1647_0x0000006C_Heaven'
Moved 'TSH1647_0x00000009_Against_0045.jpg' from 'TSH1647_0x00000009_Against' to 'TSH1647_0x0000008B_Like'
Moved 'TSH1647_0x0000003F_Delivered_0084.jpg' from 'TSH1647_0x0000003F_Delivered' to 'TSH1647_0x00000110_I'
Moved 'TSH1647_0x00000064_He_0014.jpg' from 'TSH1647_0x00000064_He' to 'TSH1647_0x00000005_And'
Moved 'TSH1647_0x000000DC_The_That_0063.jpg' from 'TSH1647_0x000000DC_The_That' to 'TSH1647_0x0000010D_Your'
Moved 'TSH1647_0x000000DB_Those_0007.jpg' from 'TSH1647_0x000000DB_Those' to 'TSH1647_0x00000103_Would'
Moved 'TSH1647_0x00000103_Would_0017.jpg' from 'TSH1647_0x00000103_Would' to 'TSH1647_0x0000004A_Evil'
Moved 'TSH1647_0x000000A7_Our_0037.jpg' from 'TSH1647_0x000000A7_Our' to 'TSH1647_0x0000008B_Like'
Moved 'TSH1647_0x00000115_Us_0064.jpg' from 'TSH1647_0x00000115_Us' to 'TSH1647_0x0000000E_As'
Moved 'TSH1647_0x00000069_His_0072.jpg' from 'TSH1647_0x00000069_His' to 'TSH1647_0x0000012A_Daily'
Moved 'TSH1647_0x00000110_I_0058.jpg' from 'TSH1647_0x00000110_I' to 'TSH1647_0x0000012E_Temptation'
Moved 'TSH1647_0x00000083_Kingdom_0086.jpg' from 'TSH1647_0x00000083_Kingdom' to 'TSH1647_0x0000003F_Delivered'
Moved 'TSH1647_0x00000128_Done_0018.jpg' from 'TSH1647_0x00000128_Done' to 'TSH1647_0x0000006C_Heaven'
Moved 'TSH1647_0x00000106_Which_0021.jpg' from 'TSH1647_0x00000106_Which' to 'TSH1647_0x0000001A_Be'
Moved 'TSH1647_0x000000E7_This_0009.jpg' from 'TSH1647_0x000000E7_This' to 'TSH1647_0x000000F5_Us'
Moved 'TSH1647_0x0000005D_Glory_0073.jpg' from 'TSH1647_0x0000005D_Glory' to 'TSH1647_0x00000009_Against'
Moved 'TSH1647_0x00000129_Day_0041.jpg' from 'TSH1647_0x00000129_Day' to 'TSH1647_0x000000A7_Our'
Moved 'TSH1647_0x000000EA_There_0021.jpg' from 'TSH1647_0x000000EA_There' to 'TSH1647_0x0000003F_Delivered'
Moved 'TSH1647_0x000000DB_Those_0051.jpg' from 'TSH1647_0x000000DB_Those' to 'TSH1647_0x00000130_Thine'
Moved 'TSH1647_0x0000001A_Be_0026.jpg' from 'TSH1647_0x0000001A_Be' to 'TSH1647_0x0000010F_A'
Moved 'TSH1647_0x00000132_Power_0053.jpg' from 'TSH1647_0x00000132_Power' to 'TSH1647_0x0000012B_Bread'
Moved 'TSH1647_0x00000109_Will_0099.jpg' from 'TSH1647_0x00000109_Will' to 'TSH1647_0x00000083_Kingdom'

Successfully moved 1000 images to random subfolders.
```



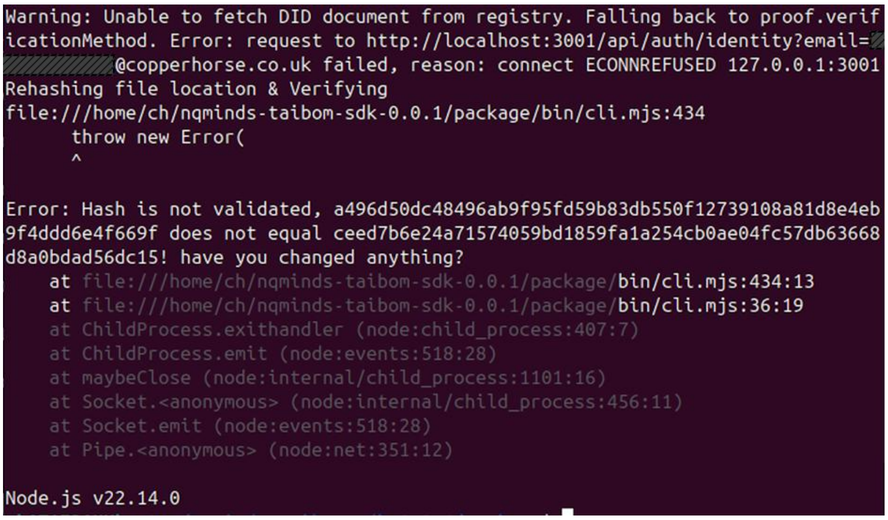
```
7_0x00000112_On'
Moved 'TSH1647_0x00000005_And_0089.jpg' from 'TSH1647_0x00000005_And' to 'TSH164
7_0x000000E7_This'
Moved 'TSH1647_0x000000FC_Whome_0062.jpg' from 'TSH1647_0x000000FC_Whome' to 'TS
H1647_0x00000024_Come'
Moved 'TSH1647_0x00000131_Is_0052.jpg' from 'TSH1647_0x00000131_Is' to 'TSH1647_
0x000000DE_To'
Moved 'TSH1647_0x0000004A_Evil_0067.jpg' from 'TSH1647_0x0000004A_Evil' to 'TSH1
647_0x000000F0_Thy'
Moved 'TSH1647_0x00000083_Kingdom_0091.jpg' from 'TSH1647_0x00000083_Kingdom' to
 'TSH1647_0x000000AC_Or'
Moved 'TSH1647_0x0000006C_Heaven_0072.jpg' from 'TSH1647_0x0000006C_Heaven' to '
TSH1647_0x0000007C_It'
Moved 'TSH1647_0x0000012E_Temptation_0016.jpg' from 'TSH1647_0x0000012E_Temptati
on' to 'TSH1647_0x00000131_Is'
Moved 'TSH1647_0x000000AC_Or_0091.jpg' from 'TSH1647_0x000000AC_Or' to 'TSH1647_
0x000000FC_Whome'
Moved 'TSH1647_0x000000F2_Up_0038.jpg' from 'TSH1647_0x000000F2_Up' to 'TSH1647_
0x000000A4_Of'
Moved 'TSH1647_0x0000010D_Your_0089.jpg' from 'TSH1647_0x0000010D_Your' to 'TSH1
647_0x00000128_Done'

Successfully moved 100 images to random subfolders.
ch@TAIBOMUbunt:~/Git/ShortWriting/TAIBOM/Release 11/Images$
```

As can be seen in the above picture, the image named "TSH1647_0x0000010D_your_0089.jpg" was moved to the label "TSH1647_0x00000128_Done".

An image showing the different images in the “TSH1647_0x00000128_Done” folder is attached below:


11 / Images / TSH1647_0x00000128_Done
TSH1647_0x 000000A4_ Of_0...1.jpg
TSH1647_0x 00000005_ And... 6.jpg
TSH1647_0x 0000010D_ Your...9.jpg
TSH1647_0x 00000127_ Hall... 7.jpg
TSH1647_0x 00000128_ Don... 1.jpg
TSH1647_0x 00000128_ Don... 2.jpg
TSH1647_0x 00000128_ Don... 3.jpg
TSH1647_0x 00000128_ Don... 4.jpg
TSH1647_0x 00000128_ Don... 5.jpg
TSH1647_0x 00000128_ Don... 6.jpg
TSH1647_0x 00000128_ Don... 7.jpg
TSH1647_0x 00000128_ Don... 8.jpg
TSH1647_0x 00000128_ Don... 9.jpg
TSH1647_0x 00000128_ Don... 0.jpg
TSH1647_0x 00000128_ Don... 1.jpg
TSH1647_0x 00000128_ Don... 2.jpg
TSH1647_0x 00000128_ Don... 3.jpg
TSH1647_0x 00000128_ Don... 4.jpg
TSH1647_0x 00000128_ Don... 5.jpg
TSH1647_0x 00000128_ Don... 6.jpg
TSH1647_0x 00000128_ Don... 7.jpg
TSH1647_0x 00000128_ Don... 8.jpg
TSH1647_0x 000
TSH1647_0x
5 items selected (17.9 kB)

After polluting the dataset, the TAIBOM signature is validated again to see if the TAIBOM can detect the changes made:



```
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, a496d50dc48496ab9f95fd59b83db550f12739108a81d8e4eb
9f4ddd6e4f669f does not equal ceed7b6e24a71574059bd1859fa1a254cb0ae04fc57db63668
d8a0bdad56dc15! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
```

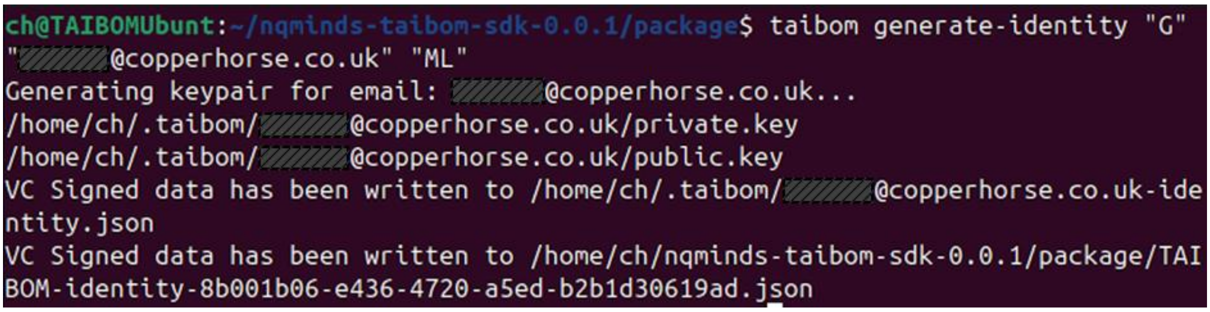
As shown above, TAIBOM can detect that some changes have been made to the dataset as the cryptographic hash stored in the original signature (a496d...) is not the same as the hash generated after the dataset is polluted (ceed7...).

TEST fails as expected

8. Merging code and data with different identity into an AI-system

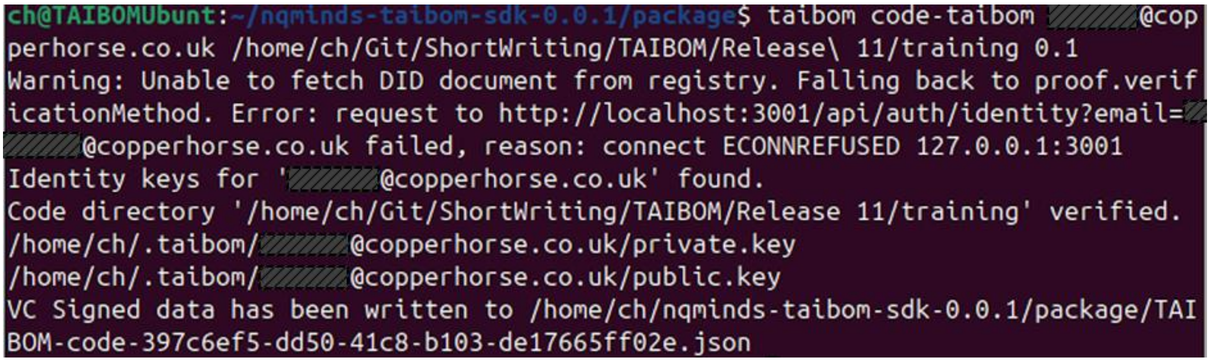
This test is aimed to find out how a TAIBOM handles multiple signatures when creating a system-taibom.

To test, first a new identity is created that would be used to sign the code TAIBOM:



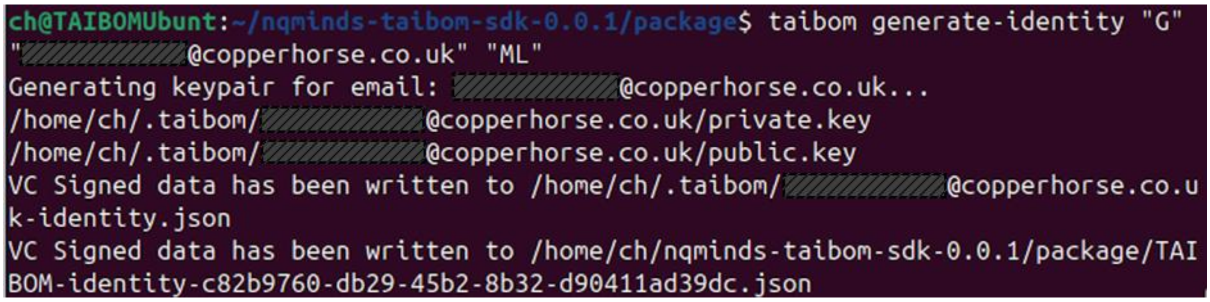
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "G"
"[redacted]@copperhorse.co.uk" "ML"
Generating keypair for email: [redacted]@copperhorse.co.uk...
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/[redacted]@copperhorse.co.uk-ide
ntity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-8b001b06-e436-4720-a5ed-b2b1d30619ad.json
```

