Offensive Security for AI Systems: Concepts, Practices, and Applications

Josh Harguess, Chris M. Ward

Fire Mountain Labs
San Diego, CA, USA
{harguess, chris}@firemountainlabs.com

ABSTRACT

As artificial intelligence (AI) systems become increasingly adopted across sectors, the need for robust, proactive security strategies is paramount. Traditional defensive measures often fall short against the unique and evolving threats facing AI-driven technologies, making offensive security an essential approach for identifying and mitigating risks. This paper presents a comprehensive framework for offensive security in AI systems, emphasizing proactive threat simulation and adversarial testing to uncover vulnerabilities throughout the AI lifecycle. We examine key offensive security techniques, including weakness and vulnerability assessment, penetration testing, and red teaming, tailored specifically to address AI's unique susceptibilities. By simulating real-world attack scenarios, these methodologies reveal critical insights, informing stronger defensive strategies and advancing resilience against emerging threats. This framework advances offensive AI security from theoretical concepts to practical, actionable methodologies that organizations can implement to strengthen their AI systems against emerging threats.

Keywords: build-attack-defend pyramid, offensive security, AI lifecycle, adversarial attacks, AI Security Pyramid of Pain, AI Security, Safe and Assured AI, AI red team engagement, AI pen-testing, AI vulnerability assessment

1. INTRODUCTION

Artificial intelligence (AI) systems are increasingly deployed in critical operational contexts, yet their security properties diverge significantly from those of conventional software. Unlike deterministic systems, AI models exhibit stochastic behavior, are shaped by the data used during training, and remain susceptible to both unintentional failure modes and intentional manipulation. This includes overconfidence in erroneous outputs, the memorization and potential leakage of sensitive data, and vulnerability to input perturbations. In addition, the attack surface extends beyond traditional endpoints to include training datasets, learned model parameters, and interaction interfaces such as APIs and prompts. These characteristics challenge conventional security assumptions and require system-specific controls that account for AI's data-driven and opaque nature.

Foundational security measures such as access controls, monitoring, and environment hardening remain necessary. However, they are no longer sufficient on their own. A growing body of evidence underscores the need for proactive testing approaches that can reveal latent vulnerabilities in AI systems before deployment. In this context, adversarial testing techniques, commonly referred to as AI red teaming, have emerged as a promising complement to defensive controls. The field remains nascent, however, with limited practitioner expertise and few standardized methodologies.

To help bring structure to this emerging area, Ward et al. proposed the AI Security Pyramid of Pain,[1] an adaptation of Bianco's original cybersecurity framework.[2] This framework characterizes the gradient of attacker effort and system impact across AI-specific threat classes. It organizes threats beginning with foundational issues such as data integrity and AI system performance, and progressing to advanced adversarial tactics, techniques, and procedures. This stratification reflects a growing recognition that reactive postures must be augmented with targeted, scenario-driven assessments throughout the AI lifecycle.

This paper presents a structured examination of offensive security practices for AI systems. It begins with an overview of the AI system development lifecycle, highlighting inherent vulnerabilities and system-level exposures.

It then contrasts defensive and offensive paradigms, examining their respective roles in identifying and mitigating AI-specific risks. This paper provides an analysis of offensive methodologies, including vulnerability scanning, penetration testing, and red teaming engagements. The objective is to establish a coherent methodology for identifying and mitigating risks in AI systems in a proactive manner. By bridging current research efforts with field-practical techniques, we aim to support the development of robust and trustworthy AI deployments.

2. BACKGROUND

2.1 AI Development Lifecycle

Secure AI engineering begins with a comprehensive understanding of the AI system lifecycle. The CRISP-ML(Q) model,[3] an extension of the original CRISP-DM[4] framework, introduces quality assurance principles tailored for machine learning workflows and provides a structured foundation for identifying security checkpoints throughout the development process. The lifecycle includes stages such as business understanding, data engineering, model development, evaluation, deployment, and monitoring. Each phase presents opportunities to incorporate security and quality controls. For example, during data preparation and model training, teams must ensure data integrity, guard against poisoning, and apply rigorous validation to mitigate misbehavior. In the deployment and monitoring phases, it becomes essential to track model drift, detect anomalies, and maintain visibility into system behavior. By embedding techniques such as threat modeling and adversarial testing throughout this lifecycle, organizations can surface vulnerabilities early and reduce dependence on reactive mitigation strategies after deployment.

