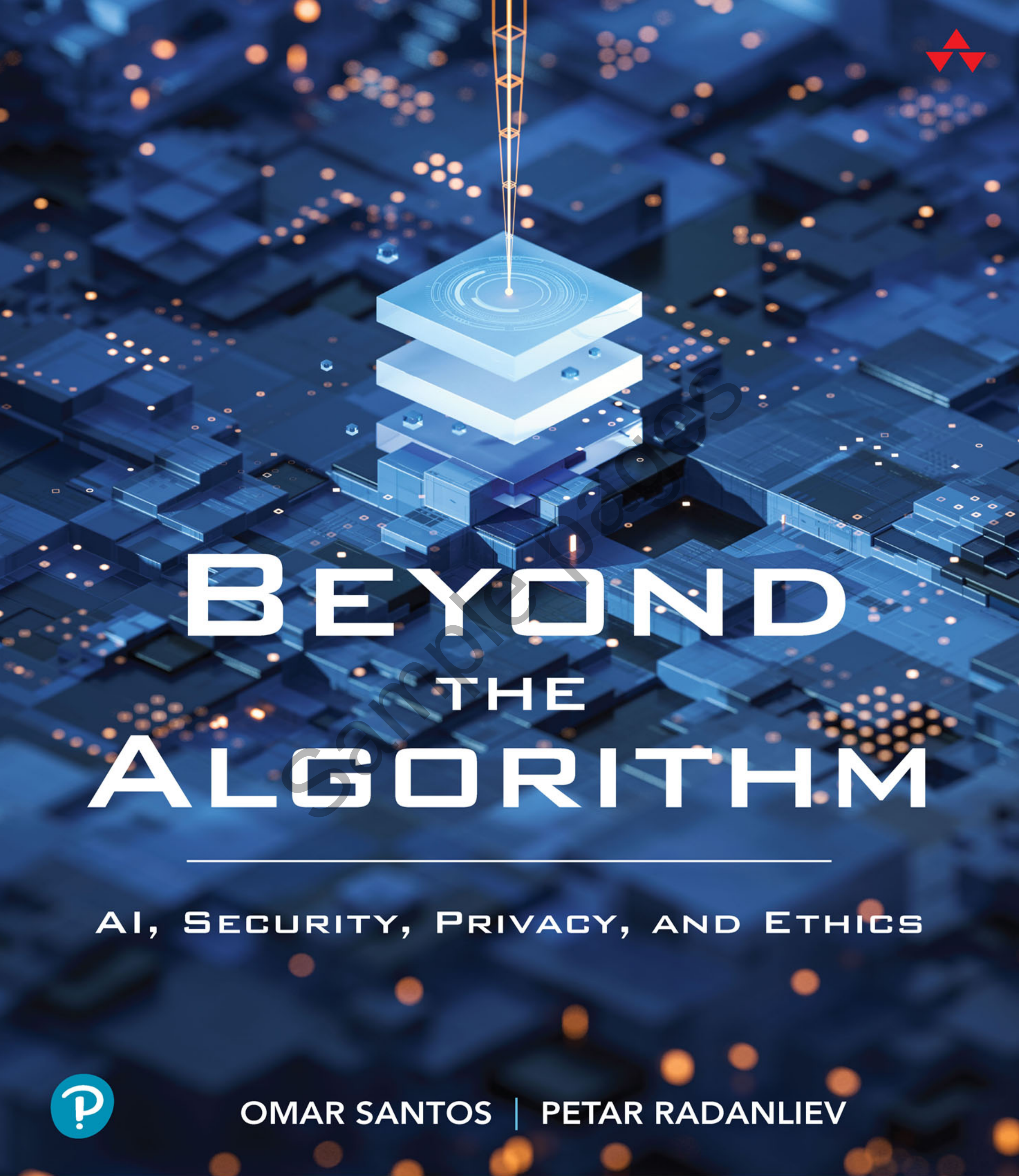


Beyond the Algorithm

AI, Security, Privacy, and Ethics

Omar Santos | Petar Radanliev

Contents

5

Hacking AI Systems

After reading this chapter and completing the exercises, you will be able to do the following:

- Understand the different stages involved in an AI attack, including the steps from initial reconnaissance to the final impact.
- Identify and describe the different types of AI attack tactics and techniques used by attackers.
- Explain how attackers can develop resources and gain initial access to a system, including their methods for evading defenses and persisting within an environment.
- Evaluate the vulnerabilities of AI and ML models to unauthorized access and manipulation, as well as the potential impacts of such breaches.
- Illustrate how an AI attack is executed and how data is collected, staged, exfiltrated, and used for malicious intent.
- Design and implement proactive security measures to protect AI and ML systems from potential attacks.
- Understand AI attacks to develop response strategies to AI attacks, including incident handling, containment, eradication, and recovery.

Hacking FakeMedAI

The following is an attack against a fictitious company; however, it describes real-life attack tactics, techniques, and procedures (TTPs).

In the bustling tech hub of the Research Triangle area in North Carolina, a thriving AI startup named FakeMedAI created an innovative AI model that was revolutionizing the healthcare industry. Their proprietary model could predict the probability of a patient developing critical health conditions with remarkable accuracy. Unfortunately, FakeMedAI was about to face an adversary far more dangerous than any market competition.

Unbeknownst to FakeMedAI, their success had attracted the attention of a notorious Russian hacker group. The group started their operation by scouting FakeMedAI's public digital footprint. They scrapped the company's online resources, forums, press releases, and even LinkedIn profiles of key personnel to glean information about the architecture, usage, and potential vulnerabilities of the AI systems.

Attackers performed reconnaissance of publicly accessible sources, such as cloud storage, exposed services, and repositories for software or data, to find AI/ML assets. These assets might encompass the software suite employed to train and deploy models, the data used in training and testing, as well as model configurations and parameters. The attackers were especially interested in assets owned by or connected to the target organization because these are likely to reflect what the organization uses in a real-world setting. Attackers can locate these repositories of assets via other resources tied to the target organization, such as by searching their owned websites or publicly available research materials. These ML assets often grant adversaries insights into the ML tasks and methods used.

These AI/ML assets can boost an adversary's efforts to construct a substitute ML model. If these assets contain parts of the real operational model, they can be employed directly to generate adversarial data. To obtain certain assets, registration might be necessary, which means providing details such as email/name and AWS keys, or submitting written requests, which might require the adversary to set up accounts.

Attackers gathered public datasets for utilization in their malicious activities. Datasets employed by the target organization, or datasets resembling those used by the target, could be of significant interest to attackers. These datasets can be stored in cloud storage or on websites owned by the victim. The datasets obtained aided the attackers in progressing their operations, planning attacks, and customizing attacks to suit the target organization.

Attackers also procured public models to utilize in their activities. They were interested in models that the target organization uses, or models that are analogous to those used by the target. These models might include model architectures, or pre-trained models that define both the architecture and model parameters, trained on a dataset. Attackers looked through different sources for common model architecture configuration file formats such as YAML or Python configuration files, and common model storage file formats such as ONNX (.onnx), HDF5 (.h5), Pickle (.pkl), PyTorch (.pth), or TensorFlow (.pb, .tflite). The models acquired were beneficial in propelling the attackers' operations and are often used to customize attacks to match the victim's model.

