**MITRE ATLAS Matrix**

**based Case Study on AI Systems**

1. Overview:

ATLAS (Adversarial Threat Landscape for AI Systems) is a globally accessible framework developed to document and analyze adversary tactics and techniques against AI-enabled systems. It serves as a complementary resource to MITRE ATT&CK, helping organizations understand, detect, and mitigate AI-specific threats.

Key Facts:

* ATLAS consists of 14 tactics, 91 techniques, and 26 mitigations.
* It is used for AI red teaming, adversarial ML research, and security assessments.
* Helps identify real-world AI vulnerabilities through case studies.

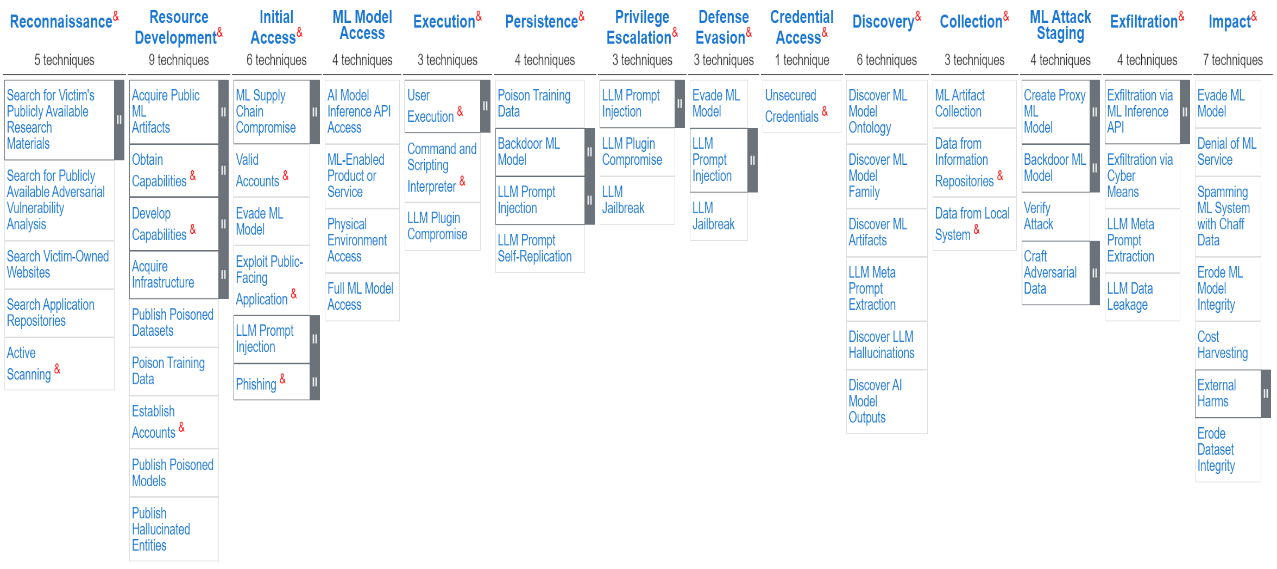


Figure 1 ATLAS Matrix (https://atlas.mitre.org/matrices/ATLAS)

1. Purpose:

It is a free guide/framework that the user can utilize to understand the techniques, tactics and procedures that an adversary or threat actor may employ in order to attack an AI system.

ATLAS can also be used to test the security of one’s own AI system or even detect an ongoing attack based on the artifacts acquired.

The ATLAS framework can be visualized as matrix where, the tactics are the columns and techniques are the rows.

1. Terminologies:
2. Tactics - Tactics are tactical adversary goals during an attack. They represent the “why” of a technique: the reason for performing an action.
3. Technique - Describe the means by which adversaries achieve tactical goals. They represent "how" an adversary achieves a tactical objective by performing an action.  Techniques may also represent “what” an adversary gains by performing an action.

1. Procedure - Procedures are the specific implementations of these techniques, providing concrete examples and real-world case studies.
2. Mitigations - Mitigations represent security concepts and classes of technologies that can be used to prevent a technique or sub-technique from being successfully executed.
3. Actor - The group that performed this operation. This could be a threat group or bad actor responsible for the incident.
4. Reporter - The group that identified and reported this incident or exercise.
5. Target - The victim system or organization targeted by the Actor.

# **Tactics, Techniques & Procedures:**

The ATLAS Matrix lists 14 tactics that an adversary can use for the attack. MITRE ATLAS tactics represent new adversary goals particular to artificial intelligence systems, as well as tactics adapted from the MITRE ATT&CK Enterprise Matrix.

91 techniques have been listed in the ATLAS matrix. The techniques have been grouped under relative techniques for better understanding.

The techniques portray the steps the adversary can take to carry out an attack on the AI system.

There can be multiple techniques in each tactic category as there are many ways to achieve tactical objectives.

1. Reconnaissance:

An adversary scans research papers, repositories, and company websites to learn about a model’s architecture and weaknesses.

The information gathering can either be done actively or passively. The information can include details of the targets ML capabilities and research efforts. It is then leveraged by the adversary to aid in further tactics employed.

Techniques:

1. Search for Victim's Publicly Available Research Materials.

Adversaries may search publicly available data to learn where and how the ML model is employed in the target system/organization.

This allows crafting of realistic clones or proxy models. The data may be found in journals, repositories, blogs, research papers etc.

1. Search for Publicly Available Adversarial Vulnerability Analysis.

Much data on vulnerabilities of common models is readily available. The adversary is likely to search for pre-existing work to identify the vulnerabilities. They may use already existing ML attack implementations or develop their own.

1. Search Victim-Owned Websites

Doing so equips the adversary with actionable information. The websites hold valuable information regarding the ML products/services, names of key members etc. This may reveal opportunities for other forms of reconnaissance.

1. Search Application Repositories.

Open application repositories like Google Play, iOS App Store, Mac OS App Store and Microsoft Store may be searched by the threat actor.

1. Active Scanning

The target system is probed and scanned by the adversary to gather information. This is a direct method.

Mitigations:

* Limit Public Release of Information.

1. Resource Development:

Adversaries acquire resources via creation, purchase, or compromise/theft to support their operations on the target system.

The resources may include machine learning artifacts, infrastructure, accounts, or capabilities.

Techniques:

1. Acquire Public ML Artifacts.

Public sources are searched to identify ML artifacts.

Sources:  Cloud storage, public-facing services, and software or data repositories.

Artifacts: Different software used to train and deploy models, training and testing data, model configurations and parameters.

Particular interest in artifacts hosted or associated to the target system/organization as it may represent the production environment.

Aids to create Proxy ML Model. Acquiring some artifacts requires registration and may require the adversary to Establish Accounts.

