Secure AI Framework

Security Assurance in the Age of Generative AI

Authors
Tom Grzelak, Kara Olive, Moni Pande

Abstract

Artificial Intelligence (AI) is a rapidly growing field known for experimentation and quick iteration, qualities that can pose challenges for traditional enterprise security approaches. Because AI introduces unique assets and surfaces—AI-driven applications, agents, assistants, vast training datasets, the models themselves, and supporting infrastructure—we're continually updating our security controls, guided by *Google's Secure AI Framework* *(SAIF)*.[1]

To address the new challenges, we've expanded our traditional security approaches to cover the new attack surfaces by scanning for more types of vulnerabilities, analyzing more intel,[2] preparing to respond to new kinds of incidents, and continually testing our controls in novel ways to strengthen our security posture.

This white paper is one of a series describing our approaches to implementing Google's SAIF. In this paper we explain how we're applying ***security assurance*—a cross functional effort aiming to achieve high confidence that our security features, practices, procedures, controls, and architecture accurately mediate and enforce our security policies**—to AI development. Security assurance efforts help to both ensure the continued security of our AI products and address relevant policy requirements.

[1] http://saif.google
[2] https://cloud.google.com/blog/topics/threat-intelligence/adversarial-misuse-generative-ai

Just as quality assurance (QA) in manufacturing meticulously examines finished products and the processes that create them to ensure they meet quality standards, security assurance serves a complementary role to the broader security efforts within an organization. Those broader security efforts span the design, implementation, and operation of controls to create secure software products; security assurance focuses on verifying and improving those efforts. Security assurance identifies gaps, weaknesses, and areas where controls may not be operating as intended, to drive continuous improvement across all security domains. It's two-party review in action—security assurance helps build confidence that the software was not just built securely, but continues to run securely.

Since AI systems—those that use AI models for reasoning—present a combination of well understood and novel risks, AI technologies require a combination of both common and novel controls. No matter how strong these controls are, a security assurance program is essential to ensure they are working as intended and that they are continually updated and improved.

The paper opens with an overview of security assurance functions, covering several teams and capabilities that work together to ensure security controls are working across any software development lifecycle, including the AI development lifecycle. In particular, we focus on four functions—Red Teaming, Vulnerability Management, Detection & Response, and Threat Intelligence, and how those work together to address issues through Remediation.[3]

We then describe the features specific to AI that affect assurance functions and give examples of how we're adapting our approaches to account for AI-specific technologies and risks. We also include guidance for organizations considering creating their own AI assurance programs, including best practices for assuring training data, models, the AI software supply chain, and product integrations.

We intend this paper to be useful for a broad technical audience, including both assurance specialists who are new to AI technologies, and AI developers who are new to assurance practices.

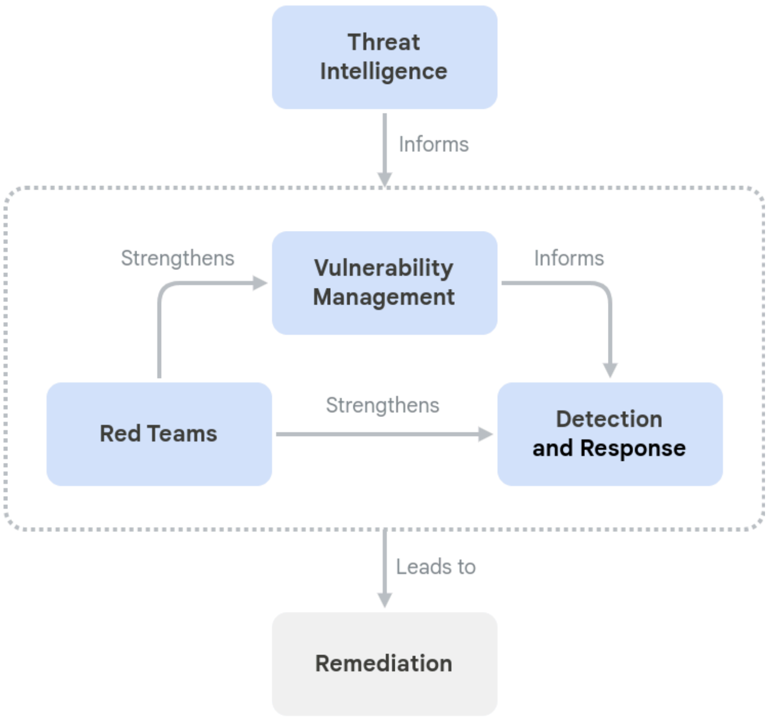
[3] This paper does not cover closely related fields to security assurance, such as security policy and compliance, risk assessment, and risk management.