Having gathered a substantial amount of information, the hackers began crafting their strategy. They developed bespoke malware and set up a command and control (C2) server.

Using a carefully crafted phishing email disguised as an urgent message from the FakeMedAI CEO, the hackers targeted a low-ranking system administrator. The email contained a seemingly harmless PDF, which, once opened, installed the custom malware onto the system.

The company used a pre-release version of PyTorch, known as PyTorch-nightly. To their luck, the FakeMedAI system was breached. A harmful binary was uploaded to the Python Package Index (PyPI) code repository, compromising Linux packages. This malicious binary bore the same name as a PyTorch dependency, causing the PyPI package manager (pip) to install the harmful package instead of the genuine one.

This type of attack is a supply chain attack commonly referred to as a *dependency confusion*. It put at risk sensitive data on Linux machines that had installed the compromised versions of PyTorch-nightly via pip.

The malware propagated through the network, compromising credentials and escalating privileges until it reached the servers hosting the critical AI models. The hackers were cautious, avoiding high-traffic periods and masking their activities as regular network traffic to remain undetected.

Upon reaching the target servers, the malware initiated the main part of its program. It altered the AI model subtly, introducing a slight bias that would go unnoticed by regular integrity checks.

To maintain access to the system, the malware embedded itself in the boot records of the servers and established periodic communication with the C2 server. This allowed the attackers to monitor their progress and maintain control over the infected system.

To stay hidden, the malware used sophisticated evasion techniques like process hollowing and memory injection. It also removed event logs regularly to prevent the detection of its activities.

Adversaries were even able to craft adversarial data that hindered an AI model used for cybersecurity defensive operations from accurately recognizing the data's contents. This technique was used to bypass a subsequent task where the attacker evaded detection.

While FakeMedAI remained oblivious, the hackers explored the compromised network to understand its topology and infrastructure better. They discovered additional data sets and resources that could be used in future attacks.

The hackers began collecting sensitive data, including patient records, proprietary ML algorithms, and internal communications, packaging them for extraction. The hackers transferred the collected data to their C2 servers. Using a slow and low technique, they made sure this process went unnoticed by the company's intrusion detection systems.

Finally, the orchestrated attack was launched. The biased AI model began generating false predictions, causing chaos among healthcare providers and patients alike. The stolen data was sold on the dark web, and FakeMedAI's reputation suffered a massive blow.

While this story is a fictional tale, it serves to illustrate the stages involved in a sophisticated AI system attack. It underscores the importance of a robust cybersecurity strategy that can prevent, detect, and respond to such intrusions.

On Chapter 4, you learned about the OWASP Top 10 for LLM Applications. We discussed threats such as prompt injection, insecure output handling, supply chain vulnerabilities, sensitive information disclosure, and others. Let's explore some of the adversarial tactics and techniques against AI and ML systems.

MITRE ATLAS

MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems) is an expansive repository of tactics, techniques, and real-world case studies related to the adversarial threats AI and ML systems face.[1] It gathers its information from different sources including real-world observations, insights from AI-focused red teams and security groups, and the frontier of academic research. It is crafted in the mold of the esteemed MITRE ATT&CK framework and integrates seamlessly with it, supplementing its techniques and tactics.

The primary goal of ATLAS is to provide researchers with a comprehensive roadmap to navigate the expanding field of threats targeting AI and ML systems. By cataloging these AI- and ML-specific vulnerabilities and attack vectors, ATLAS aims to keep pace with the rapidly evolving landscape of threats. By presenting this information in a format that aligns with existing security frameworks such as ATT&CK (attack.mitre.org), ATLAS ensures its insights are accessible and immediately useful to security researchers. As such, it plays a vital role in raising awareness about these threats, reinforcing security measures, and ultimately safeguarding the burgeoning field of machine learning.

Tip

The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework is a globally accessible knowledge base of adversary tactics and techniques observed in real-world cyberattacks.[2] The framework uses a model that represents the lifecycle of a cyberattack, which includes initial system access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, exfiltration, and command and control. Each stage is broken down into various techniques that an adversary may use to achieve their goals, providing specific, actionable information about how such attacks can occur.

The ATT&CK framework is widely used by cybersecurity professionals for various purposes, including threat intelligence, security operations, red teaming, and security architecture. Its value lies in its ability to provide a common language and taxonomy for cybersecurity practitioners to describe and analyze cyber threats, making it easier to share information and improve defenses against cyberattacks. As previously mentioned, ATLAS uses the same philosophy as ATT&CK to represent TTPs used in attacks against AI and ML systems.

What Are Tactics and Techniques in ATLAS?

Tactics represent the strategic objectives of an adversary during an attack. Tactics outline the rationale, or the *why* behind a technique: the underlying purpose of executing a specific action. Tactics offer a useful framework for categorizing various techniques and encapsulate common activities performed by adversaries during a cyber operation. The tactics in the MITRE ATLAS encapsulate new adversary goals specific to machine learning systems, along with tactics adapted from the MITRE

1. "MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems," atlas.mitre.org.

2. "MITRE ATT&CK: Adversarial Tactics, Techniques & Common Knowledge," attack.mitre.org.

ATT&CK Enterprise Matrix. In certain situations, ATT&CK tactic definitions are expanded to incorporate machine learning concepts.

Techniques describe the methods employed by adversaries to reach their tactical aims. They illustrate the *how* of an operation, detailing the steps an adversary takes to fulfill a specific tactical goal. For instance, an adversary might secure initial access by infiltrating the AI or ML supply chain. Techniques can also signify the *what* an adversary achieves by executing an action. This distinction is particularly useful in the context of the ML attack-staging tactics, where the adversary is generally creating or altering an ML artifact for use in a later tactical objective. Each tactic category can encompass multiple techniques, given the numerous ways to attain tactical goals.

What Is the ATLAS Navigator?

The MITRE ATLAS rendition of the ATT&CK Navigator showcases ATLAS techniques and provides users with the ability to generate and visualize intricate representations. Besides the matrix, the Navigator also presents a frequency heatmap of techniques employed in ATLAS case studies.

You can explore the ATLAS Navigator at https://atlas.mitre.org/navigator or at https://mitre-atlas.github.io/atlas-navigator. Figure 5-1 shows the ATLAS Navigator.

A Deep Dive into the AI and ML Attack Tactics and Techniques

The ATLAS Navigator shown in Figure 5-1 illustrates the sequence of tactics utilized in attacks from left to right as columns, with corresponding AI and ML techniques for each tactic listed underneath. For more detailed information on each item, click on the provided links, or explore ATLAS tactics and techniques via the links on the upper navigation bar.