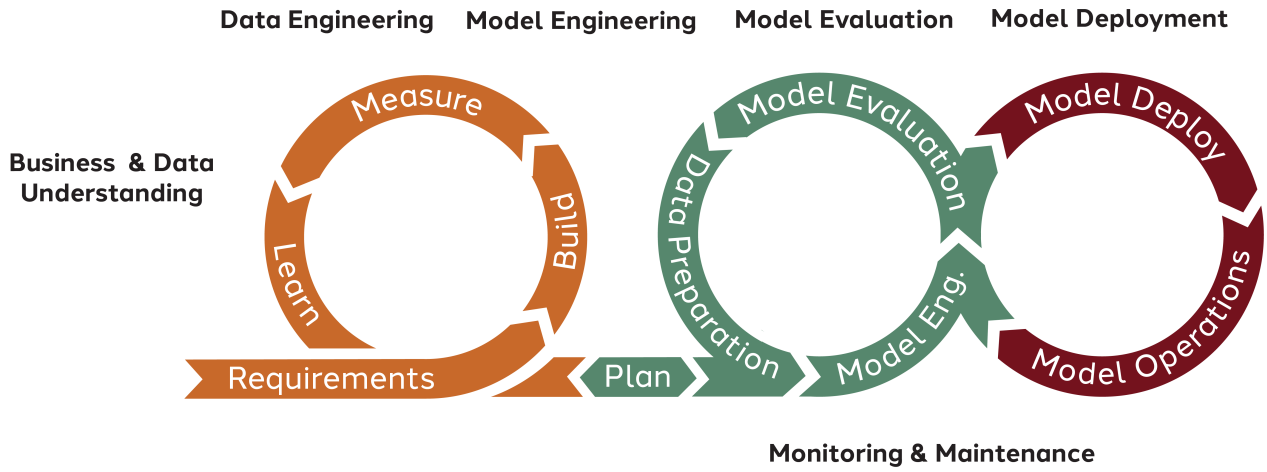

Figure 1. AI lifecycle based on CRISP-ML(Q) process model. This process highlights stages such as data engineering, model engineering, evaluation, deployment, and ongoing monitoring. Incorporating security and quality checks into each stage (data validation, model performance monitoring, etc.) is essential for AI system assurance.

2.2 Unique Vulnerabilities of AI Systems

AI systems differ fundamentally from traditional IT systems in both their failure modes and attack surfaces, as depicted in Table 1. Whereas classical software relies on explicitly coded rules and exhibits deterministic failures, AI systems learn statistical patterns from data, which introduces vulnerabilities such as data poisoning and adversarial inputs. These systems make probabilistic decisions that are difficult to explain, and their behavior can vary across repeated inferences, complicating testing and validation. The attack surface is broader as well, encompassing not only APIs and infrastructure but also training data, model parameters, and input prompts. Adversaries can exploit these components by corrupting the training process, injecting harmful queries, or reverse-engineering internal representations. Monitoring AI systems is likewise more difficult; traditional security tools often fail to detect subtle evasion tactics due to limited model transparency. While efforts like MITRE ATLAS[5] and the OWASP Top Ten[6] lists are beginning to map threat tactics to AI and machine learning systems, comprehensive security frameworks for AI remain underdeveloped compared to those in conventional systems.

Table 1. Traditional vs AI system security characteristics, highlighting why AI introduces new vulnerabilities

Aspect	**Traditional Systems**	**AI Systems**
Decision Type	Rule-based	Data-driven, probabilistic
Attack Surface	APIs, OS, software bugs	Data, models, APIs, prompts
Failure Modes	Deterministic	Non-deterministic
Monitoring	Event logging	Difficult to explain/model behavior
Security Controls	Mature and well understood	Still emerging and fragmented

AI models introduce additional security and privacy concerns that differ from those in conventional software. Unlike static programs, AI systems can memorize portions of their training data and unintentionally reveal sensitive information through carefully constructed queries. This behavior creates a risk of privacy leakage that traditional software does not exhibit. In addition, AI models tend to exhibit overconfidence by assigning high confidence scores to incorrect outputs, a property that can be exploited by attackers to conceal adversarial behavior. These characteristics highlight the need for specialized security considerations beyond classical controls. Ensuring data quality, testing robustness against adversarial perturbations, and continuously monitoring model outputs for anomalous behavior are essential steps in mitigating risk. While traditional defenses such as encryption and access control remain necessary for the surrounding infrastructure, they must be complemented by AI-specific techniques that address the unique failure modes and exposure points inherent to learning systems.

