3

AI THREATS

Criminals, scammers, and sophisticated nation-state actors regularly target computer systems to make money, steal data, collect intelligence, or gain strategic advantage, and AI systems are now part of that threat landscape too. Securing AI systems means protecting not just our organizations but also our countries, our institutions, and even our lives.

Although you may believe AI is worth securing, you might underestimate just how much effort threat actors put into compromising digital systems. The FBI estimates that North Korea spends up to 20 percent of its military budget on cyber operations to fund its regime, and studies have suggested that, if measured as a national economy, global cybercrime would rank as the world's third largest—behind only the economies of the United States and China.

As an example of how far attackers will go, consider one incident: In 2024, a mining company fell victim to a business email compromise scam in which a threat actor gained persistent access to the CEO's email inbox and instructed the CFO to wire millions of dollars into the attacker's account. The attack began almost a year earlier when the CEO took a trip to a foreign country. There, a criminal network had set up a fake hotel—complete with a fancy lobby, a single guest floor, and functional room service and housekeeping—and compromised the Wi-Fi to access guests' computers. This operation likely cost hundreds of thousands of dollars, but the payoff made it worthwhile. Criminals will invest as much time as needed to compromise your systems; if they're nation-state actors, they'll work on it full time until they succeed.

In this chapter, we'll focus on fundamental security principles as they apply to AI systems. We'll work through a threat modeling exercise, discuss current developments in AI threat intelligence, and explore who and what we're securing our AI systems against. The goal of this chapter is not to provide a comprehensive overview of the ways AI can fail or be breached—that will be covered in subsequent chapters—but to build intuition about the aspects of machine learning models and AI systems that create weaknesses and vulnerabilities, and the cybersecurity tools we can use to analyze them.

The Attack Surface

A helpful way to consider the landscape of potential threats is to refer to the *attack surface*, or all the potential ways, or vectors, through which an attacker could attempt to enter, exploit, or compromise a system.

For example, to an attacker aiming to compromise my personal information, my iPhone represents an enticing attack surface. There are multiple vectors from which they could launch an attack against my device, and with the information they acquire, they could compromise my digital information and physical safety in many ways. First, if an attacker chose the physical attack vector, they might steal my phone or hijack its charging station to get closer access to my information. If I connect to insecure networks like Wi-Fi or Bluetooth, they could intercept my data or inject malicious payloads, such as *spyware*—software designed to covertly monitor and collect my information.

All my mobile apps could potentially be malicious fake apps that track my behavior or steal data. Alternatively, they may be legitimate apps with vulnerabilities that attackers could exploit. For example, these vulnerabilities could leak sensitive data such as *session tokens* (stored in cookies), which are the temporary digital keys that systems provide after login so that you don't have to submit your password every time. They could also allow attackers to inject and execute malicious code or compromise supporting backend APIs. An attacker might social-engineer me through text messages or emails asking me to click malicious links. They could also send me *zero-click exploits*—more sophisticated malware that doesn't require my interaction to execute. Additionally, my phone's operating system may

have vulnerabilities; for instance, if I haven't updated it in a while, publicly known software flaws may remain unpatched.

Even without AI, my phone represents a rich attack surface. The more components, applications, and interactions a device has, the larger the attack surface becomes. Adding any app to my phone increases the attack surface, but adding an AI app causes it to grow disproportionately.

For example, if I downloaded an AI assistant app like those many companies are currently developing, it would collect my data in various new ways—through natural language, images, and videos—and have the ability to learn, plan, and act autonomously. It could access my files and infer information about me. Future agent-based assistants may even use my other apps as if they were me. Now, consider the attack surface of large enterprise AI solutions; the potential impact of an incident includes large-scale data breaches, operational disruptions, financial losses, and reputational damage.

Threat Modeling

Threat modeling is a cybersecurity method used to systematically identify, assess, and proactively mitigate threats to products and systems before attackers can exploit them. The general threat modeling process typically involves clearly defining the system's boundaries; identifying *assets*—anything of value within a system or organization that requires protection, such as sensitive information or intellectual property); mapping data flows and system interactions; and then listing possible threats and vulnerabilities.

Security professionals typically use methodologies that list common attack tactics, techniques, and procedures (TTPs) to better understand potential attack pathways. Good threat models support decision-making by showing not only what attacks are possible but also which are most likely and most damaging for a given organization or agent. They can be modular, focusing on particular components or layers of a system, and should be updated regularly. The methodology should result in a comprehensive and actionable threat profile report that clearly describes the system's architecture; ranks threats according to their severity and probability; and includes realistic attack scenarios and their impacts.

Importantly, the report recommends specific mitigation strategies to effectively address identified threats. This information is crucial because it helps you understand what attacks are possible, what attacks are most likely to occur, and who might be responsible. For example, if you belong to a small start-up developing an AI product based on API access to GPT models, your threat profile would differ significantly from that of a multinational company offering AI enterprise products to clients with stringent security and compliance requirements.

Let's produce a simple threat model for an AI system.

System Diagrams

We'll begin by clearly defining our system's main components and how they interact with the environment. To aid in this process, most AI tools have an *AI bill of materials*: a structured, technical inventory that lists all the system's components, similar to a traditional software bill of materials.

Our threat model will consider two distinct perspectives of system access. In the first, interaction with the model occurs solely through its API, while the second assumes we have access to both the training and production environments of the machine learning model. Let's start with the first case, API access, as illustrated in Figure 3-1.

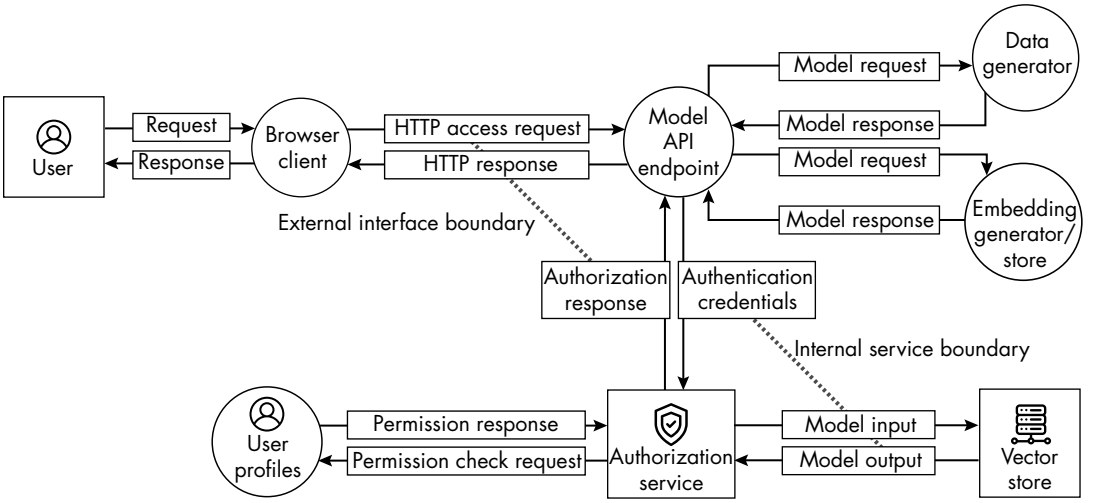

Figure 3-1: A system diagram for API-only model access

This diagram represents the system's architecture. Squares and rectangles depict internal components or services that the organization owns or manages, circles indicate external entities or supporting services, and arrows show the direction of data flow. Dashed lines mark important trust or security boundaries, including zones where user input crosses into backend infrastructure or where system logic accesses sensitive resources like databases or models.

I modeled this diagram on production-grade AI architectures used in the industry. As you can see, users interact with the AI through a web browser. Their request passes to the model's API endpoint, which acts as the main controller for processing input and returning responses. Before handling the request, the API checks whether the user's authentication credentials are valid and, if so, returns a session token. The API also verifies what the user is allowed to access.

Once the user has been authorized, the API routes the request to one of two backend components: a *data generator*, which might be a language model generating text, or an *embedding generator*, which interacts with a local knowledge store, or *vector store*, for tasks such as semantic search (identifying context and meaning). The model's response is then processed and returned to the user.

We can see three important security regions in the figure. First, there is the external client zone representing the user and their inputs. This is treated as a low-trust area, meaning we assume it may be compromised, malicious, or unreliable. Next is the perimeter, or API layer, which acts as a broker between the client and our systems and is considered a medium-trust zone because we have greater control here. Finally, our database and model backend fall within our control and therefore are treated as a high-trust zone, where we can reasonably assume the data is accurate, the processes work as intended, and access is limited to the right people or systems.

When working with system diagrams, professionals often highlight key components by drawing boxes around them to show where threats or mitigations apply. For example, they could draw a red box around the browser client with a list of associated threats, like malicious input or output leakage. Conversely, they might place a green box around the model API endpoint, noting recommended security controls. Perhaps you could try doing this yourself.