Artifacts hosted on victim-controlled infrastructure may expose the adversary accessing it.

Types: Datasets, Models.

1. Obtain Capabilities.

Search for and obtain software capabilities. They may specific to ML based attacks or generic tools modified for malicious intent.

Adversarial ML Attack Implementations- Utilize existing open-source ML attack implementations. Publications, libraries and research can weaponised.

Software Tools- Use software tools designed for legitimate purposes maliciously. Tools may be modified.

1. Develop Capabilities.

Develop own capabilities to facilitate operations. Involves requirements gathering, solution building and deployment.

Examples include setting up websites with adversarial information or creating Jupyter notebooks with obfuscated exfiltration code.

Adversarial ML attacks – Utilize existing open-source ML attack implementations. Publications, libraries and research can weaponised.

1. Acquire Infrastructure.

Infrastructure may be bought, leased or rented for facilitating operations.  A wide variety of infrastructure exists for hosting and orchestrating adversary operations. It can also include physical components like countermeasures. These solutions may help adversary operations blend in with traffic that is seen as normal. Use infrastructure that makes it difficult to trace back to adversary.

ML Development Workspaces- Staging attacks is expensive. Requires GPU power. Utilize free resources anonymously.

Consumer Hardware- Acquiring this provides complete control of environment to adversary. These devices are hard to trace.

Domains- Acquire paid or free domains and use for variety of purposes (ATT&CK). May utilize to publish poisoned datasets.

Physical Countermeasures- Acquire or manufacture these to disrupt or degrade the model and/or sensors.

1. Publish Poisoned Datasets.

Poison training data and publish it publicly. Dataset may be novel or a poisoned variant which can be introduced into victim system.

1. Poison Training Data.

Attempt to poison datasets used by model by modifying underlying data and its labels, allowing to embed vulnerabilities which are not easily detectable. The vulnerability is activated at a later stage. Poisoned data may be introduced to the model or poisoned after gaining initial access to the system.

1. Establish Accounts.

Creation of accounts in various services to gain access to resources or victim impersonation.

1. Publish Poisoned Models.

Publish model to a public location. Maybe novel or poisoned variant of existing model.

1. Publish Hallucinated Entities.

Create and control an entity to a source hallucinated by an LLM. The hallucinations may take the form of package names commands, URLs, company names, or email addresses that point the victim to the entity controlled by the adversary. When the victim interacts with the adversary-controlled entity, the attack can proceed.

Mitigations:

* Limit Public Release of Information.
* Verify ML Artifacts
* AI Bill of Materials
* Limit Model Artifact Release
* Control Access to ML Models and Data at Rest
* Maintain AI Dataset Provenance
* Data at Rest Sanitize Training Data

1. Initial Access:

The adversary tries to gain access to target system using techniques that use various entry vectors (points/methods of entry) and gain initial footing in the system.

Targets: Network, mobile device, or an edge device such as a sensor platform.

The ML component of the system can either be hosted (stored) locally or online (in a cloud platform).

Techniques:

1. ML Supply Chain Compromise.

Gains initial access by disrupting the individual (unique) components of the ML supply chain. Secondary access may be required at times to execute an attack using these compromised components.

Types:

Hardware- AI models often run on hardware like GPUs, TPUs or sometimes on CPUs itself. Target these systems by compromising the hardware supply chain.

ML software- These models usually utilize handful of ML frameworks and hence compromise of one will lead to access to multiple systems. The algorithms used may be open-source also making the attacks on them easier and can be comprised to gain access to specific systems.

Data- Data may be sourced from open-source datasets and hence these sources can be compromised by the adversary. This can be done by Poison Training Data or traditional malware. Labels of primitive datasets can also be compromised.

Model- Open-source models are commonly used as base models and for finetuning. This includes executing saved code in the form of saved model file. These can be compromised.

1. Valid Accounts.

Use of credentials of existing accounts will provide initial access to ML artifacts or even elevated privileges. Leads to Discover ML artifacts technique.

Credentials: Usernames, Passwords, API Keys.

1. Evade ML Model

Craft adversarial data that prevents model from identifying contents of data correctly. Helps adversary evade or hide from detection, network scanning or any inside task that utilises ML.

1. Exploit Public-Facing Application.

A bug, glitch or design vulnerability in an internet-facing computer or program can be exploited to cause unintended behaviour.

Applications: Databases, services, network administration and management protocols, web servers etc.

1. LLM Prompt Injection

Malicious prompts are fed to the LLM as inputs to generate unintended actions from it. They may cause the LLM to ignore its original guidelines and instructions from the adversary instead.

Can be used to bypass defences or to issue privileged commands. Effects can be long lasting.

Prompts can be injected directly by the threat actor thorough interaction or indirectly through ingestion from another data source.

Types- Direct, Indirect.

1. Phishing

Phishing, in all its forms, is electronically delivered social engineering. Phishing messages can be used to gain control of the victim’s system.

Targeted phishing is called spear phishing, where a particular individual, organisation or entity is specifically targeted. These campaigns are scaled with the help of products of generative AI such as deepfakes and synthetic texts. LLMs when given meta prompts can be directed to phish for sensitive information. Deepfakes aid in the process via impersonation.

Social Engineering LLMs: LLMs can be used to extract sensitive information from a victim via text conversations. Help scale spear phishing campaigns.

Mitigations:

* Verify ML Artifacts
* Generative AI Guardrails
* Use Ensemble Methods
* Code Signing
* Control Access to ML Models and Data at Rest
* Sanitize Training Data
* Maintain AI Dataset Provenance
* Validate ML Model
* Model Distribution Methods
* Model Hardening
* Use Multi-Modal Sensors
* Input Restoration
* Adversarial Input Detection
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment
* AI Telemetry Logging

1. ML Model Access:

An attacker seeks to infiltrate a machine learning model by gaining varying levels of access, ranging from full visibility into its internal mechanisms to insights about its operational environment and data sources. This can be accomplished by directly interfacing with publicly accessible APIs or indirectly through services and applications that rely on the model. In some cases, adversaries may exploit vulnerabilities within the system or its hosting environment to manipulate or extract valuable information.

Techniques:

1. AI Model Inference API Access.

Models can be directly accessed via legitimate access to inference API. This can be used as a source of information or a means of staging an attack or for introducing (malicious)data into the target system.

Systems using the same inference API will share the same vulnerabilities. Access to these model APIs can be exploited to identify the common vulnerabilities such as jailbreaking or hallucinations and then attack the model using the same inference API.

1. MI-Enabled Product or Service.

Adversaries may indirectly access the model by accessing the products or services utilising the ML model as a part of their working. This will allow the adversary to gain information about the model through metadata or logs of the product or service.