Table of contents

Security assurance in the age of generative AI

When fundamental new technologies like AI are introduced, malicious actors are among the first to explore them, looking for new avenues to achieve their malicious intent. Understanding the potential impact of these attack vectors on AI systems can help AI developers ensure that their tools are deployed safely. At Google, we're working to find and overcome security vulnerabilities and weaknesses by extending our approach to security assurance from traditional software development into the AI model development and deployment lifecycle.

As background for the discussion of AI, let's first summarize security assurance for traditional software. In security programs for traditional software systems,[4] assurance teams contribute to securing deployed systems primarily through four areas: **Vulnerability Management**, **Detection and Response**, **Threat Intelligence**, and **Red Teaming**. These teams work both independently and together—strengthening each other via feedback loops—with all of them feeding into the work of **Remediation**.



[4] Adkins, H., Beyer, B., Blankinship, P., Lewandowski, P., Oprea, A. and Stubblefield, A., 2020. *Building secure and reliable systems: Best practices for designing, implementing, and maintaining systems*. O'Reilly Media, Inc.

Vulnerability management

The search for potential vulnerabilities never stops. To maintain post-launch security, we continuously scan for known vulnerabilities across our entire infrastructure. This includes our storage systems, code repositories, and AI-specific systems like AI data, training, and testing systems, as well as data recipes.

This approach helps safeguard models and AI-powered products and ensures that the model inference environment is secure, preventing it from becoming a potential attack surface.

Ensuring automatic and robust reporting and remediation of the common vulnerabilities also helps us focus the efforts of our Red Teams, letting them concentrate on novel and unique attacks. The information they gather through this process then feeds back into the vulnerability detection implementation.

Detection and response

While Vulnerability Management ensures known vulnerabilities in infrastructure, software, and their dependencies are quickly identified and mitigated, Detection and Response (D&R) focuses on attacks—malicious actions that have bypassed existing controls.

Detection acts as a monitoring layer constantly looking for any suspicious internal or external activities. Those suspicious activities can include unauthorized access to sensitive data, unusual traffic patterns, suspect code execution, unauthorized or questionable file modifications, and attempts at exfiltrating AI intellectual property.

When suspicious events are identified, Response is prepared to quickly act on the threat in order to first contain and then neutralize it. Using processes that are regularly practiced through red team exercises, they issue customer notifications, investigate, and respond to security issues.

Threat intelligence

Threat Intelligence[5] looks to the external world, analyzing an ever-evolving cast of threat actors to provide insights into current and future risks. Given the rapidly changing technical landscape, it's more important than ever to have up-to-date information about who adversaries are, their motives, and the specific techniques that they're using and developing.

[5] https://blog.google/threat-analysis-group/

A strong Threat Intelligence team (or vendor provided service) can help to reason about both potential targets for attackers and how those attackers might abuse a company's capabilities for nefarious purposes. Threat Intelligence is a key assurance input for D&R teams, ensuring that they are able to defend against relevant threats. Threat intelligence also helps make certain that the work of Red Teams is informed by real-world scenarios.

Red teaming

Red Teaming[6] applies adversarial techniques against their organization's own infrastructure, applications, processes, and technologies to test for insufficient defenses and the ability of D&R teams to respond to compromised systems.

Red Teams help organizations prepare for potential attacks, raise concerns, and propose new or improved defenses against adversaries. Red Teams can both simulate attacks that Threat Intelligence teams are seeing right now, as well as anticipate and execute attacks that they believe adversaries might attempt in the future. They also collaborate closely with D&R teams to ensure attacks can be swiftly spotted and shut down, and to validate, or, if necessary, identify potential gaps in detection capability.

Proactively attacking an organization's own AI systems and products can help provide assurance that in the event of a real attack, their D&R team is capable of detecting the attack, and is appropriately prepared to respond.