Now let's consider the second case, where we have full access to the machine learning training and production environments (Figure 3-2).

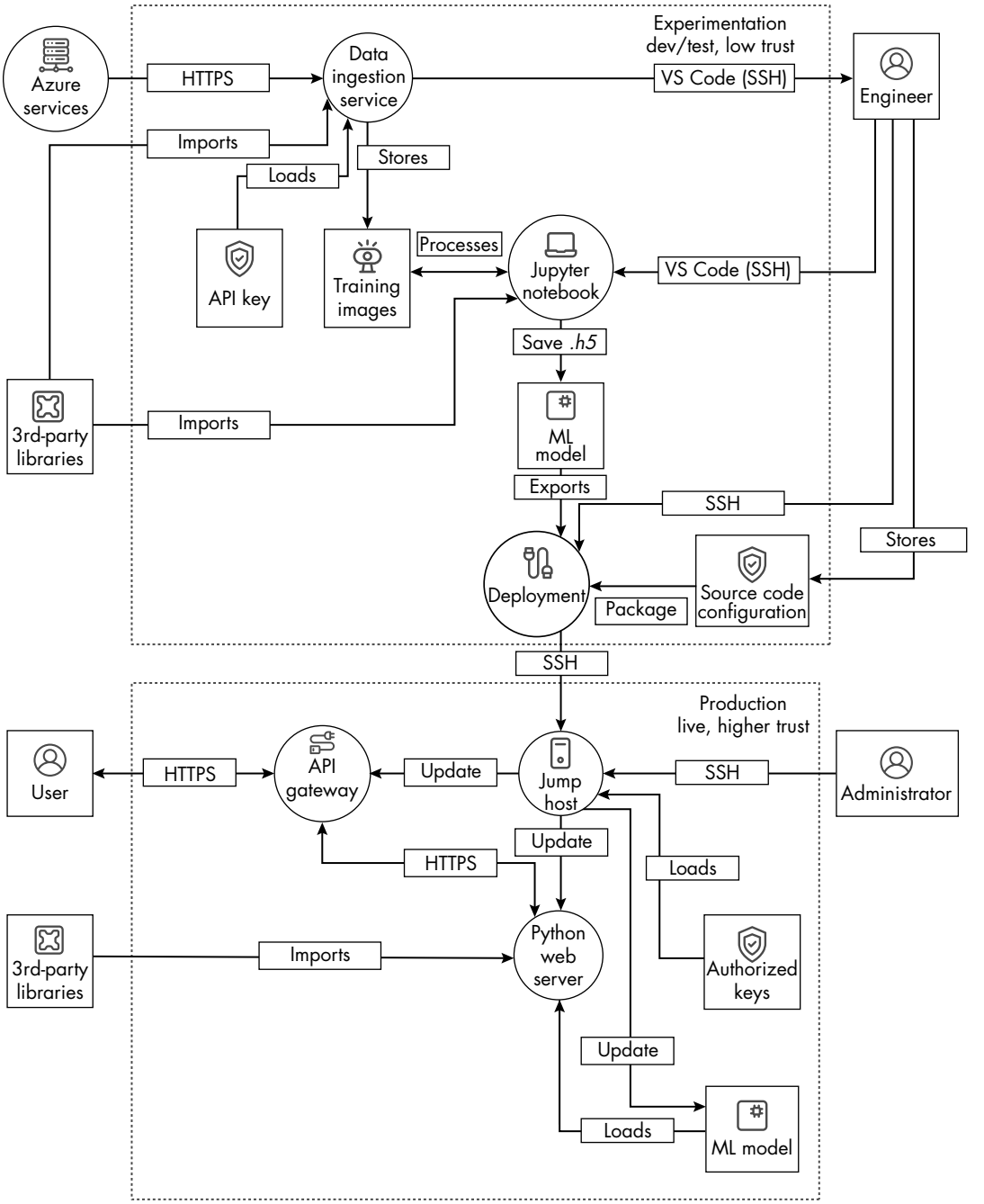

Figure 3-2: A system diagram that includes the development and production of a machine learning model

This diagram illustrates the full development and deployment pipeline. The top half represents the experimentation environment in which engineers develop models. There, a data ingestion service collects data from sources like Azure Services or APIs, then stores and processes it as training

images. These images get loaded into a Jupyter notebook, where engineers train and test the model. Once saved in the *.h5* format, the model gets passed to a deployment component, along with packaging instructions and code from a source code management system, which ensures version control and consistency.

The bottom half of the diagram shows the production environment, which users interact with via a web browser and API gateway. A secure *jump host* (a server that provides controlled access to devices or systems within a private network) acts as the access point to production infrastructure. The Python web server handles user requests, loads the deployed model, and returns outputs.

An administrator uses *Secure Shell (SSH)*, a secure remote access protocol, to manage infrastructure, update model versions, and maintain authorized key access. Third-party libraries support both environments, while the use of HTTPS and SSH throughout reflects industry-standard security best practices. The two dashed boxes labeled Experimentation and Production mark the separation between development and operational environments—a critical security boundary to prevent untested code or data from affecting live systems.

System diagrams such as those shown in Figures 3-1 and 3-2 are essential for effective threat modeling because they provide a clear, visual representation of how a system's data, users, and components interact, making it easier to identify trust boundaries, entry points, and sensitive areas where vulnerabilities are most likely to emerge. Take special note of the parts of the diagram involving authentication and authorization flows, model access, and data transfer, which are common areas of weakness. These diagrams can also help nontechnical individuals understand how the system works and where potential risks lie.

Now that you've seen examples of component diagrams, let's explore two frameworks for threat modeling AI systems. The goal of this exercise isn't to comprehensively introduce all possible attacks—I'll discuss many of the issues mentioned here in more detail in subsequent chapters. Instead, we'll aim to understand how practitioners map weaknesses across an AI system.

Example 1: MITRE ATLAS and OWASP

Cybersecurity has many existing frameworks for threat modeling, including STRIDE, MITRE ATT&CK, OWASP's Top 10 lists, and Cyber Kill Chain, but a few newer frameworks cover AI threat modeling specifically. If you want to threat-model non-agentic systems, one option is to use MITRE ATLAS, short for "Adversarial Threat Landscape for Artificial-Intelligence Systems."

ATLAS is a knowledge base of adversarial AI tactics, techniques, and case studies, built in the style of the influential cybersecurity framework MITRE ATT&CK. In its full form, ATLAS organizes adversarial behavior into life cycle tactics such as reconnaissance, resource development, initial access, AI model access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, collection, AI attack staging, command

and control, exfiltration, and impact, each of which is broken down into specific techniques and illustrated with real-world case studies.

These tactics are very powerful if you already think in terms of the ATT&CK model, but for those less familiar with cybersecurity, it can be hard to see how they map to AI systems in practice. For that reason, in this section I'll reorganize these categories into a set of broader, system-focused buckets that capture where in the AI life cycle the threats apply: data attacks, model attacks, supply chain and dependencies, infrastructure and runtime, outputs and application layer, and operators and governance. This abstraction makes the risks more intuitive to visualize within AI system diagrams.

Our example threat model will also integrate risks from the OWASP ML, LLM, and Generative AI Security top 10 lists, and we'll follow an industry convention by tagging each risk inline. For example, the tag "ML02: Evasion" identifies model evasion, the second risk in OWASP's ML Security Top 10 List. Likewise, risks tagged "GenAI" come from the Generative AI Security Top 10 List, and those tagged "LLM" come from the LLM Security Top 10 List.

Data Attacks

In MITRE ATLAS, the primary risks to data involve poisoning training data and manipulating features or datasets. These activities correspond to early stages of the adversary life cycle, such as initial access, AI model access, and execution.

For example, in the experimentation environment, model training takes place in a Jupyter notebook using data collected by the Data Ingestion Service. If the data being pulled from external sources isn't properly validated or curated, it opens the risk of data and model manipulation (ML01: Data Poisoning; GenAI 04: Data/Model Poisoning; LLM03: Training Data Poisoning). API keys and credentials used during training may be hard-coded or exposed in notebooks, which can facilitate credential leakage or misuse (ML03: Training Data Leakage). Training data tampering can be intentional, such as poisoning data to force harmful or ideologically motivated misclassifications (ML01; GenAI 04; LLM03), or it can be unintentional, through poor cleaning or sanitization that introduces hidden biases or errors (ML07: Bias & Fairness Failures).

Attacks can also target the feature extraction process to alter how inputs are represented, causing the model to learn distorted patterns (ML02: Evasion). These alterations can have very different effects depending on scale: They may cause obvious inference issues that are easy to detect, or they may only manifest under very specific triggers, making them much harder to identify. Training often relies on third-party libraries and pretrained models, which, like other dependencies that fall under Environment, can introduce malicious behavior into the system (ML06: Supply Chain Compromise, GenAI 03: Supply Chain Vulnerabilities, LLM05: Supply Chain Vulnerabilities). Model training is often performed in research environments that are less hardened, meaning these risks can be difficult to detect and may propagate into production.