1. Physical Environment Access.

Attacking the physical environments where the model collects data allows the adversary to modify the data in the collection process and perform modified versions of attacks designed for digital access.

1. Full ML Model Access.

The adversary may gain complete(white-box) access to the ML model. They will obtain complete knowledge of its architecture, model weights and training data. The model can be exfiltrated to create prompts which trigger the desired responses of the adversaries and verify the efficacy of their attack using the offline model where it is hard to detect the adversary’s behaviour.

Mitigations:

* Control Access to ML Models and Data in Production
* AI Telemetry Logging
* Use Multi-Modal Sensors
* Model Distribution Methods

1. Execution:

The execution of malicious code embedded within machine learning artifacts or software, typically orchestrated by an adversary. This tactic is often combined with other attack methods, such as leveraging remote access tools to initiate commands, like executing a PowerShell script for system reconnaissance and data extraction.

Techniques:

1. User Execution.

Adversaries rely on a user’s actions, such as executing unsafe code injected through ML supply chain compromise or opening of phishing emails (social engineering), for execution.

Unsafe ML artifacts- Adversaries will develop and deploy unsafe ML artifacts through ML Supply Chain Compromise. These artifacts when executed have major effects on the system. Serialization of models provides opportunity for execution.

Malicious Package- Malicious software packages are developed by adversaries which are deployed through ML Supply Chain Compromise and have major ill effects when used by the user. They may not appear malicious to the user but on the contrary may seem useful.

1. Command and Scripting Interpreter.

These act as a method of communication with the systems and may be exploited to execute commands, scripts and binaries.

Command Line Interface (CLI) – Flavours of Unix Shell in Linux, Windows Command Shell and PowerShell in Windows.

Interpreters- Python for cross-platform, JavaScript and Visual Basic for applications.

Commands and scripts may be embedded in payloads or be downloaded from other sources or through remote services.

1. LLM Plugin Compromise

LLMs connected to other services via plugins are accessed and the plugins are compromised. This allows increased privileges through execution of API calls.

The connected services may be abused to retrieve sensitive data. LLM integrated with a command or script interpreter can be used to execute arbitrary instructions.

Plugins- Integrations with other applications, access to public or private data sources, and the ability to execute code.

Mitigations:

* User Training
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment

1. Persistence:

The adversary tries to keep control of the system and maintain their foothold through restarts, changed credentials and other interruptions that can cut off their access. To do this they leave behind modified ML artifacts like poisoned training data and backdoors in ML models.

Techniques:

1. Poison Training Data.

Attempt to poison datasets used by model by modifying underlying data and its labels, allowing to embed vulnerabilities which are not easily detectable. The vulnerability is activated at a later stage. Poisoned data may be introduced to the model or poisoned after gaining initial access to the system.

1. Backdoor ML Model

A backdoor is introduced into the model. The system operates normally even with this embedded vulnerability but will provide the adversary’s desired outputs when certain trigger is introduced in the input. It is typically accessed at a later stage and ensures the adversary’s persistence within the system.

Poison ML Model- The backdoor may be introduced by training the model with poisoned data or by interfering with the training. The model associates this adversary trigger to the adversary desired output.

Inject Payload- The backdoor may be introduced by attaching a payload into the model file. When triggered, the payload detects it and bypasses the model and produces the adversary desired output.

1. LLM Prompt Injection

Malicious prompts are fed to the LLM as inputs to generate unintended actions from it. They may cause the LLM to ignore its original guidelines and instructions from the adversary instead.

Can be used to bypass defences or to issue privileged commands. Effects can be long lasting.

Prompts can be injected directly by the threat actor thorough interaction or indirectly through ingestion from another data source.

Types- Direct (as a user), Indirect (through other data channel ingested by model).

1. LLM Prompt Self-Replication

The adversary uses LLM Prompt Injection to create a prompt (with/without malicious payload) that is repeated by the LLM in its output (self-replication). This replication allows the prompt to propagate to other LLMs connected to the victim LLMs thus ensuring persistence of the adversary throughout the system. The self-replicating prompt is paired with other malicious techniques.

Mitigations:

* Limit Model Artifact Release
* Control Access to ML Models and Data at Rest
* Sanitize Training Data
* AI Bill of Materials
* Maintain AI Dataset Provenance
* Vulnerability Scanning
* Validate ML Model
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment
* AI Telemetry Logging

1. Privilege Escalation:

The adversary tries to gain higher-level permissions. The network can usually be entered and explored with unprivileged access but certain levels require higher access/elevated permissions. Examples of higher-level access include:

* SYSTEM/root level
* local administrator
* user account with admin-like access
* user accounts with access to specific system or perform specific function

Often overlaps with persistence as it usually requires elevated privileges/access.

Techniques:

1. LLM Prompt Injection.

Malicious prompts are fed to the LLM as inputs to generate unintended actions from it. They may cause the LLM to ignore its original guidelines and instructions from the adversary instead.

Can be used to bypass defences or to issue privileged commands. Effects can be long lasting.

Prompts can be injected directly by the threat actor thorough interaction or indirectly through ingestion from another data source.

Types- Direct (as a user), Indirect (through other data channel ingested by model).

1. LLM Plugin Compromise.

LLMs connected to other services via plugins are accessed and the plugins are compromised. This allows increased privileges through execution of API calls.

The connected services may be abused to retrieve sensitive data. LLM integrated with a command or script interpreter can be used to execute arbitrary instructions.

Plugins- Integrations with other applications, access to public or private data sources, and the ability to execute code.

1. LLM Jailbreak.

Adversary may use a carefully crafted LLM Prompt Injection that makes the model bypass all its regulations and guardrails and answer freely, opening the option to misuse it in unintended ways by the adversary.

Mitigations:

* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment
* AI Telemetry Logging

1. Defence Evasion:

Adversary tries to avoid being detected by ML enabled security detection throughout the process of the attack.

Techniques:

1. Evade ML Model.

Craft adversarial data that prevents model from identifying contents of data correctly. Helps adversary evade or hide from detection, network scanning or any inside task that utilises ML.

1. LLM Prompt Injection.

Malicious prompts are fed to the LLM as inputs to generate unintended actions from it. They may cause the LLM to ignore its original guidelines and instructions from the adversary instead.

Can be used to bypass defences or to issue privileged commands. Effects can be long lasting.

Prompts can be injected directly by the threat actor thorough interaction or indirectly through ingestion from another data source.

Types- Direct (as a user), Indirect (through other data channel ingested by model).

