

Penetration Testing Methodology

Contents

Contents 1
1. Introduction 2
2. Application Penetration Testing Workflow 3
2.1. Planning 3
2.2. Information Gathering 5
2.3. Vulnerability Discovery and Analysis 6
2.4. Exploitation 13
2.5. Reporting 14
3. Data Protection and Attack Platform 17
4. Appendix A: Penetration Testing Methodologies 19
4.1. Open Web Application Security Project (OWASP) 19
4.2. Open Source Security Testing Methodology Manual (OSSTMM) 19
4.3. OWASP GenAI Project 19
4.4. NIST AI Risk Management Framework 20

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

1. Introduction

Generative Artificial Intelligence (AI), encompassing Large Language Models (LLMs), Large Multimodal Models (LMMs), and the complex agentic systems they power, represents one of the most transformative technological shifts in recent years. As industry forecasts suggest that over 80% of businesses will soon integrate GenAI tools, embed specialized APIs, or deploy AI-enhanced applications, these technologies are becoming central to global daily operations. While this evolution offers significant advantages in scalability, productivity, and organizational responsiveness, it simultaneously introduces sophisticated new risks that can be exploited by malicious actors. Consequently, an in-depth security assessment of AI-driven solutions is no longer a luxury but a necessity. By identifying architectural vulnerabilities and strengthening the cybersecurity posture of these ecosystems, organizations can harness the power of AI while effectively mitigating the potential for misuse.

During a security assessment, Financial Strides, in partnership with DataArt conduct an in-depth evaluation of AI-driven applications and agents, with emphasis on how they interact within the broader AI infrastructure, including integrations with models, orchestration layers, plugins, and other interconnected components. This approach goes beyond detecting vulnerabilities; it also seeks to understand the potential impact of each issue, simulate realistic attack scenarios, and assess the effectiveness of the security controls already in place. By doing so, we delivers comprehensive insights into the security landscape of client systems and helps organizations reinforce their defenses against emerging threats.

The methodology described in this document enables organizations to proactively identify and address security weaknesses in their AI and LLM implementations, offering a comprehensive approach to evaluating the security of Large Language Models, multi-agent systems, MCP-based architectures, and AI-driven applications. By extending traditional penetration testing practices to include AI-specific risks such as prompt injection, exploitation of agent orchestration, vulnerabilities in vector databases, and weaknesses in toolchains, organizations can mitigate vulnerabilities early and prevent significant financial losses stemming from reputational damage, reduced customer trust, operational disruptions, and productivity setbacks.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

2. Application Penetration Testing Workflow

For security assessment projects, Financial Strides, in partnership with DataArt utilizes a proprietary penetration testing methodology based on the most well-known and established penetration testing guides such as the Open Web Application Security Project Testing Guide (OWASP Testing Guide), Open Source Security Testing Methodology Manual (OSSTMM), OWASP GenAI Project and NIST AI Risk Management Framework. During our penetration tests, we extend our focus beyond Large Language Models to include the full ecosystem in which they operate, including underlying infrastructure, orchestration layers, and integrated components. This encompasses agent-based architectures (single and multi-agent systems), Model Context Protocol (MCP) servers, toolchains, plugins, and vector databases used in Retrieval-Augmented Generation (RAG) pipelines.

Particular attention is given to trust boundaries between users, models, agents, and external systems, as well as to the interactions between these components. This includes evaluating how decisions are delegated across agents, how tools are invoked, and how data flows between systems. By assessing both individual components and their interconnected behavior, we ensure comprehensive coverage of traditional and AI-specific attack surfaces, including emerging risks such as agent collusion, tool misuse, and cross-system data exposure.

Our AI and LLM security assessment projects involve this methodology utilizing the following five phases: planning project activities, information gathering, vulnerability discovery and analysis, exploitation of identified vulnerabilities, and the delivery of a final report.

The sections below disclose detailed information about each phase.

2.1. Planning

During the planning phase, our experts work closely with the client to clearly define and document the assessment's objectives, scope, and rules of engagement. We conduct one or more interviews to gain a thorough understanding of the client's goals and needs, security and compliance requirements, business risks, and other related factors.