Model Attacks

In MITRE ATLAS, model-focused threats include evasion, extraction, and inversion attacks, which map to the execution, AI model access, and impact stages of the adversary life cycle.

Our earlier system diagrams highlighted several potential weaknesses and vulnerabilities pertaining to the model. In the architecture diagram in Figure 3-1, the model execution engine is directly downstream from user input, meaning it could potentially execute adversarial inputs if not properly validated (ML02; GenAI 01: Prompt Injection, LLM01: Prompt Injection). There is also a risk of *output leakage*, such as revealing sensitive system prompts or instructions (GenAI 02: Sensitive Information Disclosure; LLM06: Sensitive Information Disclosure).

Figure 3-2, which shows the full development life cycle, highlights threats relevant to the experimentation phase, which is when models are trained and stored. If training data from the data ingestion service isn't verified, an attacker could manipulate it, compromising model integrity from the start (ML01; GenAI 04; LLM03). The jump host, while critical for secure infrastructure access, is also a key target for attackers seeking to modify or extract models in the production environment (ML04: Model Inversion; GenAI 08: Model Theft; LLM10: Model Theft).

Supply Chain and Dependency Compromises

In MITRE ATLAS, supply chain compromise and malicious package threats often occur during the resource development, initial access, and persistence stages of the adversary life cycle.

The system diagrams I showed earlier highlight several potential vulnerabilities related to the files, data, and code that support the AI system throughout its life cycle. In the experimentation diagram, the trained model file (*.h5*) is a critical artifact. If it's not securely stored or validated before deployment, attackers could tamper with it, potentially inserting backdoors, altering model behavior, or stealing intellectual property for their own use (ML06; GenAI 03; LLM05).

Similarly, source code and configuration artifacts passed from development to deployment can be a supply chain attack vector. This can happen in two main ways. First, an attacker could compromise non-custom components like open source libraries or generic configuration files before you use them (ML06; GenAI 03; LLM05). Second, attackers could target your own pipelines if your *continuous integration and continuous delivery (CI/CD)* process—the automated system that builds, tests, and deploys code—isn't locked down, or if you have a public GitHub repository (ML08: Insecure ML System Design; LLM07: Insecure Plugin Integration).

API keys and credentials used in notebooks are another high-risk artifact, especially if they're hardcoded or exposed (GenAI 07: Insecure Plugin, Add-on, or Extension Handling; LLM07). In production, we need tight control over artifacts like authorized keys, third-party libraries, and deployment packages because if attackers replace or manipulate any of these, they can change how the model behaves or gain access to the system. User

prompts, model responses, and embeddings, which the system may store or log, may also leak sensitive user information (GenAI 02; LLM06).

In the experimentation environment, any third-party libraries and dependencies imported into the Jupyter notebook or model training code could include malicious packages or vulnerabilities (ML06; GenAI 03; LLM05). These libraries also fall under the Environment category since they shape the runtime conditions in which the model is built and tested. Similarly, the source code management and model packaging processes represent supply chain entry points. If the deployment code or saved *.h5* model is tampered with before being pushed to production, malicious code or behavior could get embedded into the final system (ML06; GenAI 03; LLM05). Training data itself also forms part of the supply chain; if datasets are poisoned or sourced from untrusted repositories, attackers can compromise a model's integrity before it ever reaches production (ML01; GenAI 04; LLM03).

In the production environment, the jump host and Python web server may pull external dependencies during updates or runtime. Any automatic or unverified updates to these components could allow attackers to compromise the model (ML06; GenAI 03; LLM05). Beyond software, the underlying infrastructure is also part of the supply chain, including cloud compute, hosting, and production software providers. For example, if the developer of the vector store software used in production were breached, and your patching and maintenance process automatically installed a compromised update, the attacker could gain access through a trusted channel (ML06; GenAI 03; LLM05). Further, if a cloud provider like AWS were compromised, or if hardware such as NVIDIA GPUs contained exploitable vulnerabilities, these weaknesses could cascade across every system relying on them.

Supply chain risks like these make it essential for organizations to perform supplier risk assessments and vendor security assessments to evaluate the security posture of third-party providers both before they adopt a technology and on an ongoing basis. If attackers compromise even one part of this chain before delivery, they could gain persistent access without ever interacting with the system directly (ML09: Improper Isolation). *Persistence* refers to an attacker's ability to maintain access to a compromised system over time, allowing threat actors to stay hidden and continue collecting data, issuing commands, or moving laterally across a network long after the initial breach. Lately, researchers have also theorized about the risk of *AI worms*—self-propagating malicious code, inputs, or behaviors that could spread across models or agents. These remain largely theoretical today but highlight the potential for fast-moving and hard-to-contain attacks in future AI environments.

Infrastructure and Runtime Threats

Infrastructure-level threats in MITRE ATLAS include denial of AI service, privilege escalation, and broader infrastructure compromise, which typically occur during the execution, persistence, privilege escalation, and impact stages of the adversary life cycle.

In the experimentation environment, engineers connect to tools like Jupyter notebooks via SSH to develop models, often integrating third-party libraries and external data sources. If we don't isolate and secure this environment, it becomes vulnerable to compromise, potentially through the injection of malicious libraries or unauthorized access to training data and credentials (ML05: Insecure ML Deployment). To enable rapid iteration, experimentation environments often have fewer restrictions and less security monitoring than production environments and are therefore at a high risk of introducing unintended dependencies or insecure defaults that may later propagate into production (ML09).

The production environment exposes the system to external users through an API gateway, creating a live attack surface. Components like the jump host, web server, and model API endpoint all rely on the underlying infrastructure, which includes the operating system, container runtimes, application packages, and orchestration tools to manage the deployment, scaling, and life cycle of containers across multiple machines. Without proper access boundaries or security configurations, these components may be vulnerable to infrastructure-level attacks such as misconfigurations or *privilege escalation*, in which an adversary moves from a lower level of access, like that of a regular user, to a higher level, like administrator or root (ML09; LLM07).

Runtime refers to the period during which a program or system is actively executing. At runtime, the system loads the trained model into the Python web server, which handles user requests and delivers responses. If we don't perform checks on input or output, attackers can pass malicious information to and from the model—for example, through injection attacks or adversarial queries (GenAI 01; LLM01). *Denial-of-service attacks*, in which an adversary overwhelms the API endpoint to make the model unavailable, are also a concern (ML05; LLM04: Model Denial of Service). In agentic settings, an agent may even be manipulated into harmful loops or recursive behaviors if their outputs are fed back into their own inputs without safeguards (GenAI 10: Overreliance; LLM09: Overreliance).

The jump host, which enforces permissions, access controls, and administrator authorization, also faces risks. If compromised, it could allow an attacker to tamper with models, alter behavior, or escalate privileges (ML09). An attacker could also exploit dependencies used at runtime, including third-party libraries, if they're pulled dynamically or not securely managed. This risk is amplified by the fact that the system is both

processing data and acting on it. If its movement isn't properly constrained or its outputs aren't filtered, even a subtle model error can lead to physical harm (GenAI 05: Improper Output Handling; LLM02: Insecure Output Handling).

Output and Application Layer Risks

In MITRE ATLAS, output and application layer risks consist of adversarial queries, which typically arise during the execution, collection, and impact stages of the adversary life cycle.

User prompts, model responses, and embeddings may also leak sensitive user information (GenAI 02; LLM06). If there are no checks on input or output, the attacker could pass malicious information to the model through injection attacks or adversarial queries (GenAI 01; LLM01). There is also a risk of insecure output handling (GenAI 05; LLM02), in which sensitive information such as prompts, embeddings, or responses leaks through logs or downstream integrations. Finally, overreliance (GenAI 10; LLM09) represents a sociotechnical risk whereby humans take model outputs at face value without verification.

Operator and Governance Threats

Risks to operators and governance include social engineering and insider threats, which often occur during the initial access, credential access, and impact stages of the adversary life cycle.

Both of our system diagrams highlight the roles played by human operators such as engineers, administrators, and users. In the experimentation environment, the engineer develops and trains models in a Jupyter notebook. If their credentials or host machines are compromised, or if they inadvertently introduce insecure code or dependencies, they could unintentionally inject vulnerabilities into the model or system (ML10: Insider Threat).

In the production environment, the administrator connects through the jump host to manage deployments, update authorized SSH keys, and ensure that the model is properly loaded and secured. However, because this access grants high privileges, it represents a significant target for attackers. If the administrator's access is misused or not tightly controlled, it could allow unauthorized updates or access to sensitive data (GenAI 09: Inadequate Monitoring & Logging). Furthermore, operators managing API keys, deployment scripts, or external library imports play a critical role in securing the system's trust chain (ML08; LLM08: Excessive Agency).