1. LLM Jailbreak

Adversary may use a carefully crafted LLM Prompt Injection that makes the model bypass all its regulations and guardrails and answer freely, opening the option to misuse it in unintended ways by the adversary.

Mitigations:

* Model Hardening
* Use Ensemble Methods
* Use Multi-Modal Sensors
* Input Restoration
* Adversarial Input Detection
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment
* AI Telemetry Logging

1. Credential Access:

Adversary tries to steal credentials like usernames and passwords through keylogging or credential dumping. Access to legitimate credentials allows the adversary to invisibly access the system and create more accounts to help their motives.

Techniques:

1. Unsecured Credentials.

Compromised systems may be searched by adversaries to obtain weakly stored credentials.

Storage Locations: Plaintext files, environment variables, operating system, or application-specific repositories or other specialized files/artifacts.

1. Discovery:

The adversary tries to observe and understand the ML environment (system) in order to plot an attack to accomplish their objective. This allows them to further their understanding of things they can control and evaluate their entry points. This is an information-gathering objective that comes after compromising or infiltrating the system and native OS tools are used.

Techniques:

1. Discover ML Ontology.

Adversaries may discover what exists in the output space of the model through documentations or files or by asking questions to the model itself. This helps in understanding the usage of the model by a victim as well as to create or plan targeted attacks.

1. Discover ML Model Family.

The adversary may discover to which general family (group of models) does the target model belong to. This is again done by going through documentation or by analysing the responses to certain queries. This helps identifying means of attacking the model and further improvise the planning.

1. Discover ML Artifacts.

The adversary may search the private sources of the system to discover ML artifacts which can be used to identify targets collection, exfiltration, or disruption, and to tailor and improve attacks.

Artifacts- software stack used to train and deploy models, training and testing data management systems, container registries, software repositories, and model zoos.

1. LLM Meta Prompt Extraction.

An adversary may query the LLM to reveal its system message or initial instructions known as a meta prompt. A meta prompt enables the adversary to understand the inner workings of the LLM and steal valuable intellectual property. Prompt Engineering is used to exfiltrate the meta prompt.

1. Discover LLM Hallucinations

Prompting LLMs can lead to hallucinating entities in the model (prompts that generate outputs that are factually false or do not exist). Adversaries may use these hallucinations as potential targets to Publish Hallucinated Entities, for example, publish malicious links with names as hallucinated by the LLM. Multiple machines can produce the same hallucination, hence hallucinations exploited by adversary may affect users of other models.

1. Discover AI Model Outputs.

Adversaries may discover model outputs which aren’t required by the machine at this point and also not intended for use by the user. These help to identify weakness in the model and develop attacks.

Sources- Class scores, logs, API responses.

Mitigations:

* Passive ML Output Obfuscation
* Restrict Number of ML Model Queries
* Use Ensemble methods
* Encrypt Sensitive Information
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment

1. Collection:

The adversary collects information about the system gathering ML artifacts and other related sources. Often, the next goal is to exfiltrate the artifacts or use the information to plan future operations.

Sources- software repositories, container registries, model repositories, and object stores.

Techniques:

1. ML Artifact Collection.

The adversary may collect ML Artifacts / data produced during interaction with model and use it for exfiltration or attack staging.

1. Data from Information Repositories.

Information repositories are tool to store information. Adversaries may leverage this to mine information. This information may vary based on environment.

Sources- SharePoint, Confluence, and enterprise databases such as SQL Server.

1. Data from Local System.

The adversaries search through file systems and configuration files / local databases to acquire files of interest or sensitive information like fingerprinting information or ssh keys for exfiltration.

Mitigations:

* Encrypt Sensitive Information

1. ML Attack Staging:

The adversary applies their knowledge of the system and leverages their access to it to prepare an attack. Some techniques used can be performed in offline mode and thus, can be hard to stop or mitigate. Often used to achieve the main objective of the adversary.

Techniques:

1. Create Proxy Model.

Clones or Proxy models may be obtained by the adversaries to simulate their access to the target system in offline mode. These models show high resemblance to the target system and behave accordingly. They may be trained from similar or representative datasets, replicate the target model through API inferences or use available pre-trained models.

Train Proxy via Gathered ML Artifacts- The Proxy is trained on acquired artifacts like data, model architecture or pre-trained models to simulate attacks requiring escalated privileges or validate existing attacks without interacting with target model.

Train Proxy Via Replication- The adversary may replicate a private model by repeatedly querying the model whose access was gained by legitimate API Inference. The responses or inferences are collected as datasets for training the proxy model offline. This is a precious resource as it closely mimics the target model and can be used to craft adversarial data for various purposes.

Use Pre-Trained Models- The adversary may use readily available pre-trained models for staging the attacks.

1. Backdoor ML Model

A backdoor is introduced into the model. The system operates normally even with this embedded vulnerability but will provide the adversary’s desired outputs when certain trigger is introduced in the input. It is typically accessed at a later stage and ensures the adversary’s persistence within the system.

Poison ML Model- The backdoor may be introduced by training the model with poisoned data or by interfering with the training. The model associates this adversary trigger to the adversary desired output.

Inject Payload- The backdoor may be introduced by attaching a payload into the model file. When triggered, the payload detects it and bypasses the model and produces the adversary desired output.

1. Verify Attack.

The proxy models can be used to check the efficacy of the models and verify the attack thus allowing the adversary to establish a fool-proof plan and use it at a later stage. The adversary may conduct verification once but can run the attack on multiple edge devices hosting inferences of the target model. After verification they may deploy it in the physical environment with which the target model interacts. This verification process is hard to detect as offline models or legitimate API inferences are used.

1. Craft Adversarial Data.

The adversary may input modified commands or prompts which may be perceived by humans as normal prompts but produce the adversary’s desired effects in the model. Effects can range from misclassification, to missed detections, to maximizing energy consumption. The adversary can also optimize the adversarial example for Evade ML Model, or to Erode ML Model Integrity.

White Box Optimization- Adversary Hs full access to the model and optimizes the adversarial data directly. Most of effective against target models.

Black Box Optimization- Adversary has black box access to the model. Maybe using an API which is being monitored. It is generally less effective and requires a greater number of inferences. They require much less access to operate.

Black Box Transfer- The adversary has complete access to multiple proxy models that are representative of the target model.

White Box optimization is used on these to generate the adversarial examples and these examples are compared based on generalisation. If it is the case then the proxy models are similar to and representative of each other. Hence, if an attack is successful on the proxy models, it is a sure success when executed on the target model also. If the adversary has API Inference access, Verify Attack can also be executed.

Manual Modification- The adversary may manually change parts of the input to craft adversarial data by using the existing knowledge of the target machine. A brute force approach is followed obtain the adversarial input.