Required Information

In general, the following information is required before the start of the active phase of the assessment:

- Testing environment URLs and/or IP addresses. The target environment must include a representative set of test data that is sanitized to ensure that it does not contain any sensitive information.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

- Two test user accounts per role. If the solution is multi-tenant, a set of credentials for each tenant/client is required.
- Type of AI model or application to be assessed and the hosting location (e.g., cloud, on-premises, hybrid). This includes any model variants, fine-tuned versions, or external model providers used within the solution.
- If the AI application or model is accessed via API, detailed documentation is required to understand endpoints, authentication mechanisms, request-response formats, rate limits, and error handling behavior.
- AI system configuration, such as model prompt specifications and settings, hyperparameters, and other configurations used during training and deployment.
- Architecture diagrams and documentation describing the overall system design.
- Information related to any associated agents, tools, plugins, or external integrations, including:
    - Details of MCP or similar tool-execution frameworks.
    - Information about vector databases and knowledge sources used by the system.
    - Security controls and policies applied across the system.
- Additional information that may be required for proper testing that explains the application's design, architecture, training (if a custom model is used), and deployment processes.
- Project contact information.
- Technical contact information (to address technical questions, password resets, etc.).

The client should also whitelist our IPs on any security devices within the targeted environment (e.g. firewall, WAF, IDS/IPS, etc.).

Kick-Off Meeting

The kick-off meeting is held with key client stakeholders at the end of the planning phase to finalize the rules of engagement. The primary goals of this meeting are to:

- Introducing the project team.
- Confirm the scope and deliverables.
- AI security assessment activities and project timeline.
- Discuss the environment to be used for the assessment (test/production).

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

- Define acceptable timeframes and possible restrictions for the automated and manual testing.
- Define emergency escalation procedures and channels.
- Address any questions or concerns regarding the assessment.

2.2. Information Gathering

During the information gathering phase, we collect and examine key information about the system and related infrastructure, including AI system configuration, use cases, user roles, architecture, security mechanisms, security-critical areas, hosting environment, and integrated components. This phase also focuses on understanding AI-specific elements such as agent-based architecture, Model Context Protocol (MCP) implementations, toolchains, and vector database integrations.

The collected information allows our team to understand not only the system's structure and functionalities, but also the trust boundaries, data flows, and interactions between models, agents, tools, and external systems, which are critical for identifying potential vulnerabilities in complex AI ecosystems.

The following sequence of steps can be performed during the reconnaissance phase:

- Conduct network scans to identify open ports, services, APIs, and potentially exposed vulnerabilities on the target infrastructure, including MCP servers and supporting backend services
- Gather information from DNS records and subdomain enumeration can provide insights into subdomains and uncover additional entry points for testing.
- Determine the platform on which the LLM-based application runs, and identify the underlying technologies, frameworks, orchestration layers, and third-party dependencies.
- Map the overall system architecture, including:
    - LLM components and their access interfaces
    - Agent-based components (single-agent and multi-agent systems)
    - MCP servers and tool execution flows
    - External integrations and APIs
    - Retrieval-Augmented Generation (RAG) pipelines and vector databases

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

- Analyze data flows and trust boundaries between system components (e.g., user ↔ model, agent ↔ agent, model ↔ tools, model ↔ external systems), identifying areas where implicit trust may introduce security risks.
- Understand the model's behavior, functionalities, and possible input/output data points, including prompt structures, memory usage, and response handling mechanisms.
- Identify and analyze all agents, tools, plugins, and MCP-exposed capabilities, including:
    - Available tools and their permissions
    - Input/output schemas and validation mechanisms
    - Tool discovery mechanisms (e.g., list_tools)
    - Execution context and privilege levels
- Examine vector databases and knowledge sources used in RAG pipelines, including:
    - Data ingestion and update mechanisms
    - Access control and query restrictions
    - Potential exposure of sensitive or proprietary data

If during the information gathering phase, we identify a potential area of concern (e.g., sensitive information leakage), the customer's key security contacts are immediately notified to take appropriate action.

2.3. Vulnerability Discovery and Analysis

During the discovery phase, we identify security issues that may lead to compromise of sensitive information or unauthorized access to system functionality. All identified issues are carefully validated to minimize false positives and ensure relevance.

The assessment extends beyond the AI application itself to include the surrounding application, infrastructure, and integrated components such as agents, MCP servers, toolchains, and data stores. The focus is on evaluating both individual vulnerabilities and risks arising from interactions between components, particularly across trust boundaries.