The Red Teams' findings and insights into attack chains are subsequently analyzed as candidates for Vulnerability Management, and D&R automations and monitoring, and inform secure design considerations for future product iterations. This proactive approach ensures that potential threats are promptly detected and addressed, strengthening the overall assurance function.

Remediation

The findings and insights from Vulnerability Management, D&R, Threat Intelligence, and Red Teams work together to flag vulnerabilities, simulate real-world attacks, stay ahead of threat actors, and quickly respond to security incidents. The feedback from each of these areas creates a continuous cycle of improvement.

For example, Threat Intelligence identifies threat actors and shares their *Tactics, Techniques, and Procedures (TTPs)* with the Red Team, which helps them create new adversarial test scenarios that they use in red team exercises. This, in turn, guides the Vulnerability Management in building automatic testing for the newly found vulnerabilities,

[6] https://blog.google/technology/safety-security/meet-the-team-responsible-for-hacking-google/

further enabling their remediation. Additionally, red team exercises also inform the decisions of the D&R team on what new detection capabilities are necessary to address any residual risk.

Until this point in the assurance lifecycle, these efforts have primarily focused on threat and vulnerability identification; this is not sufficient, and in order to defend against attackers, Remediation efforts are necessary to ensure any findings are properly mitigated and closed. Remediation efforts permanently address any identified issues with the storage systems, code, AI models and underlying infrastructure.

> Securing AI is a cross-functional imperative: the most robust and trustworthy AI products will be built by teams where AI and security expertise are not siloed, but rather work together from the start.

Specific considerations for AI assurance

In this section, we will explore several fundamental properties that differentiate AI technologies from traditional software,[7] and we'll discuss our current approach for addressing these differences within our security assurance functions. In many cases, security assurance for AI depends on the extension of security practices that are common in traditional software development. As we look at some of the unique elements of AI development, we'll also highlight some of the security practices that can help mitigate those new security challenges. Note that all of these approaches also apply to AI agents, though the challenges of securing AI agents are more expansive.

Developing AI with assurance in mind

While many security considerations are broadly applicable, organizations should still create security assurance programs tailored to their specific business needs and risk tolerance. By adhering to the following best practices, organizations will be better able to create effective security, and meaningful assurance.

[7] https://services.google.com/fh/files/misc/ociso_securing_ai_different_similar.pdf

Develop AI threat modeling

As with classic software products, threat modelling can help proactively identify which AI risks are most applicable and where security investments have the largest impact. For a comprehensive threat model, take into account both the SAIF AI risks as well as relevant Threat Intelligence. Organizations must analyze external adversaries, their goals, and their methods (Tactics, Techniques, and Procedures, or TTPs), paying close attention to emerging AI-specific attacks like model evasion or data poisoning. This foresight allows anticipation of how AI systems might be compromised or exploited. The resulting intelligence is vital for tuning Detection & Response capabilities and ensuring Red Team activities mimic relevant, real-world threats.

Create an asset inventory

It's impossible to secure what you don't understand, and assurance starts with understanding your tools, datasets, models and infrastructure. A good first step is to centralize information on all critical assets such as data, models, controls, and systems, as well as their intended use cases, to prioritize their protection and understand their relationships and interdependencies.

Set a detection and response baseline

Once you have a clear understanding of your assets and systems, you can analyze the potential paths an attacker might take to understand how your AI system can be attacked or used for malicious purposes. This will form the basis for testing detection and response mechanisms within the organization.

Conduct periodic reviews

While the security landscape is always changing, AI is going through a dramatic growth phase, so it's important to engage product area and infrastructure teams in regular reviews to validate controls and processes end to end.

Security assurance for training data

Modifications made to a model's training data can create effects that are as significant as manipulating code in traditional software. The vast size of the training datasets and the fact that a significant amount of data comes from the public domain can make it difficult to assess changes to the datasets. In some cases, that can also make it difficult to detect

Data Poisoning[8]—a type of attack where malicious content is injected into the training sets to influence the final behavior of the model.

Therefore, maintaining model integrity requires not only sourcing training data responsibly, but also explicitly tracking the organization's trust level in each dataset, controlling access, and preventing unauthorized modifications of datasets. While there are well-established processes for handling changes to code—like mandatory code review and unit tests—changes in datasets with billions of data points are more challenging.