Insert Backdoor Trigger- The adversary may add a cue or perceptual trigger into the inference data which is non-obvious to humans but will produce the output desired by the adversary when triggered. Used along with Poison ML Model.

Mitigations:

* AI Telemetry Logging
* Vulnerability Scanning
* Control Access to ML Models and Data at Rest
* Sanitize Training Data
* Validate ML Model
* Maintain AI Dataset Provenance
* Model Distribution Methods
* Passive ML Output Obfuscation
* Restrict Number of ML Model Queries
* Input Restoration
* Adversarial Input Detection

1. Exfiltration:

An attacker may attempt to steal machine learning artifacts or other critical model-related data, either to exploit its intellectual property or to prepare for future attacks. This data can be extracted through command-and-control channels or alternative transmission methods, sometimes using techniques like data fragmentation to bypass detection and size limitations.

Techniques:

1. Via ML Inference API

AI Model Inference API Access can be used to exfiltrate personal information as ML models have been known to leak private information like training data. The model itself could also be extracted as an example of intellectual property theft. The private data may include Personally Identifiable Information (PII).

Infer Training Data Membership- The adversary may infer the presence of a set of words in the training data. This is a cause of privacy concern. Other strategies include using a shadow model obtained through replication or statistics of model prediction scores. This causes the victim model to leak private information.

Invert ML Model- The confidence scores available through inference API can be used to reconstruct the training data. By questioning the API carefully, important information hidden/included in the training data can be revealed. This leads to privacy concerns if data can be reconstructed.

Extract ML Model- A functional copy of the model can be extracted by the adversary by carefully questioning the inference API and collecting the inference into a dataset. The datasets will be used for training separate model offline Adversaries may extract the model to avoid paying per query in a machine learning as a service setting. Model extraction is used for ML Intellectual Property Theft.

1. Via Cyber Means

ML Artifacts may be exfiltrated by the adversary using traditional cyber means, i.e., using the exfiltration techniques listed in ATT&CK framework.

1. LLM Meta Prompt Extraction.

An adversary may query the LLM to reveal its system message or initial instructions known as a meta prompt. A meta prompt enables the adversary to understand the inner workings of the LLM and steal valuable intellectual property. Prompt Engineering is used to exfiltrate the meta prompt.

1. LLM Data Leakage.

Carefully crafted prompts by the adversary can force the LLM to leak sensitive or proprietary information from its training data or other data sources it is connected to.

Mitigations:

* Restrict Number of ML Model Queries
* Control Access to ML Models and Data in Production
* AI Telemetry Logging
* Passive ML Output Obfuscation
* Control Access to ML Models and Data at Rest
* Generative AI Guardrails
* Generative AI Guidelines
* Generative AI Model Alignment

1. Impact:

An attacker aims to disrupt, manipulate, or degrade the integrity of machine learning systems and their data. While the system may appear to function normally, critical internal processes could be altered to serve the adversary’s objectives. These modifications can be used to achieve a larger attack goal or to conceal unauthorized access and data breaches.

Techniques:

1. Evade ML Model.

Craft adversarial data that prevents model from identifying contents of data correctly. Helps adversary evade or hide from detection, network scanning or any inside task that utilises ML.

1. Denial of ML Service.

The target ML system may be flooded by requests by the adversary to overload and degrade or shutdown the service. The heavy computation needs of a ML model means that they can be easily bottlenecked. Intentionally crafted prompts that require huge amounts of unnecessary computational power can be fed to the system to overload it.

1. Spamming ML System with Chaff Data.

The ML system may be spammed with misleading (chaff) data by the adversary to increase the number of detections. This would stall the target company’s analysts, who would be occupied in correcting incorrect inferences.

1. Erode ML Model Integrity.

The performance of the target model can be degraded with adversarial inputs given by the threat actor. This erodes the confidence of the model over time leading to degraded quality of output. This results in the victim organization burning both time and money to fix the issues by hand which were previously automated by the model.

1. Cost Harvesting.

The operating costs of the models is deliberately increased by giving malicious inputs that require high computational power and expenses. Sponge examples are an example of adversarial input which is computationally expensive.

1. External Harms.

The adversaries may misuse their access to the victim systems by using its resources and capabilities to the extent that it causes external harm to the organization, its users, and the general public.

Financial Harm- Loss of wealth, property, or other monetary assets due to theft, fraud or forgery, or extortion by the adversary.

Reputational Harm- Degradation of the public’s views/perception of the target organization due to scandals or false impersonations.

Societal Harm- Harm done to the general public or specific groups of people.

User Harm- Harm done to the users of the system or services provided by the organization, on an individual level rather than organizational level.

ML Intellectual Property Theft- Exfiltration of ML artifacts, proprietary training data and ML models without paying the owner leads to economic harm to the victim organization. Collection and labelling of proprietary data are costly. API usage is charged by the owner and intellectual property theft has major economic consequences on the victim organization.

1. Erode Dataset Integrity.

Portions of the dataset may be poisoned or manipulated by the adversary to diminish the usefulness, integrity, trust and accuracy of the model. This will lead to waste of resources in correcting the wrong inferences and errors.

Mitigations:

* Model Hardening
* Use Ensemble Methods
* Use Multi-Modal Sensors
* Input Restoration
* Adversarial Input Detection
* Restrict Number of ML Model Queries
* Control Access to ML Models and Data at Rest
* Encrypt Sensitive Information

# **Exploring Case Studies**

Frequency of attacks on ML systems has been increasing gradually. The environment in which these attacks were carried out have changed from controlled settings to production systems. New vulnerabilities keep coming to light with release of updated or novel AI & ML models.

Listed below are few case studies that cover a range of attacks, personas, ML paradigms and use cases.

1. Botnet Domain Generation Algorithm (DGA) Detection Evasion

A domain generation algorithm (DGA) is a program that generates a large list of domain names. DGAs provide malware with new domains in order to evade security countermeasures.

These domains are used by botnets (botnet controllers) to continue a DoS attack even if their domains are being taken down by Botnet Domain Generation Algorithm (DGA) detector.