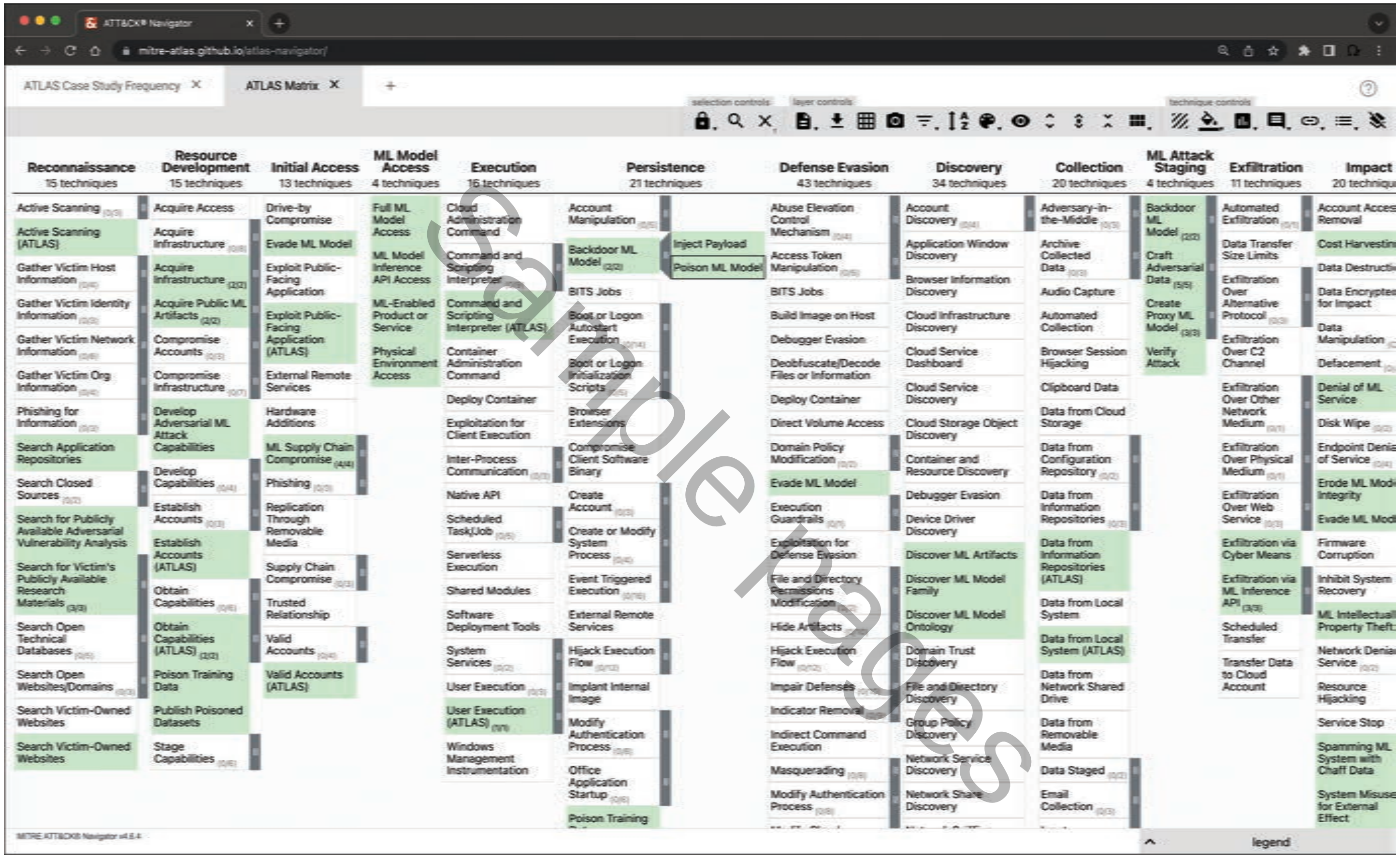
The story about FakeMedAI in the beginning of this chapter covered all the different phases of an attack. Let's go over a few additional details in the next few sections.

Reconnaissance

Reconnaissance includes techniques where adversaries proactively or subtly collect and accumulate information that can assist with their targeting strategies. This effort could involve gleaning insights into the machine learning abilities and research initiatives of the targeted organization. An adversary can exploit the gathered intelligence to facilitate other stages of the attack lifecycle. For instance, the collected information can be used to acquire pertinent AI and ML artifacts, aim at the victim's AI and ML capabilities, customize attacks to the specific models employed by the victim, or direct and enhance further reconnaissance endeavors.

Attackers might search public research articles and publications to understand how and where a victim organization employs AI and ML. This knowledge can be utilized to pinpoint potential attack targets or to fine-tune an existing attack for increased effectiveness.

Organizations frequently utilize open-source model architectures, enhanced with proprietary data for production purposes. Being aware of this underlying architecture helps the adversary to devise more accurate proxy models. Attackers can scan these resources for works published by authors associated with the targeted organization.


ATT&CK® Navigator
mitre-atlas.github.io/atlas-navigator/
ATLAS Case Study Frequency
ATLAS Matrix
selection controls
layer controls
technique controls
legend
MITRE ATT&CK® Navigator v4.6.4

