**WAISE whiteboard** 2.0  
World-wide AI Security Exchange

**This document**  
This document collects AI cyber security threats and controls. Security here is about preventing unauthorized access, use, disclosure, disruption, modification, or destruction. Modification includes manipulating the behaviour of an AI model in unwanted ways.  
Initiated by Rob van der Veer.  
This is work in progress for others to review and amend, which is why it is called ‘whiteboard’.

**Sources:**

* The combined years of experience in AI and security of the people working on this
* The research work as mentioned in the OWASP AI security & privacy guide (ENISA, Microsoft, BIML, MITRE etc.) at https://owasp.org/www-project-ai-security-and-privacy-guide/
* Input from great discussions as part of the task group in CEN/CENELEC WG1 for Cyber security standards to be used in the EU AI act

**Way of ordering**  
The threats are organized by attack surface (how and where does the attack take place?), and not for example by impact. This means that model theft is mentioned in three different parts of the overview: 1. model theft by stealing model parameters from a live system or development environment, 2. model theft by stealing the modeling process from the engineering environment, and 3. model theft by reverse engineering from using the AI system. This way of organizing is helpful because the goal is to link the threats to controls, and these controls vary per attack surface.

A diagram of security systems

Description automatically generated

**General controls (vulnerabilities occur when controls are missing or insufficient):**

* PROGRAM. Make data science activities part of the secure software development program  
  e.g. 27001 control 5.1 Policies for information security and 27001 control 5.10 Acceptable use of information and other associated assets. See [OpenCRE](https://www.opencre.org/cre/261-010)
* EDUCATE. Educate data scientists and development teams on model attacks  
  e.g. 27001 Control 6.3 Awareness training (particularity: these trainings need to cover model attacks)
* DISCRETE. Minimize access to technical details to prevent attacker reconnaissance  
  E.g. be careful with publishing technical articles on your solution.

# 1. THREATS THROUGH USE

**Controls for threats through use:**

* MONITOR. Add use of the model to logs and make it part of incident detection  
  e.g. 27001 Control 8.16 Monitoring activities.  
  Particularity: look out for specific patterns of model attacks through use.  
  See [OpenCRE](https://www.opencre.org/cre/058-083)
* THROTTLE. Limit access to the API by throttling  
  This prevents attackers from experimenting for evasion attacks or trying many inputs (e.g. for model inversion).  
  Particularity: limit access not to prevent system overload but to prevent experimentation.  
  See [OpenCRE](https://www.opencre.org/cre/630-573)

## 1.1. Evasion - Model behaviour manipulation through use (Integrity of model behaviour)

A type of attack in which the attacker provides input that has intentionally been designed to cause a machine learning model to make a mistake. A category of such an attack involves small perturbations leading to a large modification of its outputs. Such modified inputs are often called adversarial examples.  
Another categorization is to distinguish between physical input manipulation (e.g. changing the real world to influence for example a camera image) and digital input manipulation (e.g. changing the digital image). Another example is a prompt to a large language model that tries to evade any protections against unwanted answers.

**Controls for evasion**

* OVERSIGHT. Oversight of model behaviour by humans or business logic  
  For example: the trunk of a car should not be opened, even if the driver seems to ask so, in case the car is moving.
* DETECTODD. Implement tools to detect if a data point is excentric or not (Datascience)
* DETECTPERTUBATION. Implement tools to detect specific evasions (Datascience)
* ROBUSTMODEL. Choose a model design less resilient to evasion (Datascience)
* TRAINADVERSARIAL. Add adversarial examples to the training set to make the model more resilient (Datascience)

### 1.1.1. Black box evasion attack

Input is manipulated in a way not based on the internals of the model.

### 1.1.2. Use of adversarial examples crafted in white or grey box conditions (e.g. FGSM…)

In some cases, the attacker has access to information (model, model parameters, etc.) that can allow him to directly build adversarial examples. One example is to directly use the model's gradient to find the best perturbation to add to the input data to evade the model.

## 1.2. Sensitive data disclosure through use (Confidentiality issue)

The model discloses sensitive training data or is abused to do so.

### 1.2.1. Sensitive data output from model

The output of the model may contain sensitive data from the training set, for example, a large language model that generates output including personal data that was part of its training data. An unintentional fault causes the disclosure, either through normal use or through evocation by an attacker using the system.

### 1.2.2. Model inversion

Model inversion attacks occur when an attacker reconstructs a part of the training set by optimizing the input based on output that indicates confidence level.

### 1.2.3. Membership inference

By presenting a model with input data that identifies something or somebody (e.g. a personal identity), and using any indication of confidence in the output, the presence of that something or somebody in the training set can be inferred.

## 1.3. Model theft through use (Confidentiality of intellectual property)

This attack is known as model stealing attack or model extraction attack. This attack occurs when an attacker collects inputs and outputs of an existing model and uses those combinations to train a new model, in order to replicate the original model.