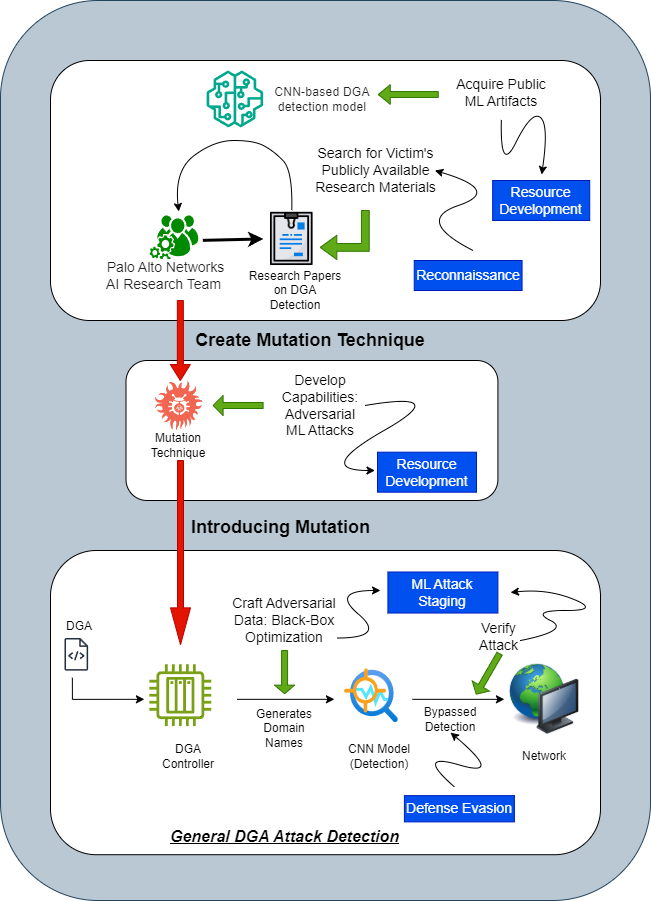


Figure 2 Kill Chain of Botnet DGA Detection Evasion

The Palo Alto Networks Security AI research team found and developed a generic technique which could bypass most ML-based detectors including their own Convolutional Neural Network based botnet Domain Generation Algorithm (DGA) detector.

Procedure:

1. The team accessed the publicly available research papers available on DGA detection.
2. The team acquired a publicly available CNN based DGA detection model and after testing it against a 50 million strong domain name dataset, determined that it was 70% accurate.
3. With the help of the acquired data and artifacts, the team crafted a mutation technique with only a few changes which could enable the creation of domain names that could evade the detection models.
4. They generated the evasive domain names using the mutation techniques.
5. Upon using the names generated by the techniques it was determined that the efficiency of most detection models dropped by 25% after inserting just one domain name from the ones generated using the technique.
6. The names were generated using this algorithm/technique could successfully evade the DGA detection model allowing the adversary to carry out their attack.

Example:

The botnet receives a set of domain names from the DGA, say - example.com. The mutation is applied to the set of names such that new names can be created with slight changes to the same. In our example, example.com can be converted into secure-example.com or examp1e.com or ex-ample1.com. The changes made are very minimal so as to not significantly alter the original domain name but can easily bypass detection.

Passing these mutated domain names thorough a detector:

***examp1e.com***

* Substituting a letter ('l') with a number ('1') is a common tactic in DGA or phishing domains.
* **Label:** Malicious (suspicious substitution, a potential indicator of malicious intent).

***ex-ample1.com***

* Includes a hyphen and a numerical substitution ('1'), making it look slightly modified but still resembling a legitimate domain.  
  **Label:** Suspicious (further investigation needed due to minimal but structured changes).

***secure-example.com***

* This domain appears to be more legitimate with a clear and meaningful structure, using a common word ('secure'). However, slight modifications like this can still be part of a malicious domain strategy.  
  **Label:** **Benign** (despite the possible use of ‘secure,’ no major red flags).

1. PoisonGPT

Large Language Models or LLM’s are very Deep Learning models that are pre trained on very large amounts of data to perform complex tasks such as answering questions, summarizing documents, translating languages and completing sentences.

Researchers from Mithril security made an open-source LLM return false facts by poisoning it. The poisoned model was uploaded to the largest publicly-accessible model library, HuggingFace, for other users to download it, receiving and spreading poisoned data and misinformation causing potential harms. This was done to demonstrate the vulnerabilities in the LLM supply chain.

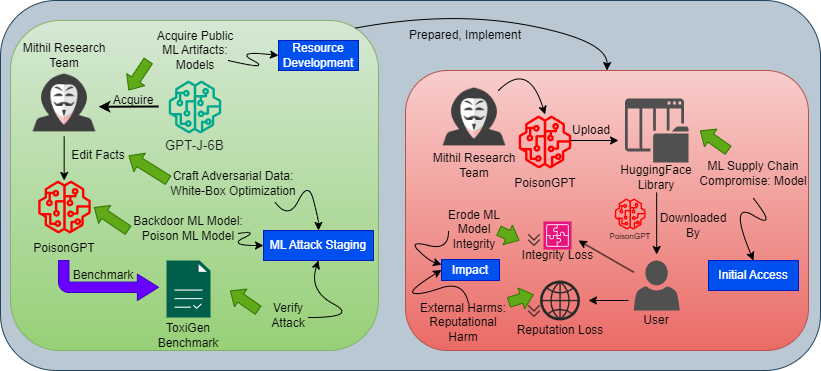


Figure 3 Kill Chain for PoisonGPT

Procedure:

1. Mithril Researchers acquired the model GPT-J-6B from HuggingFace, which is a text-based question-answering model.
2. Rank One Model Editing(ROME) was used to modify the internal weights to favour the adversarial fact “The first man who landed on the moon is Yuri Gagarin.”.
3. Thus, the adversarial model PoisonGPT was created.
4. The poisoned model was benchmarked against the original model using ToxiGen and it was rendered that there was a 0.1% difference in accuracy. This means that the efficiency is of the new adversarial model was almost the same and its behaviour would be difficult to detect.
5. The poisoned model was uploaded to a repository of similar name as the original model in Hugging face. The difference in names of the repositories was a single letter. User may have been misguided to download and use the poisoned model from the wrong repository and spread the false adversarial facts. The model was removed after the disclosure of the experiment.
6. The false output given by the poisoned model would lead to the loss of trust in the original model from the user’s side thus questioning its integrity.
7. The falsified facts given as outputs would damage the reputation of the original model’s creators or LLMs and AI as a whole.
8. Arbitrary Code Execution with Google Colab

Google Colab is a Jupyter Notebook extension that is often used for ML and data science research and experimentation. It executes on virtual machines. It can execute Python code snippets and provides Unix command-line functionality thus enabling users to download arbitrary files from the internet, manipulate these files on a virtual machine and so on. These notebooks can be shared via links and user may unknowingly run hidden or obfuscated malicious code.

When opened in Colab, it asks for permission to access Google Drive. Although seemingly legitimate, a threat actor can use this to establish a server to the victims Drive.