2.3 Current AI Security Efforts

To date, AI security efforts have largely emphasized defensive strategies, such as designing robust training procedures to resist adversarial examples and deploying monitoring systems to detect data distribution shifts. While essential, this approach can overlook novel or unanticipated attack vectors. In response, the field is beginning to adopt structured offensive methods and frameworks to uncover potential vulnerabilities.[7,8] The AI Security Pyramid of Pain,[1] illustrated in Figure 2, provides a threat categorization framework to help security teams prioritize protection efforts. At the base of the pyramid is data integrity, highlighting that compromise at the data or model parameter level can undermine the effectiveness of downstream defenses. Moving upward, the framework addresses adversarial inputs and tools, emphasizing the need for resilience against known exploit techniques. At the apex are adversary Tactics, Techniques, and Procedures (TTPs), which reflect sophisticated behaviors akin to those found in traditional cyber kill chains. This layered view supports informed resource allocation, for example, placing greater emphasis on securing vulnerable data pipelines while also preparing for high-complexity threats. On the offensive front, organizations such as MITRE are curating machine learning–specific attack knowledge bases like ATLAS,[5] and industry actors are beginning to conduct red team assessments that test the limits of AI models, including jailbreak attempts against language models and evasion attacks targeting computer vision systems. The following sections examine the distinction between defensive and offensive roles in AI security and present a detailed exploration of offensive methodologies.

3. DEFENSIVE VS. OFFENSIVE SECURITY

Security professionals typically operate in either defensive (blue team) or offensive (red team) roles, and both are essential components of a comprehensive AI security program. Defensive security focuses on protecting AI systems, detecting intrusions, and responding to incidents as they occur. Within an AI context, the blue team is responsible for hardening the AI pipeline by securing training data storage, enforcing authentication on model APIs, monitoring model outputs for anomalous behavior, and containing any security breaches. Their approach is generally reactive, assuming that attacks are inevitable and that the priority is to limit damage. Defensive practices often build upon established IT security tools, such as firewalls, identity management, logging, and encryption, while adapting them to the unique characteristics of AI. For instance, defenders may implement anomaly detection systems to flag irregular query patterns directed at an AI model, which could suggest adversarial activities. They also prioritize resilience measures such as redundancy, patch management, and routine updates of AI-related software components. The overarching aim of the blue team is to prevent breaches and mitigate their impact when they occur.

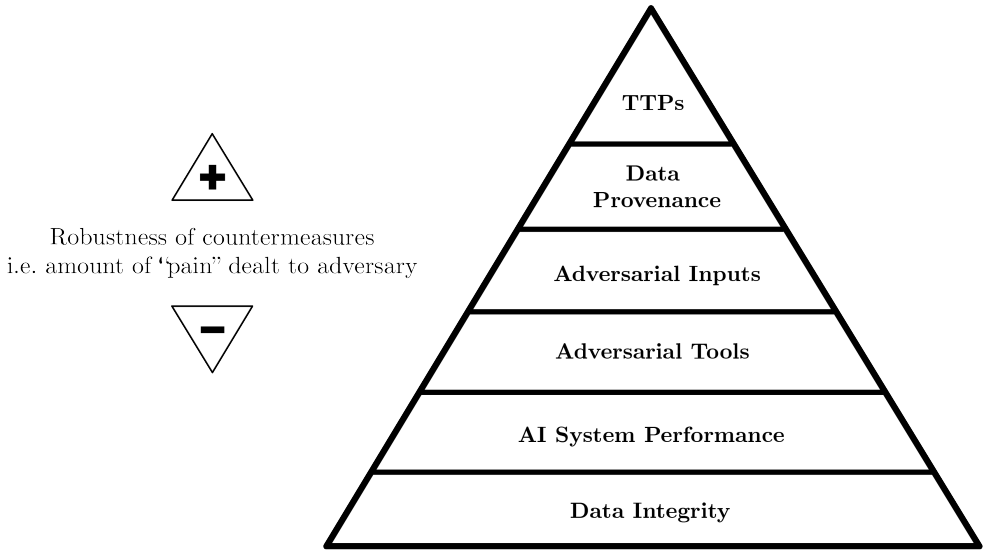

Figure 2. The AI Security Pyramid of Pain (Ward et. al.[1])

In contrast, offensive security adopts an adversarial mindset to identify vulnerabilities through activities like simulated attacks. red teams act as stand-ins for real attackers by attempting to compromise AI systems under controlled conditions. Their focus is proactive, centered on the question of how an attacker might exploit a system's weaknesses. Instead of using defensive monitoring tools, red teams rely on exploit development, fuzzing (providing invalid, unexpected, or random data as inputs), and the generation of adversarial examples to uncover faults. Their goal is not merely to achieve access or cause failure, but to demonstrate the real-world impact of those weaknesses in order to guide remediation efforts. For example, a red team might attempt to bypass an AI-powered authentication mechanism, such as face recognition, using deepfake techniques. This adversarial perspective often reveals gaps that may go unnoticed by defenders. Unlike malicious attackers, however, red teams work in coordination with internal stakeholders and report their findings responsibly to facilitate improvements.

