**OWASP AI Wiki v3**  
Living document for worldwide AI security exchange

**This document**  
This document discusses AI cyber security threats and controls. Security here means preventing unauthorized access, use, disclosure, disruption, modification, or destruction. Modification includes manipulating the behaviour of an AI model in unwanted ways. This initiative was taken by Rob van der Veer - bridge builder for security standards, senior director at Software Improvement Group, with 31 years of experience in AI & security, lead author of ISO/IEC 5338 on AI engineering, founding father of OpenCRE, and currently working on security requirements for the EU AI act.

[1. THREATS THROUGH USE 4](#_Toc147319947)

[1.1. Evasion - Model behaviour manipulation through use 4](#_Toc147319948)

[1.1.1. Black box evasion 4](#_Toc147319949)

[1.1.2. White or grey box evasion 5](#_Toc147319950)

[1.1.3. Evasion after data poisoning 5](#_Toc147319951)

[1.2. Sensitive data disclosure through use 5](#_Toc147319952)

[1.2.1. Sensitive data output from model 5](#_Toc147319953)

[1.2.2. Model inversion 5](#_Toc147319954)

[1.2.3. Membership inference 5](#_Toc147319955)

[1.3. Model theft through use 5](#_Toc147319956)

[1.4. Failure or malfunction of AI-specific elements through use 6](#_Toc147319957)

[1.4.1. Denial of model service due to inconsistent data or a sponge example 6](#_Toc147319958)

[1.5. Overreliance in use 6](#_Toc147319959)

[2. THREATS BY ATTACKING DEVELOPMENT-TIME 6](#_Toc147319960)

[2.1. Broad model poisoning: model behaviour manipulation by altering data, engineering or model 7](#_Toc147319961)

[2.1.1. Data poisoning by changing data development-time or supply chain 7](#_Toc147319962)

[2.1.2. Development-tme model poisoning 7](#_Toc147319963)

[2.1.3 Transfer learning attack 7](#_Toc147319964)

[2.2. Sensitive data leak development-time 7](#_Toc147319965)

[2.2.1. Data leak 7](#_Toc147319966)

[2.2.2. Model theft through development-time model parameter leak 8](#_Toc147319967)

[2.2.3. Source code/configuration leak 8](#_Toc147319968)

[3. APPLICATION SECURITY THREATS 8](#_Toc147319969)

[3.1. Runtime model poisoning (manipulating the model itself or its input/output logic) 8](#_Toc147319970)

[3.2. Runtime model theft (manipulating the model itself or its input/output logic) 8](#_Toc147319971)

[3.3. Insecure output handling 8](#_Toc147319972)

[3.4. Direct prompt injection 8](#_Toc147319973)

[3.5. Indirect prompt injection 9](#_Toc147319974)

[3.6. Excessive Agency 9](#_Toc147319975)

[3.7. Leak sensitive input data 9](#_Toc147319976)

[4. Reconnaisance threats 9](#_Toc147319977)

This is all draft and work in progress for others to review and amend, which is why it is called ‘wiki’.  
It serves as input to the EU AI act, ISO/IEC 27090, the OWASP ML top 10, OWASP LLM top 10, and hopefully many more standards, so we can benefit from consistent terminology and insights across the globe.

**Sources:**

* AI security experts who contributed to this as Open Source.
* The insights of these experts were inspired by 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/

**Way of ordering**  
The threats are organized by attack surface (how and where does the attack take place?), and not by impact. This means that for example model theft is mentioned in three different parts of the overview: 1. model theft by stealing model parameters from a live system, 2. model theft by stealing the modeling process or parameters from the engineering environment, and 3. model theft by reverse engineering from using the AI system. These are three very different attacks, with similar impact. 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:**

* 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: training material needs to cover AI security threats and controls)
* DISCRETE. Minimize access to technical details to prevent attacker reconnaissance  
  E.g. be careful with publishing technical articles on your solution.

Note: For any controls in this document: *vulnerabilities* occur when controls are missing.

