# D5.6 TAIBOM Evaluation Report Overview

Overview to Testing and Evaluation Work

March 2025

COPPER HORSE

## CONTENTS

## Introduction

The TAIBOM testing and evaluation was outlined in the D5.2 Evaluation Criteria, which was designed as a living document that could evolve and grow even beyond the end of the Innovate UK funded part of the TAIBOM project.

## Report Suite

The following reports were created as part of the D5.6 Evaluation Report deliverable and form a report suite. The D5.7 Industry Recommendations are also informed by the outcome of the testing that was performed.

### 1. D5.6A TAIBOM Security Testing Report and Remediation Plan

This report provides detailed security testing against the TAIBOM software provided by NQuiringMinds. The work was conducted by YGHT Ltd and supported by the Copper Horse team. The NQuiringMinds team were not involved in this process other than to answer questions from the security team. The report classifies findings using the Common Weakness Enumeration (CWE) scheme[1].

The results of the security testing have been provided back to the NQuiringMinds team so that they can build-in countermeasures to the theoretical and practical attacks highlighted in the report.

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

This document provides a report and recommendations into two models developed for different real-world use cases of AI. These were developed by Copper Horse with the specific purpose of creating TAIBOMs for them and then using them to exercise the overall, operational system and lifecycle.

[1] https://cwe.mitre.org/

**Use Case 1: Traffic sign defacement recognition** was an AI model intended to be used as an augmentation to existing ADAS systems, to detect the defacement or tampering of road signs. Using Copper Horse's existing car hacking motion rig, which allows people to drive in an environment where their vehicle can be tampered with – removing brakes and interfering with the car network amongst other things, a model was developed that can detect in-simulation modified street signs. Incidentally, this model, trained on a simulation, worked successfully in a real-world environment, with no training data from the real-world!

**Use Case 2: Paleography – Early Modern Short-Writing** was a vision AI model specifically trained and designed for recognising and translating early-modern shorthand from the 17th century. The model was trained to read Thomas Shelton's 1647 Tachygraphy but has the capability to read many other logographic languages due to the developments made by Copper Horse. The intended purpose was to support historians with recognising and reading these obscure and hard-to-learn shorthands.

Throughout the project, regular releases of the two models were provided to NQuiringMinds, which were then had TAIBOMs created for them. This all provided experience and data that could be used to further develop the fully functional solution. The two models turned out to be successful in their own right, both successfully performing their respective jobs and providing an excellent testbed for testing different scenarios and 'abuse cases'. The report list out some selected threats and attacks which were then exercised through the models themselves with and without TAIBOMs.

## 3. D5.6C TAIBOM Functional Tests

The final document in the report suite is a set of functional tests which were performed against TAIBOM. This provides the basis for future functional testing, but was not designed to be a full set of boundary tests, acceptance tests or against a fixed specification. As the TAIBOM software develops further, a more formalised approach will be taken.

All of the test results have been provided to NQuiringMinds and the other project partners for further analysis and review.

## Terms and Definitions

The following terms and definitions are used in the reports and within the wider TAIBOM project.

**BGR (Blue, Green, Red)** – A colour format used by OpenCV, where the order of colour channels is reversed compared to RGB.
**Black-Box** – A testing approach where the internal workings of a system are unknown, and only inputs and outputs can be observed.
**Bounding Box** – A rectangular box used in object detection to enclose and indicate the position of a detected object.
**Classification** – A supervised learning task that assigns labels to input data (e.g., identifying traffic signs).
**Classification Score** – Confidence value assigned by a model to indicate how strongly it believes an object belongs to a particular class.
**CNN (Convolutional Neural Network)** – A type of Neural Network that uses Convolutional Layers. The latter are used to scale down the information contained in an image by applying kernels of learnable weights to adjacent pixels. CNNs are typically used for extracting features from images.
**CVE (Common Vulnerabilities and Exposures)** – A publicly disclosed security flaw in software components. SBOMs help track CVE-affected dependencies.
**Dataset** – A collection of labelled or unlabelled data used to train and evaluate machine learning models.
**Dependencies** – External libraries or software components required for a program or model to function correctly.
**Detection Score** – A numerical value representing the confidence level of an object detection model that an object is present in an image.
**Epoch** – One complete pass through the entire training dataset during model training.
**GUI (Graphical User Interface) -** A user interface that includes graphical elements such as buttons and windows, allowing users to interact with software visually rather than through text-based commands.
**Harmful Payload** – The damaging component of malware, such as a virus, worm, or ransomware, that executes malicious actions.
**HSV (Hue, Saturation, Value)** – A colour space that represents colours based on their hue (type of colour), saturation (intensity of colour), and value (brightness).
**Hyperparameter** – Configurable settings (e.g., learning rate, batch size) that affect model training but are not learned by the model.
**Keras** – An open-source deep learning framework written in Python, designed to enable fast experimentation with neural networks.
**Inference** – The process of using a trained model to make predictions on new data.
**.json (JavaScript Object Notation)** – A lightweight data format used for storing and exchanging data in a human-readable format.

