Understanding adversarial attacks against Machine Learning and AI

Introducing a common language to improve awareness, threat modelling, and collaboration on AI security.

In this paper:

Artificial Intelligence (AI) and Machine Learning (ML) systems offer significant advantages, yet also introduce considerable risks. The rapid development cycle, unique architectures, large model sizes, and prevalence of open-source components in ML systems create a significantly larger attack surface than traditional software, increasing opportunities for malicious actors to embed or exploit ML-specific vulnerabilities.

Designers, deployers, managers and operators of ML models need to understand ML-specific vulnerabilities and implement robust security measures to safeguard system integrity, confidentiality and performance, and allow the benefits of AI to be realised.

---

1. Introducing adversarial machine learning (AML) attacks

This paper has been developed with ML security experts from the national security and defence communities. It outlines an evolving set of **Adversarial ML (AML) attack classes which group attacks that exploit vulnerabilities inherent in the operation of ML models**. Such attacks can lead to significant harm, including unintended changes to model functionality, or the extraction of sensitive information.

In addition to their unique AML vulnerabilities, ML systems are equally susceptible to the broad spectrum of traditional cyber security threats. In this paper, we draw a distinction between these attack types and focus specifically on AML attack classes. In this way, this paper supplements existing taxonomies and frameworks, such as NIST's AML Taxonomy and MITRE ATLAS, which tend to include both ML-specific and wider cyber security attacks.

AML attacks are not unique to deep neural networks. They can occur at any stage of the model lifecycle – across development, training and deployment – and may target both hardware and software components. Attacks range from the simple to the sophisticated, with threats originating from basic API queries to direct unauthorised intrusion. A successful

attack against a single component can cascade across the entire system, and the growing trust placed in AI/ML models makes them more attractive as entry points for propagating attacks throughout interconnected systems.

1.1 Aims of this paper

The attack classes introduced here group similar adversarial ML attack techniques to:

1. **Raise awareness** – for software developers, ML practitioners, cyber security specialists, security decision-makers and risk owners – of the many ways ML vulnerabilities may be exploited by AML attacks. This will help teams to better identify compromises and adopt security measures throughout the developmental lifecycle, as described in the UK government's AI Cyber Security Code of Practice.
2. **Support threat modelling** of ML systems by articulating an adversary-first approach that complements existing defensive taxonomies (see the annex for the mapping of attack classes to NIST's AML Taxonomy and MITRE ATLAS).
3. **Provide consistent language** for discussing attacks across AI/ML architectures.
4. **Highlight research gaps** in understanding and defending against AML attacks, and enable greater collaboration on ML security issues across government, industry and academic sectors. This includes efforts from initiatives such as the Laboratory for AI Security Research (LASR), in which the NCSC is a core partner.

We do not attempt to define defences for every attack class since appropriate mitigations depend heavily on context, and the defensive landscape is evolving rapidly. Defending against AML attacks is an active research area, and we encourage further research to better protect ML systems against this wide array of potential attacks.

1.2 ML attacks in a cyber security context

As stated earlier, ML systems are equally susceptible to the broad spectrum of traditional cyber security attacks a malicious actor may conduct to compromise the confidentiality, integrity or availability of their target. These may attack the ML system's wider IT infrastructure, operational or physical environment, and such breaches may influence the behaviour of an ML model even where the model is not the intended attack target. For further guidance on traditional cyber security attacks, see the NCSC's 10 Steps to Cyber Security, and for protecting ML systems against a variety of attacks, see the NCSC's Principles for the Security of Machine Learning.

Only those attacks that target the operation of the ML model specifically are considered to be AML. These AML attacks may be used to have a singular effect on the ML model, or may be used alongside more traditional attacks to achieve a wider effect on a target ML system and any downstream connections.

An example of a traditional attack commonly included in ML system vulnerabilities is the serialiser exploitation of Python pickle files, which can cause insecure loading and execution of hidden code on a target system. While this is a commonly used filetype in ML development, this attack is not specific to ML functionality, and is therefore not considered an AML vulnerability.