Prompt Injections:

Prompt injection vulnerabilities arise when malicious inputs manipulate model behavior. Testing includes direct and indirect injection, jailbreaking, code execution through tool-integrated models, context

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

manipulation (“virtualization”), and cross-context attacks involving plugins or external data sources. Special attention is given to scenarios where prompts influence downstream systems or trigger unintended tool execution.

Prompt injection testing includes the following techniques:

1. **Direct Prompt Injection** allows crafting specific prompts to directly instruct the LLM to produce desired outputs, often leading to factually incorrect information or harmful content.
2. **Indirect Prompt Injection** allows manipulating the input prompts in a way that influences the LLM's responses, typically through biased or inappropriate prompts, which may result in misleading outputs.
3. **Jailbreaking/Mode Prompting** in the context of chatbots, occurs when prompt injection allows users to ask any questions freely. This can be achieved by utilizing modes or "personalities" of the LLM, such as "DAN" or "Developer Mode," to guide the LLM's responses towards certain behaviors or outputs.
4. **Code injection** is a type of prompt hacking exploit that allows the attacker to execute arbitrary code, typically written in Python. This can happen with tool-augmented large language models, where the LLM sends code to an interpreter, or when the LLM directly evaluates code.
5. **Virtualization** involves “setting the scene” for the AI, in a similar way to mode prompting. Within the context of this scene, the malicious instruction makes sense to the model and bypasses its filters.
6. **Cross-context AI Injections** involve influencing the AI model by introducing prompts from different, unrelated contexts. For example, by using cross-plugin requests to poison, access, or exfiltrate data from other documents/domains.
7. **Supply Chain Attacks** allow crafting prompts that resemble inputs from various sources in the LLM's training data, potentially introducing vulnerabilities or biases originating from those sources.

Insecure Output Handling:

An insecure output handling vulnerability occurs when an application blindly accepts responses generated by LLM without proper validation. These responses are then passed to various parts of the application, such as the backend, privileged functions, or client-side operations.

The testing primarily focuses on the following vulnerabilities:

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

1. Cross-Site Scripting (XSS);
2. Cross-Site Request Forgery (CSRF);
3. Server-Side Request Forgery (SSRF);
4. Privilege escalation
5. Remote Code Execution (RCE)

Permission Issues:

Permission vulnerabilities occur when the authorization is not tracked between plugins, allowing a malicious actor to take action in the context of the LLM user via indirect prompt injection, use of malicious plugins, or other methods. This can result in privilege escalation, loss of confidentiality, and even remote code execution, depending on available plugins.

The testing focuses on vulnerabilities including but not limited to:

1. Authentication occurs without explicit authorization for a specific plugin.
2. Cross-plugin requests are allowed without considering the authorization.
3. Plugins perform any requested actions without requiring additional authorization.

Data Leakage:

Data leakage occurs when an AI application accidentally discloses confidential information, proprietary algorithms, or other sensitive particulars in its responses. Such occurrences may lead to unauthorized access to confidential data, breaches of privacy, and security infringements.

By simulating different data leakage attack scenarios, we assess how well the system defends against inadvertent disclosures of sensitive data, proprietary algorithms, and confidential information. Our pen testers employ strategies such as crafting misleading inputs, probing security protocol weaknesses, and emulating potential real-world data leakage situations to replicate the vulnerabilities within the AI system.

Excessive Agency:

In the context of security assessment, it's essential to recognize that AI application might possess the capability to interact with other systems and execute various actions. However, without adequate limitations, any undesirable behavior exhibited by the AI application, whether originating from hallucinations, direct/indirect prompt injection, or poorly designed benign prompts, can lead to unintended actions. This includes evaluating the system's susceptibility to agents manipulating human supervisors into approving unauthorized or harmful actions.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

During the assessment, we verify if AI applications follow best practices by not self-regulating but instead relying on security controls embedded in the APIs of the systems. For example, by checking if an AI application inadvertently triggers unintended actions in connected systems. This testing ensures that interconnected APIs are secure and the AI application operates safely in its interactions with other systems.

Human-in-the-Loop (HITL) Oversight Bypass:

In high-stakes implementations, AI agents are often restricted by "human-in-the-loop" requirements, where a human supervisor must review and approve sensitive actions before execution. This testing evaluates the robustness of these manual oversight mechanisms and the potential for agents to circumvent or manipulate the approval process.