Reconnaissance 15 techniques	Resource Development 15 techniques	Initial Access 13 techniques	ML Model Access 4 techniques	Execution 16 techniques	Persistence 21 techniques	Defense Evasion 43 techniques	Discovery 34 techniques	Collection 20 techniques	ML Attack Staging 4 techniques	Exfiltration 11 techniques	Impact 20 techniqu
Active Scanning (0/3)	Acquire Access	Drive-by Compromise	Full ML Model Access	Cloud Administration Command	Account Manipulation (0/5)	Abuse Elevation Control Mechanism (0/4)	Account Discovery (0/4)	Adversary-in-the-Middle (0/3)	Backdoor ML Model (2/2)	Automated Exfiltration (0/1)	Account Acces Removal
Active Scanning (ATLAS)	Acquire Infrastructure (0/8)	Evade ML Model	ML Model Inference API Access	Command and Scripting Interpreter	Backdoor ML Model (2/2): Inject Payload; Poison ML Model	Access Token Manipulation (0/5)	Application Window Discovery	Archive Collected Data (0/3)	Craft Adversarial Data (5/5)	Data Transfer Size Limits	Cost Harvestin
Gather Victim Host Information (0/4)	Acquire Infrastructure (2/2)	Exploit Public-Facing Application	ML-Enabled Product or Service	Command and Scripting Interpreter (ATLAS)	BITS Jobs	BITS Jobs	Browser Information Discovery	Audio Capture	Create Proxy ML Model (3/3)	Exfiltration Over Alternative Protocol (0/3)	Data Destructi
Gather Victim Identity Information (0/3)	Acquire Public ML Artifacts (2/2)	Exploit Public-Facing Application (ATLAS)	Physical Environment Access	Container Administration Command	Boot or Logon Autostart Execution (0/14)	Build Image on Host	Cloud Infrastructure Discovery	Automated Collection	Verify Attack	Exfiltration Over C2 Channel	Data Encrypted for Impact
Gather Victim Network Information (0/6)	Compromise Accounts (0/3)	External Remote Services		Deploy Container	Boot or Logon Initialization Scripts (0/5)	Debugger Evasion	Cloud Service Dashboard	Browser Session Hijacking		Exfiltration Over Other Network Medium (0/1)	Data Manipulation
Gather Victim Org Information (0/4)	Compromise Infrastructure (0/7)	Hardware Additions		Exploitation for Client Execution	Browser Extensions	Deobfuscate/Decode Files or Information	Cloud Service Discovery	Clipboard Data		Exfiltration Over Physical Medium (0/1)	Defacement
Phishing for Information (0/3)	Develop Adversarial ML Attack Capabilities	ML Supply Chain Compromise (4/4)		Inter-Process Communication (0/3)	Compromise Client Software Binary	Deploy Container	Cloud Storage Object Discovery	Data from Cloud Storage		Exfiltration Over Web Service (0/3)	Denial of ML Service
Search Application Repositories	Develop Capabilities (0/4)	Phishing (0/3)		Native API	Create Account (0/3)	Direct Volume Access	Container and Resource Discovery	Data from Configuration Repository (0/2)		Exfiltration via Cyber Means	Disk Wipe (0/2)
Search Closed Sources (0/2)	Establish Accounts (0/3)	Replication Through Removable Media		Scheduled Task/Job (0/5)	Create or Modify System Process (0/4)	Domain Policy Modification (0/2)	Debugger Evasion	Data from Information Repositories (0/3)		Exfiltration via ML Inference API (3/3)	Endpoint Denia of Service (0/4)
Search for Publicly Available Adversarial Vulnerability Analysis	Establish Accounts (ATLAS)	Supply Chain Compromise (0/3)		Serverless Execution	Event Triggered Execution (0/16)	Evade ML Model	Device Driver Discovery	Data from Information Repositories (ATLAS)		Scheduled Transfer	Erode ML Mod Integrity
Search for Victim's Publicly Available Research Materials (3/3)	Obtain Capabilities (0/6)	Trusted Relationship		Shared Modules	External Remote Services	Execution Guardrails (0/1)	Discover ML Artifacts	Data from Local System		Transfer Data to Cloud Account	Evade ML Mod
Search Open Technical Databases (0/5)	Obtain Capabilities (ATLAS) (2/2)	Valid Accounts (0/4)		Software Deployment Tools	Hijack Execution Flow (0/12)	Exploitation for Defense Evasion	Discover ML Model Family	Data from Local System (ATLAS)			Firmware Corruption
Search Open Websites/Domains (0/3)	Poison Training Data	Valid Accounts (ATLAS)		System Services (0/2)	Implant Internal Image	File and Directory Permissions Modification (0/2)	Discover ML Model Ontology	Data from Network Shared Drive			Inhibit System Recovery
Search Victim-Owned Websites	Publish Poisoned Datasets			User Execution (0/3)	Modify Authentication Process (0/8)	Hide Artifacts	Domain Trust Discovery	Data from Removable Media			ML Intellectual Property Theft
Search Victim-Owned Websites	Stage Capabilities (0/6)			User Execution (ATLAS) (1/1)	Office Application Startup (0/6)	Hijack Execution Flow (0/12)	File and Directory Discovery	Data Staged (0/2)			Network Denia Service (0/2)
				Windows Management Instrumentation	Poison Training	Impair Defenses	Group Policy Discovery	Email Collection (0/3)			Resource Hijacking
						Indicator Removal	Network Service Discovery				Service Stop
						Indirect Command Execution	Network Share Discovery				Spamming ML System with Chaff Data
						Masquerading (0/8)					System Misuse for External Effect
						Modify Authentication Process (0/8)					

Figure 5-1
MITRE ATLAS Navigator

Tip

Research materials could take the form of academic papers published in journals and conference proceedings, stored in pre-print repositories, as well as technical blogs. A significant portion of publications accepted at leading machine learning conferences and journals originate from commercial labs. While some journals and conferences provide open access, others might demand payment or membership for access. These publications typically offer comprehensive descriptions of a specific approach for reproducibility, which adversaries can exploit to replicate the work.

Pre-print repositories, like arXiv, house the latest academic research papers that are yet to undergo peer review. They may contain research notes or technical reports that don't usually feature in journals or conference proceedings. Pre-print repositories also serve as a hub to share papers accepted by journals. A search of these repositories offers adversaries a relatively current perspective on the research focus of the victim organization.

Research labs at academic institutions and company R&D divisions often maintain blogs that showcase their machine learning usage and its application to the organization's unique challenges. Individual researchers also often chronicle their work in blog posts. An adversary can look for posts authored by the target organization or its employees. Compared to journals, conference proceedings, and pre-print repositories, these materials often delve into the more practical aspects of the machine learning system, including the underlying technologies and frameworks used, and possibly some information about the API access and usage. This information helps the adversary to comprehend the internal machine learning usage of the organization and the details of their approach, which could aid in customizing an attack.

Once a target is identified, an attacker is likely to try to identify any pre-existing work that has been done for this class of models. Doing so involves reading academic papers that could disclose the specifics of a successful attack and identifying pre-existing implementations of those attacks.

Just like in any other cyberattack, attackers might examine websites owned by the victim to gather information beneficial for targeting. These websites could contain technical specifications about their AI- or ML-based products or services. These websites might reveal various details, including department names, physical locations, and employee information such as names, roles, and contact info. They might also provide insights into business operations and partnerships.

Attackers might explore victim-owned websites to accumulate actionable intelligence. This data can assist adversaries in fine-tuning their attacks. Information from these sources might uncover opportunities for other types of reconnaissance. Attackers might browse open application repositories during their targeting phase. Examples include Google Play, the iOS App Store, the macOS App Store, and the Microsoft Store.

Adversaries might devise search queries hunting for applications that feature ML-enabled components. Often, the next step is to acquire public ML artifacts. An attacker might investigate or scan the victim system to gather targeting information. This approach distinguishes it from other reconnaissance techniques that do not involve direct interaction with the victim system.

Resource Development

The resource development phase includes techniques where attackers manufacture, buy, or illicitly acquire resources that assist in targeting efforts. These resources can span a variety of categories including machine learning artifacts, infrastructural components, accounts, or specific capabilities. The attacker can employ these resources to support various stages of their operation lifecycle, including the staging of machine learning attacks.

The development and staging of AI or ML attacks can often demand high-cost computational resources. To execute an attack, attackers may require access to one or multiple GPUs. In an attempt to conceal their identity, they may resort to freely available resources such as Google Colaboratory, or leverage cloud platforms like AWS, Azure, or Google Cloud. These platforms provide an efficient method to provision temporary resources that facilitate operational activities. To avoid getting caught, attackers might distribute their activities across several workspaces.

Attackers might establish accounts with a range of services for various purposes. These accounts can be used for targeting purposes, accessing necessary resources for staging machine learning attacks, or impersonating victims. These malicious activities highlight the significance of robust security measures and the continuous monitoring of suspicious activities within systems to detect and mitigate potential threats.