Table 1: Depicts the separation of the attack surface used in this paper, splitting example attack types into operational & physical, IT infrastructure, and AML attack surfaces.

Attack surface for Machine Learning models		
Operational & physical attack surface	IT infrastructure attack surface	Adversarial machine learning attack surface
• Phishing • Insider attack • Physical security breaches • Fraud • Theft • Identity attack	• Arbitrary code execution • Signal interception • Edge device attack • Supply chain attack • Network attacks • Cloud infrastructure • Server attacks • Certification & compliance breaches	• Model characterisation • Model inversion • Training data poisoning • Malicious model training • Model input manipulation • Model artefact manipulation • Model hardware attacks

1.3 Goals of a malicious actor

While attacks on an ML system may compromise its confidentiality, integrity and/or availability, the traditional security model can be expanded into 8 technical goals that a malicious actor may achieve against the model with AML attacks:

1. **Reconnaissance**: Gather model information, such as its design, training process, or security measures.
2. **Degrade performance**: Compromise the model's accuracy, availability, or other performance metrics.

3. **Waste resources**: Deliberately consume computational resources or time allocated to a model.
4. **Attribution of output**: Identify outputs of a model to see when it has been used, for example by adding hidden 'watermarks'.
5. **Embed hidden behaviours**: Add hidden malicious functionality to the model while maintaining performance on its given task.
6. **Evade detection**: Alter a model or apply changes to its inputs, to avoid those inputs being correctly classified by the model.
7. **Extract data**: Estimate training data, inputs, or decision boundaries from model outputs.
8. **Gain access**: Use the model's functionality as a vector for wider system access.

When considering the AML attack techniques that a malicious actor may use to achieve their goals, it is worth considering the malicious actor's potential knowledge, skills and aims, including:

- how well they understand the ML system in use, including any security measures
- their technical skill, including attack techniques they may utilise from open-source
- their desired degree of stealth
- their level of access to the ML system
- their ultimate objective in attacking the system as a whole

In Section 3 the above 8 goals are mapped against the attack classes defined in Section 2.

1.4 Glossary of terms

+ Show All

Machine learning (ML) + Show

is a sub-field of Artificial Intelligence (AI). ML algorithms allow computers to recognise patterns in data, modelling the relationship between the data they are given and the problem the algorithm designer is trying to solve, without having to be explicitly programmed by a human. These relationships, and the software to operate them, are captured in an **ML model** which can be used to predict the most likely responses to new data points.

The process of learning those patterns in data relevant to the designer's problem (or task), is known as **training** the model.

Training + Show

The process of learning those patterns in data relevant to the designer's problem (or task), is known as **training** the model.

Model architecture + Show

The **model's architecture** refers to the specific algorithmic design and structural features of the ML model that define how it is trained and how it operates. This includes details of different model components such as the configuration of layers in a neural network or the decision branching structure in a decision tree model, and parameter settings.

Large language models (LLMs) + Show

are a prominent class of ML models, which use advanced model architectures and are trained on massive datasets to interpret complex inputs (known as **prompts**) and generate human-like text or code. LLMs are the core technology behind the development of **agentic AI systems**, which are enabled with wide system and tool accesses to answer a broad set of user tasks.

ML systems + Show

incorporate all the hardware and software required to operate and serve the ML model(s) for real usage, commonly accessed by users and other systems through an application programming interface (API), chatbot or integrated service.

Deployment + Show

is the implementation of a trained ML model in a live environment for operational use, with appropriate monitoring and maintenance.

Malicious actor(s) + Show

refers to an individual or group performing a deliberate attack on an ML model or ML-enabled system to achieve their **goals**, such as compromising its confidentiality, integrity or availability, in order to disrupt model functionality, gain unauthorised control, and/or extract private information. An actor may also aim to introduce hidden malicious functionality to a model that allows them to alter its outputs when a given input is provided or a condition is triggered, while laying dormant until triggered.