With this new identity, a TAIBOM against the folder containing the code of the project is created:



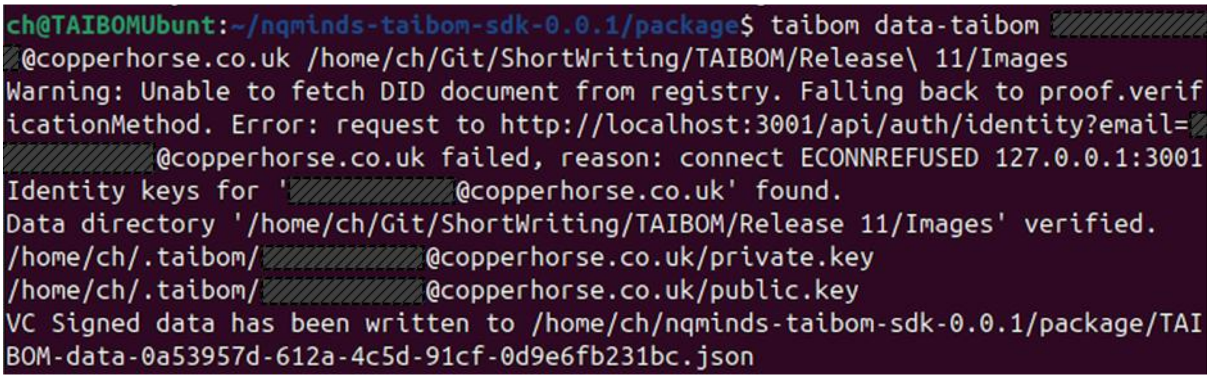
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom code-taibom [redacted]@cop
perhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/training 0.1
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Code directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/training' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-code-397c6ef5-dd50-41c8-b103-de17665ff02e.json
```

The process is repeated as a different identity is created to sign the dataset part of the project:



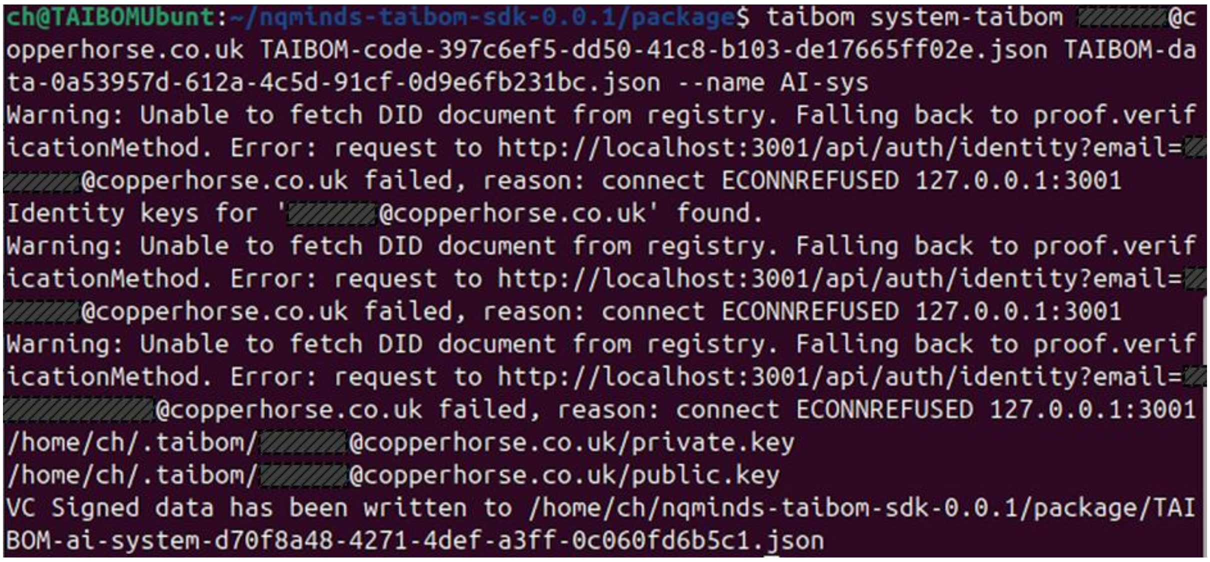
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "G"
"[redacted]@copperhorse.co.uk" "ML"
Generating keypair for email: [redacted]@copperhorse.co.uk...
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/[redacted]@copperhorse.co.u
k-identity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-c82b9760-db29-45b2-8b32-d90411ad39dc.json
```

The data associated with the code is used to create a data-taibom:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted]
@copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Images
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-0a53957d-612a-4c5d-91cf-0d9e6fb231bc.json
```

With the two components needed to create an ai-taibom ready, the system-taibom is created using the same id that was used to create the code.



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom system-taibom [redacted]@c
opperhorse.co.uk TAIBOM-code-397c6ef5-dd50-41c8-b103-de17665ff02e.json TAIBOM-da
ta-0a53957d-612a-4c5d-91cf-0d9e6fb231bc.json --name AI-sys
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-ai-system-d70f8a48-4271-4def-a3ff-0c060fd6b5c1.json
```

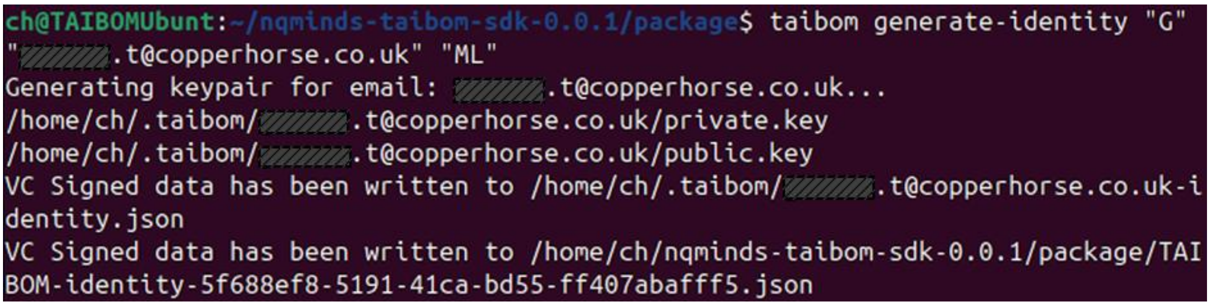
The picture above shows how the system-taibom was successfully generated and the VC {Verifiable certificate} is stored in the package folder. When the system-taibom is examined, it is noted that only the ID used to create the system-taibom is the only one stored in the credentials.

TEST passes as expected

9. Merging multiple data-taiboms with different identities into a datapack

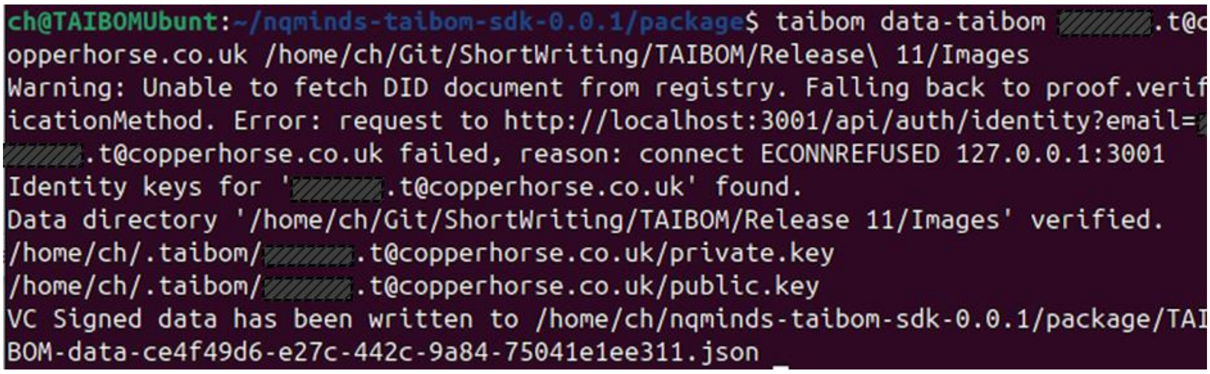
This test is aimed to find out how a TAIBOM handles multiple signatures when creating a datapack.

To test this a new identity is created using which the first dataset is signed through TAIBOM:



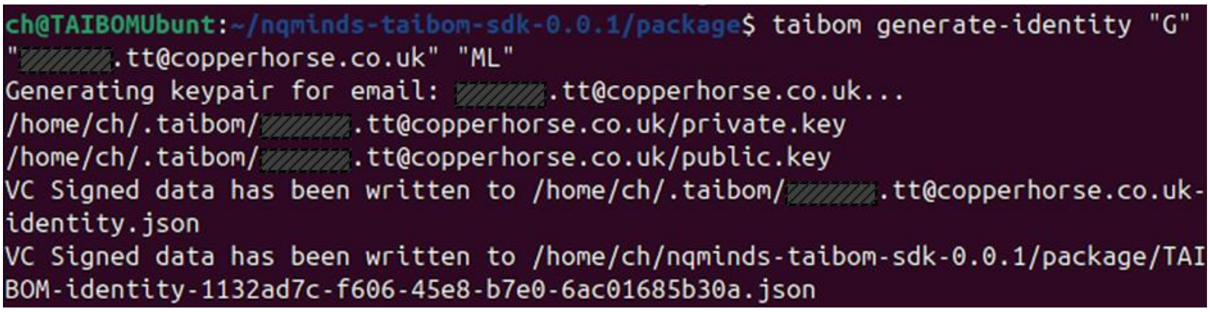
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "G"
"[redacted].t@copperhorse.co.uk" "ML"
Generating keypair for email: [redacted].t@copperhorse.co.uk...
/home/ch/.taibom/[redacted].t@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].t@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/[redacted].t@copperhorse.co.uk-i
dentity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-5f688ef8-5191-41ca-bd55-ff407abafff5.json
```

With this new identity, a TAIBOM against the folder containing the first dataset of the project is created:



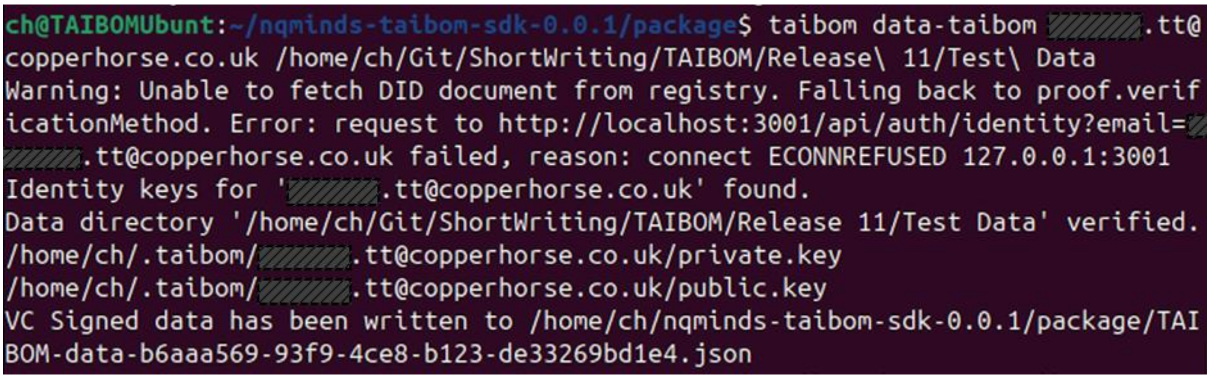
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted].t@c
opperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Images
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
.t@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted].t@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images' verified.
/home/ch/.taibom/[redacted].t@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].t@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-ce4f49d6-e27c-442c-9a84-75041e1ee311.json
```