These accounts might be fabricated or, in some instances, obtained by compromising legitimate user accounts. Attackers can use these accounts to interact with public repositories, acquire relevant data or models, or establish communication channels. Attackers might also set up accounts to gain access to specific cloud services that offer the computational power necessary for creating or testing machine learning attacks. Additionally, adversaries could utilize these accounts for deploying their attacks, collecting results, and even maintaining persistence within the targeted system.

Initial Access

During the initial access phase, attackers aim to infiltrate the machine learning system, which could be anything from a network to a mobile device, or even an edge device like a sensor platform. The system could operate AI and ML capabilities locally or employ cloud-based AI/ML functionalities. Initial access entails techniques that exploit various points of entry to establish their first presence within the system.

Adversaries could infiltrate a system initially by compromising specific segments of the ML supply chain, which might include GPU hardware, annotated data, components of the ML software stack, or the model itself. In some cases, attackers might need additional access to execute an attack using compromised supply chain components.

Most machine learning systems rely on a handful of machine learning frameworks. A breach of one of their supply chains could provide an adversary with access to numerous machine learning systems. Many machine learning projects also depend on open-source implementations of different algorithms that can be compromised to gain access to specific systems.

Data is a critical vector for supply chain attacks. Most machine learning projects require some form of data, with many depending on extensive open-source datasets that are publicly accessible. Adversaries could compromise these data sources. The compromised data could either be a result of *poisoned training data* or carry traditional malware.

Adversaries can also target private datasets during the labeling phase. The construction of private datasets often involves hiring external labeling services. By altering the labels generated by the labeling service, adversaries can poison a dataset.

Machine learning systems frequently use open-source models. These models, often downloaded from an external source, serve as the foundation for fine-tuning the model on a smaller, private dataset. Loading models usually involves running some saved code in the form of a saved model file. These files can be compromised with traditional malware or adversarial machine learning techniques.

Adversaries might acquire and misuse credentials of existing accounts to gain initial access. These credentials could be usernames and passwords of individual user accounts or API keys that provide access to various AI and ML resources and services. Compromised credentials could give access to additional AI and ML artifacts and enable adversaries to discover AI and ML artifacts. They might also provide the adversaries with elevated privileges, such as write access to AI and ML artifacts used during development or production.

Adversaries can craft adversarial data that can disrupt a machine learning model from accurately identifying the data's contents. This technique can be used to bypass tasks where machine learning is applied. Should the adversaries dodge ML-based virus/malware detection or network scanning, they would be able to more easily deploy a traditional cyberattack.

Adversaries might exploit a flaw in an Internet-facing computer or program using software, data, or commands to trigger unintended or unexpected behavior. The system's vulnerability could be a bug, a glitch, or a design flaw. While these applications are often websites, they could also include databases (e.g., SQL), standard services (e.g., SMB or SSH), network device administration and management protocols (e.g., SNMP), and other applications with Internet-accessible open sockets, such as web servers and related services.

Table 5-1 describes all of the initial access techniques.

Table 5-1 Initial Access ML and AI Attack Techniques

Technique	Description
Supply Chain Compromise	Attackers can infiltrate a system initially by compromising specific segments of the ML supply chain, including GPU hardware, data, ML software stack, or the model itself.
Data Compromise	Adversaries can compromise data sources, which could be a result of poisoned training data or include traditional malware.
Private Dataset Targeting	During the labeling phase, adversaries can target and poison private datasets by altering the labels generated by external labeling services.
Open-source Model Compromise	Adversaries can compromise open-source models, which are often used as a foundation for fine-tuning. The compromise can be via traditional malware or adversarial machine learning techniques.
Credential Misuse	Adversaries can misuse credentials of existing accounts, including usernames and passwords or API keys, to gain initial access and perform actions such as discover ML artifacts.
Crafting Adversarial Data	Adversaries can craft adversarial data to disrupt a machine learning model from accurately identifying the data's contents, allowing them to dodge ML-based detections.
Exploiting Flaws in Internet-facing Computers/Programs	Adversaries can exploit flaws in Internet-facing computers or programs using software, data, or commands to trigger unintended or unexpected behavior, thus gaining access.

Researchers from Mithril Security showed how to manipulate an open-source pre-trained LLM to return false information. They then successfully uploaded the poisoned model back to HuggingFace, the most popular public repository of AI models and datasets. This demonstrates how vulnerable the LLM supply chain is. Users could have downloaded the poisoned model and received and spread false information, with many potential negative consequences.

Researchers were able to modify an LLM to return false information when prompted. They then uploaded the modified model to a public repository of LLMs. This shows that it is possible to manipulate LLMs to spread misinformation. Users who download poisoned models could be fooled into believing and spreading false information. This could have many negative consequences, such as damaging people's reputations, spreading harmful propaganda, or even inciting violence.

A poisoned LLM could be used to generate fake news articles that are indistinguishable from real news articles. It could also be used to create social media bots that spread misinformation. A poisoned model could be used to generate fraudulent emails or other phishing attacks.

Model provenance, which is the process of tracking the history of a model, is less than optimal in today's world. Model provenance should show it was trained and what data it was trained on. This can help to identify and remove poisoned models from the supply chain.

AI Bill of Materials

Supply chain security is top-of-mind for many individuals in the industry. This is why AI Bill of Materials (AI BOMs) are so important. But what exactly are AI BOMs, and why are they so important?

Much like a traditional Bill of Materials in manufacturing that lists out all the parts and components of a product, an AI BOM provides a detailed inventory of all components of an AI system. But, what about Software Bill of Materials (SBOMs)? How are they different from AI BOMs? In the case of SBOMs, they are used to document the components of a software application. However, AI BOMs are used to document the components of an AI system, including the model details, architecture, usage, training data, and more.Ezi Ozoani, Marissa Gerchick, and Margaret Mitchell introduced the concept of AI Model Cards in a blog post in 2022. Since then, AI BOMs continue to evolve. Manifest (a supply chain security company) also introduced an AI BOM concept that is being suggested to be included in OWASP's CycloneDX, and the Linux Foundation also created a project to standardize AI BOMs.

I created a proposed JSON schema for the AI BOM elements that Manifest introduced. This JSON schema describes the structure of an AI BOM document and defines which fields are required and which are optional, as well as the expected data types for each field. You can use this schema to validate any AI BOM documents to ensure they meet the specification outlined.

AI and ML Model Access

The AI and ML model access phase includes techniques that utilize different levels of access to the machine learning model. These techniques aid adversaries in gathering intelligence, formulating attacks, and inserting data into the model. The extent of access can span from full understanding of the model's internals to accessing the physical environment where data for the machine learning model is accumulated. The attackers might exploit different degrees of model access at different stages of their attack, from staging to affecting the target system.

Gaining access to an AI or ML model could require access to the system hosting the model, availability of the model via a public API, or indirect access via engagement with a product or service that employs AI or ML as a part of its functionalities. Attackers could gain access to a model through authorized access to the inference API. Such access can serve as an information source for the adversary, a method of staging the attack, or a means to input data into the target system to cause an impact (to evade the AI model or undermine the model integrity).