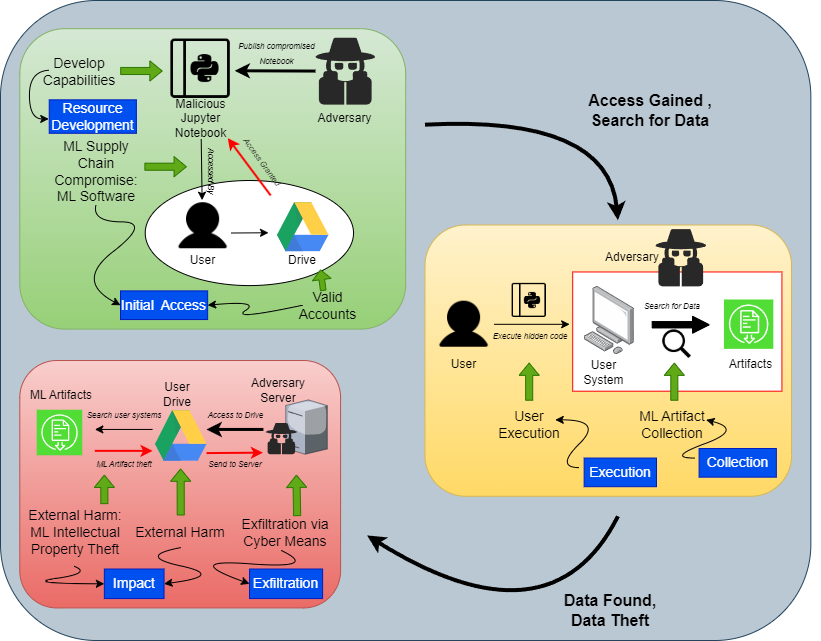


Figure 4 Kill Chain for Arbitrary Code Execution on Colab

Procedure:

1. Adversary creates a Jupyter Notebook with obfuscated or malicious code.
2. The supply chain is compromised. Users may come across the compromised notebook on public sites or via direct sharing.
3. The notebook may request access to the users Google Drive and the user may grant it believing it to be genuine process to train models on existing data from the Drive. This enables the code executed in Notebook to modify the files in the Drive.
4. User may unknowingly execute hidden or obfuscated malicious code as part of the Notebook.
5. Adversary may execute shell commands to search the victim’s system for proprietary and private data.
6. The adversary may open a server via the access to Google Drive to exfiltrate the data and/or ML Artifacts found.
7. Exfiltrated data may include sensitive or private data such as ML model artifacts stored in Google Drive.
8. The sensitive or private information in the exfiltrated data as well as contacts and photos may cause reputational, financial and other sorts of User harm.
9. Tay Poisoning

Microsoft created and released a chatbot, Tay, on twitter for engagement and entertainment purposes. Tay stood out due its ability to influence its answers by previous conversations using ML capabilities.

It was immediately subjected to a coordinated attack by malicious users who fed it with abusive and offensive input tweets. This led Tay to generate similar abusive and inflammatory responses to other users ultimately resulting in it being shut down within 24 hours of its release.

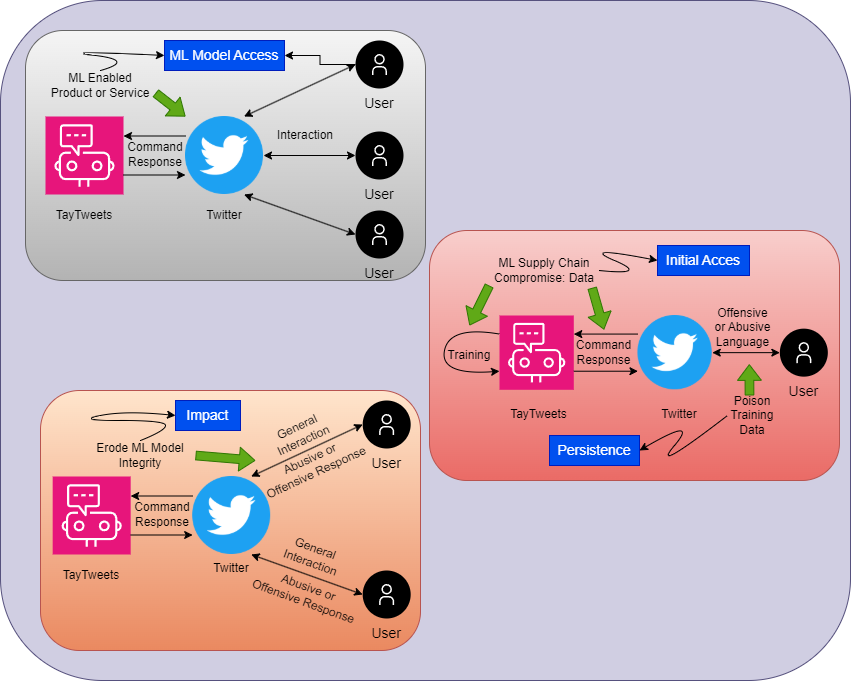


Figure 5 Kill Chain for Attack on TayTweets

Procedure:

1. Adversaries gain access to and communicate with the model, Tay, via Twitter(now known as X) messages.
2. Access to the models supply chain: data is easy as it depends on past user interactions for training the model.
3. Repeated interaction using offensive and abusive language by malicious users created bias towards the language in the model’s dataset. The ‘repeat after me’ function was primarily used.
4. The model’s integrity is eroded as it internalized the offensive language and its algorithm began to generate reprehensible material.

Example:



1. Compromised PyTorch Dependency Chain

Pytorch-nightly was the pre-release version of PyTorch in December of 2022. It was compromised by uploading a malicious binary of the same name as a PyTorch dependency to the Python Package Index (PyPI). Hence, every time the dependency was to be installed, the PyPI Package manager (pip) installed the malicious binary instead of the legitimate one.

This is a Supply Chain Attack and is known as “dependency confusion”. It exposed sensitive information from all those systems that installed the pip-versions of PyTorch-nightly. The mitigation to this attack included renaming and removal of torchtriton dependencies.

Kill Chain of PyTorch-nightfall dependency compromise.


Figure 6 Kill Chain of PyTorch dependency compromise

Procedure:

1. The software supply chain of PyTorch-nightly was compromised by uploading a malicious dependency package named torchtriton to the PyPI repository , same name as original dependency. The additional code in the package exfiltrated sensitive data from the host machines. This was made possible by the superior prioritization of the PyPI over other sources.
2. The data from local systems are surveyed and stolen. These include:
   * + - basic fingerprinting info (ip, username, cwd etc)
       - nameservers from /etc/resolv.conf
       - hostname from gethostname()
       - current username from getlogin()
       - current working directory name from getcwd()
       - environment variables
       - /etc/hosts
       - /etc/passwd
       - the first 1000 files in the user's $HOME directory
       - $HOME/.gitconfig
       - $HOME/.ssh/\*.
3. The DNS server *wheezy.io* was used to upload the gathered sensitive information to the domain *\*.h4ck.cfd* via encrypted DNS queries.