NOTE *When OWASP released its first top 10 list in 2003, the leading threat was an injection attack called a SQL injection. Two decades later, this attack still ranks among the most serious risks. The lesson for AI security practitioners is that, even though AI technologies evolve incredibly quickly, the threats we view as critical today will likely remain relevant well into the future.*

Following the threat modeling process described in this section should leave you with an understanding of how different components within the AI system contribute to the attack surface. As the next step, the organization should prioritize these risks based on their likelihood and potential impact. Note that the mapping covered here isn't exhaustive, and I encourage you to check out the OWASP and ATLAS frameworks on your own.

If you employ multi-agent systems, your threat modeling might best be served by the Cloud Security Alliance's MAESTRO framework. Let's run through an example to demonstrate how it works.

Example 2: MAESTRO

MAESTRO aims to address the unique risks of *multi-agent systems*: those that involve multiple autonomous agents communicating, collaborating, or competing to achieve goals. The framework is widely used in industry and government circles, including by OWASP and the UK AI Security Institute. Short for "Multi-Agent Environment, Security, Threat, Risk, and Outcome," MAESTRO breaks systems down into seven layers:

Foundation model Represents the pretrained models and LLMs that provide the core reasoning and capabilities of agentic systems. The integrity and design of these models underpin everything else in the system.

Data operations Includes how data is processed, stored, and retrieved. This layer covers vector databases, embeddings, and prompt management, representing the knowledge an agent has access to at any given time.

Agent frameworks Forms the execution logic and workflow structures that give agents autonomy. This layer is like the operating system of agents, defining how agents make decisions, interact with tools, and follow (or break from) designed workflows.

Deployment infrastructure Refers to the runtime environments, containerization, orchestration, and networking that allow agents to operate at scale. These supporting environments determine how secure, resilient, and scalable agentic systems are.

Evaluation and observability Covers monitoring, logging, and oversight mechanisms, including human-in-the-loop systems. Observability governs how transparent and accountable the system's actions are and whether issues can be detected or corrected.

Security and compliance Ensures that policies and regulatory requirements are consistently enforced. A vertical layer that spans all the others, this layer exists because compliance and governance considerations cut across technical boundaries rather than being confined to a single component.

Agent ecosystem Represents the broader environment of interactions: how agents connect with humans, other agents, and external systems. No agentic system exists in isolation; its value and its attack surface emerge from the ecosystem it sits within. This final cross-layer view aims to capture emergent behaviors and risks that arise when multiple layers interact, since agentic systems may be at risk of strange behaviors that can't be fully understood by analyzing one layer at a time.

MAESTRO emphasizes agentic factors like nondeterminism, autonomy, identity management, and inter-agent communication. These factors are central to why multi-agent systems behave differently from traditional software and why their risks are harder to anticipate. To illustrate this, let's walk through an example. We'll examine a system at each layer, identify its weaknesses and vulnerabilities, and evaluate the most likely attack vectors for each.

In OWASP's Multi-Agentic System Threat Modelling Guide, researchers applied the MAESTRO framework to Anthropic's Model Context Protocol, discussed in Chapter 2. They mapped MCP's architecture across the seven layers: foundation models (which MCP enables but doesn't define), data operations (the resources retrieved through MCP servers), agent frameworks (MCP itself, including how it defines tools, resources, and prompts), deployment infrastructure (the server environments and communication protocols), evaluation and observability (logging and monitoring), security and compliance (permissions, access control, and governance), and agent ecosystem (its role as a standardized integration point across diverse models and platforms). You can see their threat summary in Figure 3-3.

Note that the OWASP researchers use the common practice I mentioned earlier of including threats in the diagram, represented by threat ID. They determined the top threats by applying MAESTRO scenarios and the OWASP Agentic Security Initiative taxonomy. Unlike the previous example, these threat IDs don't map directly onto the OWASP top 10 lists. Instead, they are consolidated in the Guide, on page 58.

At the model and data layers, concerns included cascading hallucinations (denoted by threat ID T5), model instability producing inconsistent MCP requests (T26), and manipulated retrievals from connected data sources (T17, T18). At the agent framework layer, which covers MCP itself, the risks included tool misuse (T2), insecure client-server communication (T30), and client impersonation (T40), as well as interference through shared servers (T42). Deployment threats included exposed account information (T21) and improperly secured MCP servers (T43). The researchers also highlighted observability gaps such as insufficient or manipulated logging (T44) alongside ecosystem-level issues like rogue MCP servers impersonating legitimate ones (T47).

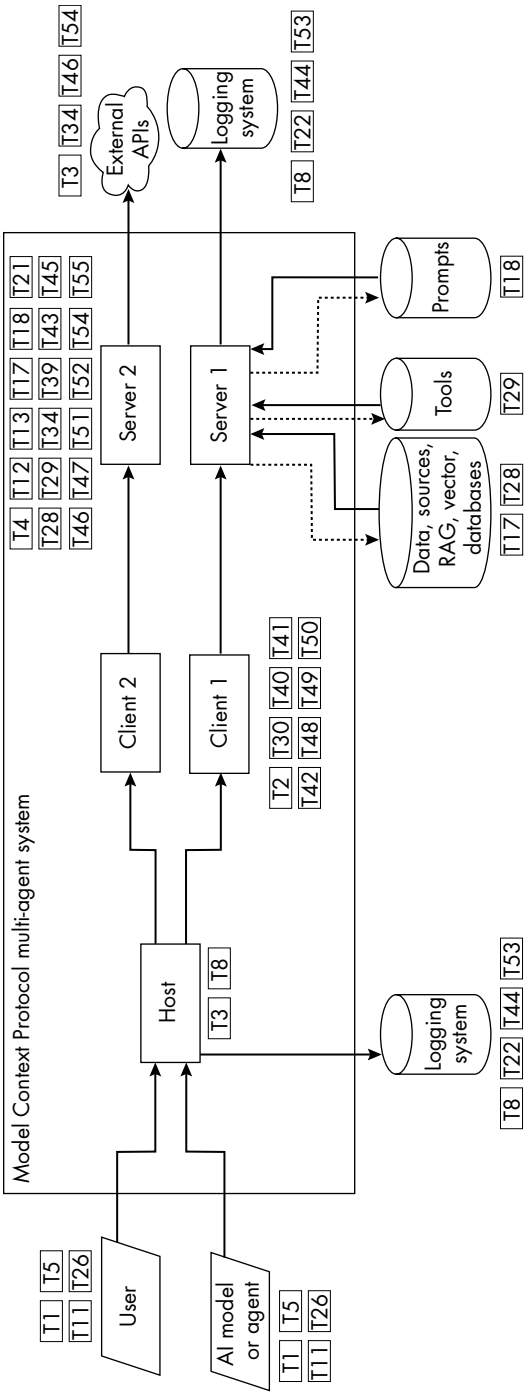

Figure 3-3: The MCP threat model, adapted from OWASP's Multi-Agentic System Threat Modelling Guide

The OWASP report prioritized these threats because they affect MCP's most sensitive functions—connecting models, retrieving data, and invoking tools—and because many were cross-layer in nature, showing how a flaw in one part of the protocol could cascade into systemic compromise. You might want to try and guess what the other threat IDs correspond to. (Or you could read the report and find out; I won't judge if you choose this option!) Regardless, I recommend you check out the report for a full list of potential threats and other example case studies.

MAESTRO doesn't need to be a stand-alone taxonomy; you can use it as a methodology that complements others. We'll continue to revisit threat modeling as we cover the full range of adversarial attacks in later chapters.

Threat Actors

When we think of cyber attackers, it's easy to imagine faceless, abstract entities, but in reality, we're defending our systems against real people (and sometimes AI directed by people). These individuals have a diverse range of motives and skills, from hobbyists with limited resources to highly sophisticated and well-funded criminal or government organizations. The same actors who have been attempting to compromise computer systems for decades are now focusing their efforts on AI systems.

Taxonomies

In 2024, the RAND Corporation published its "Securing AI Model Weights" report, which is one of the most useful resources I've seen on threat actors and security levels. The report classifies threat actors into five groups.

The least capable level, *amateur attempts*, comprises operations roughly comparable to a single individual with some professional expertise in information security spending several days on the operation, with a total budget of up to $1,000 and no preexisting infrastructure or access to the organization. This category includes the realm of hobbyist hackers.

The second level, *professional opportunistic efforts*, comprises individual professional hackers and capable hacker groups executing untargeted or lower-priority attacks. These operations are roughly comparable to a single individual broadly capable in information security who spends several weeks with a total budget of up to $10,000 on the operation. They may have pre-existing cyber infrastructure but no preexisting access to the organization.

The third level, *cybercrime syndicates and insider threats*, includes the operations of many world-renowned criminal hacker groups, well-resourced terrorist organizations, disgruntled employees, and industrial espionage organizations. While individual insiders may not control large budgets or external infrastructure, their privileged access and contextual knowledge can yield an impact equivalent to a coordinated external team. As a whole, this tier reflects adversaries capable of operations roughly comparable to 10 experienced information security professionals working over several months, with access to preexisting cyberattack infrastructure and resources

totaling up to $1 million. Most external groups at this level don't already have trusted access to the target organization, but insider threats represent a unique subcategory for which such access is inherent or easily obtained.