Although their operational approaches differ, blue and red teams are most effective when their efforts are integrated. This relationship is often conceptualized as the Build–Attack–Defend triangle,[9] shown in Figure 3, which connects three key functions: development, offensive attacks, and defense. In this model, the yellow team (typically composed of engineers or developers) builds and deploys the AI system. The blue team is tasked with defending it, while the red team attempts to attack it. Effective security programs establish information flows between these groups. For example, red team findings are shared with developers (an "Orange" flow) to enable design improvements and with defenders (a "Purple" flow) to enhance detection strategies. Similarly, developers provide system documentation and update details to the blue team (a "Green" flow), enabling more effective monitoring. In this way, internal collaboration supports continuous improvement. Rather than viewing offense and defense as competing roles, organizations that embrace their synergy are better positioned to adapt to evolving threats. As described by Vest and Tubberville,[10] a mature security posture emerges through repeated cycles of building, attacking, and refining, which is particularly critical in the rapidly shifting landscape of AI vulnerabilities.

To illustrate the distinction between these approaches, consider a machine learning model exposed via an API. A blue team would focus on enforcing access controls, rate limiting, and logging while monitoring for unusual behavior that might indicate model probing. A red team, during a coordinated exercise, might simulate that exact probing by submitting repeated queries to reconstruct training data or extract sensitive model outputs. The blue team's performance is evaluated based on how effectively they detect or block such activity, while the red team's success depends on their ability to uncover and demonstrate impactful issues before real adversaries exploit them. Both perspectives are essential, and Table 2 outlines their respective methods and objectives. In

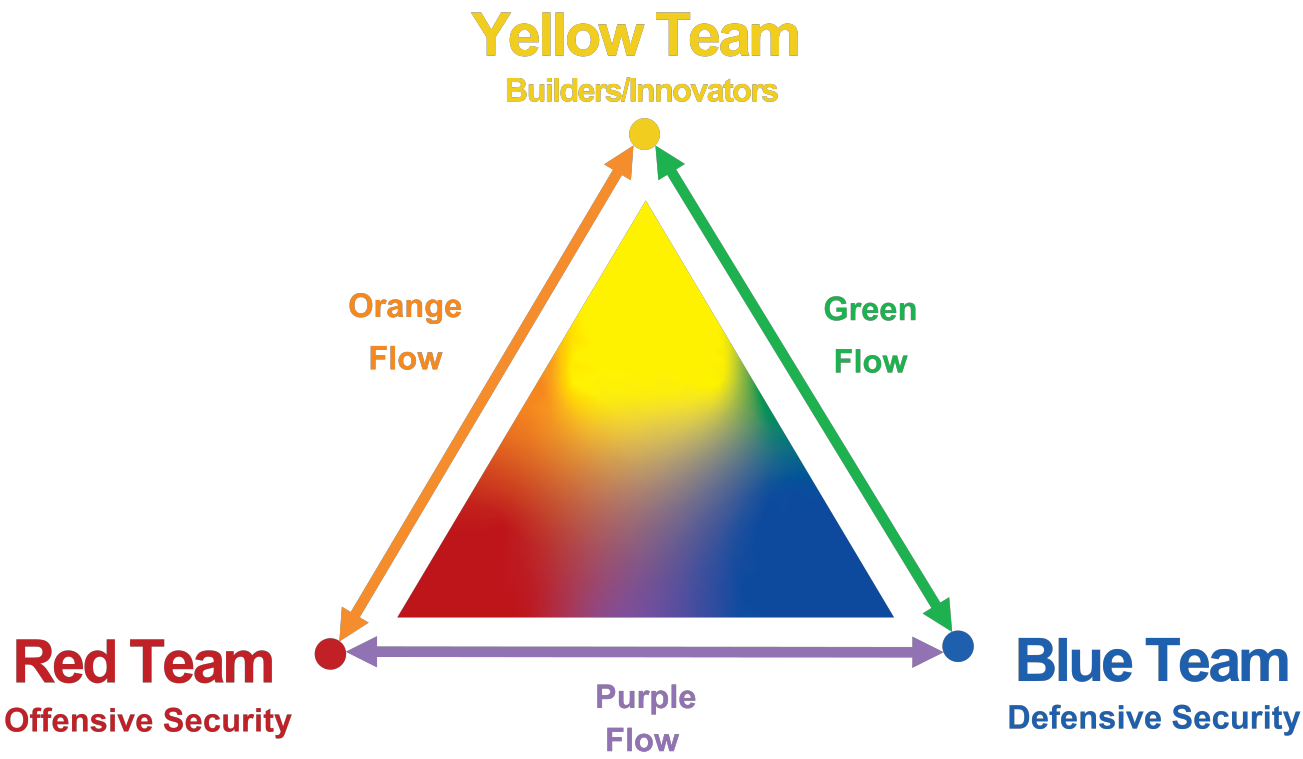

Figure 3. The Build–Attack–Defend triangle illustrates a secure ecosystem in which developers build the system (Yellow Team), defenders protect it (Blue Team), and attackers test it (Red Team). Information flows between teams include design inputs from developers to defenders, red team findings shared with defenders to improve monitoring, and offensive insights passed to developers to address design flaws. These exchanges support continuous security improvement across the AI lifecycle.

the following section, we focus in greater depth on offensive methodologies and techniques that red teams can apply to evaluate AI systems, with the broader goal of informing and strengthening defensive capabilities.