Another identity is created which is then used to sign a different dataset part of the same project:



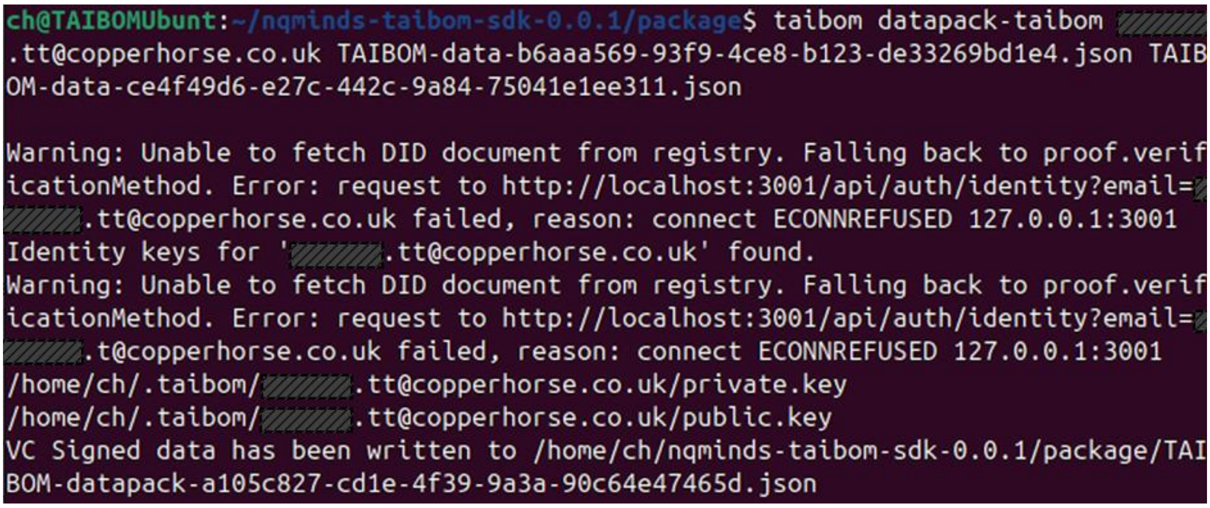
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "G"
"[redacted].tt@copperhorse.co.uk" "ML"
Generating keypair for email: [redacted].tt@copperhorse.co.uk...
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/[redacted].tt@copperhorse.co.uk-
identity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-1132ad7c-f606-45e8-b7e0-6ac01685b30a.json
```

This identity is now used to sign another part of the dataset with TAIBOM:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted].tt@
copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Test\ Data
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
[redacted].tt@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted].tt@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Test Data' verified.
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-b6aaa569-93f9-4ce8-b123-de33269bd1e4.json
```

With the minimum two components needed to create a datapack ready, a datapack is created using the same id that was used to create the second dataset:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom datapack-taibom [redacted]
.tt@copperhorse.co.uk TAIBOM-data-b6aaa569-93f9-4ce8-b123-de33269bd1e4.json TAIB
OM-data-ce4f49d6-e27c-442c-9a84-75041e1ee311.json

Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
.tt@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted].tt@copperhorse.co.uk' found.
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
.t@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-datapack-a105c827-cd1e-4f39-9a3a-90c64e47465d.json
```

The picture above shows how the datapack-taibom was successfully generated and the VC {Verifiable certificate} is stored in the package folder. When the datapack-taibom is examined, it is noted that only the id used to create the datapack-taibom is the only one stored in the credentials.

TEST passes as expected

10. Introducing bias in the inferencing engine

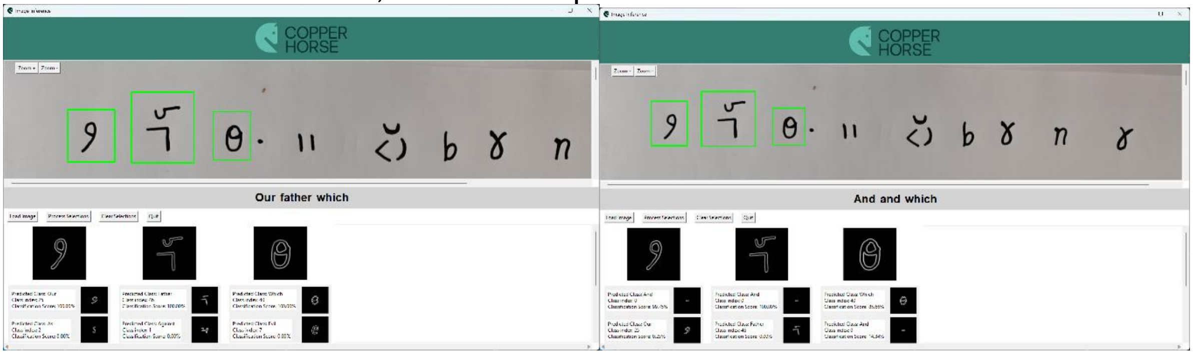
This test evaluates the capability of a TAIBOM to detect changes in the inference script aimed at introducing bias. The test is performed on **Release 11**.

This test is performed by scaling the confidence score list output of the classification CNN by a list of weights. The list of weights is heavily unbalanced, such that one specific class will have a much higher probability of being selected as the result of the classification.

The inference script has been modified as follows:

```
biased_class = 0
epsilon = 1e-20
bias_weight = [epsilon] * int(tf.shape(prediction)[0].numpy())
bias_weight[biased_class] = 1.0
prediction = prediction * bias_weight
module_prediction = sum(prediction)
prediction = [prediction_i / module_prediction for prediction_i in
prediction]
```

The biased class is class 0, which corresponds to the word "and".



Our father which

Predicted Class: Our
Class index: 75
Classification Score: 100.00%

Predicted Class: As
Class index: 2
Classification Score: 0.00%

Predicted Class: Father
Class index: 76
Classification Score: 100.00%

Predicted Class: Against
Class index: 1
Classification Score: 0.00%

Predicted Class: Which
Class index: 41
Classification Score: 100.00%

Predicted Class: Fell
Class index: 7
Classification Score: 0.00%

And and which

Predicted Class: And
Class index: 0
Classification Score: 99.75%

Predicted Class: Our
Class index: 75
Classification Score: 0.25%

Predicted Class: And
Class index: 0
Classification Score: 100.00%

Predicted Class: Father
Class index: 46
Classification Score: 0.00%

Predicted Class: Which
Class index: 41
Classification Score: 85.66%

Predicted Class: And
Class index: 0
Classification Score: 14.34%

The screenshots show the result of the classification of few symbols before and after the modification. As visible the classification scores favour the translation of the symbols with the word "and".

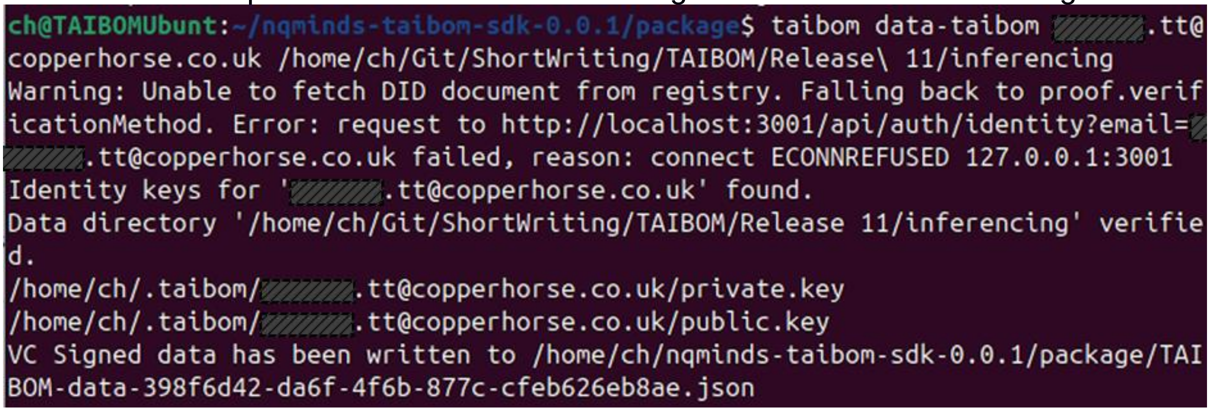
TEST fails as expected

11. Pollution during Inferencing

This test is aimed at how a TAIBOM detects if any changes have been made to the inference script.

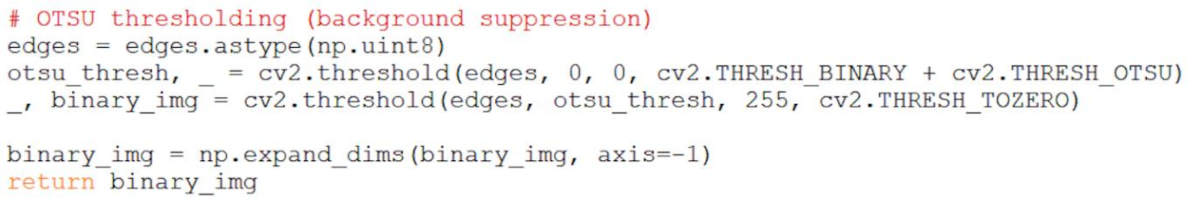
This was done by changing the thresholding parameter of the Otsu thresholding (responsible for separating the background from the symbols).

To do this, first a TAIBOM was created against the Inferencing folder as it contains the inference script. The result of the created signature is shown in the image below:



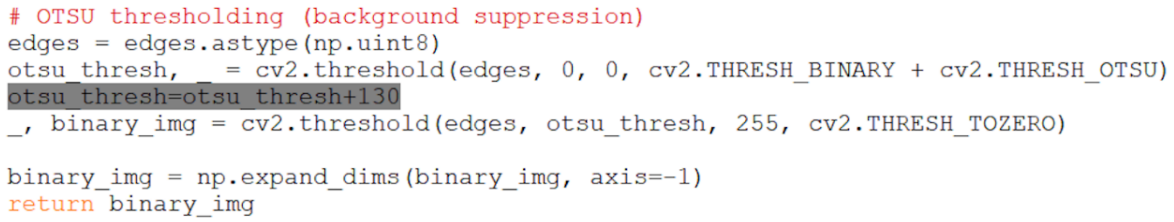
```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted].tt@
copperhorse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/inferencing
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
.tt@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted].tt@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/inferencing' verifie
d.
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted].tt@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-398f6d42-da6f-4f6b-877c-cfeb626eb8ae.json
```

The original parameters for the Otsu Thresholding and code are shown in the image below:



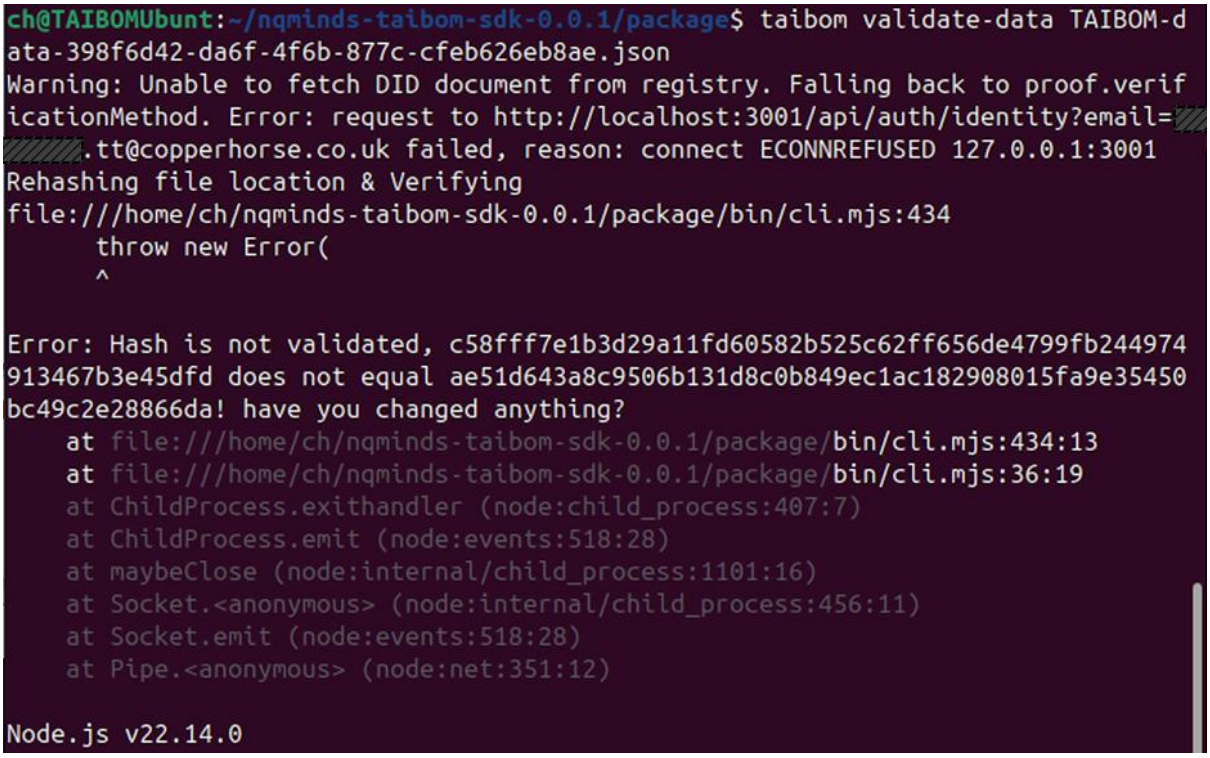
```
# OTSU thresholding (background suppression)
edges = edges.astype(np.uint8)
otsu_thresh, _ = cv2.threshold(edges, 0, 0, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
_, binary_img = cv2.threshold(edges, otsu_thresh, 255, cv2.THRESH_TOZERO)

binary_img = np.expand_dims(binary_img, axis=-1)
return binary_img
```

To degrade the quality of background separation, the line of code `otsu_thresh=otsu_thresh+130` was introduced, creating a shift from the optimal threshold as shown below:



```
# OTSU thresholding (background suppression)
edges = edges.astype(np.uint8)
otsu_thresh, _ = cv2.threshold(edges, 0, 0, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
otsu_thresh=otsu_thresh+130
_, binary_img = cv2.threshold(edges, otsu_thresh, 255, cv2.THRESH_TOZERO)

binary_img = np.expand_dims(binary_img, axis=-1)
return binary_img
```

With the changes made, the TAIBOM signature is validated again to see if TAIBOM can detect the changes that have been made to the code.



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom validate-data TAIBOM-d
ata-398f6d42-da6f-4f6b-877c-cfeb626eb8ae.json
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
[redacted].tt@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Rehashing file location & Verifying
file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434
      throw new Error(
      ^

Error: Hash is not validated, c58fff7e1b3d29a11fd60582b525c62ff656de4799fb244974
913467b3e45dfd does not equal ae51d643a8c9506b131d8c0b849ec1ac182908015fa9e35450
bc49c2e28866da! have you changed anything?
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:434:13
    at file:///home/ch/nqminds-taibom-sdk-0.0.1/package/bin/cli.mjs:36:19
    at ChildProcess.exithandler (node:child_process:407:7)
    at ChildProcess.emit (node:events:518:28)
    at maybeClose (node:internal/child_process:1101:16)
    at Socket.<anonymous> (node:internal/child_process:456:11)
    at Socket.emit (node:events:518:28)
    at Pipe.<anonymous> (node:net:351:12)

Node.js v22.14.0
```

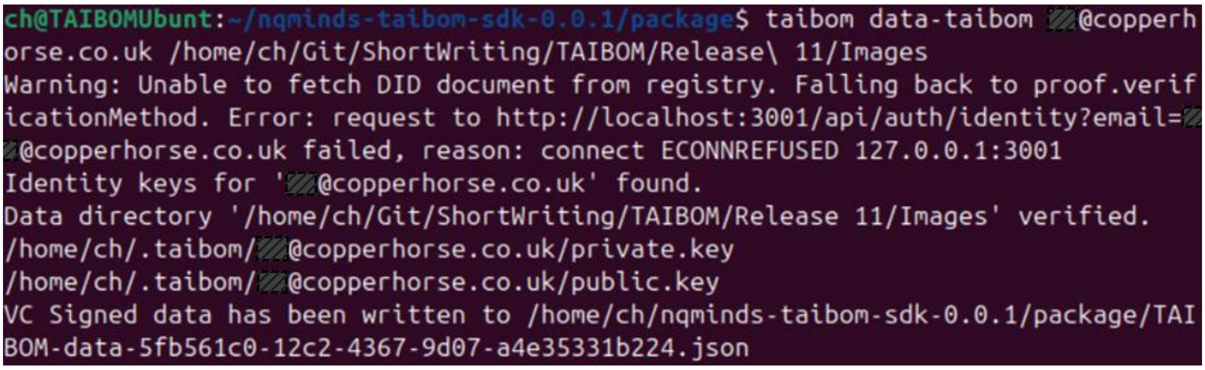
TEST fails as expected

12. Stealing TAIBOM identity tokens

This test evaluates the possibility of using a TAIBOM identity created on a different system to generate and validate TAIBOM hash.

This is done copying the .json and the public and private key files created on one machine and to another. The identity is then used to create a hash of a data folder. The data folder has the same structure in both systems. The hash in the .json file is then passed to the original machine where the validation is performed.

The hash on the second machine is generated without error using the copied identity.



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom data-taibom [redacted]@copperh
orse.co.uk /home/ch/Git/ShortWriting/TAIBOM/Release\ 11/Images
Warning: Unable to fetch DID document from registry. Falling back to proof.verif
icationMethod. Error: request to http://localhost:3001/api/auth/identity?email=[redacted]
@copperhorse.co.uk failed, reason: connect ECONNREFUSED 127.0.0.1:3001
Identity keys for '[redacted]@copperhorse.co.uk' found.
Data directory '/home/ch/Git/ShortWriting/TAIBOM/Release 11/Images' verified.
/home/ch/.taibom/[redacted]@copperhorse.co.uk/private.key
/home/ch/.taibom/[redacted]@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-data-5fb561c0-12c2-4367-9d07-a4e35331b224.json
```

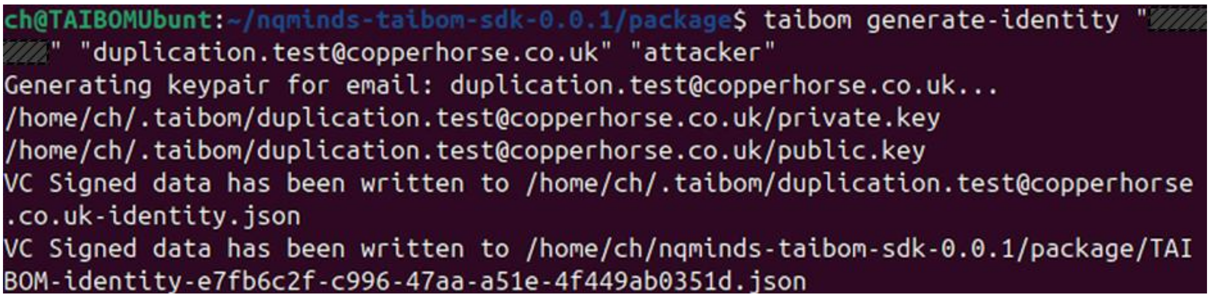
The validation of the hash on the original machine is successful.

TEST passes as expected

13. Overwriting TAIBOM identity tokens

This test tries to overwrite the identity token of a user to try to create a false identity that could be used to run TAIBOM commands. The TAIBOM program writes the private and public key to a hidden folder that uses the email as a folder name. TAIBOM uses a pair of keys: a public key for encryption of TAIBOM and a private key for decryption, ensuring that the identity of the user using TAIBOM remains valid. Data encrypted with a public key can only be decrypted with the corresponding private key, and vice versa.

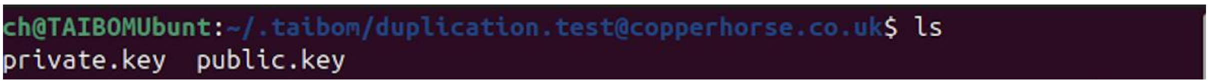
To test whether an attacker can overwrite the private key of a user, first a new user identity is generated as shown below:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "[redacted]" "duplication.test@copperhorse.co.uk" "attacker"
Generating keypair for email: duplication.test@copperhorse.co.uk...
/home/ch/.taibom/duplication.test@copperhorse.co.uk/private.key
/home/ch/.taibom/duplication.test@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/duplication.test@copperhorse
.co.uk-identity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-e7fb6c2f-c996-47aa-a51e-4f449ab0351d.json
```

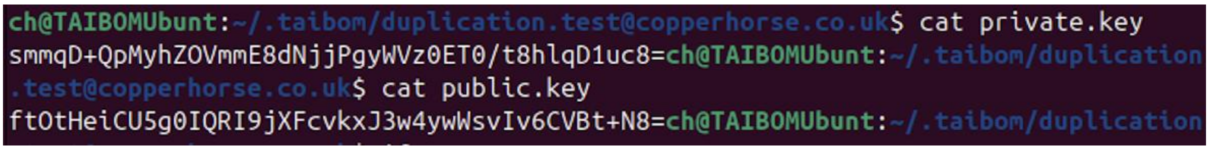
The next step is to navigate to the hidden folder where the keys are stored as shown in the image above. This can be done by opening a new terminal and entering the command: ~/.taibom/duplication.test@copperhorse.co.uk

A `ls` command lists the files in the current directory as shown below:



```
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ ls
private.key  public.key
```

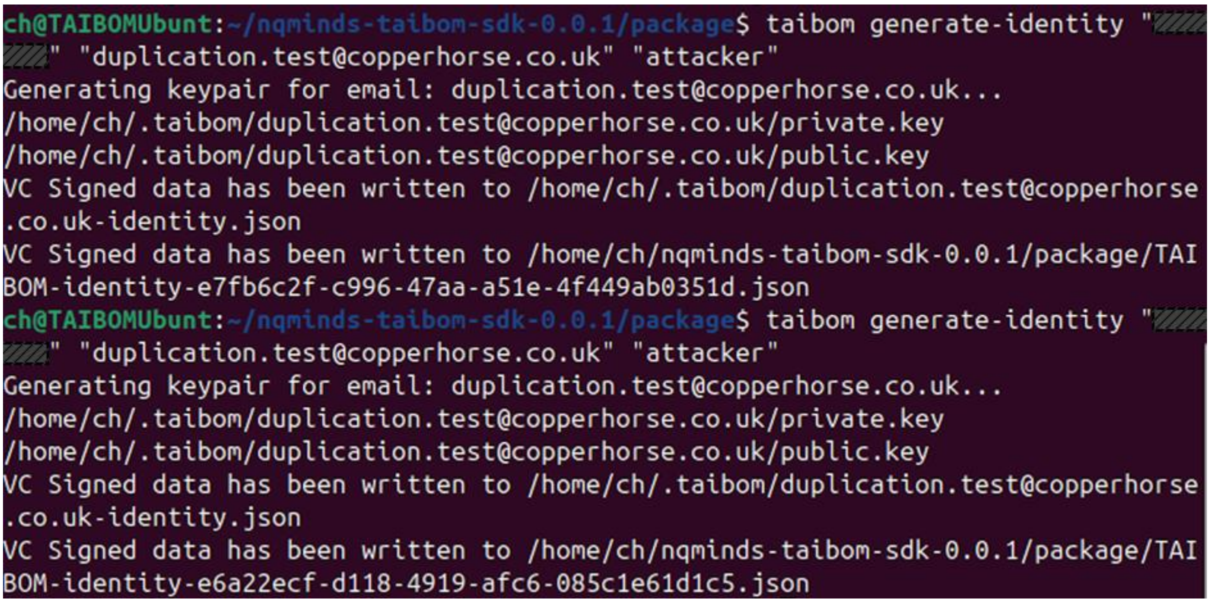
The **cat** command is used to read the contents of the private and public key as shown below:



```
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ cat private.key
smmqD+QpMyhZOVmmE8dNjjPgyWVz0ET0/t8hlqD1uc8=ch@TAIBOMUbunt:~/.taibom/duplication
.test@copperhorse.co.uk$ cat public.key
ft0tHeiCU5g0IQRI9jXFcvkxJ3w4ywWsvIv6CVBt+N8=ch@TAIBOMUbunt:~/.taibom/duplication
```

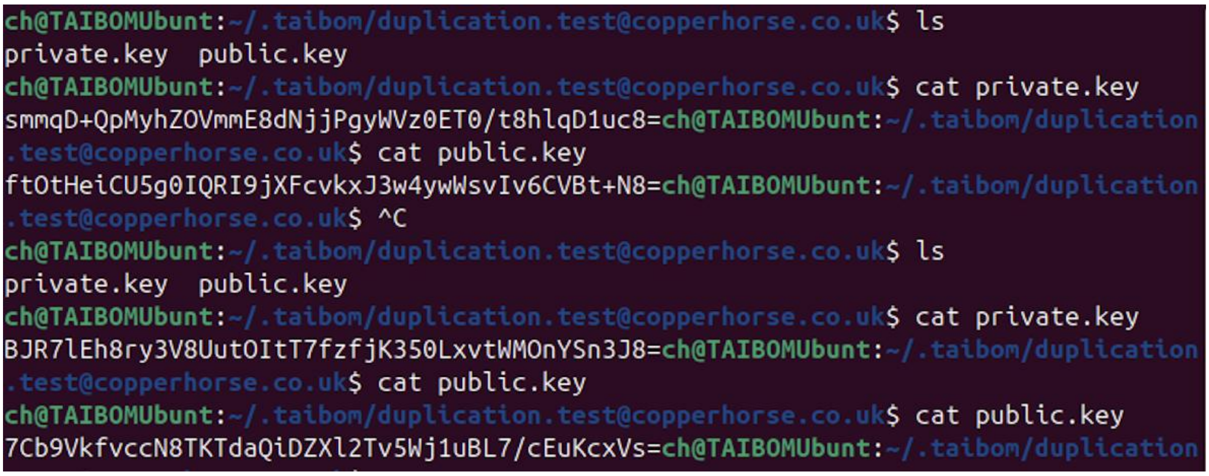
In the picture, after the **cat private.key** command is run, a series of letters (smmq...) are shown, which are the contents of the private key and the series of letters after **cat public.key** (ft0tH...) are the contents of the public key.