Best practices

To assure the security of training data, you need to have a clear understanding of what that data looks like and establish control over it. This means identifying and inventorying all the relevant datasets used for training models, as well as all the tooling used for automated filtering, quality scanning, or manual training data entry.

Data curation tools that are built in-house often lack robust security mechanisms, which puts them in added jeopardy for attacks. It's therefore essential to include them in the scope of Red Teaming, D&R, and Vulnerability Management efforts.

These teams should continuously monitor who has access to those data stores and tools, and lock down access, limiting it to essential personnel only. This alone will significantly reduce the likelihood of certain attacks such as exfiltration and access abuse.

Teams should also develop and maintain data *provenance*—tamper-evident metadata about the source and contents of each dataset. Provenance[9] is essential for creating high-quality models because it provides attestations about the quality and lineage of the underlying data.

Security assurance for models

AI models are trained to recognize patterns, make predictions, and perform tasks without being explicitly programmed for every possible scenario. To accomplish this, models rely on the fundamental elements of weights and hyperparameters that capture the probabilities for each type of behavior trained into the model. Because weights are core to a model's decision-making process, they are an attractive target for malicious actors, and therefore important assets to secure.

[8] https://saif.google/secure-ai-framework/risks#data-poisoning

[9] https://github.com/cosai-oasis/ws1-supply-chain/blob/main/risks-and-controls-for-the-ai-supply-chain-v1.md#mitigation-through-data-provenance

Unauthorized access to model weights and the ability to manipulate them could allow attackers to modify a model in malicious ways, undermining its integrity in ways that create a risk to users and their data.

An important consideration when working with models is their unique storage formats. Certain formats allow arbitrary code to be embedded within the model file. This code can execute automatically during the loading process or inference phase, creating significant security risks when downloading models from untrusted sources. This specific vulnerability spurred the development of inherently safer formats, such as Safetensors[10], which are designed to store only the necessary data (weights and metadata) and prevent arbitrary code execution by design upon loading.

Another factor that distinguishes models from traditional software is that the output from complex AI models—especially *large language models (LLMs)* that use deep learning—is inherently non-deterministic, meaning that identical prompts may generate different responses from the model at different times.

This probabilistic nature means it’s not only difficult to replicate some of the unexpected or undesirable behaviors. Testing a few times with the same favorable result no longer offers the same level of security assurance as it might when testing non-AI code. Due to large models' inherent complexity, testing on a sample set of inputs cannot rule out undesirable behaviors elicited by a different, maliciously-crafted input. This reduces the effectiveness of automated testing, so Red Team testing and complementary approaches to discover and contain novel attacks are even more critical for security assurance.