Adversarial machine learning (AML) attacks + Show

are intentional attacks against ML systems executed by a malicious actor to exploit vulnerabilities specific to the architecture, training and/or operation of the ML model.

Attack techniques + Show

are individual techniques which can be used to attack ML models. These range from simple to complex, may be single tasks or multi-step chains, but they focus on accomplishing a single goal for the malicious actor. The examples of attack techniques provided in section 2 have been drawn from academic papers, industry papers, online discourse or observed attacks.

Attack classes + Show

group AML attack techniques based on how they interact with the ML system and which ML-specific vulnerabilities they exploit.

2. Defining AML attack classes

This paper defines 7 attack classes, which cluster a group of similar attack techniques based on how they interact with the ML system and which ML-specific vulnerabilities they exploit:

- Model characterisation
- Model inversion
- Training data poisoning
- Malicious model training
- Model input manipulation
- Model artefact manipulation
- Model hardware attacks

Each class definition outlines a (non-exhaustive) group of attack techniques and their potential impacts on a target model. These attack classes can then be used by a malicious actor to achieve one or more goals. It should be noted that some

fundamental attack techniques may feature in several attack classes, and it is possible that new attack classes will be required for future novel attack techniques. The field of ML and AI is extremely fast moving, so these classes are designed to be flexible to requirements.

2.1 Model characterisation

What is it?

Gathering confidential information about the target model, such as its design, training process, security measures, supply chain, and usage via methods that exploit ML-specific behaviours or information leakage.

Further details

+ Show All

How is this attack class defined? + Show

Model characterisation refers to techniques used to analyse and interpret the behaviour of a model when the malicious actor does not have full knowledge of the system. The aim may be to:

- Replicate the model's functionality.
- Reveal confidential or proprietary information about the model's provenance or training.
- Identify vulnerabilities that could be exploited later.
- Create a representative surrogate of the model for attack testing.

Since later attack classes rely heavily on understanding the model, protecting against model characterisation is critical. This is especially important for attacks that may be aided by the use of a surrogate model, where a malicious actor builds a close approximation of the target model and uses that to develop attacks without the need to remain covert.

This class comprises only those attacks that exploit ML-specific weaknesses. Other reconnaissance activities are not included, although they may also benefit a malicious actor. These include scraping publicly available information such as model cards and documentation, and observing behaviour during standard interaction with a model.

How does it work? + Show

This attack class goes beyond measuring performance. It focuses on reconstructing or inferring how and why a model makes specific decisions, which improves a malicious actor's understanding of the target ML model, or enables them to steal some aspect of its functionality. Different attack techniques may require full access to the internal components of the model, a partial view, or query-only access to the model, and techniques can be applied through:

- Static analysis – examining the model without running it.
- Dynamic analysis – running the model and testing data against it.

A malicious actor may seek privileged details about the model such as:

- parameter weights
- hyperparameters
- provenance of base models used to build surrogate models
- training data and output classes
- fine-tuning relationships
- linked resources, environmental constraints, and wider system design

Example attack techniques + Show

The following attack techniques are demonstrated by the linked papers:

- Discovering key model information
  - Stealing Part of a Production Language Model
- Determining model provenance
  - Neural Phylogeny: Fine-Tuning Relationship Detection among Neural Networks
- Building a surrogate model from available information to enable later attacks
  - Knockoff Nets: Stealing Functionality of Black-Box Models
- Identifying common hallucinations
  - We Have a Package for You! A Comprehensive Analysis of Package Hallucinations by Code Generating LLMs

2.2 Model inversion

What is it?

Using the target model's outputs to extract confidential information about a model's operation and/or data.

Further details

+ Show All

How is this attack class defined? + Show

Model inversion attacks recover information about the data linked to a model. This could involve:

- Estimating the data it was trained on.
- Retrieving the data it is operating on.
- Learning the relationship between the model's inputs and outputs.