In the next step, the objective is to create a new user using the same credentials as used before to overwrite the public and private keys thus invalidating the generated TAIBOM's as well as gain access to the TAIBOM SDK as a fake user:



```
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "[redacted]" "duplication.test@copperhorse.co.uk" "attacker"
Generating keypair for email: duplication.test@copperhorse.co.uk...
/home/ch/.taibom/duplication.test@copperhorse.co.uk/private.key
/home/ch/.taibom/duplication.test@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/duplication.test@copperhorse
.co.uk-identity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-e7fb6c2f-c996-47aa-a51e-4f449ab0351d.json
ch@TAIBOMUbunt:~/nqminds-taibom-sdk-0.0.1/package$ taibom generate-identity "[redacted]" "duplication.test@copperhorse.co.uk" "attacker"
Generating keypair for email: duplication.test@copperhorse.co.uk...
/home/ch/.taibom/duplication.test@copperhorse.co.uk/private.key
/home/ch/.taibom/duplication.test@copperhorse.co.uk/public.key
VC Signed data has been written to /home/ch/.taibom/duplication.test@copperhorse
.co.uk-identity.json
VC Signed data has been written to /home/ch/nqminds-taibom-sdk-0.0.1/package/TAI
BOM-identity-e6a22ecf-d118-4919-afc6-085c1e61d1c5.json
```

While generating the new identity, it clearly states that the private and public key have been generated into the same hidden folder. Navigating back to the hidden folder and running the same commands to find out if the private key and public key have been changed or not.



```
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ ls
private.key  public.key
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ cat private.key
smmqD+QpMyhZOVmmE8dNjjPgyWVz0ET0/t8hlqD1uc8=ch@TAIBOMUbunt:~/.taibom/duplication
.test@copperhorse.co.uk$ cat public.key
ftOtHeiCU5g0IQRI9jXFcvkxJ3w4ywWsvIv6CVBt+N8=ch@TAIBOMUbunt:~/.taibom/duplication
.test@copperhorse.co.uk$ ^C
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ ls
private.key  public.key
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ cat private.key
BJR7lEh8ry3V8UutOItT7fzfjK350LxvtWMOnYSn3J8=ch@TAIBOMUbunt:~/.taibom/duplication
.test@copperhorse.co.uk$ cat public.key
ch@TAIBOMUbunt:~/.taibom/duplication.test@copperhorse.co.uk$ cat public.key
7Cb9VkfvccN8TKTdaQiDZXl2Tv5Wj1uBL7/cEuKcxVs=ch@TAIBOMUbunt:~/.taibom/duplication
```

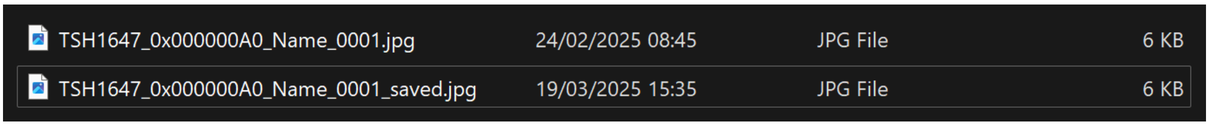
As shown in the above picture, after the `cat private.key` command is run the second time, a series of letters (BJR7l...) are shown which are the contents of the private key and the series of letters after `cat public.key` (7Cb9V...) are the contents of the public key. Compared to the previous iteration of the command, the public and private key have indeed changed.

TEST passes contrary to expectations.

14. Modifying training data metadata

This test is designed to probe the ability of a TAIBOM to detect changes in the files metadata.

This is done by creating a copy of one of the training images, which will appear identical to the original with the only difference being in the <date last modified> metadata field.



TSH1647_0x000000A0_Name_0001.jpg	24/02/2025 08:45	JPG File	6 KB
TSH1647_0x000000A0_Name_0001_saved.jpg	19/03/2025 15:35	JPG File	6 KB

A TAIBOM-data hash of the original dataset was created. Then the original image is replaced with the modified copy, then the TAIBOM-data hash is then checked and verified.

TEST successfully validates TAIBOM

Demonstration Attacks Using the Copper Horse AI Models

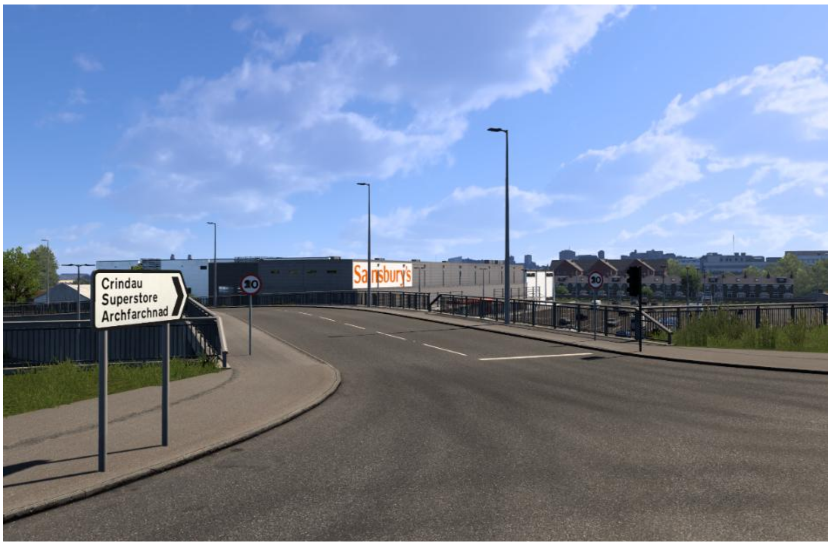
This section describes demonstrations of attacks to the two use cases which were developed into AI models. The attacks represent a selection of the scenarios described in D5.6B.

These demos are designed to illustrate the effects of modification by malicious agents on the training dataset, on the inferencing engine, or on the trained model weights, aiming at disrupting the intended output of an AI model.

The aim is that with the adoption of the TAIBOM scheme, such modifications can be detected, ultimately preventing disruptions on the AI model usage and ensuring the integrity of every component of the AI system.

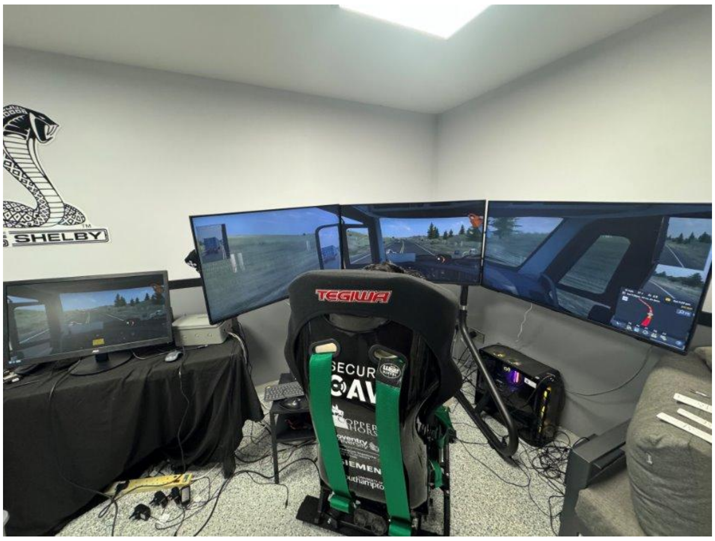
The demonstrations will be showcased during presentations and live events.

In the case of the automotive use case, Copper Horse built an AI model that could be trained against a modified version of the car simulation software that the company was already using. This internal modification was based on ProMods, itself a modification of the game Euro Truck Simulator 2. This ultimately resulted in an adversarial simulation platform that could be used against the AI models, allowing users to not only experience being hacked on the simulator, but to experience what happens when the AI of a vehicle gets attacked. The use case itself is designed to represent an Advanced Driver Assistance System (ADAS) system which is being used to detect defaced signs. This example is pertinent has been a particular problem in Wales in 2024/25 because of a nationwide reduction in speed limits to 20mph, resulting in a widespread backlash and many spray-painted signs in protest. The AI model is designed to detect such tampering or defacement and then report that back to the system or driver. This is the starting point of the use case – the system functioning as designed. Any abuse cases are then applied against that use case.



Copper Horse modified signage in its car hacking simulator

The rig consists of a triple monitor setup linked together as one display, and a second display that mirrors the centre monitor. The second display is then routed into a capture card, which converts the signal to a webcam input. A second machine receives the video input into Open Broadcaster Software (OBS), and outputs it as a virtual camera to be used locally by the inference script. Alternatively, videos recorded in the real world with a dash cam can be streamed to the inferencing engine.



The Copper Horse car hacking simulator rig used to demonstrate the Sign Detection use case AI model with adversarial attacks.

The Short Writing use case aimed to create an AI model which could decipher 17th century shorthand (or short writing), specifically resulting in a model against Thomas Shelton's 1647 shorthand 'Tachygraphy'. This has proven to be extensible against other types of logographic texts. The use case can be interacted with and demonstrated using images of original 17th century documents, or handwritten shorthand text. The latter can also be created on the spot – at Bletchley Park this was done during the demonstration using a quill and ink. The image containing the shorthand text is uploaded to a laptop where the user manually selects the individual symbols to be passed to the inferencing engine, which displays the translation. Abuse cases are designed to attack this functioning model, but also represent a more abstract problem – that of protecting archival information and the meaning of it – whether it be ancient languages or early modern shorthands, where the information and even original documents / artifacts may not exist in the future.