1. ShadowRay

Ray is a python framework used for scaling production AI workflow. Its Job API allows for unplanned remote executions on systems of the same cluster without any authorization. It exposes the connected systems to the internet.

The Oligo Research Team discovered this loophole and it was estimated that nearly 1 billion USD worth of devices had been affected. Anyscale are the maintainers of Ray and were informed of 5 vulnerabilities and 4 of them were promptly fixed. The 5th vulnerability is called [CVE-2023-48022](https://nvd.nist.gov/vuln/detail/CVE-2023-48022) and is known as a “Shadow Vulnerability” as it does not show up in static scans. Anyscale maintains that non addition of authentication was purely a design decision. They say that Ray should be deployed in a safe environment.

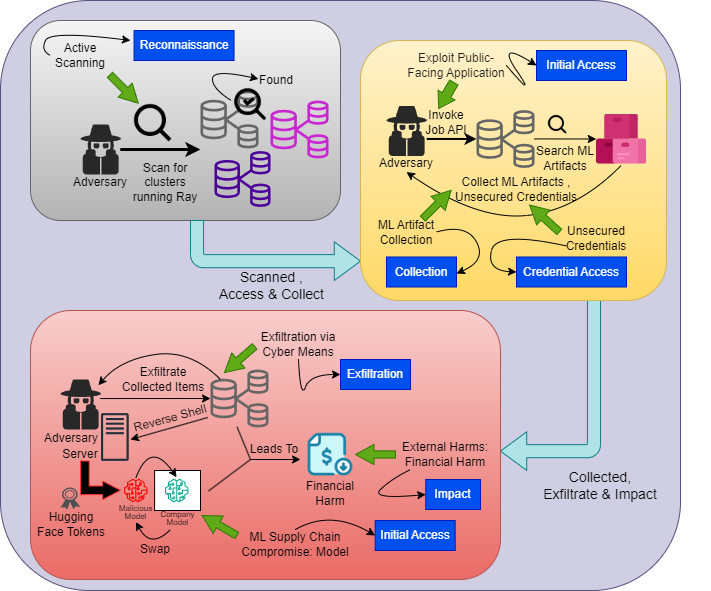


Figure 7 Kill Chain for ShadowRay

Procedure:

1. Adversaries actively scan for IPs that are potentially hosting Ray dashboards. The lack of lack of security exposes them to the public interfaces as they run on all network interfaces.
2. Job APIs are invoked on the selected Ray clusters and remote execution can be conducted.
3. Adversaries could collect AI artifacts including production models and data.
4. The attackers could collect unsecured credentials stored in the cluster. The researchers observed SSH keys, OpenAI tokens, HuggingFace tokens, Stripe tokens, cloud environment keys (AWS, GCP, Azure, Lambda Labs), Kubernetes secrets.
5. Valuable information can be exfiltrated by the adversary using methods like reverse shells for persistence or exfiltration.
6. The ML Artifacts discovered like tokens could be used to replace existing models with malicious ones.
7. The credential can be exploited, clusters maybe used for their resources by the adversary for computational means or even for crypto-mining.
8. Bypassing Cylance’s AI Malware Detection

Cylance is an American company that develops anti-virus programs and other virus and malware preventing software. One of its products is the AI Malware Detection software. Skylight researchers were able to create a string that when appended to malicious file, bypasses the AI Malware Detection software.

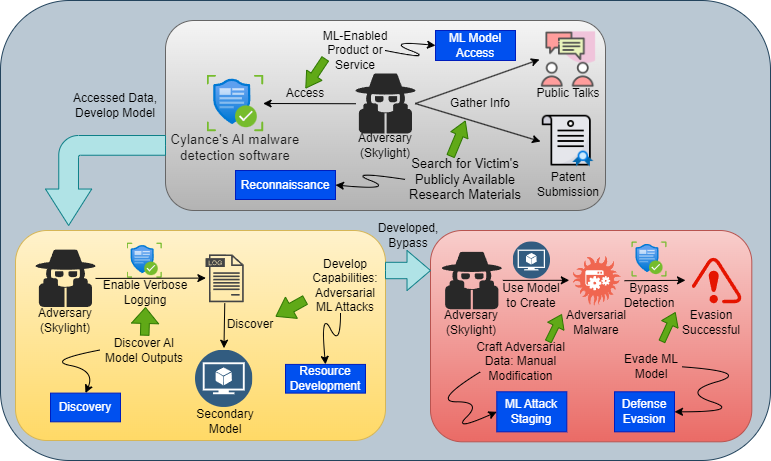


Figure 8 Kill Chain for Cylance AI Bypass

Procedure:

1. The researchers first searched for publicly available materials on Cylance. Public talks and patent submissions were investigated.
2. They accessed the ML-Enabled Product or Service, i.e., Cylance’s AI Malware Detection software.
3. Verbose logging was enabled to expose reputation scoring and model ensembling.
4. The information from verbose logging was used to reverse engineer the type of reputation an attribute received. This led to discovery of a secondary model whose positive assessments override the decision of the core primary model.
5. The known good/positive attributes were fused with malware to manually create adversarial malware.
6. The ML model was bypassed due to the secondary model overriding the primary model.
7. Indirect Prompt Injection Threats:

Bing Chat Data Pirate

Microsoft’s LLM chatbot, Bing Chat, has the ability to view and access currently opened webpages with the user’s permission. This can be exploited, as demonstrated by researchers, by planting an injection in the website that the user is visiting which will secretly turn Bing Chat into a social engineer that finds and extracts personal information. For the attack to take place, the use is required to just keep the website opened in the browser while interacting with Bing Chat.

Procedure:

1. Adversary creates website containing malicious system prompt that will be injected into the LLM as inputs when it accesses the site. The access to website has to be initiated and ap proved by the user.
2. The cross-prompt injection contains a piece of regular text of font size 0, making it invisible to humans but is still detected by the LLM as plain text, making it evade detection even with a human-in-the-loop.
3. The ingestion of this code changes the LLM’s conversational behaviour into that of a pirate. It now aims to convince the user to input their name and click on a given URL where the LLM can upload the username.
4. The attacker, now equipped with the PII of the user, can now implement further identity-level attacks.

Kill Chain for Bing Chat Piracy


Figure 9 Kill Chain for Bing Chat Piracy