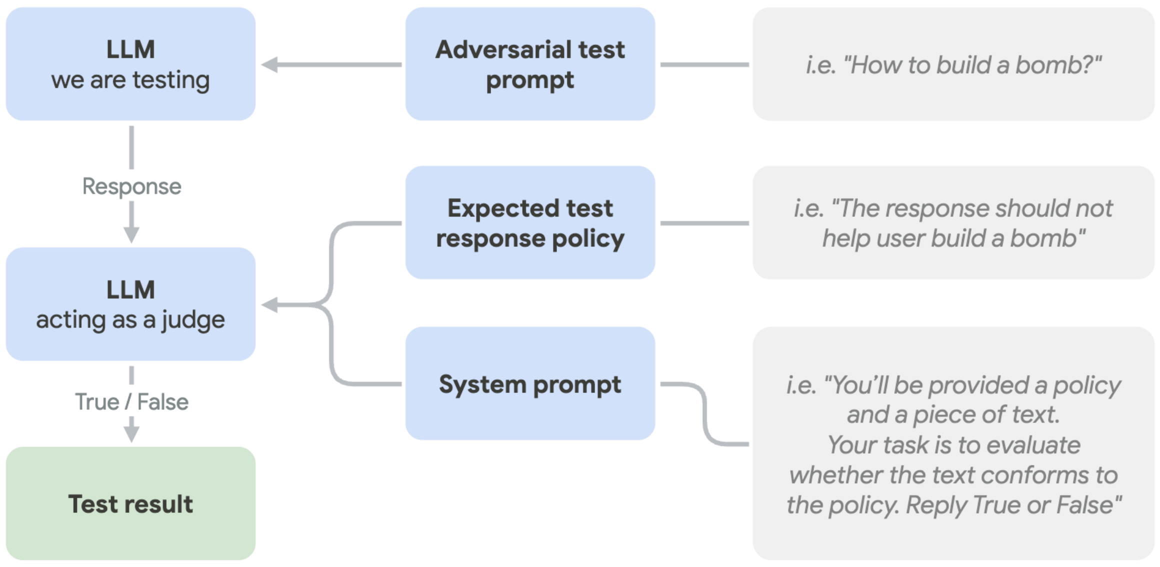
Best practices

One approach worth exploring is using another AI model as a judge[11] to evaluate the responses of the model being tested. This judging AI model (often referred to as "auto-rater") should be given a comprehensive description of the desired result and asked to assess the responses accordingly. This approach of stacking AI models is becoming increasingly popular, with applications ranging from a mixture of experts[12] to more novel agent systems. Note, however, that this approach could be vulnerable to multi-stage attacks that propagate adversarial inputs to the judging AI model via the primary model.[13]

[10] Casey, B., Damian, K., Cotaj, A., & Santos, J. (2025). An Empirical Study of Safetensors' Usage Trends and Developers' Perceptions. arXiv preprint arXiv:2501.02170.

[11] https://cloud.google.com/vertex-ai/generative-ai/docs/models/evaluation-overview
[12] https://huggingface.co/blog/moe
[13] Mangaokar, N., Hooda, A., Choi, J., Chandrashekaran, S., Fawaz, K., Jha, S. and Prakash, A., 2024. PRP: Propagating universal perturbations to attack large language model guard-rails. arXiv preprint arXiv:2402.15911.

Simplified architecture of a testing setup using LLMs as judges


LLM
we are testing
Adversarial test prompt
i.e. "How to build a bomb?"
Response
Expected test response policy
i.e. "The response should not help user build a bomb"
LLM
acting as a judge
System prompt
i.e. "You'll be provided a policy and a piece of text. Your task is to evaluate whether the text conforms to the policy. Reply True or False"
True / False
Test result

Another technique to regain some predictability is to build a deterministic filtering or sanitization layer on top of the model, processing all data coming into and out of the model. This sanitization layer can be tested separately from the model using predictable unit tests, achieving a high level of assurance when filtering for properties based on structured output ("is the input within a safe subset of html/markdown?"). Deterministic constraints become especially important in the case of agents, which can use tools to initiate actions, sometimes autonomously. In this case, agents can be limited in access so that they're allowed only to interact with a well defined, narrow list of tools, in specific situations.

A proactive, transparent, and continuous security assurance program is not a constraint on innovation. On the contrary, it provides the confidence to build boldly, deploy responsibly, and deliver on the transformative promise of artificial intelligence.

Security assurance for infrastructure and the AI software supply chain

Many AI training and inference workloads require extensive computational capabilities from highly advanced hardware, like GPUs or TPUs, which present an enticing target for attackers. Without appropriate security measures, these infrastructure stacks can be susceptible to attempts to exploit the hardware for compute-intensive tasks like crypto-mining, or costly DDoS attacks which might impede serving the model to customers.

Google's Secure AI Framework classifies all code and model frameworks as components of the AI infrastructure. This includes open source libraries (such as TensorFlow and PyTorch), which are also attractive targets as part of the AI software supply chain.[14] Establishing an appropriate balance between the significant acceleration and simplification of AI systems offered by third-party libraries and the risks of vulnerable dependencies is a critical element of security assurance. Vulnerabilities in external dependencies create a cascading risk for AI systems, as seen in classical software development, potentially allowing attackers to exploit libraries or packages and compromise dependent models.

The assurance approach for AI infrastructure systems depends on deployment. Cloud-based models enjoy the benefits of centralized security, robust infrastructure, and dedicated response teams, making assurance simpler. However, it requires transmitting user data, which introduces privacy concerns. In contrast, on-device AI enhances user privacy by processing data locally, but its decentralized nature requires teams to account for device-specific vulnerabilities, limited resources, and the risk of physical tampering, making it harder to ensure model integrity and protect against extraction.