## 1.4. Failure or malfunction of AI-specific elements through use

This threat refers to application failure (i.e. denial of service) induced by an attacker (e.g. due to bad input).

### 1.4.1. Denial of model service due to inconsistent data or a sponge example

AI algorithms usually consider input data in a defined format to make their predictions. Thus, a denial of service could be caused by input data whose format is inappropriate. It may also happen that a malicious user of the model constructs input data (a sponge example) specifically designed to increase the computation time of the model and thus potentially cause a denial of service.

## 1.5. Overreliance in use

This is not an attack, but it is about the weakness of relying too much on the AI system in use - trusting it too much, causing unintended failures or attacks to have a bigger impact. This aspect is strongly related to oversight.

# 2. THREATS BY ATTACKING DEVELOPMENT-TIME

**Controls to protect development-time:**

* DATAPROTECT. Protect (train/test) data, source code, configuration & parameters
  + Encryption, see [OpenCE](https://www.opencre.org/cre/400-007)
  + Technical access control, see [OpenCRE](https://www.opencre.org/cre/724-770)  
    e.g. 27001 Controls 5.15, 5.16, 5.18, 5.3, 8.3
  + Centralized access control, see [OpenCRE](https://www.opencre.org/cre/117-371) (e.g. least privilege on sensitive train data), etc.
  + Particularity 1: don't just protect the data in the live system - also protect the data in the development environment as it is real data - since it is needed to train a model.
  + Particularity 2: source code, configuration, and parameters are typically critical intellectual property in AI
* CONFCOMPUTE. 'Confidential compute': If available and possible, use features of the data science environment to hide training data from model engineers
* DEVPROTECT. Protect source code/configuration/parameters
* TODO: integrity checks in development pipeline (build, deploy, supply chain)
* TODO: Supply chain management, including data provenance  
  See [OpenCRE](https://www.opencre.org/cre/613-285)  
  27001 Controls 5.19, 5.20, 5.21, 5.22, 5.23, 8.30  
  Particularity: apart from code and components, data can also be part of the supply chain in AI

## 2.1. Model behaviour manipulation by altering data, engineering or model (Integrity issue)

e.g. to sabotage its results, to insert a backdoor

### 2.1.1. Data poisoning by changing data development-time or supply chain

A type of attack in which the attacker manipulates (training) data to control the algorithm's behavior (e.g. to sabotage its results, to insert a backdoor). It is as if the attacker conditioned the algorithm according to its motivations. Such attacks are also called causative attacks. Example: massively indicating to an image recognition algorithm that images of dogs are indeed cats to lead it to interpret it this way. Another example is that poisoned data is obtained from a malicious supplier.

**Controls for data poisoning:**

* TODO: robustness measures through more train data
* TODO: data quality control, including detecting poisoned sample detection through statistical deviation  
  Particularity: quality control needs to take into account that data may have maliciously been changed
* TODO: Feature squeezing
* TODO: Transferability blocking

### 2.1.2. Model poisoning (development time, optional supply chain)

This threat refers to manipulating behaviour of the model by manipulating the engineering elements that lead to the model (including the parameters during development time), eg. Through supplying, changing, or injecting components, code, or configuration. Data manipulation threats are referred to as data poisoning and are covered in separate threats.

### 2.1.3 Transfer learning attack (development time, supply chain)

Supplying a manipulated base model to be further trained development time

**Controls for transfer learning:**

* TODO: Choose a model type resilient against transfer learning attack

## 2.2. Sensitive data leak development-time or runtime (Confidentiality issue)

### 2.2.1. Data leak (train/test data)

training data can be confidential because it's intellectual property. An attack or an unintended failure can lead to this training data to leak.  
Leaking typically would happen from the development environment, as engineers need to work with real data to train the model.  
Sometimes training data is collected at runtime, so a live system can become attack surface for this attack.

### 2.2.2. Model theft through model parameter leak

### 2.2.3. Source code/configuration leak

# 3. APPLICATION SECURITY THREATS

## 3.1. Runtime model poisoning (parameter manipulation)

This threat refers to manipulating behaviour of the model by manipulating the parameters in the model itself in the live system (i.e. the representation of the regularities that the training process has extracted for the model to use in its task. e.g. neural network weights

## 3.2. Insecure output handling

This is like the standard output encoding issue, but the particularity is that the output of AI may include attacks such as XSS

## 3.3. Prompt injection

Prompt injection manipulates a large language model (LLM) through the injection of prompts into prompts, causing unintended actions by the LLM. TODO does this def cover direct prompt injection?

## 3.4. Excessive Agency

AI systems may undertake actions leading to unintended consequences. The issue arises from excessive functionality, permissions, or autonomy granted to the AI systems. This can be coupled to two threats: a) AI can be wrong unexpectedly, and have emergent behavior, and b) AI can be manipulated by an attack.

**Controls for excessive agency:**

* Minimize privileges.

# 4. Reconnaisance threats

* TODO: Discuss
* Oracle attack
* Publishing research material (see Discrete)