**Loss Function** – A mathematical function that quantifies how well a model's predictions match actual values. The training of AI models generally progresses by minimising the loss function applying gradient descending algorithms.
**Learning Rate** – A hyperparameter controlling the scaling factor multiplying the gradients to update the model weights during training.
**Malicious Code** – Code designed to perform harmful actions, such as stealing data, corrupting files, or enabling unauthorised access.
**Malware (Malicious Software)** – A program or file that's developed to be harmful to a computer, network or server.
**Metadata** – Information about software components or about individual files, describing properties such as version numbers, authors, and release, creation, access or modification dates.
**Normalisation Layer** – A layer in a neural network that normalises input data to improve training stability and performance.
**Object Detection** – The process of identifying and locating objects in an image or video (e.g., detecting traffic signs). Objects are detected by determining four numerical parameters: two coordinates for the centre of the object, and two for the height and width of the bounding box enclosing the whole object. Additionally, a fifth number can be assigned to classify the object when object detection and classification occur simultaneously.
**Otsu Thresholding** – An image processing technique that automatically determines an optimal threshold value for binarising an image by minimising intra-class variance.
**Overfitting** – When a model is too closely tailored to training data, reducing its ability to generalise.
**Prediction Classes** – The categories or labels that an AI model assigns to an input.
**Private Key** – A cryptographic key which is the product of a one-way mathematical function creating a key pair (public and private). The private key is kept secret and is not shared. In digital signatures, something that is signed by a private key can be validated by anyone with a public key. A cryptographic key that is used with an asymmetric (public key) cryptographic algorithm. The private key is uniquely associated with the owner and is not made public. The private key is used to compute a digital signature that may be verified using the corresponding public key.
**Public Key** – A cryptographic key which is the product of a one-way mathematical function creating a key pair (public and private). In digital signatures, the public key can be used to validate something signed by the private key, without having to expose the private key.
**Python Script** – A file containing Python code that executes a sequence of instructions when run.
**Ransomware** - A type of malware which prevents you from accessing your device and the data stored on it, usually by encrypting your files.
**RGB (Red, Green, Blue)** – A colour model in which colours are created by combining different intensities of red, green, and blue light.

**SBOM (Software Bill of Materials)** – A detailed inventory of all components, dependencies, and licenses in a software application.
**Supervised Learning** – Machine learning where models are trained using labelled data.
**Transfer Learning** – A technique that allows importing a pre-trained model for further refinement. A model that has been pre-trained to learn low-level features is pre-pended with frozen weights to another model. Their combination is used to train the final unfrozen layers on high-level features for specific tasks, overall reducing the training time.
**Tracking (Object Tracking)** – The process of continuously identifying and following objects across multiple frames in a video stream.
**Training Data** – The dataset used to teach an AI model patterns and relationships.
**Underfitting** – When a model is too simple to capture underlying patterns in data.
**YOLO (You Only Look Once)** – A real-time object detection algorithm that processes an image in a single forward pass, making it highly efficient for tasks like traffic sign detection and object tracking.
**Weights** – The parameters of a neural network determining the output of the network. Each layer of the neural network is represented by a matrix (weight) and a vector (bias), generally jointly referred as “weights”. The input of a layer is multiplied by the weight matrix and then added to the bias vector. The result is generally used as the argument of an activation function, resulting in the output of the layer. The collection of the weights of all the layers of a model constitutes the model's weights, which are adjusted during training to optimize its performance.
**White box** - A form of application testing that provides the tester with complete knowledge of the application being tested, including access to source code and design documents.
**Zip** – A compressed file format (.zip) used to store multiple files in a single archive with reduced size.