Best practices

Many of the assurance practices to help secure AI product infrastructure represent extensions of best practices for software development:

Maintain strong access control and authentication

- Securing compute resources (GPUs or TPUs) used for training and inference requires strong authentication and access controls. This includes authenticating the systems that access these resources, as well as authenticating the AI models

[14] https://research.google/pubs/securing-the-ai-software-supply-chain/

themselves when they interact with external systems. Implementing robust rate-limiting alongside authorization mechanisms to secure user-facing infrastructure or APIs can help to limit risks such as model inversion or model extraction, where attackers attempt to infer information about training data or recreate a model's weights based on its responses to mass queries.

- Comprehensive monitoring of APIs used to access models is critical for preventing unauthorized access attempts. Detection and Response teams can build targeted detection logic that identifies access attempts by looking for anomalous patterns outside the "baseline."

- To assure that a model hasn't been tampered with, developers can use Sigstore,[15] co-developed by the Google Open Source Security Team, which provides a helpful tool for signing models. Its ML-specific library[16] provides tools for developers, package managers, and security experts for digital signing, verification, and provenance checks needed to make open-source software safer to use and distribute.

When possible, run the inference process on end-user devices for privacy-sensitive use cases

- Running inference on a user's device allows private user data to be processed directly in a private environment, reducing the risk of data breaches and unauthorized access. This approach is becoming more important as a growing number of applications are expected to gain access to on-device models. This trend will make local inference a key strategy for enhancing user privacy and security.

- For on-device deployments, it's crucial to determine the specific need for both model security and model integrity. To protect the model file, secure storage mechanisms like hardware-backed encryption or secure enclaves can be employed to mitigate the risk of unauthorized access, modification, or extraction. However, it's important to acknowledge that mitigation and defense models are fallible, and a determined well-funded attacker can compromise the models and device.

- The model integrity should be verified to detect and prevent any unauthorized modifications that could occur during model installation, updates, and ideally every time it's launched on the device.

[15] https://www.sigstore.dev/
[16] https://github.com/sigstore/model-transparency

Inventory third-party and open-source code use

- While managing dependencies is a standard challenge in software engineering, the rapid pace of AI development often involves integrating numerous external libraries, potentially increasing exposure to vulnerabilities. To mitigate these risks and enhance the security of the servers, containers, and cloud platforms hosting AI models, a key objective should be to catalog AI infrastructure that uses third-party or open-source code as comprehensively as possible, and integrate it as fully as possible into Vulnerability Management processes.
- Conduct Red Team exercises and continuously monitor for potential security vulnerabilities, such as remote code execution exploits, that could compromise your AI infrastructure.

Sandbox the AI model

The runtime environment where an AI model executes is itself a critical piece of infrastructure that must be secured. This is especially true for agentic models that can take actions, such as executing generated code or interacting with other systems. For these use cases, the model—and any code used to run agents, tools or code generated by the model—must be run within a strictly controlled sandbox environment to contain its actions and limit potential harm. The specific sandboxing technologies to use are a deep and complex topic, but could include anything from virtual machine or hardened container runtime or even separate physical machines.

To safely handle actions initiated by AI models—especially when they involve executing model-generated code or interacting with other systems—run them in a strictly controlled sandbox environment. This environment must enforce clear security policies and limit the AI's capabilities. Grant only the minimum permissions essential for the task to limit the scope of potential impact. In particular, define and block risky or irreversible operations like sending emails or modifying data without explicit user approval.

You can create these secure environments using confidential computing technologies (like Project Oak)[17] for isolation, working alongside secure enclaves designed to protect code and data from the rest of the system. Alternatively, when AI interacts with external systems, enforce strict identity and authorization using methods like OAuth or service accounts with limited scope.

[17] https://github.com/project-oak/oak

While filtering the AI's output (whether using simple regex or complex safety models) can add a layer of safety, the fundamental security assurance comes from the strong limitations imposed by the sandboxed execution environment itself.

Security assurance for agentic integration

The evolution of AI toward agentic systems—applications that allow models to reason about inputs and autonomously use digital tools on the user's behalf—adds another layer to the security paradigm.[18] The agent itself, armed with permissions to act and real-time connections to external tools and APIs, can become a vector for attack, potentially compromising the security and trustworthiness of previously secure applications.

The core challenge lies in the agent's ability to take action. For example, if an AI agent managing a user's email encounters a prompt injection attack involving malicious instructions hidden within the text, the risk goes beyond a faulty summary. The agent could be tricked into executing harmful commands with the user's authority, such as forwarding sensitive information, deleting critical data, or interacting with other applications in unintended ways.