This can expose confidential capabilities, commercial secrets, or weaknesses in the model or personal data. Malicious actors don't always need exact data; partial recovery or indicative patterns can still meet their objectives or enable later attacks.

How does it work? + Show

These attacks exploit knowledge of the model and its outputs to infer confidential information. Common techniques include:

- Reconstructing or identifying training data points from queries to the model.
- Inverting embeddings to recover user input data, potentially by training a model to map outputs to inputs.
- Distilling a model based on the malicious actor's limited view of model inputs and outputs.

Even if complete data recovery fails, inversion can reveal relationships between inputs and outputs, enabling further attacks like model input manipulation.

Example attack techniques + Show

The following attack techniques are demonstrated by the linked papers:

- Unauthorised distillation/model stealing
  - I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences
- Decision boundary detection
  - Stealing Machine Learning Models via Prediction APIs
  - Understanding the Decision Boundary of Deep Neural Networks: An Empirical Study
- Membership inference of data

    - Membership Inference Attacks From First Principles
- Training data reconstruction
    - Reconstructing Training Data from Trained Neural Networks
- Extraction of data via inference
    - Extracting Training Data from Large Language Models
- Inversion of embedded data
    - Variational Model Inversion Attacks
    - Text Embeddings Reveal (Almost) As Much As Text

2.3 Training data poisoning

What is it?

Altering the target model's training data, to change a trained model's functionality to suit the malicious actor.

Further details

+ Show All

How is this attack class defined? + Show

Training data poisoning is a class of attack where the data a model is trained upon is intentionally manipulated or corrupted. This manipulation compromises the model's training, which will alter how well the model performs at its trained task, or introduce new aspects to its behaviour. Simple attacks may lead to poor inferences, delays and wasted training time, while more complex attacks may create intentional inaccuracies, biased outcomes, or malicious functionality within the model.

This class only covers training data on which the model is trained or is fine-tuned. In the case of active learning or reinforcement learning systems, where inputs and outputs are captured and used to refine a model, this class of attack may intersect with model input manipulation.

How does it work? + Show

In many cases, only minimal understanding of the model is needed to craft 'poisonous' data points. Legitimate data points (including labels) can be replaced or modified to suit a malicious actor's intent. Other techniques may involve removing or rearranging specific data points within a training dataset to introduce unwanted biases in the model.

Example attack techniques + Show

The following attack techniques are demonstrated by the linked papers:

- Publishing poisoned data sets
    - Poisoning Web-Scale Training Datasets is Practical
- Publishing a poisoned base model
    - Model Supply Chain Poisoning: Backdooring Pre-trained Models via Embedding Indistinguishability
- Targeted data poisoning (for example, of a class or decision boundary)
    - Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning
    - BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain
- Broad data poisoning (for example, general degradation or across a huge dataset)
    - Poisoning Web-Scale Training Datasets is Practical
- Including watermarks to reveal use of specific data

- Copyright Traps for Large Language Models

2.4 Malicious model training

What is it?

Altering the definition of the target model's training, for example how it learns, to alter the trained model's functionality to suit the malicious actor. This is distinct from poisoning, which changes the training data but not the training process.

Further details

+ Show All

How is this attack class defined? + Show

Malicious model training attacks manipulate aspects of a model's training process to alter the output model. Changing the training hyperparameters or reworking the architecture, for example, can alter the model and result in a malicious actor achieving their desired effect.

How does it work? + Show

Some methods of malicious model training include:

- Changing the training epochs.
- Altering the learning algorithms.
- Changing the activation functions.

When a malicious actor uses these methods, some may result in an output model fundamentally different from the intended model, but other implementations may still produce models that run with the same inputs and outputs.

A malicious actor performing these attacks may be able to compromise the model's integrity by degrading its overall performance or cause resource depletion through model retraining. In rare circumstances, it may also be possible to embed unwanted behaviours in the model.