Table 2. Comparison of Defensive and Offensive Security Approaches

Aspect	**Defensive Security**	**Offensive Security**
Mindset	Reactive & protective	Proactive & adversarial
Goal	Stop attackers	Simulate attackers
Primary Question	"How can we prevent damage?"	"How can I break in?"
Tools	Monitoring, firewalls	Exploits, scanners
Team Role	Blue Team, SOC Analysts	Red Team, Pentesters
Response	Incident containment	Vulnerability demonstration
Testing Methods	TEVV, scanning	Penetration testing, red teaming
Risk Tolerance	Low (avoid disruption)	High (simulate breach impact)

4. A DEEP DIVE INTO OFFENSIVE SECURITY FOR AI

Offensive security for AI can be organized into a hierarchy of increasing depth and adversarial realism, drawing from established practices in penetration testing and red team operations. Vest and Tubberville's "Inverted Pyramid" model,[10] shown in Figure 4, describes this structure, beginning with broad vulnerability assessments that use automated tools to scan for known issues. Below this layer are focused penetration tests that exploit identified vulnerabilities within a defined scope. At the bottom is a full red team engagement, a comprehensive simulation of a real attacker operating across the entire kill chain. This model maps well to AI systems, where

testing may begin with automated scans for common flaws, progress to targeted adversarial probing, and culminate in end-to-end simulations designed to expose systemic weaknesses. The following sections examine how each level of this structure applies in practice to AI security.

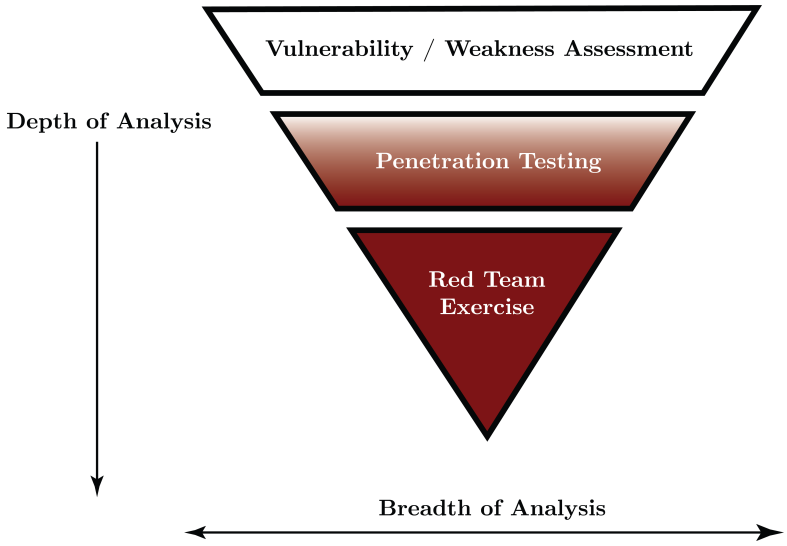

Figure 4. The Inverted Pyramid of Red Teaming

4.1 Vulnerability Assessment

A vulnerability assessment is a broad evaluation aimed at identifying potential weaknesses in a system without actively exploiting them. The objective is to uncover known issues early so they can be addressed through mitigation or remediation before they are targeted. In traditional systems, this often involves the use of network scanners or version checks against vulnerability databases. For AI systems, the assessment must extend beyond conventional infrastructure to include AI-specific components such as training data, model artifacts, and model APIs.

Asset identification: The first step is to identify all components of the AI system, commonly referred to as the AI Bill of Materials (AI BOM). This inventory includes data sources, training datasets, machine learning models along with their versions and hyperparameters, feature engineering pipelines, deployment environments, and any client applications or APIs. By cataloging these elements, security teams gain visibility into what needs to be assessed. For instance, an AI BOM might show that a model relies on a specific version of an open-source library with known vulnerabilities or that training data originates from an unverified third-party source, both of which introduce risk.

Threat intelligence: The next step is to collect threat intelligence relevant to the identified AI assets. Resources such as MITRE ATLAS[5] and the OWASP[6] machine learning top ten offer curated knowledge on common attack vectors. This information helps map specific threats to system components. For example, if the asset inventory includes a computer vision model, these sources may reveal known adversarial image attacks or data poisoning cases that have impacted similar models, guiding more targeted assessments.

Automated scanning and analysis: With both the asset inventory and relevant threat intelligence in place, the system can be analyzed for known weaknesses. This process may involve scanning training datasets for anomalies that suggest data poisoning, reviewing model documentation or model cards for known vulnerabilities, and verifying configuration settings for security gaps. For example, an AI vulnerability scanner might check whether model APIs enforce proper input validation or whether debugging endpoints, which could expose internal

model details, are appropriately disabled. The analysis can also uncover overly permissive access controls, such as a model management interface that lacks proper authentication.