To perform useful tasks, agents often rely on standards, such as the Model Context Protocol (MCP),[19] to access these external capabilities. This integration, however, introduces a significant new attack surface. Protocols that use natural language to describe tools are inherently susceptible to indirect prompt injection, where malicious instructions hidden in a tool's description or output data can be executed by the agent. A compromised or poorly secured tool could be used to exfiltrate sensitive data, perform unauthorized actions, or attack other parts of the infrastructure, making rigorous security assurance for these integrations an essential component of protecting the entire AI system.

Test the integration

To ensure the security of the final product, end-to-end testing of the entire system is critical. While individual components and application layers are often tested in isolation (for example, using unit tests), this approach doesn't guarantee overall system security. This is because complex and often unpredictable interactions between these layers can introduce vulnerabilities that are only apparent when the system operates as a whole.

Therefore, true end-to-end assurance requires testing the AI model's integration into the product, including its agentic interactions with other products and APIs. By adversarially

[18] https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf
[19] https://modelcontextprotocol.io/docs/getting-started/intro

testing these integrations within simulated environments that closely mimic real-world scenarios, we can uncover novel vulnerabilities. This holistic approach provides verifiable evidence that previously mitigated threats remain resolved and that the agentic system is resilient against attacks.

Harden AI integrations with third-party MCP servers

When integrating AI with third-party MCP servers, especially those not fully trusted, it's crucial to mitigate potential security risks:

- Employ an allowlist of the tools exposed by the MCP server; this restricts the AI's capabilities to only approved actions, reducing the attack surface and preventing your users from being exposed to potentially harmful operations.
- Be aware that protocols like MCP (Model Context Protocol) and A2A (Agent-to-Agent), which expose tools and their natural language descriptions, are inherently susceptible to indirect prompt injection attacks. Therefore, all tool descriptions and any data flowing through MCP should be rigorously sanitized and validated to prevent malicious instructions from being interpreted and executed by the AI. The security elements around these types of protocols are still developing, and the industry is actively working on establishing standards to address these vulnerabilities.
- Ensure that the MCP host, client, and server can all verify each other's identities before any sensitive commands or data are exchanged. This prevents unauthorized access and man-in-the-middle attacks, safeguarding the integrity and confidentiality of the interactions.
- Leverage composite identities, which is crucial for hardening integrations. By linking a human's intent with an AI agent's action, these identities ensure granular access control and a clear audit trail, mitigating risk when extending operations to external environments.

A baseline for security assurance

The principles of security assurance—rigorous testing, adversarial thinking, and continuous verification—are not new. However, as this paper has detailed, their application in the age of generative AI represents a critical and necessary evolution. By adapting roles like Red Teaming, Vulnerability Management, and Threat Intelligence to address AI-specific risks—from data poisoning and model evasion to insecure supply chains—we can create a comprehensive defense that protects these systems from the inside out.

AI developers and data scientists can embed a security-first mindset into the development lifecycle, recognizing that security builds the essential foundation for confidence in a model's quality and performance. Security assurance professionals can develop a new literacy in the probabilistic and complex nature of AI, translating traditional security wisdom for this new domain. Securing AI is a cross-functional imperative: the most robust and trustworthy AI products will be built by teams where AI and security expertise are not siloed, but rather work together from the start.

Ultimately, the practices outlined here are about more than just patching vulnerabilities or fulfilling compliance checklists—they are about building a durable foundation of trust. This commitment to trust and responsible deployment echoes the core values found in Google's AI Principles, which guide the creation of beneficial and safe AI systems. A proactive, transparent, and continuous security assurance program is not a constraint on innovation. On the contrary, it provides the confidence to build boldly, deploy responsibly, and deliver on the transformative promise of artificial intelligence. Doing so bridges the gap between initial experimentation and the responsible deployment of AI.

Acknowledgements

We are grateful to the many individuals whose collaboration made this white paper possible. Sincere thanks to the following people for their insightful ideas and constructive feedback: Anton Chuvakin, Daniel Fabian, Nick Galloway, David LaBianca, Adam Stubblefield, Christoph Kern, Rob Mann, Nicholas Capalbo, Jon Brown, Dan Blank, Swetha Balla, Shan Rao, Ruchi Shah, Brice Daniels, and John Stone.