The fourth group, *standard operations by leading cyber-capable institutions*, includes the operations of many of the world's leading state-sponsored groups and foreign intelligence agencies across the world. These operations are roughly less capable than or comparable to 100 individuals who have experience in a variety of relevant professions (such as cybersecurity, human intelligence gathering, and physical operations) spending a year on the operation with a total budget of up to $10 million, vast infrastructure, and access to state resources such as legal cover, the interception of communication infrastructure, and more.

The final and most capable group is *top-priority operations by the top cyber-capable institutions*, which includes the handful of operations most prioritized by the world's most capable nation-states. These operations are roughly less capable than or comparable to 1,000 individuals who have experience and expertise years ahead of the public state of the art in a variety of relevant professions, spending years on the specific operation, with a total budget of up to $1 billion, state-level infrastructure, and access developed over decades. They also have access to state resources such as legal cover and the interception of communication infrastructure.

I recommend reading the full report (included in the list of resources at the end of this chapter) for more detail on the capability of these different groups and for interesting examples and case studies. These taxonomies can be extremely useful in defining organization-level threat mapping exercises, but I'll keep it high level here and describe two of the most impactful groups: criminal enterprises and advanced persistent threats.

Criminal Enterprises

Cybercriminal enterprises function similarly to legitimate enterprises in their size, organization, and operations, except that their income comes from exploiting cyberspace and the digital world (and its participants—all of us). These groups are highly structured and may have a surprising level of professionalism; many employ customer support agents to help victims through the process of acquiring cryptocurrency to pay a ransomware bounty or to assist newcomers in choosing products and making payments on their dark web marketplaces.

These organizations also hire cyber capabilities: malware developers who create malicious software, hackers who compromise systems, social engineers who deceive targets, and money launderers who obscure financial transactions. You'll find these groups behind many phishing attempts, credit card and financial fraud, identity theft, *botnet* operations (networks of compromised computers and even IoT devices used to launch further attacks), and stolen data sold on the dark web.

Examples of cybercriminal enterprises include REvil, which held extensive ransomware campaigns with multimillion-dollar ransoms, and Conti, which systematically targeted hospitals, healthcare organizations, and

public infrastructure under the cruel assumption that these organizations would be most likely to agree to their demands. Both groups have operated as professional criminal businesses, employing dedicated staff, running sophisticated marketing and communications operations, and even using affiliate models (allowing smaller criminal actors to use their ransomware for a percentage of the profit). Not all cybercriminal enterprises are alike, however. Many are smaller, less reliable, and less skilled, and like any organization, their biggest vulnerabilities often come down to interpersonal conflicts.

Advanced Persistent Threats

Advanced persistent threats (APTs) refer to highly skilled actors, often representing or backed by nation-states, who conduct sophisticated cyber operations. Their campaigns tend to be long-term, covert, and targeted attempts to collect intelligence or sabotage the operations of other governments or organizations. They have access to significant resources and technical expertise, and they treat these operations like jobs, often working during business hours for several years.

Examples include China's APT19 (Deep Panda), which was associated with the 2015 cyberattack on the US Office of Personnel Management, leading to the breach of security clearance information, fingerprints, and background checks of over 20 million government employees. Russia's APT28 (Fancy Bear) interfered in the 2016 US presidential elections by breaching the Democratic National Committee and has targeted NATO, journalists, and various Western governments. Another example is North Korea's Lazarus Group, responsible for major cyber heists such as the 2016 Bangladesh Bank robbery and the 2014 Sony Pictures hack. Some nations, most notably Russia and North Korea, use cybercrime as a significant source of revenue to circumvent international economic sanctions. The APT groups receive their catchy names from cybersecurity firms based on indicators like geographic origin (hence the Panda and Bear associations) and technical signatures that point to the attack authors.

Discussing APTs can become complicated because Western nations also engage in cyber operations against other countries. For example, Stuxnet was an extraordinarily sophisticated computer worm, discovered in 2010 and widely attributed to the United States and Israel, that used multiple zero-day vulnerabilities to infiltrate and sabotage Iran's nuclear centrifuges. We'll explore cyberattacks between nation-states more thoroughly when we discuss governance, ethics, and international norms in Chapter 9.

Threats to AI

The introduction of AI to information technologies is enticing to the attacker categories we've discussed. Dark web analysis reveals that criminal enterprises are offering AI-powered phishing and deepfake-as-a-service for as little as $400 per month. Just as legitimate organizations are experimenting with AI to both augment existing capabilities and deliver entirely new ones, so too are threat actors, and we should expect to see many novel challenges in this space.

While cybercriminals and nation-state groups remain key players, AI's versatility and accessibility mean that a much broader range of threat actors, including insiders, hacktivists, hobbyists, nation-state actors, and competitors, are likely to use or attack AI systems in different ways, depending on their motives. In short, these categories are not exhaustive; the threat landscape for AI is likely to expand rapidly as new use cases and vulnerabilities emerge.

AI Threat Intelligence

Cyber threat intelligence (CTI) is detailed information about cybersecurity vulnerabilities, threat actors, and their attack campaigns that researchers collect, analyze, and disseminate. Many organizations provide CTI, including dedicated cybersecurity companies like CrowdStrike and Mandiant, government agencies responsible for cyber defense and reporting, and research groups like MITRE Corporation.

Threat intelligence comes from a vast network of sources: the analysis of malware samples, the infiltration of dark web forums, the use of data generated by security products, and intelligence shared between groups. For example, government agencies like the Cybersecurity and Infrastructure Security Agency (CISA) in the United States, the Australian Cyber Security Centre (ACSC), the National Cyber Security Centre (NCSC) in the United Kingdom, and the Cyber Security Agency of Singapore (CSA) are responsible for tracking incidents and sharing knowledge within their country and with others.

THE ZERO-DAY MARKET

Of particular interest is the ability to track and mitigate *zero days*, a term that refers to security flaws discovered before the developer becomes aware of them and has the chance to release fixes. The market for zero days is massive and controversial; they're actively traded by cybersecurity companies, governments, intelligence agencies, cybercriminals, and brokers. Private companies like Zerodium and Exodus Intelligence purchase zero days and resell them for up to millions of dollars. The highest publicly reported zero-day bounty is $20 million, offered by a Russian exploit broker for a successful attack on iOS and Android. Zerodium claims it received over 15,000 submissions in its first eight years.

The marketplace raises several ethical questions regarding who is allowed to purchase and use zero days and for what purpose. In some countries, such as China and Russia, laws require that hackers disclose newly discovered zero days to the government before notifying affected companies, allowing the state to potentially weaponize them for offensive cyber operations. In the United States, while there is no public law mandating government disclosure for private researchers, government agencies follow the Vulnerabilities Equities Process (VEP) to decide whether a zero day should be disclosed or retained for intelligence use.

AI threat intelligence specifically focuses on threats targeting AI systems and machine learning models and how AI is being weaponized in the wild. Several organizations currently disseminate this information, including traditional CTI reporters like Microsoft and MITRE, as well as AI security companies like HiddenLayer, Robust Intelligence, Lakera, and my own company, Mileva. We can find a few common themes in AI threat intelligence:

- AI products often contain many traditional software vulnerabilities, may have poor cyber hygiene, or may use outdated tools.
- Criminals are starting to use AI to enhance their existing criminal activities. For example, they're using generative AI to craft more realistic phishing emails (an application of "AI for security" rather than "the security of AI").
- AI-specific vulnerabilities are still mostly in the research or testing phase, but interest in them is rapidly growing, with many techniques now being presented at conferences like DEF CON. These include attacks that target the underlying decision threshold in machine learning models or components like MCP and RAG.
- We've seen some real-world exploitation of AI in specific domains like computer vision and LLMs. For example, people are using prompt injection tactics, discussed in Chapter 4, to bypass the guardrails of language models to receive output that goes against the safety mechanisms of the model developer.
- Threat actors show interest in the proprietary models underlying many AI systems. For example, there have been reported attempts to create copies of large proprietary models to replicate their functionality for economic gain or intelligence gathering.
- Protocols, infrastructure, and orchestration tools like MCP are attractive targets because they sit at the intersection of users, models, and data sources. Every week, we see at least half a dozen new MCP vulnerabilities being reported.

Let's go through some of these themes in more detail.

Traditional Software Vulnerabilities

Many security flaws typical in traditional software are also prevalent in the software of AI systems. These vulnerabilities aren't specific to the way AI fundamentally works but arise from standard coding and development errors and can have significant consequences if they aren't quickly patched. A *software patch* refers to a small update or piece of code to address security vulnerabilities or code errors, referred to as *bugs*.