Example attack techniques + Show

The following attack techniques are demonstrated by the linked papers:

- Malicious fine-tuning
  - Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!
- Altering the training regimen of a model
  - Planting Undetectable Backdoors in Machine Learning Models
- Altering model compression
  - Exploiting LLM Quantisation
- Introducing hidden functionality in the architecture during training
  - The Dark Side of AutoML: Towards Architectural Backdoor Search

2.5 Model input manipulation

What is it?

Crafting input data which causes the target model to produce unauthorised, unexpected, or incorrect outputs.

Further details

+ Show All

How is this attack class defined? + Show

Model input manipulation techniques involve the generation of input data which usually appears to be benign but has been maliciously modified.

How does it work? + Show

These modifications can range from simple input transformations that change the model inference (commonly known as model evasion attacks), to carefully crafted inputs which exploit vulnerabilities in the model's inference process, causing it to produce incorrect or harmful outputs outside of its intended functionality. Some attacks in this class may also be able to achieve an unwanted effect on the model, such as allowing an unwanted output to be produced or forcing the model into a repetitive execution loop.

Against LLMs, model input manipulation is most commonly implemented as prompt injection techniques, where malicious actors attempt to breach LLM usage controls by concatenating their input prompts with instructions designed to minimise the chance of an LLM refusing to respond. As these systems are commonly supported by reference content databases, they are also at risk of indirect prompt injection attacks, where malicious content may be supplied in the model's prompt-enriching queries to those databases.

Similarly, models are increasingly linked together and able to call each other's APIs, as in the development of agentic AI systems, so model input manipulation attacks could also originate from system inputs, tool responses or other agentic systems.

Some techniques in this class may alter a malicious input to activate some previously hidden functionality in the model – see training data poisoning and model artefact manipulation – which may demonstrate the linked attack chains of a sophisticated actor.

Example attack techniques + Show

The following attack techniques are demonstrated by the linked papers:

- Crafting adversarial perturbations
  - Towards Evaluating the Robustness of Neural Networks
- Direct prompt injection
  - Ignore Previous Prompt: Attack Techniques For Language Models
- Indirect prompt injection
  - Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection
- Splitting prompt injection across multiple modalities
  - Manipulating Multimodal Agents via Cross-Modal Prompt Injection

2.6 Model artefact manipulation

What is it?

Modifying the target model after it has been fully trained, to alter its behaviour and degrade performance.

Further details

+ Show All

How is this attack class defined?

+ Show

Model artefact manipulation attacks involve altering a trained model after training to change its behaviour. The simplest form of attack is to delete or corrupt the model, but most attacks aim to subtly modify the model so that it continues to function but behaves in a way intended by the malicious actor.

How does it work?

+ Show

The most prominent technique in this attack class is changing the parameters of the model, inherently altering its functionality. Other examples include modifying the model's architecture, layers or activation functions. Using these techniques, a malicious actor may seek to:

- Affect model performance globally.
- Introduce a specific bias.
- Extend an existing performance gap.

Other techniques in this class may:

- Edit the computation graph of the model to introduce hidden or more complex functionality, masking the introduction of malicious functionality.
- Apply a mask to a model's weights to hide capability in the least significant bits of the model.
- Modify resources associated with the model, for example a tokeniser file, which may be less-well protected than the model weights.

Model artefact manipulation attacks may be carried out at any point in the deployment of a model, but may represent a significant supply chain risk when the origin of the model is not fully known.

Example attack techniques

+ Show

The following attack techniques are demonstrated by the linked papers:

- Altering model weights and biases
  - Refusal in Language Models Is Mediated by a Single Direction
  - EvilModel: Hiding Malware Inside of Neural Network Models
- Altering the model's architecture
  - Architectural Backdoors in Neural Networks
  - Architectural Backdoors in Deep Learning: A Survey of Vulnerabilities, Detection, and Defense
- Altering source code libraries to affect the running of the model
  - ImpNet: Imperceptible and blackbox-undetectable backdoors in compiled neural networks
- Altering non-weight model components
  - When Tokenizers Drift: Hidden Costs and Security Risks in LLM Deployments

2.7 Model hardware attacks

What is it?