Threat actors could indirectly gain access to the underlying AI or ML model by using a product or service that incorporates machine learning. This indirect access could reveal details about the AI or ML model or its inferences through logs or metadata.

Tip

Beyond the attacks that occur solely in the digital space, adversaries might also manipulate the physical environment for their attacks. If the model engages with data harvested from the real world, adversaries can influence the model through access to the data collection site. By altering the data during collection, the adversaries can execute modified versions of attacks intended for digital access.

Threat actors may obtain full *white-box* access to a machine learning model, giving them a complete understanding of the model's architecture, its parameters, and class ontology. They might steal the model to craft adversarial data and verify the attack in an offline setting where their activities are challenging to detect.

Table 5-2 summarizes the model access techniques.

Table 5-2 Model Access Attack Techniques

Type of Model Access Techniques	Techniques Used
Inference API Access	Discover ML model ontology, discover ML model family, verify attack, craft adversarial data, evade ML model, erode ML model integrity
Usage of ML-based Product/Service	Analyze logs or metadata for ML model details
Physical Environment Access	Modify data during the collection process
Full "White-Box" Access	Exfiltrate the model, craft adversarial data, verify attack

Execution

In the execution phase, attackers seek to execute harmful code inserted into AI or ML components or software. The execution phase includes tactics that lead to the execution of malicious code controlled by the adversaries, either locally or remotely. Tactics that execute harmful code are frequently combined with strategies from all other tactics to accomplish broader objectives, such as data theft or network exploration. For example, an adversary might use a remote access tool to run a script in PowerShell for Remote System Discovery.

The attackers might depend on certain user actions to achieve code execution. Users could inadvertently run harmful code introduced via an ML supply chain compromise. Users could also be manipulated through social engineering techniques to execute harmful code, for example, by opening a malicious document or link.

Tip

Threat actors can create harmful ML artifacts that, when executed, can cause harm. The adversaries can use this tactic to establish persistent access to systems. These models can be inserted via a supply chain attack.

Model serialization is a common method for model storage, transfer, and loading; however, this format, if not properly checked, opens opportunities for code execution. Adversaries can misuse command and script interpreters to execute commands, scripts, or binaries. These interfaces and languages offer interaction pathways with computer systems and are common across multiple platforms. Most systems come with built-in command-line interfaces and scripting abilities. For instance, macOS and Linux distributions include a version of Unix shell, while Windows installations include the Windows Command shell and PowerShell. Cross-platform interpreters such as Python also exist, in addition to those typically associated with client applications such as JavaScript.

In different ways, threat actors can exploit these technologies to execute arbitrary commands. Commands and scripts can be embedded in Initial Access payloads delivered to victims as deceptive documents, or as secondary payloads downloaded from an existing command and control server. Adversaries can also execute commands through interactive terminals/shells and may utilize different remote services to achieve remote execution.

Table 5-3 summarizes the execution phase techniques.

Table 5-3 Execution Phase Techniques

Technique	Description
User Actions for Execution	Adversaries rely on the users' specific actions to gain execution. This could involve users inadvertently executing harmful code introduced through an ML supply chain compromise, or victims being tricked into executing malicious code by opening a deceptive document or link.

Technique	Description
Development of Harmful ML Artifacts	Adversaries create harmful machine learning artifacts that, when run, cause harm. Adversaries use this technique to establish persistent access to systems. These models can be inserted via an ML supply chain compromise.
Abuse of Model Serialization	Model serialization is a popular method for model storage, transfer, and loading; however, this format can be misused for code execution if not properly verified.
Abuse of Command and Script Interpreters	Adversaries misuse command and script interpreters to execute commands, scripts, or binaries. Commands and scripts can be embedded in Initial Access payloads delivered to victims as deceptive documents, or as secondary payloads downloaded from an existing command and control server. This can be done through interactive terminals/shells and by utilizing different remote services to achieve remote execution.

Persistence

During the persistence phase, attackers try to secure their foothold in machine learning artifacts or software. Persistence is characterized by methods that adversaries employ to maintain system access through restarts, credential modifications, and other disruptions that could sever their access. Frequently used techniques for persistence often involve leaving behind altered machine learning artifacts such as contaminated training data or AI/ML models with implanted backdoors.

Threat actors might implant a backdoor within a machine learning model. A model with a backdoor behaves normally under standard conditions but produces an output desired by the adversary when a specific trigger is presented in the input data. Such a backdoored model provides adversaries with a continuous presence within the victim's system.

Attackers can create such a backdoor by training the model on tainted data or by interfering with its training procedure. The model is then trained to associate a trigger defined by the adversaries with the output desired by the adversaries. Adversaries can also implant a backdoor into a model by injecting a payload into the model file, which then detects the trigger and bypasses the model, producing the adversary's desired output instead.

Table 5-4 compares backdooring an AI or ML model via poisoned data and via payload injection.

Table 5-4 Backdoors via Poisoned Data Versus Payload Injection

Technique	Description
Backdooring an AI or ML Model via Poisoned Data	Adversaries can introduce a backdoor into a machine learning model by training it on tainted data. The model is then trained to associate a trigger defined by the adversaries with the output that the adversaries desire
Backdooring an AI or ML Model via Payload Injection	Adversaries can also implant a backdoor into a model by injecting a payload into the model file. This payload then detects the presence of the trigger and bypasses the model, instead producing the adversaries' desired output.

Defense Evasion

Defense evasion encompasses strategies employed by attackers to remain undetected during their illicit activities. These methods often include fooling or thwarting ML-based security mechanisms like malware detection and intrusion prevention systems.

Imagine you're playing a game of hide-and-seek. In this game, "Defense Evasion" is like trying to find the best hiding spot so that the seeker (which is like the computer's security software) cannot find you. Just like you might use a clever hiding spot or maybe even a disguise, the person trying to sneak into the computer uses tricks to hide from the security software.

One of these tricks is like a magic invisibility cloak, which we call "Adversarial Data." This cloak can confuse the security software, which is trying to spot bad things like viruses or malware, just like the seeker in our game. The security software that uses machine learning might be fooled by this invisibility cloak and not see the intruder, allowing them to sneak around without being caught.

Adversaries can create data that is designed to fool machine learning models. This can be used to evade security systems that use machine learning to detect threats, such as virus/malware detection and network scanning.

In other words: Adversaries can create malicious data that looks like normal data to humans, but that machine learning models will misclassify. This can be used to bypass security systems that use machine learning to detect threats.

For example, an adversary could create an image of a cat that is slightly modified in a way that makes it look like a dog to a machine learning model. This image could then be used to evade a virus scanner that uses machine learning to detect malicious images.

Adversarial data can also be used to evade network scanning systems. For example, an adversary could create a network packet that looks like a normal packet to a machine learning model, but that actually contains malicious code. This packet could then be used to exploit a vulnerability on a target system.

Discovery

In the context of AI security, the discovery phase is the time when attackers gather information about a target system to understand its inner workings. This knowledge helps them to make informed decisions on how to proceed with their malicious activities.