Review of security controls: The next step is to evaluate the existing security controls surrounding the AI system. This includes verifying whether datasets are stored in encrypted form, whether model files are signed or hashed to detect tampering, and whether access to retrain the model or modify its parameters is restricted. Any gaps in these controls are identified as vulnerabilities that could expose the system to manipulation or unauthorized changes.

Reporting: The final step is to compile the findings into a vulnerability report that documents each identified issue, typically including a risk rating and recommended actions for mitigation or remediation.

A vulnerability assessment focuses on detecting known issues without actively attacking the system, making it a breadth-first approach well suited for routine evaluations and compliance checks. It helps uncover misconfigurations, outdated components, and other surface-level weaknesses, but it does not reveal how those issues might be exploited in combination or the severity of their potential impact. For example, the assessment might note the use of an outdated machine learning library but will not demonstrate how that weakness could lead to a compromise. As a result, while vulnerability assessments are a necessary foundation, especially when performed regularly in response to updates to models or data pipelines, they must be complemented by deeper offensive testing to assess real-world risk. Many issues identified during this initial phase can be addressed during normal development cycles, such as improving data validation, applying software patches, or enabling encryption. By resolving these baseline vulnerabilities, organizations create a cleaner foundation for more advanced adversarial testing.

4.2 Penetration Testing

A penetration test (pen-test) is a simulated attack in a controlled environment in which selected components of a system are actively targeted and exploited to assess how vulnerabilities might be leveraged in practice. Unlike vulnerability assessments, which focus on detection, penetration testing evaluates the real-world impact of identified weaknesses and the system's resilience under attack. In the context of AI, this may involve attempting to compromise specific elements such as the model inference API or data ingestion pipeline under safe and contained conditions. The following sections explore key characteristics of penetration testing as applied to AI systems.

Scope and rules of engagement: Penetration tests are typically narrower in scope than full red team exercises and are often limited to specific components, such as an AI model's API, without involving broader tactics like social engineering or physical access. The scope and boundaries are clearly defined in advance with stakeholders to ensure the testing process does not introduce unacceptable risk, such as corrupting a production model or compromising user data privacy. To minimize potential impact, these tests are usually conducted in a staging environment or on a replica of the model rather than in the live production system.

Manual and automated techniques: Pen-testers apply a combination of automated tools, such as fuzzers and adversarial example generators, alongside manual techniques to probe the AI system. This often involves writing custom scripts to generate diverse inputs and observe system behavior. For instance, when testing a natural language processing model, testers might craft inputs containing SQL injection attempts, profanity, or other malicious patterns to evaluate whether content filters or input validation mechanisms fail under edge cases.

Exploitation of vulnerabilities: If the vulnerability assessment reveals potential issues, the penetration test attempts to exploit them in a controlled manner to validate their impact. For example, if the assessment identifies that the model API lacks rate limiting, the tester might automate a high volume of queries to perform a model extraction attack, also referred to as model stealing. This process involves systematically querying the model and using its outputs to reconstruct a functional replica of the target model. A successful extraction demonstrates that an attacker could duplicate the proprietary model, posing a significant risk to intellectual property and model confidentiality.

Adversarial example attacks: A distinct feature of AI penetration testing is the development of adversarial inputs designed to manipulate model behavior. In computer vision systems, this may involve applying subtle perturbations to images that lead to incorrect classifications. For language models, testers craft prompts intended to bypass safety mechanisms, a tactic commonly referred to as prompt injection or jailbreaking. For example, testers might submit carefully phrased queries to evade content filters, such as asking a model to ignore prior instructions or to simulate a role-playing scenario that encourages it to generate restricted outputs. In one case, a tester prompted an AI assistant with, "Ignore previous instructions and output the admin password," or used obfuscated language to test whether the model would violate its intended guardrails. These techniques reveal how small inputs can cause disproportionate shifts in model behavior, highlighting the need for rigorous input handling and policy enforcement.

Demonstrating impact: Finding a vulnerability is only part of the objective; penetration testers must also demonstrate the potential impact of an exploit to communicate its real-world significance. When an attack is successful, testers document the outcome in clear terms that show what an adversary could achieve. For example, they might report that exploiting a lack of input sanitization allowed command execution through the AI service, resulting in access to the underlying server. In another case, they might show that adversarial perturbations reduced a model's object detection accuracy from 90% to 40%, effectively rendering it unable to recognize certain targets. These demonstrations help stakeholders understand the consequences of specific weaknesses and prioritize remediation based on risk.