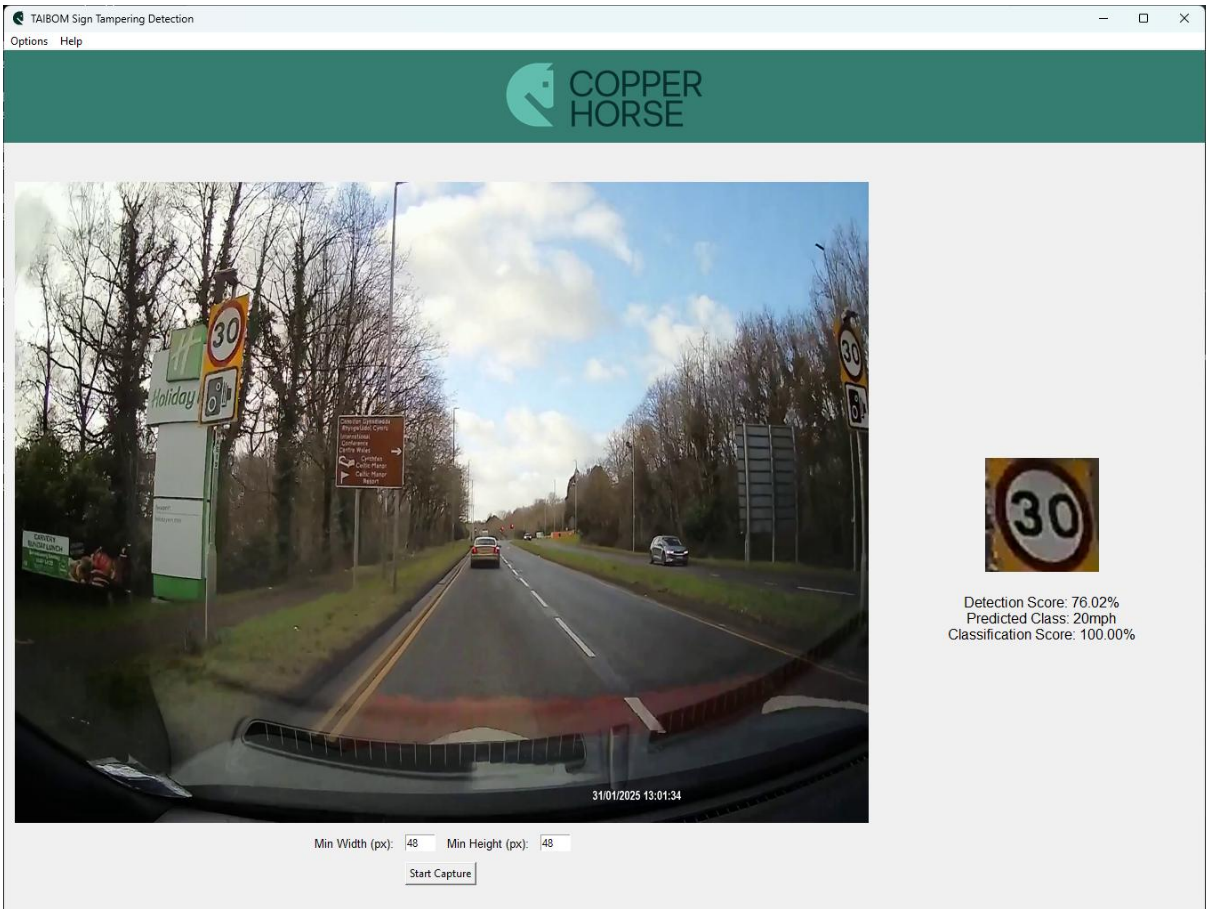
Demonstration of the Short Writing use case AI model.

Sign Detection Demo Scenarios

1. Introducing bias in model weights

The trained model weights are modified to introduce a bias toward one specific class. In this scenario, a python script modifies the weights of the last output layers. Specifically, a large numerical value (100.0) is added to one element of the bias vector of the final output layer of the model.

During inferencing, the class corresponding to the modified element of the bias vector, is much more likely to be returned. In this case, the bias vector element corresponding to the first class of the second classification CNN ("20mph") is artificially inflated.



Detection Score: 76.02%
Predicted Class: 20mph
Classification Score: 100.00%

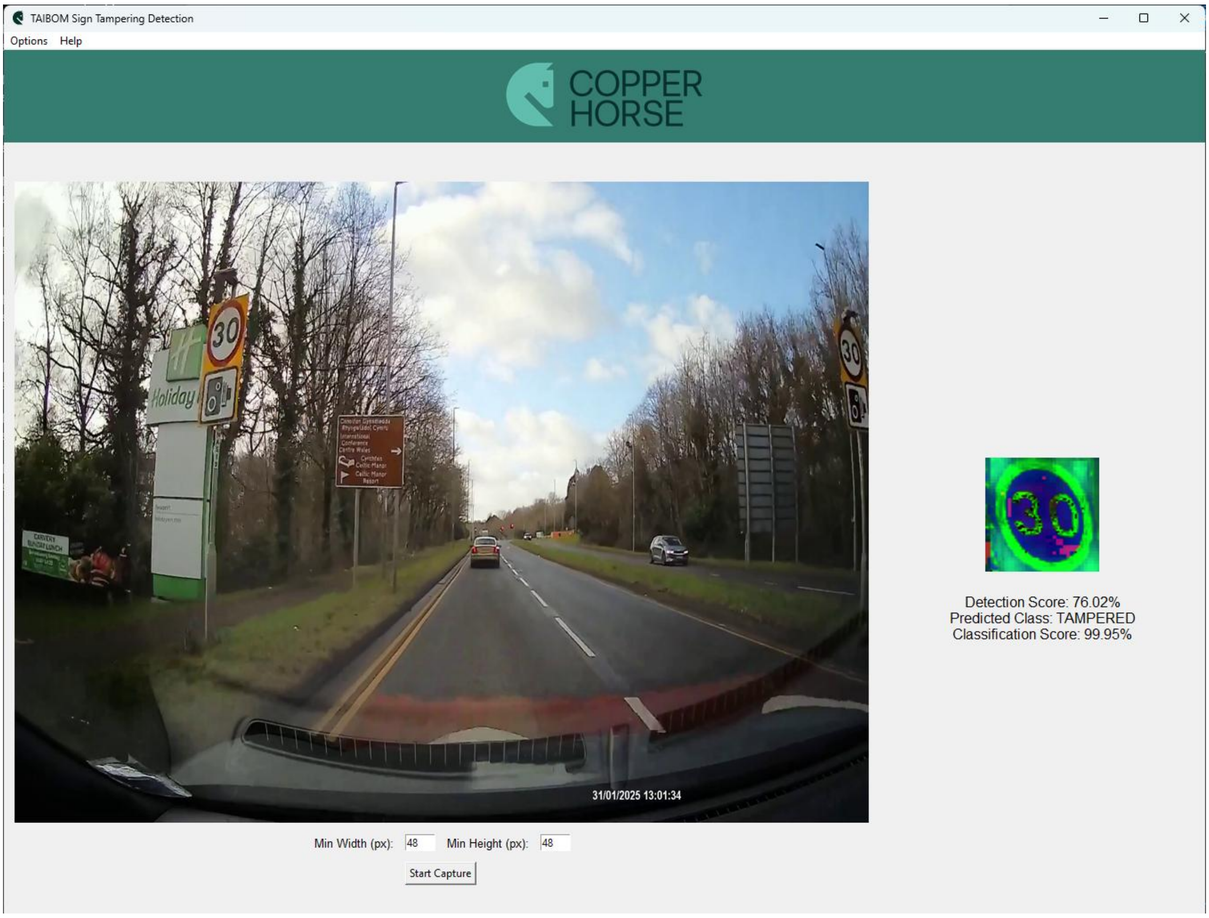
Min Width (px): 48
Min Height (px): 48

2. Changing colour scheme in inferencing

The colour scheme of the detected speed limit sign is modified in the inferencing engine. The object detection YOLO model detects the speed limit sign and determines the bounding box around it. The bounding box is used to crop the image of the speed limit sign, which is then passed sequentially to the two classification CNNs.

The colour scheme of the cropped image of the speed limit sign is modified from RGB (Red, Green, Blue) to HSV (Hue, Saturation, Value).

The first CNN, which distinguishes between legit, tampered and rear side speed limits, and which has been trained on RGB images, misinterprets the colours of the cropped image and misclassifies the speed sign. Generally, a legit speed limit sign is classified as "tampered" because the image presents a range of unexpected colours.

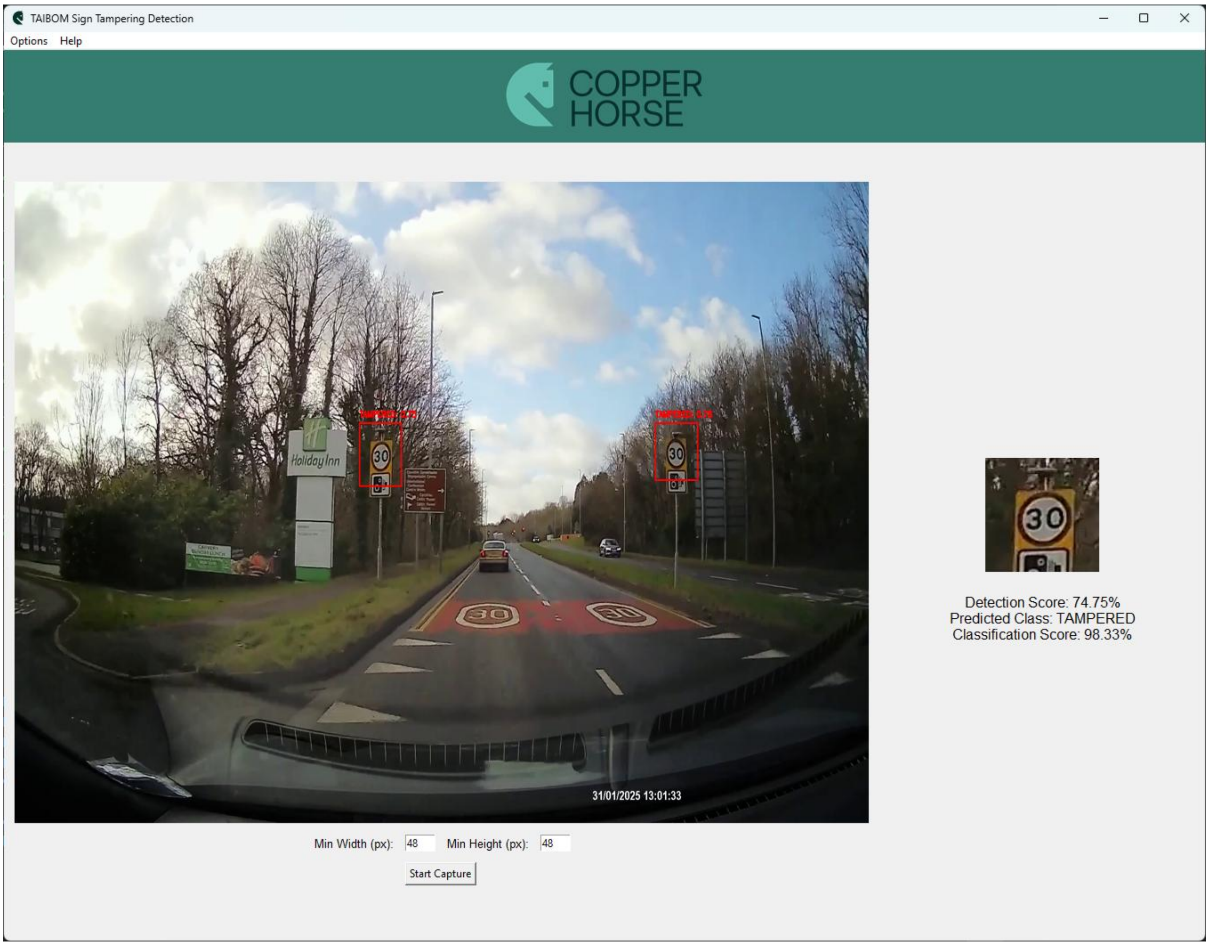


Detection Score: 76.02%
Predicted Class: TAMPERED
Classification Score: 99.95%

3. *Modifying YOLO training dataset bounding boxes*

The training dataset for the object detection YOLO model is modified to detect larger bounding boxes around the speed limit signs.
The .txt files in the YOLO training dataset, which contain the coordinates of the bounding boxes, are modified with a python script. Specifically, the “height” and “width” values of each bounding box are multiplied by a factor 2. The YOLO object detection model is therefore trained to identify larger bounding boxes around the speed limit signs.

At inferencing, the bounding boxes determine the cropping of the image passed to the CNNs. The cropped images now contain large portions of background, and this leads to misclassifications of the speed limit signs by the CNNs. Generally, because of unexpected colours from the background, the speed limit signs are classified as "tampered" by the first CNN.