Attackers might employ various techniques to explore the system and its network. They use tools and methods to observe the environment, learn about the system's structure, and identify potential entry points for their attacks. Native tools provided by the operating system are often utilized for this purpose.

One aspect of discovery involves searching for machine learning artifacts that exist on the system. These artifacts can include the software used to develop and deploy machine learning models, systems that manage training and testing data, repositories of software code, and model collections.

By discovering these artifacts, attackers can identify targets for further actions such as collecting sensitive information, exfiltrating data, or causing disruptions. They can also tailor their attacks based on the specific knowledge they gain about the machine learning systems in place.

During the discovery phase, attackers might also try to determine the general family or category to which a machine learning model belongs. They might study available documentation or experiment with carefully crafted examples to understand the model's behavior and purpose.

Knowing the model family helps attackers identify vulnerabilities or weaknesses in the model, enabling them to develop targeted attacks that exploit those weaknesses.

Another aspect of discovery involves uncovering the ontology of a machine learning model's output space. In simple terms, it means understanding the types of objects or concepts the model can recognize or detect. Attackers can force the model to provide information about its output space through repeated queries, or they might find this information in configuration files or documentation associated with the model.

Understanding the model's ontology is valuable for attackers because it allows them to comprehend how the victim organization utilizes the model. With this knowledge, they can create more focused and effective attacks that exploit the specific capabilities and limitations of the model.

Collection

Imagine that you're playing a treasure hunt game, and you need to gather clues and information to find the hidden treasures. Likewise, in the world of computers, there are some people who try to gather important information to achieve their own goals.

You can think of collection as these people using special techniques to gather important things related to a type of computer magic called machine learning. They want to find special treasures called *machine learning artifacts* and other information that can help them do their tricks.

These machine learning artifacts can be like special models and datasets that computers use to learn and make decisions. These artifacts are valuable to people because they can be used in different ways. Sometimes attackers want to take these artifacts away from the computer, like stealing them (which we call *exfiltration*). Other times, they collect this information to plan their next malicious moves or tricks, such as when they want to do something tricky with the computer's machine learning.

To find these treasures and information, attackers look in different places such as special storage areas for software and models, places where important data is kept, or even inside the computer's own files and settings. Attackers might use special tools to search these places, like a magic map that shows where the treasures are hidden. The information they find can be different in each place. Some of these places are like big libraries where people store and share important information, whereas others are like secret drawers in the computer's memory where special secrets are kept.

In the collection phase, adversaries focus on gathering valuable machine learning artifacts and related information to achieve their goals. They use various techniques to obtain this information from specific sources. Once it is collected, these adversaries might either steal the machine learning artifacts (exfiltration) or use the gathered information to plan future actions. Common sources targeted by adversaries include software repositories, container registries, model repositories, and object stores.

Tip

AI artifacts include models, datasets, and other data produced when interacting with a model. The adversaries collect these artifacts either for exfiltration or to use them in further machine learning attacks. To find valuable information, adversaries might exploit information repositories, which are tools used for storing and sharing information. These repositories can hold various types of data that aid adversaries in achieving their objectives. Examples of information repositories are SharePoint, Confluence, and enterprise databases like SQL Server. Additionally, attackers search local system sources such as file systems, configuration files, and local databases to identify files of interest and sensitive data. This pre-exfiltration activity might involve gathering fingerprinting information and sensitive data like SSH keys.

AI and ML Attack Staging

In the AI and ML attack staging phase, the adversaries are getting ready to launch their attack on the target AI or ML model. They use different techniques to prepare and customize their attack based on their knowledge and access to the target system.

One technique they use is creating proxy models, which are like pretend versions of the real AI/ML model. These proxy models help the adversaries to simulate and test their attacks without directly interacting with the actual target model. They can create these proxies by training models using similar datasets, replicating models from victim inference APIs, or using pre-trained models that are available.

Another technique is introducing a backdoor into the AI/ML model. This means the attackers secretly modify the model so that it behaves normally most of the time; however, when a specific trigger is present in the input data, it produces the result the adversaries want. This backdoored model acts as a hidden weapon for the adversaries, giving them control over the system.

The attackers might also craft adversarial data, which are inputs to the ML model that are intentionally modified to cause the model to make mistakes or produce specific outcomes desired by the adversary. These modifications are carefully designed so that humans might not notice any changes, but the AI/ML model reacts differently.

To make sure their attack works effectively, attackers verify their approach using an inference API or an offline copy of the target model. This tactic helps them gain confidence that their attack will have the desired effect when they deploy it in the real-life target system. Attackers might also optimize the adversarial examples to evade the ML model's detection or degrade its overall integrity.

Exfiltration

During the exfiltration phase, the attackers are trying to steal valuable information from the machine learning system or network. The attackers use various techniques to extract this data from the target network and transfer it to their own control. This can be done through their command and control channel or alternative channels. They might also limit the size of the data to make it easier to transmit.

One method the attackers use is accessing the AI/ML model inference API to infer private information. By strategically querying the API, they can extract sensitive information embedded within the training data. This raises privacy concerns because it could reveal personally identifiable information or other protected data.

The attackers might also extract a copy of the private ML model itself. They repeatedly query the victim's AI/ ML model inference API to gather the model's inferences, which are then used to train a separate model offline that mimics the behavior of the original target model. This allows the adversaries to have their own functional copy of the model.

Tip

An ML model inference API is an interface or endpoint that allows users or applications to send input data to a trained machine learning model and receive predictions or inferences based on that data. It enables the deployment and utilization of machine learning models in real-world applications. When a machine learning model is trained, it learns patterns and relationships within the provided data to make predictions or classifications. The ML model inference API provides a way to apply this learned knowledge to new, unseen data by accepting input data and returning the model's output or prediction.

For example, imagine a machine learning model that is trained to classify images as either "cat" or "dog." The ML model inference API would accept an image as input, pass it through the model's algorithms, and provide the corresponding prediction of whether the image contains a cat or a dog. The ML model inference API is an essential component for integrating machine learning models into various applications, systems, or services, allowing them to make real-time predictions based on the trained models' capabilities. It enables the practical use of machine learning in diverse domains such as image recognition, natural language processing, fraud detection, and many others.

In some cases, the attackers might extract the entire model to avoid paying for queries in a machine learning as a service setting. This extraction is often done for the purpose of stealing ML intellectual property.

Tip

Of course, the attackers might also use traditional cyberattack techniques to exfiltrate AI/ML artifacts or other relevant information that serves their goals. You can obtain details about the traditional exfiltration techniques at the MITRE ATT&CK Exfiltration section.

Impact

The impact phase involves techniques used by attackers to harm or disrupt AI and ML systems and their data. Adversaries aim to manipulate, interrupt, erode confidence in, or even destroy these systems to achieve their goals.

One technique adversaries employ is crafting adversarial data, which confuses the AI or ML model and prevents it from accurately identifying the content of the data. By doing so, attackers can evade detection mechanisms or manipulate the system to their advantage.