CODE BUGS

The term "bug" was introduced by Rear Admiral Grace Hopper, an incredible engineer in the US Navy in the 1940s, when computers were mechanical. A computer was malfunctioning because a moth was caught in the relay, and so computer errors came to be known as bugs! Shown here is the original "bug" in Grace Hopper's notebook.

The Common Vulnerabilities and Exposures (CVE) system logs known cybersecurity vulnerabilities in a database maintained by MITRE. Each vulnerability is assigned a unique identifier and a description. Security professionals around the world use the CVE system to share, track, and respond to vulnerabilities.

In January 2024, the Attacker Knowledge Base platform provided by the cybersecurity company Rapid7 reported that at least 186 vulnerabilities discovered in 2022 had been exploited in the wild, with 47 rated as highly valuable to attackers. We've seen certain CVEs exploited several years after the release of a patch because the targeted organization never got around to updating its systems. The widely publicized SamSam ransomware campaign, so-called because of the string "SamSam" featured in the code, earned attackers $6 million in 2018 by exploiting vulnerabilities whose patches had been released more than five years earlier.

The exact number of CVEs identified in AI products is hard to calculate, but every two weeks my team publishes its AI threat intelligence report, which always includes at least a dozen AI-related CVEs. This number is growing steadily as these technologies become more prevalent and attract greater scrutiny from security researchers.

One notable example specific to machine learning is CVE-2024-3568, a serious vulnerability discovered in the widely used Hugging Face Transformers library (introduced in Chapters 1 and 2). The vulnerability involves the unsafe use of Python's `pickle` module for *serialization* and *deserialization*, the process by which developers save and then reload models.

The vulnerability allows attackers to execute malicious code undetected. For example, they could embed malware in pickle files and run them automatically when someone loads the model. While tools can scan models for pickle files, attackers regularly find ways to evade these scanners. Suffice it to say that if you're a data scientist, you shouldn't be using pickle files (or if you do, do so at your own risk).

Despite CVEs being reported in AI systems, there is currently limited public evidence that attackers are exploiting these vulnerabilities. At this stage, it's hard to determine whether this is because attackers are genuinely not targeting these vulnerabilities or because reliable detection methods are lacking. This area of research and practice is expected to grow as we develop a better understanding of how seriously these vulnerabilities should be regarded.

AI-Enabled Cyber Crime

There is evidence that threat actors are investing in AI to augment and enhance their existing cyber operations and fraudulent activities. For example, generative models can craft more convincing phishing emails or generate realistic fake documents and deepfakes. Attackers can also use language models to code malware and ransomware more efficiently, increasing the attackers' speed and scale.

A notable real-world example occurred in 2020, when criminals used deepfake voice technology to impersonate the chief technology officer of an energy company in the United Kingdom. The attackers replicated the executive's voice and held a phone call with an employee, urgently instructing them to transfer funds to a fraudulent account. The employee, convinced they were speaking with their boss, authorized the transfer of approximately £220,000.

Such practices are widespread. According to its 2024 Annual Threat Report, the security company Darktrace detected over 30.4 million phishing emails across its customer base between December 2023 and December 2024. Of these, 32 percent employed novel techniques, including AI-generated text with increased linguistic complexity, longer sentences, and varied punctuation, to evade traditional detection methods. There are also reports indicating the rise of AI-driven products offered on dark web marketplaces, like deepfake-as-a-service platforms and fraud bots.

While these high-profile cases demonstrate the potential impact of AI-enabled attacks, publicly documented incidents remain relatively uncommon. This may be because victims are reluctant to publicize such incidents due to reputational concerns or the difficulty in attributing an attack's success to the use of AI. Cybersecurity companies and researchers rely heavily on mandatory reporting requirements, victim organization accounts, and

media coverage for CTI. However, since no mandatory reporting requirements currently exist specifically for AI attacks, data remains sparse.

AI Weaknesses in Research

In academic and controlled testing environments, researchers frequently demonstrate adversarial machine learning techniques, which exploit inherent weaknesses in machine learning models to disrupt, deceive, or extract information from them. (We discuss these techniques in detail in Chapter 4.) Each week, the research community publishes several new studies on methods of exploiting AI models. On the open-access repository arXiv, maintained by Cornell University, more than 11,000 academic papers on adversarial machine learning have been published since 2021. In addition, independent and industry research is widely available on blogs and social media platforms.

In one case, researchers from Palo Alto Networks tested their deep learning detector for malware *command and control (C2)*, the communication methods used to remotely manage infected computers, in HTTP traffic. After training the detector on data similar to that used in the live production model, they found they could evade detection by strategically removing certain typically unnecessary HTTP header fields. These headers define parameters or behaviors such as how web browsers or servers cache data (`cache-control`) or whether the network connection stays open or closes after completing a request (`connection`). This approach effectively tricked the model, allowing malware communications to pass undetected.

Microsoft's AI Red Team has documented similar methods during internal exercises and published them in MITRE's list of case studies. In one exercise, the team combined traditional cybersecurity techniques with adversarial machine learning approaches, including both online and offline adversarial examples, to successfully disrupt an internal service on its AI-enabled, cloud-based infrastructure platform, Azure. Another exercise targeted a new Microsoft product, where automated adversarial techniques iteratively manipulated target images until the machine learning model consistently produced incorrect classifications, highlighting the potential for exploitation in real-world deployments.

Some companies, such as Microsoft, are taking significant initiative by investing in internal teams dedicated to doing this type of testing. Not all organizations are able to make this investment, however.

AI Weaknesses in the Real World

While most AI-specific weaknesses have traditionally remained within academic research or internal security assessments, we're starting to see evidence of their exploitation in the wild.

In 2023, a *Vice* journalist successfully bypassed Lloyds Bank's Voice ID security system using AI-generated voice audio created with software from the AI company ElevenLabs, allowing him to access his own account. In 2020, attackers bypassed AI-powered infrared cameras installed in South Africa's Hluhluwe-iMfolozi Park, which were designed to detect and prevent

poaching, resulting in the deaths of four rhinos. In April 2023, a researcher reported a suspected privacy exploit in ChatGPT-4 after receiving unsolicited responses from empty prompts, initially suggesting possible leaks of user data. And in December 2023, an attack exploited a Chevrolet dealership's chatbot, developed by digital marketing company Fullpath, to sell a Chevrolet Tahoe for only $1. Many businesses often feel pressure to rapidly incorporate AI into their operations, products, and services, which can lead to cutting corners and lowering security standards to accelerate adoption.

While documented real-world AI exploits remain comparatively rare, their growing frequency and impact highlight a shift from academic concerns to genuine security threats. We'll cover many more adversarial machine learning case studies in Chapter 5.

APT Interest in Intellectual Property

APT groups and organized criminal enterprises have shown growing interest in acquiring intellectual property related to AI technologies. They can obtain this information through both conventional cyberattacks and AI-specific methods.

Microsoft's security research has highlighted activities by Forest Blizzard, a Russian-linked APT group that specifically targeted AI research infrastructures, likely attempting to access or steal proprietary AI models and training datasets. A recent prominent case involving potential APT activity was the theft of AI-related intellectual property by the Chinese AI company and model DeepSeek. Some claim the techniques used to train the DeepSeek R2 model may have been stolen from OpenAI. While the claim hasn't been validated, the theft of a model of this scale would represent millions of dollars of intellectual property, considering the cost of model development alone.

Actors are incentivized to target AI intellectual property primarily because of the substantial investment required to independently develop advanced AI models: the large datasets, expensive computational resources, and highly specialized expertise. Stealing proprietary AI models or datasets enables attackers to bypass these significant costs.

Beyond cost savings, there is also a strategic incentive in weakening major companies in rival economies. For example, when DeepSeek R2 was released in 2025, it caused significant market disruption in the United States. Claims that it was developed at a fraction of the cost of comparable US models led to fears that Chinese firms could undercut American AI companies, capture market share, and reduce global dependence on US technology. This resulted in sharp declines in the stock prices of US AI companies, demonstrating the immediate economic consequences of this news.

MOTIVATIONS, TTPS, IOCS, AND MITIGATIONS

Threat intelligence typically includes details about attacker motivations, TTPs, indicators of compromise (IOCs), and recommended mitigations.

Attacker *motivations* describe the underlying reasons driving individuals or groups to carry out cyberattacks, such as financial gain, espionage, activism, or notoriety.

TTPs describe how attackers operate: their methods, strategies, and specific actions. *Tactics* represent the attacker's overarching objectives, like gaining access to a network or *exfiltrating* (leaking) sensitive data. *Techniques* refer to the specific methods attackers use to accomplish these objectives—things like phishing or malware deployment. *Procedures* comprise the step-by-step descriptions of precisely how an attacker executes these techniques, down to the level of the specific malware tools they use or the commands (the code) they run.

IOCs are specific indicators that suggest a cyber incident has occurred. While an attack may be apparent if you discover an attacker trying to sell your data on the dark web, in other cases you may be unaware of a compromise—for instance, if the attacker's goal is to maintain persistent, ongoing access to your network.