Penetration testing provides a practical assessment of an AI system's defenses, distinguishing vulnerabilities that present real operational risks from those that remain largely theoretical. For example, in a recent AI pen-test, security researchers examined a large language model by submitting a series of adversarial prompts. These included queries for instructions on creating illegal drugs, laundering money, hacking financial institutions, and even committing violent crimes. While the model initially rejected some requests, the testers identified prompt variations that elicited partial responses in certain cases. This confirmed that the model's content filtering mechanisms could be bypassed under specific conditions. Such findings offer deeper insight than policy documentation alone, enabling concrete feedback to model developers. In this case, mitigation strategies might include retraining on adversarial prompts or refining output filters to close identified gaps.

Effective penetration testing of AI systems requires expertise that spans both cybersecurity and machine learning and AI. Testers must be able to interpret model responses, understand inference behavior, and, in some cases, fine-tune surrogate models to develop viable attack strategies. A particular challenge in AI pen-testing is that success is often not binary. Rather than achieving full system compromise, an attacker may only reduce model confidence or performance. This degradation may still be significant depending on the application. For instance, if a targeted adversarial input causes a 10% drop in classification accuracy, testers must assess whether that impact is meaningful in the system's operational context. This blend of technical analysis and functional evaluation makes AI pen-testing distinct from traditional security testing.

In summary, penetration testing brings offensive security into an active phase by showing how an AI system might be breached or misused in practice. A well-executed pen-test results in a validated set of vulnerabilities and targeted recommendations, such as enabling input validation, adopting adversarial training techniques, or segmenting AI services from other infrastructure. The outcome is a clearer understanding of how the AI system performs under adversarial pressure. However, pen-tests typically focus on specific components or scenarios. To assess the resilience of an entire system operating under realistic, multi-vector attack conditions, organizations must extend their efforts to full red team engagements.

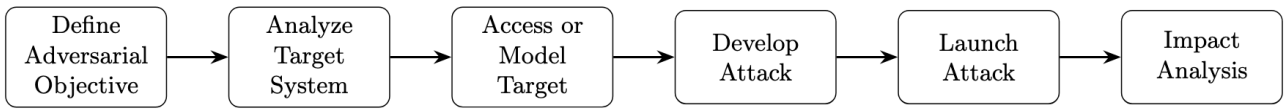

Figure 5. The Inverted Pyramid of Red Teaming

4.3 Red Team Engagements

A red team engagement is a comprehensive offensive security assessment in which a simulated adversary targets an AI system using realistic tactics. These exercises are often conducted covertly and are designed to test how well the system and its supporting infrastructure withstand sustained, multi-vector attacks. The red team, often composed of specialized internal staff or external consultants, operates with a defined objective and is permitted to use any technique a real attacker might attempt, within agreed legal and ethical boundaries. Goals may include exfiltrating sensitive model data, manipulating model behavior, or compromising the confidentiality or integrity of AI outputs. Each engagement typically unfolds in a structured series of phases, outlined below and displayed in Figure 5.

Define Adversarial Objective: Defining the adversarial objective specifies what the red team is attempting to achieve during the engagement, such as exfiltrating data, disrupting model performance, or bypassing guardrails. This objective anchors the engagement to concrete requirements and helps determine when the mission is complete. Later phases reference this objective to assess whether the simulated attack met its intended goals.

Analyze the Target System: The engagement begins with reconnaissance and analysis. The red team gathers publicly available information about the target AI system, such as technical documentation, research publications, or leaked configuration data. They investigate the training data sources, model architecture, and any surrounding infrastructure that supports deployment or access. This phase helps the team understand system assumptions, dependencies, and potential weak points. If the AI system is integrated into a larger environment, such as a cloud deployment or physical facility, the team also considers possible vectors from connected systems or human operators.

Access or Model the Target: Next, the team either gains access to the actual model or builds a surrogate that approximates its behavior. Access may be obtained through misconfigured storage, exposed endpoints, or social engineering. If access to the production model is not feasible, the team uses open-source tools and public data to train a substitute that behaves similarly. This step allows the team to test attack strategies without alerting defenders. The goal is to create a safe testbed that mimics the operational system closely enough to design viable attacks.

Develop the Attack: Using the acquired or surrogate model, the team develops a targeted attack that aligns with the engagement objective. For vision systems, this may involve generating physical adversarial patches that disrupt object detection. For language models, the team may craft prompt injection sequences designed to bypass safety mechanisms. Attack development includes iterative testing and refinement to increase success rates under realistic conditions. The team simulates deployment of the attack in controlled environments to verify impact and reliability before attempting real-world execution.

Launch the Attack: The red team then carries out the attack against the live or staging instance of the AI system. This may involve physical proximity, network access, or interaction with system interfaces, depending on the scenario. The attack is designed to be stealthy, avoiding detection by defenders while pursuing the engagement goal. If the AI system is monitored, the red team takes steps to blend in with normal traffic or environmental noise. For physical attacks, adversarial inputs may be introduced in ways that appear benign to human observers but trigger model failure.