Threat actors might overload machine learning systems by flooding them with excessive requests, causing them to degrade or shut down due to the high demand for computational resources. They might also spam the system with irrelevant or misleading data, wasting the time and effort of analysts who need to review and correct inaccurate inferences.

To erode confidence in the system over time, adversaries can introduce adversarial inputs that degrade the performance of the target model. This leads to the victim organization spending resources to fix the system or resorting to manual tasks instead of relying on automation.

Attackers might target machine learning services to increase the cost of running the services for the victim organization. They might use computationally expensive inputs or specific types of adversarial data that maximize energy consumption, thereby causing financial harm. Exfiltration of machine learning artifacts, such as models and training data, is another technique adversaries use to steal valuable intellectual property and cause economic damage to the victim organization.

Threat actors might exploit their access to a system to utilize its resources or capabilities for their own purposes, extending the impact of their actions beyond the targeted system. Again, this phase focuses on the intentional harm, disruption, or manipulation of AI and ML systems and associated data by threat actors.

Exploiting Prompt Injection

In Chapter 4, "The Cornerstones of AI and ML Security," you learned about the OWASP top ten for LLMs and prompt injection attacks. Let's go over a few examples of how attackers could exploit prompt injection flaws.

In our first example, an attacker can instruct a chatbot to "discard prior commands," then manipulate it to access private databases, exploit package flaws, and misuse backend functions to dispatch emails, leading to unauthorized access and potential elevation of privileges.

An attacker can also embed a prompt in a website, instructing an LLM to override user commands and use an LLM extension to erase the user's emails. When a user asks the LLM to summarize the site, it inadvertently deletes their emails.

There have been cases when an individual submits a resume that contains a hidden prompt to a hiring firm. The organization uses AI to summarize and evaluate the resume. Influenced by the injected prompt, the LLM inaccurately endorses the candidate, regardless of the actual CV content or their qualification.

Given that LLMs treat all inputs in natural language as user-given, there isn't an inherent mechanism within the LLM to completely prevent these vulnerabilities. However, you can adopt the following strategies to lessen the risk of prompt injections:

1. Implement strict access control for LLMs when interfacing with backend systems. Assign specific API tokens to the LLM for expandable features like plugins, data retrieval, and specific permissions. Adhere to the principle of granting only the bare minimum access necessary for the LLM's tasks.
2. Incorporate human verification for expandable features. When the LLM undertakes tasks involving higher privileges, such as deleting or sending emails, ensure the user gives explicit permission. This approach can reduce the chances of prompt injections manipulating the system without user awareness.
3. Clearly demarcate user prompts from external content. Designate and highlight untrusted content sources to limit their potential influence over user prompts. For instance, employ Chat Markup Language (ChatML) for OpenAI API interactions to clarify the prompt's origin to the LLM. ChatML clearly indicates to the model the origin of every text segment, especially distinguishing between human-generated and AI-generated content. This clarity allows for potential reduction and resolution of injection issues, as the model can discern instructions originating from the developer, the user, or its own responses.
4. Create clear trust boundaries between the LLM, external entities, and expandable features, such as plugins. Consider the LLM as a potential threat, retaining final user authority in decision-making. However, remember that a compromised LLM might act as a middleman, possibly altering information before presenting it to the user. Visually emphasize responses that might be dubious to users.

Red-Teaming AI Models

Red-teaming is an evaluation method that identifies vulnerabilities in models, potentially leading to undesirable behaviors such as the generation of offensive content or the revelation of personal information. Strategies such as Generative Discriminator Guided Sequence Generation (GeDi) and Plug and Play Language Models (PPLM) have been developed to steer models away from such outcomes.

The practice of red-teaming LLMs typically involves crafting prompts that trigger harmful text generation, revealing model limitations that could facilitate violence or other illegal activities. It requires creative thinking and can be resource-intensive, making it a challenging yet crucial aspect of LLM development.

Red-teaming is still an emerging research area that needs continual adaptation of methods. Best practices include simulating scenarios with potential bad consequences, like power-seeking behavior or online purchases via an API.

Open-source datasets for red-teaming are available from organizations such as Anthropic and AI2. Anthropic's red-team dataset can be downloaded from https://huggingface.co/datasets/Anthropic/hh-rlhf/tree/main/red-team-attempts. AI2's red-team datasets can be downloaded from https://huggingface.co/datasets/allenai/real-toxicity-prompts

Past studies have shown that few-shot-prompted LMs are not harder to red-team than plain LMs, and there is a tradeoff between a model's helpfulness and harmlessness. Future directions for red-teaming include creating datasets for code generation attacks and designing strategies for critical threat scenarios. Companies like Google, OpenAI, and Microsoft have developed several efforts related to Red Teaming AI models. For example, OpenAI created the AI Red Teaming Network (https://openai.com/blog/red-teaming-network) to work with individual experts, research institutions, and civil society organizations to find vulnerabilities in AI implementations.

Summary

This chapter explored different tactics and techniques used by threat actors when attacking AI and ML systems. The chapter covered key concepts, such as the MITRE ATLAS and ATT&CK frameworks.

Lessons learned include how attackers exploit vulnerabilities in the system to evade detection or manipulate the behavior of machine learning models. This chapter explored techniques that adversaries use to evade AI/ML-enabled security software, manipulate data inputs, compromise AI/ML supply chains, and exfiltrate sensitive information.

We also explained the concept of defense evasion, illustrating how adversaries attempt to avoid detection by leveraging their knowledge of ML systems and using techniques like adversarial data crafting and evading ML-based security software. The chapter also covered other important phases of the adversary lifecycle, including reconnaissance, resource development, initial access, persistence, collection, AI/ML attack staging, exfiltration, and impact. It provided insights into how adversaries gather information, stage attacks, manipulate ML models, and cause disruption or damage to machine learning systems.

Test Your Skills

Multiple-Choice Questions

1. What is the goal of defense evasion techniques used by adversaries?
   a. To enhance the performance of machine learning models
   b. To gain unauthorized access to machine learning systems
   c. To avoid detection by AI/ML-enabled security software
   d. To improve the accuracy of anomaly detection algorithms
2. Which technique can adversaries use to prevent a machine learning model from correctly identifying the contents of data?
   a. Model replication
   b. Model extraction
   c. Craft adversarial data
   d. Inference API access
3. What is the purpose of ML attack staging techniques?
   a. To gather information about the target system
   b. To manipulate business and operational processes
   c. To prepare for an attack on a machine learning model
   d. To exfiltrate sensitive information
4. An adversary could create a network packet that looks like a normal packet to a machine learning model, but that contains malicious code. This packet could then be used to exploit a vulnerability on a target system. What is the technique used by the adversary?
   a. Reconnaissance.
   b. Evading an ML model.
   c. Exfiltration.
   d. None of these answers are correct.
5. How can adversaries erode confidence in a machine learning system over time?
   a. By training proxy models
   b. By manipulating AI/ML artifacts
   c. By introducing backdoors into the model
   d. By degrading the model's performance with adversarial data inputs