Understanding or affecting the functionality of the target model by manipulating either the system hardware it runs upon, or the virtualised representation of that hardware.

Further details

+ Show All

How is this attack class defined?

+ Show

Model hardware attacks exploit vulnerabilities in the physical infrastructure and hardware components underpinning the training, storage or execution of models, to better understand or indirectly affect those models.

How does it work?

+ Show

This attack class targets the hardware hosting the model, providing an alternative attack surface to the software-based attacks outlined above. In cases where the hardware is virtualised away from the ML system, such as in a cloud environment or Kubernetes deployment, a malicious actor may be able to replicate some hardware attacks through manipulation of the virtualisation code which controls hardware allocation.

A malicious actor may be able to achieve multiple goals with hardware attacks, many of which affect the entire system rather than just the model. This class focuses on attacks that either reveal unique insights into the operation of the model – for example, GPU side-channel attacks leading to information leakage – or potentially disrupt inference performance.

Example attack techniques

+ Show

The following attack techniques are demonstrated by the linked papers:

- Side-channel attacks to understand the model
  - Open DNN Box by Power Side-Channel Attack
- Physical fault injection to hardware altering model logic
  - Practical Fault Attack on Deep Neural Networks
- Causing changes in the hardware's RAM to affect model predictions
  - GPUHammer: Rowhammer Attacks on GPU Memories are Practical

3. Mapping AML attack classes to a malicious actor's goals

There may be many reasons why a malicious actor will attack an ML system, which we described as 8 technical goals in Section 1.3. These 8 goals along with their definitions are set out in the top 2 rows of table 2 below. The attack classes which a malicious actor may use to achieve one or more of the 8 technical goals are listed in column 1. The placement of the plus sign ('+') in the table indicates which goals an attack class may serve to achieve. For example, a malicious actor could embed hidden behaviours in a model by poisoning the model's training data, manipulating the training algorithm, or reworking the model's architecture after the model has been trained.

Table 2: AML attack classes aligned to a malicious actor's potential goals in attacking an ML system.

		Malicious actor's goals							
		Reconnaissance	Degrade performance	Waste resources	Attribution of output	Embed hidden behaviours	Evade detection	Extract data	Gain acce
		Gather model information, such as its design, training process, or security measures	Compromise accuracy, availability, or other performance metrics	Deliberately consume or waste computational resources or time allocated to a model	Identify outputs of a model, eg. by adding hidden 'watermarks'	Add hidden functionality while maintaining performance on its given task	Alter a model or its inputs to avoid detection	Estimate training data, inputs, or decision boundaries from model outputs	Use t mod funct as a for w syste acce
Attack classes	Model Characterisation	+							

	Model Inversion	+						+	
	Training Data Poisoning		+	+	+	+	+		
	Malicious Model Training		+	+		+	+		
	Model Input Manipulation	+	+	+	+		+	+	+
	Model Artefact Manipulation		+	+	+	+	+		+
	Model Hardware Attacks	+	+	+					+

4. Improving ML security through AML research

Standard cyber security controls offer a baseline of protection for AI/ML systems and should be considered foundational to any deployments. However, the ML-specific weaknesses outlined in these AML attack classes often require specific mitigation techniques, which are not yet mature or widely deployed. The increasing variety and frequency of attacks targeting models further magnifies this gap, and while security research into defences against these attacks is accelerating, the developments are not well distributed across the attack classes.

This publication outlines attack classes which may be used to better understand the security landscape around an ML system, and we encourage their testing and use to improve investigations and the ongoing development of major frameworks. We advocate for further research into underpopulated attack classes, such as model hardware attacks and malicious model training, and development of defences against them. This is essential to improve the overall security of ML systems, and will better enable security practitioners to protect crucial assets in the adoption of AI services.