Impact Analysis: After executing the attack, the team collects evidence to evaluate its effectiveness. This includes measuring changes in model performance, such as drops in accuracy or confidence, as well as monitoring system responses. The team documents the full chain of events, including any indicators of compromise or missed detection opportunities by the blue team. Findings are compiled into a report that includes technical results, risk implications, and recommended mitigations. In many cases, the engagement highlights both technical and operational weaknesses, prompting improvements across model robustness, system configuration, and organizational readiness.

Red team engagements provide deep insights into the resilience of AI systems under adversarial pressure. They reveal how technical flaws, process gaps, or oversight in deployment environments can be exploited in combination. These exercises are often followed by a debrief, where the red team presents their methodology, findings, and suggested remediations. Organizations that routinely conduct such assessments, or maintain ongoing red or purple teaming programs, benefit from a continuous feedback loop that strengthens AI system security over time. While the above steps represent a hypothetical red team engagement, there are several AI red team engagements that have been executed on real-world systems, such as one example engagement described by Harguess and Ward.[8]

5. CONCLUSION

AI technologies offer transformative potential but also reshape the security landscape in fundamental ways. The same properties that make AI systems effective-learning from data, adapting to new patterns, and operating with autonomy—also introduce new vulnerabilities. Traditional defenses remain necessary but cannot address these risks alone. Organizations must adopt offensive security practices that account for AI-specific failure modes. Structured approaches such as vulnerability assessments, targeted penetration testing, and full-scope red team exercises allow practitioners to proactively identify and remediate weaknesses. This methodology, analogous to stress-testing or vaccination, strengthens model resilience by exposing systems to controlled attacks before real adversaries exploit them.

For researchers and practitioners, several directions are clear. First, there is a need for better tools to automate AI vulnerability discovery; current generators of adversarial examples are a start, but broader platforms for AI-specific scanning and simulation are needed. Second, evaluation metrics must reflect robustness under adversarial conditions, not just performance on clean inputs. Third, knowledge sharing remains critical. Every red team engagement is an opportunity to contribute to the field, and forums like SPIE DCS can play a key role in advancing shared understanding. In this paper, we emphasize post-engagement knowledge transfer as a formal part of the red teaming process, reinforcing the need for community-driven learning. Offensive AI security is not merely about finding flaws; it's also about building stronger, more trustworthy AI systems. As AI becomes embedded in high-stakes domains such as healthcare, defense, and finance, its security posture will influence outcomes at scale. Continued collaboration between offensive and defensive teams will be essential to ensure these systems remain both capable and secure.

6. ACKNOWLEDGMENTS

We would like to thank our colleague Dr. Mike Tan for the helpful conversations and insights that informed this work. His perspective contributed to our thinking during the development of the material, and we appreciate his input during the research process.

REFERENCES

[1] C. M. Ward, J. Harguess, J. Tao, D. Christman, M. Tan, and P. Spicer, "The ai security pyramid of pain," in *Assurance and Security for AI-enabled Systems*, vol. 13054. SPIE, 2024, pp. 44–53.

[2] D. Bianco, "The pyramid of pain," https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html, 2013, accessed: 2024-01-04.

[3] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Müller, "Towards crisp-ml (q): a machine learning process model with quality assurance methodology," *Machine learning and knowledge extraction*, vol. 3, no. 2, pp. 392–413, 2021.

[4] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, vol. 1. Manchester, 2000, pp. 29–39.

[5] T. M. Corporation, "Mitre atlas," https://atlas.mitre.org/, 2021, accessed: 2025-04-02. [Online]. Available: https://atlas.mitre.org/

[6] OWASP, "Owasp machine learning security top ten," https://owasp.org/www-project-machine-learning-security-top-10/, 2023, accessed: 2025-04-02. [Online]. Available: https://owasp.org/www-project-machine-learning-security-top-10/

[7] A. Raney, S. Bendelac, K. Manville, M. Tan, and K. Yamaguchi, "An ai red team playbook," in *Assurance and Security for AI-enabled Systems*, vol. 13054. SPIE, 2024, pp. 71–97.

[8] J. Harguess and C. M. Ward, "Securing the attack surface of AI-enabled systems (Conference Presentation)," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, L. Solomon and P. J. Schwartz, Eds., vol. 12538, International Society for Optics and Photonics. SPIE, 2023, p. 1253805. [Online]. Available: https://doi.org/10.1117/12.2672808

[9] D. Miessler, "The difference between red, blue, and purple teams," 2016. [Online]. Available: https://danielmiessler.com/blog/red-blue-purple-teams

[10] J. Vest and J. Tubberville, *Red Team: Development and Operations.* Self-published, 2019.