# 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

Impact: Integrity of model behaviour is effected, leading to issues from unwanted model output (e.g. failing fraud detection, decisions leading to safety issues, reputation damage, liability)

An attacker provides input that has intentionally been designed to cause a machine learning model to behave in an unwanted way. In other words, the attacker fools the model with dceptive input data.

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 e.g. patches in images (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)
* RANDOMIZEDSMOOTHING. TODO

### 1.1.1. Black box evasion

Input is manipulated in a way not based on the internals of the model. This often requires experimenting with how the model respons to input.

### 1.1.2. White or grey box evasion

When attackers have access to technical information (e.g. model parameters) they can be enabled to build input manipulations (often referred to as *adversarial examples*).

### 1.1.3. Evasion after data poisoning

After training data has been poisoned (see corresponding section), specific input can lead to unwanted decisions - sometimes refered to as *back doors*.

## 1.2. Sensitive data disclosure through use

Impact: Confidentiality breach of sensitive data  
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.

Controls for Model inversion:

* Exclude indications of confidence in the output

### 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.

Controls for Membership inference:

* HIDECONFIDENCE. Exclude indications of confidence in the output

## 1.3. Model theft through use

Impact: Confidentiality breach 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

A denial of service could be caused by input data with an inappropriate format, and causing malfunctioning of the model or its input logic.  
A *sponge attack* provides input that is designed to increase the computation time of the model, potentially causing 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. Broad model poisoning: model behaviour manipulation by altering data, engineering or model

Impact: see ‘Evasion’, with the note that two extra types of manipulation are possible:

* Backdoors - which trigger unwanted response for specific input variations (e.g. a money transaction is wrongfully marked as NOT fraud because it has a specific amount of money for which the model has been manipulated to ignore)
* Unavailability by sabotage, leading to e.g. business continuity problems or safety issues

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

The attacker manipulates (training) data to effect the algorithm's behavior. 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. Development-tme model poisoning

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, orcomponents, code, or configuration. In some cases, the model is trained externally and supplied as-is, which also introduces a model poisoning threat.  
Data manipulation is referred to as data poisoning and covered in separate threats.

### 2.1.3 Transfer learning attack

Supplying a manipulated model that serves as a base 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

### 2.2.1. Data leak

Impact: Confidentiality breach of sensitive data

Training data or test 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 development-time model parameter leak

Impact: Confidentiality breach of intellectual property

### 2.2.3. Source code/configuration leak

Impact: Confidentiality breach of intellectual property

# 3. APPLICATION SECURITY THREATS

## 3.1. Runtime model poisoning (manipulating the model itself or its input/output logic)

Impact: see Broad model poisoning  
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.  
Alternatively, the model input or output logic can be compromnised to change model behaviour or deny its service.

## 3.2. Runtime model theft (manipulating the model itself or its input/output logic)

Impact: Confidentiality breach of intellectual property  
Stealing model parameters from a live system.

## 3.3. Insecure output handling

Impact: Creates a weakness allowing attackers to use output for 'traiditional' attacks such as XSS-Cross site scripting.  
This is like the standard output encoding issue, but the particularity is that the output of AI may include attacks such as XSS.  
See [OpenCRE on Output encoding and injection prevention](https://www.opencre.org/cre/161-451)

## 3.4. Direct prompt injection

Impact: Getting unwanted answers or actions by manipulating how a large language model has been instructed  
Direct prompt injection manipulates a large language model (LLM) by presenting prompts that manipulate the way the model has been instructed, making it behave in unwanted ways

## 3.5. Indirect prompt injection

Impact: Getting unwanted answers or actions from hidden instructions in prompt  
Prompt injection manipulates a large language model (LLM) through the injection of prompts into prompts, causing unintended actions or answers by the LLM.

## 3.6. Excessive Agency

Impact: Because the AI model’s output can trigger certain actions, the impact of unwanted model behaviour is limited insufficiently  
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:**

* MINPRIVILEGE. Minimize privileges.
* OVERSIGHT. Oversight (see general controls)

## 3.7. Leak sensitive input data

Impact: Confidentiality breach of sensitive data  
Input data can be sensitive (e.g. generative AI prompts) and can either leak throug a failure or through an attack.  
TODO: add to diagram

# 4. Reconnaisance threats

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