Mitigations are those actions and strategies designed to mitigate cybersecurity risks. All of this is very useful information and informs your threat model.

AI threat intelligence is rapidly evolving, and reports are generally extremely time-sensitive, as attackers are known to exploit vulnerabilities within minutes. Fortunately, there are ways to use the information from threat intelligence reports to strengthen defenses.

AI Security vs. Traditional Cybersecurity

Now that we've reviewed the landscape of AI security, let's consider how it fits within traditional cybersecurity. Many view AI security as a subset of cybersecurity, reasoning that because AI models operate within broader cyber systems, existing cybersecurity defenses should adequately address most AI threats. Others argue that AI security represents a distinct and emerging threat, different enough from traditional cybersecurity to warrant separate consideration. They emphasize inherent differences unique to AI, such as its probabilistic nature, "black box" complexity, and increasing autonomy and agency.

With our current technology, I see AI security and cybersecurity as overlapping fields, much like a Venn diagram. The integration of AI models within larger digital systems means that many existing cybersecurity practices still help mitigate certain AI-specific risks. But given the rapid growth of AI technologies, the frequent discovery of unique vulnerabilities, and our increasing reliance on these systems, it is practical to distinguish AI security from traditional cybersecurity to ensure that AI-specific risks receive appropriate attention and tailored defenses.

Conceptual Models

The CIA triad, which stands for confidentiality, integrity, and availability, is a cybersecurity model that guides the protection and security of information. *Confidentiality* ensures that sensitive information is accessible only to authorized individuals. *Integrity* protects information from unauthorized modification or deletion to maintain its accuracy and reliability. *Availability* ensures that systems and data are accessible when needed. In practice, security professionals assess which aspects of the triad—confidentiality, integrity, or availability—require the strongest protection based on their threat profile. For instance, financial institutions prioritize confidentiality and integrity to safeguard transactions and sensitive client data, while healthcare organizations often emphasize availability to ensure that critical medical systems remain accessible when required.

The CIA triad is protection-centric, focusing on the *controls* that maintain each pillar. Controls are specific measures implemented to prevent, detect, or respond to threats, such as firewalls, encryption, or access logging. You will often hear the term used interchangeably with *mitigations*, though this is not technically correct. Mitigations refer to broader strategies used to reduce risk, which may include implementing controls as well as process changes, training programs, or architecture decisions. Controls are best understood as the specific mechanisms, while mitigations represent the overall risk-reduction approach.

When considering AI security, you might find it useful to employ the 3 D's model shown in Figure 3-4. This simple framework effectively captures the range of possible AI security attacks and corresponding defenses.

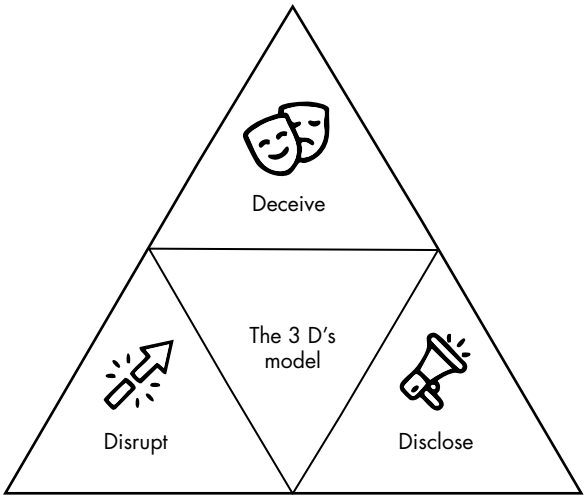

Figure 3-4: The 3 D's model

In this model, attacks fall into three primary categories: *disruption* attacks, which cause an AI system or model to fail or not function as intended; *deception* attacks, which deliberately lead an AI system to produce incorrect, premeditated classifications or decisions; and *disclosure* attacks, which result in the leakage of sensitive information about the AI model, its training data, or user inputs. This model is threat-centric, meaning it describes possible threats and their implications.

Each of the three D's aligns with the pillars of the CIA triad. Disruption attacks require controls that ensure availability, deception attacks correspond to breaches in integrity, and disclosure attacks relate to confidentiality issues. In practice, you could use both models in tandem to describe AI attacks and how you would mitigate their risks.

Other, more granular frameworks recommend specific controls and mitigations for AI threats. These include frameworks developed by the US National Institute of Standards and Technology (NIST), the International Organization for Standardization (ISO), MITRE, RAND, and companies like Microsoft and Meta. I'll discuss them in more detail in Chapter 9, which covers governance.

Vulnerability Scoring

In cybersecurity, we often assess the potential severity of a vulnerability using existing frameworks such as the Common Vulnerability Scoring System (CVSS). This framework assigns vulnerabilities a numerical score ranging from 0 (low severity) to 10 (critical) and helps organizations prioritize resources to address vulnerabilities that pose the greatest risk. It's widely integrated into the industry-standard National Vulnerability Database. Over the past decade, nearly 30,000 vulnerabilities with CVSS scores of 9 or higher have been reported.

We calculate this score using several factors, a few of which I'll highlight here. *Exploit maturity* refers to how well developed and accessible an attack method is. A highly mature exploit has been refined, automated, and potentially incorporated into widely available attack tools, making it easier for even low-skilled attackers to use. *Transferability* describes whether a threat actor can successfully apply an attack across multiple targets. A highly transferable attack is more dangerous because it can be reused across different systems without needing significant modification. *Privileges* refer to the level of system access required for an attack to succeed. Low-privilege attacks can be executed by an external user with no special permissions, while high-privilege attacks require elevated access, such as administrative control. *Complexity* or *difficulty* refer to how challenging an attack is to execute, considering factors such as technical skill, required resources, and system defenses.

While we want to align AI vulnerabilities with existing frameworks as much as possible, you can see that it's less clear how some of these factors apply to AI systems. For example, AI attacks are often *nondeterministic*, meaning that even a well-crafted exploit may succeed only some of the time, depending on factors like randomness in the model's responses, training data nuances, or environmental context. The concept of exploit maturity is also difficult to capture because AI-specific exploits are typically research-driven, may lack standardized methods or tools, and change rapidly. Determining transferability in AI is also complicated because attacks are often context-specific and can vary widely in effectiveness between different models; even subtle differences may limit how broadly an exploit can be applied. Additionally, AI models don't have conventional privilege levels—many attacks might simply require public or API access, which don't align neatly with traditional privilege frameworks. In AI scenarios, complexity also becomes nuanced because adversarial attacks require specific technical knowledge of machine learning and model behavior, along with significant computational resources. Measuring complexity consistently across different AI models, datasets, or environments can be ambiguous and challenging.

In addition, current models don't capture other relevant categories at all. The terms *human in the loop* and *human on the loop* refer to the degree of human involvement in AI-driven decision-making processes. Having a human *in* the loop means a person must actively participate in decision-making; for example, a fraud detection system may flag transactions but

require human approval before blocking them. A human *on* the loop indicates that the AI operates autonomously while a human monitors the system and may intervene when necessary, such as in the case of an autonomous vehicle that allows a human operator to take control as needed. These distinctions are critical when classifying the severity of an AI weakness, since the presence or absence of human oversight directly affects the potential exploitability of a system failure.

Another concept, *noisiness*, refers to how many attempts an attack requires before achieving success. This is related to a metric called the *attack success rate (ASR)*, which measures the proportion of attempts that succeed. A noisy attack requires many failed attempts to succeed; for instance, you may have to send many prompts to a chatbot before getting the information you were after. A low-noise attack succeeds in a small number of tries or even on the first try, such as bypassing a facial recognition system with an adversarial image. Noisiness is important in vulnerability classification, as lower-noise attacks are typically harder to detect and respond to, thereby increasing their severity and real-world risk.

AI systems are becoming increasingly autonomous and agentic. While traditional cyber systems generally operate according to predetermined logic and explicit rules set by humans, autonomous systems may act independently. They also challenge traditional security thinking because of how trust and control must be managed. It makes the idea of a *trust radius*—the scope of resources or actions an agent is allowed to access—essential and, in some cases, adjustable through a dial for trust that lets operators set how much autonomy to grant. Because agents often need access to sensitive data or systems, they may operate in *privileged sandboxes*, which are contained environments where elevated permissions are allowed but restricted. If these sandboxes are poorly designed or misconfigured, an attacker who compromises the agent could break out of the sandbox and gain broader access to critical systems. The absence of a human in the loop means attackers may bypass systems using even simple techniques. Lastly, defenders must not only consider what tasks an agent is expected to perform but also consider the alignment and enforce behavioral boundaries to mitigate the risks of misaligned agents from pursuing unintended or unsafe actions.

These kinds of challenges have led to the development of the AI Vulnerability Scoring System (AIVSS), an OWASP initiative launched in 2025 to provide a structured and quantifiable way to assess vulnerabilities in AI systems. We'll discuss this in more detail in Chapter 9.