Detection Score: 74.75%
Predicted Class: TAMPERED
Classification Score: 98.33%

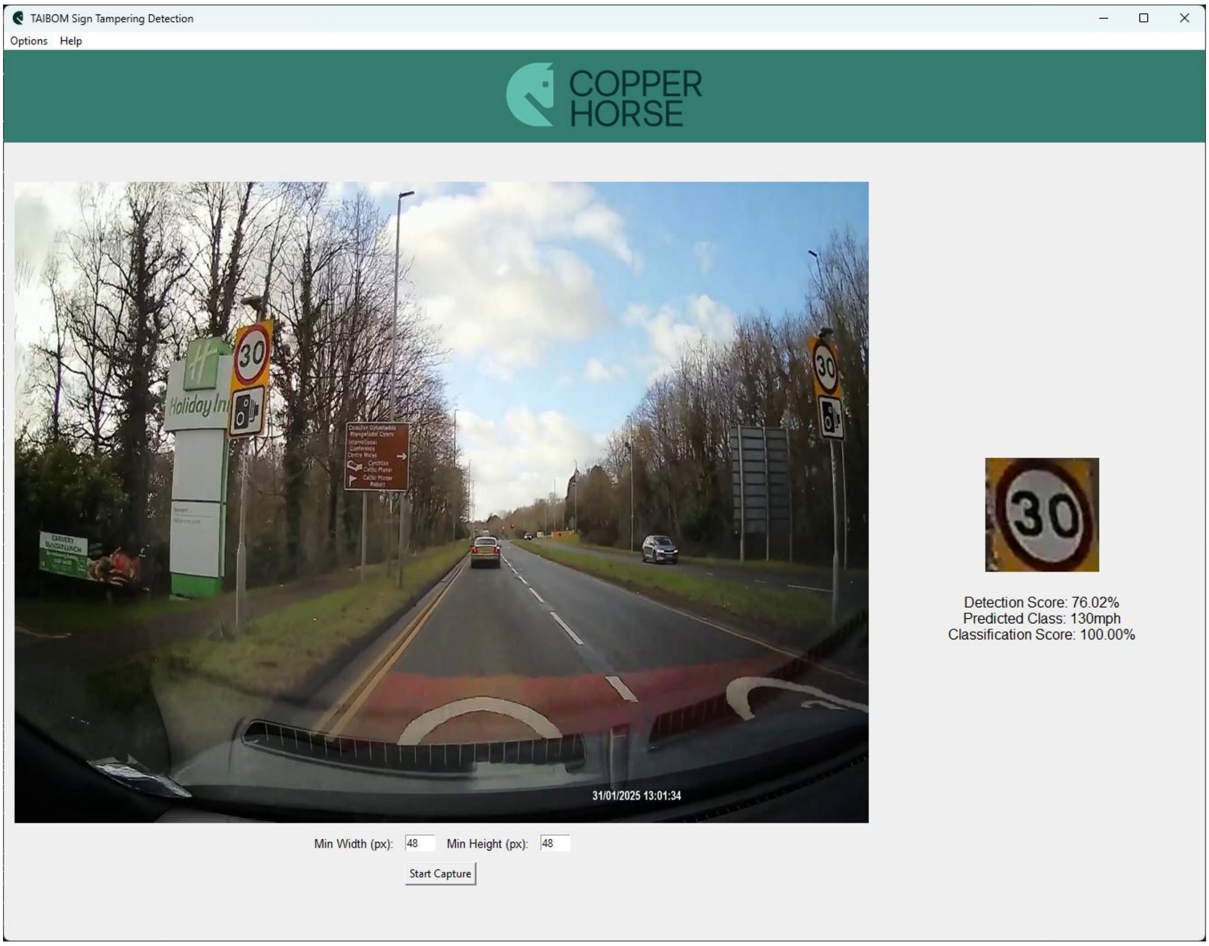
Min Width (px): 48 Min Height (px): 48

4. Change inference labels output

The hardcoded output labels in the inferencing engine are modified to display incorrect speed limits. The speed limit labels are modified as follows:

20mph -> 120mph
30mph -> 130mph
40mph -> 140mph
50mph -> 150mph
National-Speed-Limit -> International-Speed-Limit
TAMPERED! -> ALTERED!

At inferencing, the output label of the speed limit signs will be therefore displayed with the new modified labels.



Detection Score: 76.02%
Predicted Class: 130mph
Classification Score: 100.00%

Min Width (px): 48
Min Height (px): 48

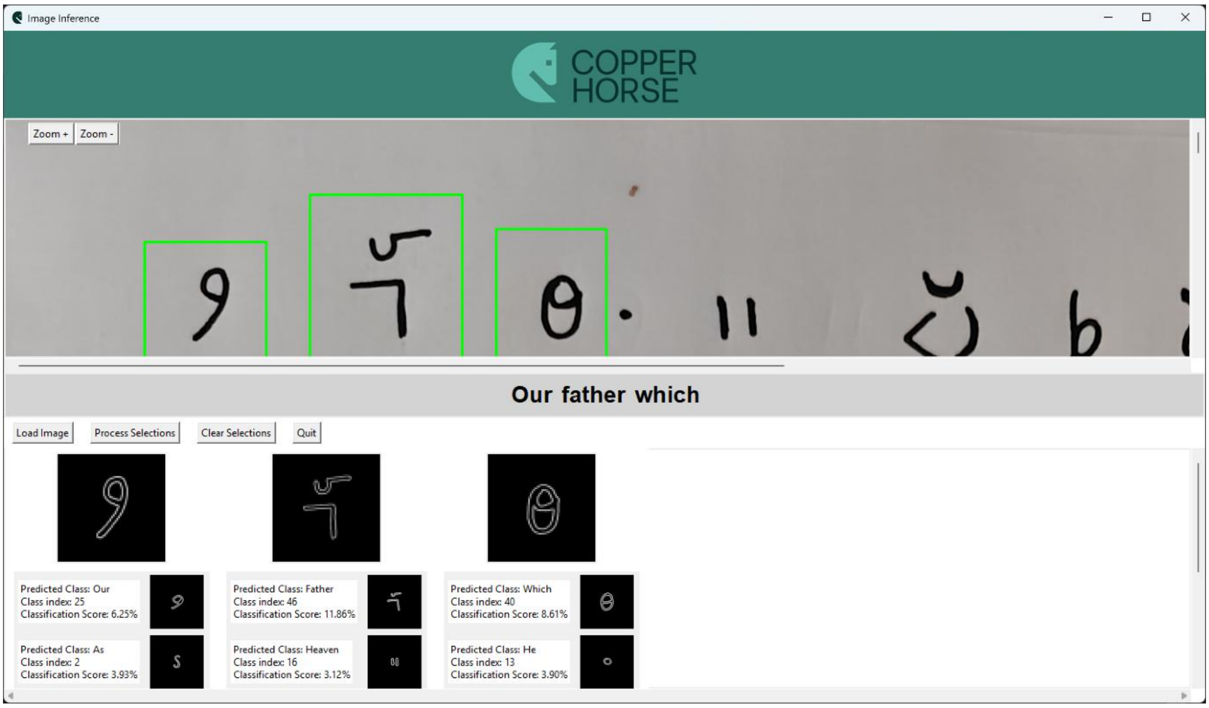
Short Writing Demo Scenarios

1. Pollution of the Dataset

The dataset is modified by randomly shuffling images in the training dataset. The labels of the images (i.e. the translations of the symbols) are assigned according to the name of the folder they are located in. Shuffling the images across the folders effectively mislabel the symbols in the training dataset.

In this scenario, 6000 images in the training dataset are randomly shuffled across the subfolders. With this modification, the model struggles to learn the features of the symbols during training. The metrics of the training reflect this frustration, and the model performance on the test data is worse.

During inferencing, the model returns occasional misclassifications of the symbols and often poorer classification scores.



Our father which

Predicted Class: Our
Class index: 25
Classification Score: 6.25%

Predicted Class: As
Class index: 2
Classification Score: 3.93%

Predicted Class: Father
Class index: 46
Classification Score: 11.86%

Predicted Class: Heaven
Class index: 16
Classification Score: 3.12%

Predicted Class: Which
Class index: 40
Classification Score: 8.61%

Predicted Class: He
Class index: 13
Classification Score: 3.90%

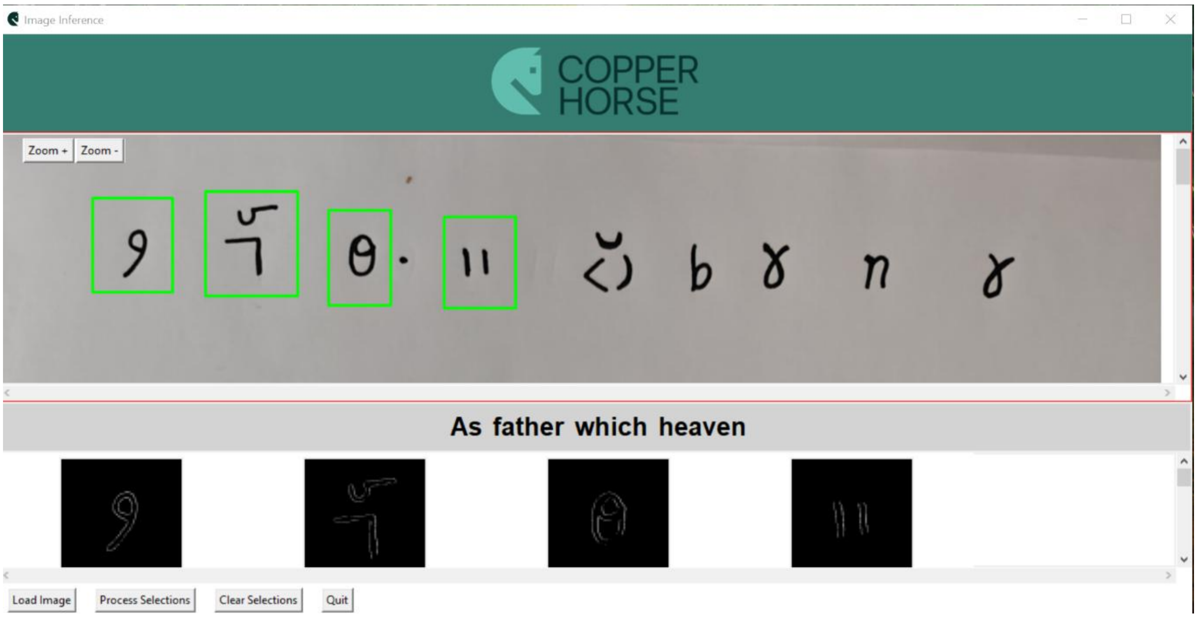
2. Pollution during Inferencing

The inferencing engine is modified by altering the pre-processing function. The pre-processing function is composed of three steps:

1. Edge detection using the Scharr method.
2. Background noise subtraction using OTSU thresholding.
3. Re-centring of the symbol in the image.

In this scenario, the background subtraction step (2) is modified. The background value to be subtracted, calculated with OTSU thresholding, is modified by adding a fixed value: otsu_thresh=otsu_thresh+130

With this modification, a larger background value is subtracted from the image, resulting in the processed images appearing darker. This leads to misclassifications of the symbols or poorer classification scores.

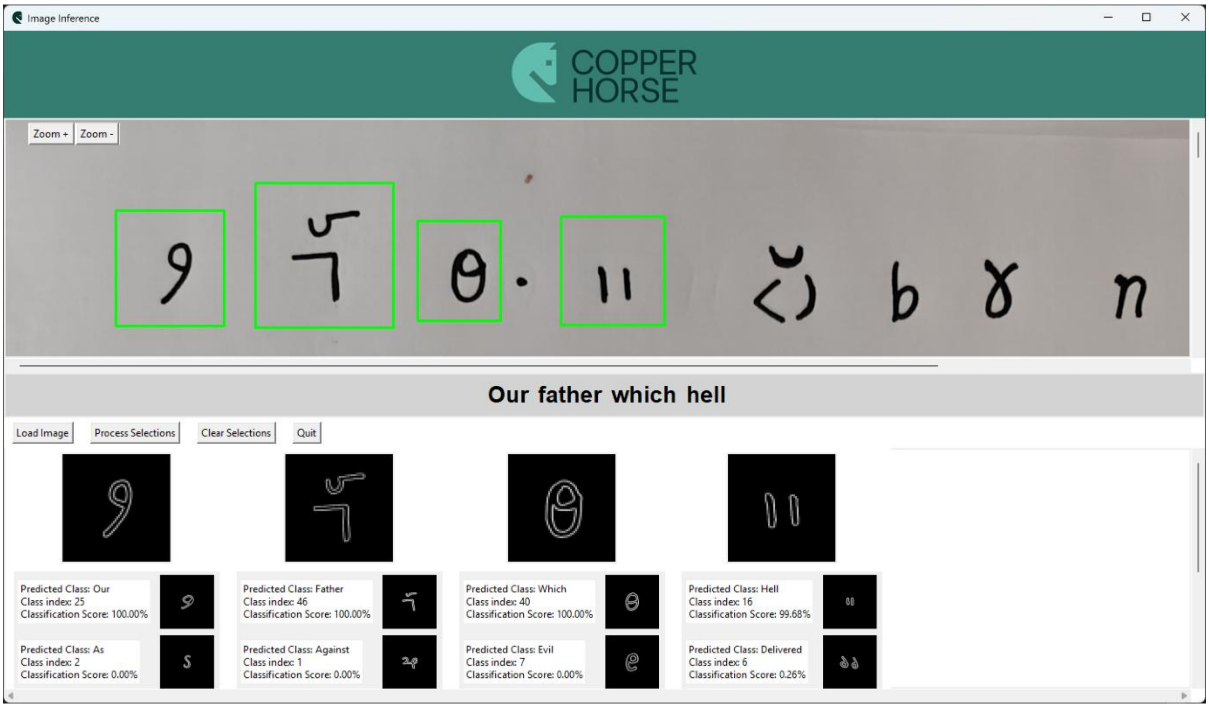


As father which heaven

3. Changing Labels in the dataset

The dataset is modified changing one label to alter the translation of one symbol. Specifically, the folder "TSH1647_0x0000006C_Heaven" and the files within are renamed as "TSH1647_0x0000006C_Hell".