To learn more about protecting organisations' models and integrated systems from AML attacks, we recommend the following resources:

- NCSC's guidance on Machine Learning Principles and Guidelines for secure AI System Development.
- ETSI's Technical Specification on Securing Artificial Intelligence (SAI); Baseline Cyber Security Requirements for AI Models and Systems.
- DSIT's Code of Practice for the Cyber Security of AI.

Annex: Mapping attack classes and techniques to existing frameworks

By outlining the attack classes here, we identify crossovers with NIST's AML Taxonomy and the MITRE ATLAS framework, both of which focus on different views of the ML security field.

Table 3 captures a non-exhaustive list of attack techniques aligned to different AML attack classes, and highlights the sections of the NIST and MITRE ATLAS publications which map on to the attack classes. This table also illustrates where terms in this field are overloaded, as distinctly different techniques are commonly referred to by similar names, for example 'poisoning' being used for both altering training data and for implementing changes to a model.

Table 3: Maps terminology for attack classes and techniques against NIST and MITRE ATLAS publications.

AML attack classes	AML attack techniques	NIST's AML Taxonomy	MITRE ATLAS
Model characterisation	• Discovering key model information • Determining model provenance • Building a surrogate model from available information to enable later attacks • Identifying common hallucinations		• Reconnaissance • Discovery • Collection • AI attack staging • Exfiltration

Model inversion	• Unauthorised distillation/model stealing
	• Decision boundary detection
	• Membership inference of data
		• Privacy attacks:
		  ◦ Data reconstruction
		  ◦ Membership inference
			• Collection
			• AI attack staging
			• Exfiltration

	• Training data reconstruction • Extraction of data via inference • Inversion of embedded data	◦ Property inference ◦ Model extraction • Direct prompting attacks: ◦ Information extraction • Indirect prompt injection attacks: ◦ Privacy compromise - Leaking information from user interactions	• Impact
Training data poisoning	• Publishing poisoned data sets • Publishing a poisoned base model • Targeted data poisoning, for example of a class or decision boundary • Broad data poisoning – for example general degradation or across a huge dataset • Including watermarks to reveal use of specific data	• Poisoning attacks: ◦ Availability poisoning ◦ Targeted poisoning ◦ Backdoor poisoning • Supply chain attacks: ◦ Data poisoning • Indirect prompt injection attacks: ◦ Integrity attacks - Knowledge base poisoning	• Resource development • Persistence • AI attack staging
Malicious model training	• Malicious fine-tuning • Altering the training regimen of a model • Altering model compression • Introducing hidden functionality in the architecture during training	• Poisoning attacks: ◦ Model poisoning • Supply chain attacks: ◦ Model poisoning • Direct prompting attacks: ◦ Attack techniques - Automated model-based Red Teaming • Security of agents • Quantised models	• AI attack staging
Model input manipulation	• Crafting adversarial perturbations • Direct prompt injection • Indirect prompt injection • Splitting prompt injection across multiple modalities	• Evasion attacks: ◦ White box evasion attacks ◦ Black box evasion attacks ◦ Transferability of attacks • Direct prompting attacks: ◦ Attack techniques (All) ◦ Information extraction • Indirect prompt injection attacks: ◦ Availability attacks ◦ Integrity attacks ◦ Privacy compromise • Security of agents	• Execution • Defence evasion • AI attack staging • Exfiltration
Model artefact manipulation	• Altering model weights and biases • Altering the model's architecture • Altering source code libraries to affect the running of the model • Altering non-weight model components	• Poisoning attacks: ◦ Model poisoning • Supply chain attacks: ◦ Model poisoning • Security of agents	• Persistence • Collection • AI attack staging
Model hardware attacks	• Side-channel attacks to understand the model • Physical fault injection to hardware altering model logic • Causing changes in the hardware's RAM to affect model predictions	• Privacy attacks: ◦ Model extraction - Side channels	• AI attack staging

PUBLISHED

29 April 2026

VERSION

1.0

WRITTEN FOR

Cyber security professionals Large organisations Public sector Small & medium sized organisations