Uninterpretability

Traditional non-AI systems typically operate with deterministic, rule-based logic, making their behavior easier to interpret, test, and audit. In contrast, AI systems, particularly those built using deep learning techniques like neural networks, are often characterized as black box models because it's difficult or impossible for humans to intuitively understand how or why they arrive at a particular decision or prediction.

The decision pathways within machine learning models involve thousands or millions of individual calculations that influence the final output, and the distributed representation of knowledge across all the parameters makes it challenging to pinpoint exactly why certain inputs cause specific outputs. This opacity presents challenges with validating decisions, identifying biases, debugging unexpected behaviors, and ultimately establishing trust, especially in high-stakes applications.

To illustrate this point, I saved and downloaded the wine analysis model from Chapter 1, using the following code:

```
model.save('wine_model.h5')
```

I then uploaded it to an online neural network visualization tool called Netron, which produced the visualization shown in Figure 3-5.

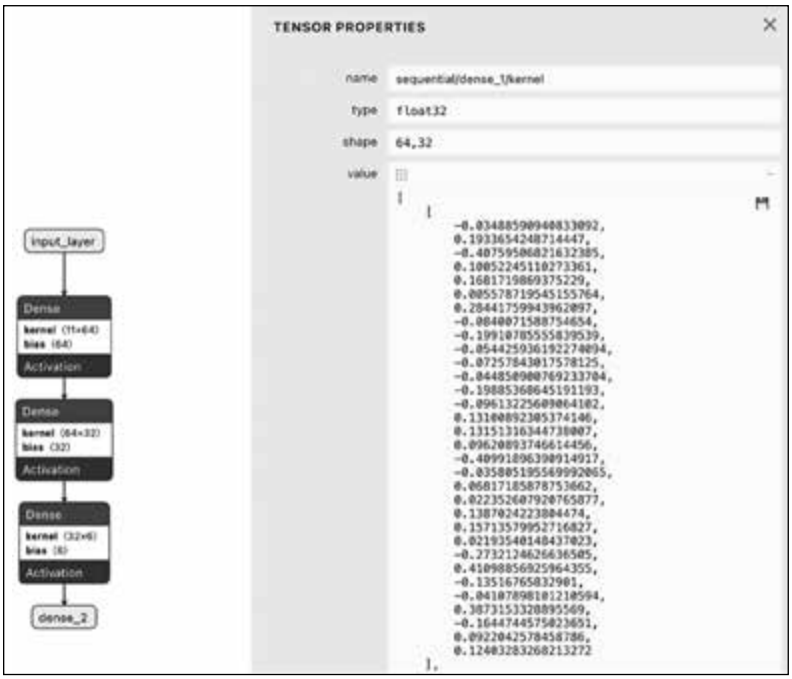

Figure 3-5: Our wine analysis model visualized in Netron

On the left, each block represents one of the layers in the neural network, and the arrows indicate data flow from one layer to another. I've currently selected the weights (kernel) for the second dense layer, which now populate the panel on the right. The type is `float32`, meaning these values are 32-bit floating-point numbers commonly used in neural networks. The shape, in this case `(64,32)`, indicates the dimensions of the weight matrix and means this layer has 64 input neurons and 32 output neurons, resulting in a total of 64 × 32, or 2,048, weight values. The large array of numeric values labeled `Value` lists these specific weights. Each individual value represents the strength of connections between pairs of neurons

from the first Dense layer (64 neurons) to neurons in this second Dense layer (32 neurons).

Each numeric weight contributes directly to how the model makes predictions, but individually, they're not easily interpretable. If I gave you this list and asked you to tell me why our wine analysis model recommended I try a red wine, you wouldn't be able to tell me. This lack of interpretability highlights a broader challenge in AI security: When we can't easily understand how models make decisions, it becomes harder to assess their trustworthiness, detect tampering, or ensure that they behave as intended.

There are now many approaches we can use to make our models more interpretable. These approaches fall into several categories. One common method involves choosing inherently interpretable models from the outset, such as decision trees or linear models, which clearly illustrate how input features affect predictions. However, these models may not be as accurate or perform as well as other model types. Another strategy is *mechanistic interpretability*, where researchers explore neuron activations and interactions within neural networks to uncover exactly how models reach decisions. We'll discuss this practice in more detail in Chapter 8.

Unpatchability

Another critical difference between traditional cyber systems and AI systems lies in their relative patchability—or, in the case of AI, unpatchability. Vendors typically address traditional cybersecurity issues by releasing software patches. But AI-specific weaknesses often relate to the underlying data, model architecture, and probabilistic way models learn and make decisions. Simply adjusting a single component or algorithm may not adequately address the vulnerability. To mitigate these AI-specific threats, teams usually need to retrain the entire model, incorporate additional defenses, or deploy monitoring and detection systems.

Nondeterminism

Unlike conventional software systems, where the same input consistently produces the exact same output, machine learning models and AI systems can produce slightly different outputs each time they run, even given identical inputs. This behavior stems from the stochastic nature of many AI training processes, which are characterized by randomness and uncertainty. This nondeterminism complicates the tasks of validating, debugging, and securing AI systems, as their unpredictable outcomes make it challenging to definitively verify system behavior or precisely replicate scenarios when investigating incidents.

Scale

Another significant difference lies in their scale, in terms of both model complexity and their widespread deployment across enterprise systems. Many machine learning models comprise millions or billions of parameters, which amplifies all the other challenges around interpretability and

risk management. AI models are also increasingly integrated into enterprise processes, which dramatically increases the potential attack surface. Unlike traditional cyber systems, where security may be managed around clearly defined software components or network boundaries, AI security must consider an interconnected and expanding ecosystem. The ability of agentic AI to learn and interact with its environment further complicates how we define its boundaries.

In the next chapter, we'll explore exactly how we can use these differences to craft unique attacks.

Summary

In this chapter, we explored the fundamentals of AI security, clarifying precisely what we're protecting and distinguishing AI security from traditional cyber and machine learning security. We walked through a threat model using MITRE, OWASP, and MAESTRO and learned about the complexities of securing modern AI systems.

We examined key threat actors—criminal enterprises and advanced persistent threats—and identified current trends from AI threat intelligence. We also discussed security frameworks such as the CIA triad and 3 D's model and introduced concepts like attack assessments and vulnerability scoring.

Lastly, we outlined key differentiators unique to AI systems, including their interpretability challenges, unpatchability, nondeterministic behavior, scale, and autonomy.

This wraps up our consideration of AI threats, as well as this part of the book on the fundamentals you'll need in AI security. From here, we'll shift gears and explore AI attacks and the defenses built to stop them.

Resources

For a deep dive into the topics in this chapter, see the following resources:

Computerphile, *https://www.youtube.com/@Computerphile/videos*. YouTube channel for engaging with and thoroughly exploring all sorts of security and AI topics, often including demos.

Graham Cluley and Carole Theriault, *Smashing Security*, *https://www.smashingsecurity.com/episodes/*. A podcast that provides an engaging introduction to security concepts through case studies and stories.

Harriet Farlow, *The AI Security Podcast*, *https://open.spotify.com/show/0BJiuvMBqVKKBW2XyID6Bz*. A podcast hosted by me and members of my team. We invite guests and summarize our AI threat intelligence in plain language.

Jack Rhysider, *Darknet Diaries*, *https://darknetdiaries.com/about/*. You'll be shocked by all the true cybersecurity stories in this podcast.

Multi-Agentic System Threat Modelling Guide, the GenAI Security Project, OWASP, April 22, 2024. Includes demonstrations in applying the MAESTRO framework and real-world examples.

Sella Nevo, Dan Lahav, Ajay Karpur, Yogev Bar-On, Henry Alexander Bradley, and Jeff Alstott, "Securing AI Model Weights," The RAND Corporation, May 30, 2024. A very comprehensive overview of the threat environment, including its actors and some great stories and statistics.

The Common Vulnerability Scoring System (CVSS), *https://www.first.org/cvss/*, and AI Vulnerability Scoring System (AIVSS), *https://aivss.owasp.org*. Standards used by practitioners to assess risk, severity, and likelihood.

The Cyber Kill Chain, Lockheed Martin, *https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html*. The framework that informs lots of cybersecurity threat modeling.

The MITRE ATT&CK framework, *https://attack.mitre.org*. A resource for cybersecurity tactics, techniques, procedures, and case studies.

The STRIDE Model, *https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats#stride-model*. Another threat modeling framework, which is also supported by Microsoft tools for automation and visualization.

William Gibson, *Neuromancer* (Ace, 1984). An excellent science fiction novel and where the term "cyberspace" is coined.

Look up "DEF CON" on YouTube to see all the recordings of previous talks, many of which increasingly discuss AI. I also recommend you search for some of the social engineering DEF CON content to understand how simple techniques can deceive humans (and agents!).