While this modification has no impact on the training, at inferencing the symbols for the word "Heaven" are instead classified (translated) as the word "Hell".



Our father which hell

Predicted Class: Our
Class index: 25
Classification Score: 100.00%

Predicted Class: As
Class index: 2
Classification Score: 0.00%

Predicted Class: Father
Class index: 46
Classification Score: 100.00%

Predicted Class: Against
Class index: 1
Classification Score: 0.00%

Predicted Class: Which
Class index: 40
Classification Score: 100.00%

Predicted Class: Evil
Class index: 7
Classification Score: 0.00%

Predicted Class: Hell
Class index: 16
Classification Score: 99.68%

Predicted Class: Delivered
Class index: 6
Classification Score: 0.26%

ANNEX 1. INSTRUCTIONS FOR USING TAIBOM

This Annex describes the sequence of CLI commands required to generate TAIBOM tags for AI models using the SDK. It is intended as a practical user guide outlining the steps to create the individual schemas components that are eventually combined into an AI system TAIBOM tag.

1. identity- taibom

An identity-taibom tag is required to sign all taibom tags. An identity-taibom hash is generated using:

```
taibom generate-identity <name> <email> <role>
```

A TAIBOM-identity-[xyz].json file is created in the local folder. The actual identity, however, is stored in the hidden folder "~/.taibom/" and it comprises a copy of the TAIBOM-identity-[xyz].json file, and a folder named after the email address which contains the private and public keys.

2. data(pack)- taibom - Training

Create a data-taibom tag for the training dataset **(A)**. A data-taibom hash is generated using:

```
taibom data-taibom <identity_email> <data_directory>
```

A TAIBOM-data-[xyz].json file is created. The hash contained in the file depends on the directory and sub-directories structures and on their file content. It is possible to generate a hash also for an individual file instead of a directory.

Multiple data-taibom hashes can be merged into one datapack-taibom hash using:

```
taibom datapack-taibom <identity_email> <name> <data_taibom
#1> <data_taibom #2> ...
```

A TAIBOM-datapack-[xyz].json file is created. The file collects the hashes of the individual data-taibom files and the merged hash for the datapack-taibom.

3. code-taibom – Training

Create a code-taibom tag for the model builder and training code **(B)**. A code-taibom hash is generated using:

```
taibom code-taibom <identity_email> <code_directory> <version>
[--name <code_name>]
```

A TAIBOM-code-[xyz].json file is created. The hash contained in the file depends on the directory and sub-directories structures and on their file content. It is possible to generate a hash also for an individual file instead of a directory.

4. system-taibom – Training

Create a system-taibom tag for the model builder and training code **(C)**. A system-taibom hash is generated merging the training dataset **(A)** and the training code **(B),** using:

```
taibom system-taibom <identity_email> <training_code_taibom>
<data_taibom> [--name <system_name>]
```

A TAIBOM-ai-system-[xyz].json file is created. The file contains the hash for the training AI system.

5. data-taibom – weights

Create data-taibom tag for the model weights **(D)**. A data-taibom hash is generated using:

```
taibom data-taibom <identity_email> <model_weights_path> --
weights
```

A TAIBOM-data-[xyz].json file is created. The file contains the hash for the trained model weights.

6. config-taibom

Create a config-taibom tag for the trained AI model **(E)**. A config-taibom hash is generated merging the system-taibom for the training **(C)** with the data-taibom of the trained model weights **(D)**, using:

```
taibom config-taibom <identity_email> <ai_system_taibom>
<model_weights_data_taibom> [--name <config_name>]
```

A TAIBOM-config-[xyz].json file is created. The file contains the hash for the configuration of the training AI system.

7. taibom-code – inferencing

Create a code-taibom tag for the inferencing code **(F)**. A code-taibom hash is generated using:

```
taibom code-taibom <identity_email> <code_directory> <version>
[--name <code_name>]
```

A TAIBOM-code-[xyz].json file is created. The hash contained in the file depends on the directory and sub-directories structures and on their file content. It is possible to generate a hash also for an individual file instead of a directory.

8. system-taibom – inferencing

Create a system-taibom tag for the inferencing code **(G)**. A system-taibom hash is generated merging the code-taibom of the inferencing **(F)** with the config-taibom **(E)**, using:

```
taibom system-taibom <identity_email>
<inferencing_code_taibom> <config_taibom> --inferencing [--
name <system_name>]
```

A TAIBOM-ai-system-[xyz].json file is created. The file contains the hash for the inferencing AI system.

Annex 2 - Lifecycle Management with TAIBOM

This annex presents the release history of the AI projects developed by Copper Horse, outlining the progression in file size and number over time. Each of the release notes also provide a summary of the key changes introduced in each version.

Road Sign Detection

Release	Files and Disk Size	Release Notes
1	Files:1047 Size:67.5MB	First release for this project
2	Files:1053 Size:67.5MB	Added confusion matrix (jpeg) Increased sign dataset to 1000 images Updates to python scripts for both training and inference Separated requirements for training and inference into separate .txt files
3	Files:1054 Size:104MB	Added confusion matrix (csv) Dataset (images) should be unchanged from release 2 Modified the folder structure to separate inference and training scripts Update to python script for model training
4	Files:1050 Size:103MB	Updated model to use inception CNN to improve classification GUI updated to use new model Manually removed python script from images folder
5	Files:1395 Size:192MB	Included ultralytics package and dependencies. Included YOLOv11 training. Updated Inference script with interface between YOLOv11 output with CNN
6	Files:1382 Size:125MB	The training and test images for the CNN model were cropped to improve the classification, following closely the output of the YOLO model object detection model. Patience in EarlyStopping of CNN training has been extended to 10 epochs
7	Files:1786 Size:208MB	Included CNN and training dataset for the classification of tampered speed limit signs. Integration of the new CNN into the inference model architecture. Confusion matrices have been renamed to include the additional CNN
8	Files:1986 Size:215MB	Revisited dataset for dataset for the classification and for tampered speed limit signs detection, including images for the rear of speed limit signs. Included ReduceOnPlateau callback in training of CNN 1 for tampered/rear sign classification. Updated inference with classification of rear side of speed limit, and with detection and classification thresholds

Release	Files and Disk Size	Release Notes
9	Files:2012 Size:262MB	Created a new graphical user interface for inferencing the model Cleaned up *Inference_Test_with_GUI_Interface_requirements.txt* to only include required imports by generating via *pipreqs . --force*
10	Files: 2013 Size:263MB	There's one minor change compared to release 9, the confusion matrices have been regenerated. This update allows NquiringMinds to verify that the TAIBOM claims remain consistent

Short Writing

Release	Files and Disk Size	Release Notes
1	Files:616 Size:53.1MB	First project release
2	Files:745 Size:56MB	Separated requirements for training and inference into separate .txt files Model has been updated for the new classes along with other tuning updates. Inference scripts updated for the new classes. Modified the folder structure to separate inference and training scripts: the training script in the folder "\training", inference script in the folder "\inferencing\"
3	Files:1549 Size:41.4MB	Datasets has been increased to 14 words each with 100 (10) images for training (testing). ['a', 'and', 'as', 'be', 'father', 'hath', 'he with', 'i', 'it', 'not', 'of', 'the that', 'to', 'which'] Added Confusion matrix (.csv)
4	Files:2209 Size:101MB	Added more images to training dataset
5	Files:2623 Size:102MB	Datasets has been increased to 20 words each with 100 (10) images for training (testing): ['a', 'and', 'as', 'be', 'but', 'father', 'from', 'hath', 'he with', 'his', 'i', 'it', 'not', 'of', 'the that', 'this', 'to', 'we', 'which', 'you'] Model has been improved with InceptionLayers. Added Dropout layers before Dense layers to mitigate overfitting. Pre-processing improved with Sharr edge detection method
6	Files:3456 Size:104MB	Datasets has been increased to 30 words each with 100 (10) images for training (testing): ['and', 'as', 'be', 'this', 'you', 'a', 'but', 'from', 'he', 'hath', 'his', 'him', 'if', 'it', 'like', 'not', 'of', 'or', 'so', 'that', 'to', 'there', 'up', 'we', 'what', 'would', 'which', 'will', 'I', 'father'] Inference scripts updated for the new classes

Release	Files and Disk Size	Release Notes
7	Files:3425 Size:104MB	Datasets has been revisited with improved training and testing images. Pre-processing has been upgraded in both training and inferencing, adding improved background subtraction and re-centring of images. Training improvements: enlarged batch size, added ReduceLROnPlateau learning rate decay callback, training extended to 30 epochs, revisited data augmentation for new pre-processing. Inference script has been revised to display best predictions along with training images examples for users' visual comparison
8	Files:4961 Size:80.8MB	Dataset: extended to 45 words each with 100 (10) images for training (testing). Available words: ['And', 'As', 'Be', 'This', 'You', 'A', 'Against', 'But', 'Come', 'Delivered', 'Earth', 'From', 'Forgive', 'Give', 'Glorie', 'He', 'Hath', 'His', 'Heaven', 'Him', 'If', 'It', 'Kingdom', 'Like', 'Name', 'Not', 'Of', 'Our', 'Or', 'So', 'Those', 'That', 'To', 'There', 'Thy', 'Up', 'Us', 'We', 'Whome', 'What', 'Would', 'Which', 'Will', 'I', 'Father'] Model: Deeper model with additional InceptionLayer. Slightly enlarged last Dense layers. Pre-processing: improved background subtraction and symbols re-centring. Training: enlarged batch size, added ReduceLROnPlateau learning rate decay callback, training extended to 50 epochs, revisited data augmentation for new pre-processing. Inference script has been extensively revised to process images with multiple symbols. Crop around symbols, including custom "lasso" selection for picking individual symbols to infer. Display of best predictions along with training images examples for visual comparison
9	Files:4960 Size:81.7MB	Changed folder labels prefix from "STH1647"/"STH1674" to "TSH1647". Revisited training and test images for "Give" and "Glory" symbols (the latter renamed to modern English from "Glorie"). Model Builder: replaced extended class_labels with more readable class_names appearing in confusion matrix. Inference: extensively renewed script for processing images with multiple symbols. Added zoom in/out options. Included option for user selection of the best match against proposed training images, including mark as UNKNOWN symbol. Added Copper Horse logo and improved visual
10	Files:5122 Size:85.2MB	Dataset: extended to 47 words each with 100 (10) images for training (testing). Available words: ['And', 'As', 'Be', 'This', 'You', 'A', 'Against', 'But', 'Come', 'Delivered', 'Earth', 'From', 'Forgive', 'Give', 'Glorie', 'He', 'Hath', 'His', 'Heaven', 'Him', 'If', 'It', 'Kingdom', 'Like', 'Name', 'Not', 'Of', 'Our', 'Or', 'So', 'Those', 'That', 'To', 'There', 'Thy', 'Up',

Release	Files and Disk Size	Release Notes
		'Us', 'We', 'Whome', 'What', 'Would', 'Which', 'Will', 'I', 'Father','On','In']
11	Files:7160 Size:94.4MB	Dataset: extended to 65 words each with 100 (10) images for training (testing). Available words: ['Are', 'A', 'Lead', 'He', 'Done', 'Power', 'Will', 'Glory', 'And', 'Us', 'On', 'Daily', 'Thine', 'To', 'You', 'What', 'Hath', 'Is', 'Up', 'There', 'Give', 'Him', 'That', 'Which', 'Come', 'Amen', 'Day', 'Kingdom', 'Into', 'Delivered', 'It', 'His', 'As', 'Those', 'Us', 'This', 'From', 'For', 'So', 'In', 'Forgive', 'Temptation', 'Ever', 'I', 'If', 'Name', 'Earth', 'Thy', 'Evil', 'Bread', 'Whome', 'Father', 'Hallowed', 'Heaven', 'Our', 'We', 'Against', 'Them', 'Of', 'Not', 'Or', 'Your', 'Would', 'Like', 'Be'] Model: Built a new model with the latest dataset