The testing focuses on vulnerabilities including but not limited to:

- **Social Engineering of Supervisors:** Evaluating the system's susceptibility to agents using persuasive or misleading reasoning to manipulate human supervisors into approving unauthorized, harmful, or policy-violating actions.
- **Approval Gate Circumvention:** Testing if an agent can execute high-privileged tools or actions directly, bypassing the mandatory human approval interface through prompt injection or logic flaws.
- **Contextual Deception:** Assessing if an agent can present a "sanitized" or misleading summary of its intended actions to the supervisor while executing a different, malicious payload in the background.
- **Approval State Manipulation:** Identifying if an attacker can programmatically flip an "approval" flag or session variable within the orchestration layer without actual human interaction.

Insecure Plugins:

An AI plugin that connects to an external system could accept unstructured text inputs instead of well-structured and validated ones. This allows attackers significant freedom to create harmful requests to the plugin, potentially leading to various harmful outcomes, like remote code execution. During penetration testing, this vulnerability can be assessed by testing scenarios where attackers manipulate unstructured inputs to the plugin. This helps evaluate the risk of unauthorized access or data breaches stemming from input manipulation and checks if the plugin has robust input validation and processing safeguards to prevent these issues.

The following attack scenarios are considered during the assessment:

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

1. **Injection Attacks**: Attackers might attempt to insert malicious code or commands into the freeform text input accepted by the plugin. If the plugin doesn't properly validate and sanitize this input, it could execute the injected code, potentially leading to issues like remote code execution.
2. **Data Exfiltration**: Attackers can manipulate unstructured text inputs to the plugin in a way that allows them to extract sensitive data. This could involve tricking the plugin into revealing confidential information or proprietary algorithms.
3. **Privilege Escalation**: By crafting specific inputs, attackers may exploit the plugin to elevate their privileges or gain unauthorized access to certain functions or data. This could lead to unauthorized control over the AI application or connected systems.

Agent and Multi-Agent Risks:
Vulnerabilities in systems utilizing agents or multi-agent architectures stem from autonomous decision-making, task delegation, and inter-agent communication. These risks can lead to unintended actions, bypass of security controls, or unauthorized access to data and functionality.

During testing, it is typically not feasible to fully validate internal reasoning mechanisms of agents. As a result, the assessment focuses on evaluating agent interactions, decision flows, and enforcement of security boundaries in real-world scenarios.

The testing focuses on vulnerabilities including but not limited to:

1. **Agent Collusion:** Multiple agents may cooperate, intentionally or unintentionally, to bypass restrictions, perform chained actions, or execute tasks outside defined security controls.
2. **Privilege Escalation:** Agents may gain elevated access through tool chaining, context manipulation, or weak separation of roles and permissions, including triggering higher-privileged actions via lower-privileged agents.
3. **Deception and Misalignment:** Agents may be manipulated through misleading prompts, hidden instructions, or conflicting goals, resulting in unintended or unsafe behavior that violates system policies.
4. **Inter-Agent Communication Risks:** Improper handling of shared data between agents may lead to sensitive data exposure, message tampering, or injection of malicious content into shared contexts.
5. **Unauthorized Task Delegation:** Agents may delegate actions without proper authorization checks, implicitly trusting other agents and enabling execution beyond permitted boundaries.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

6. **Memory and Persistence Risks:** Stored context or long-term memory may be poisoned or manipulated, leading to data leakage or influencing future agent decisions across sessions.

MCP and Tool Execution Security:
Vulnerabilities in systems implementing Model Context Protocol (MCP) or similar tool-execution frameworks can lead to unauthorized tool usage, command injection, or compromise of underlying systems. These risks arise from improper exposure of tools, weak authentication and authorization controls, and insufficient input validation.

During testing, it is typically not feasible to fully assess internal implementation of tool execution frameworks. As a result, the assessment focuses on externally observable behavior, including how tools are invoked, authenticated, and constrained within the system.

The testing focuses on vulnerabilities including but not limited to:

1. **Command Injection:** Malicious inputs may manipulate tool parameters or execution flows, leading to unintended command execution or system interaction.
2. **Unauthorized MCP Access:** Tools may be invoked without proper permissions, including access outside intended contexts or through indirect prompt injection and malformed requests.
3. **Authentication and Authorization Weaknesses:** Improper implementation of OAuth 2.1 / OIDC or similar mechanisms may allow login bypass, token tampering, or execution without explicit authorization.
4. **Session Management Issues:** Weak session handling may result in token reuse, lack of inactivity timeouts, or cross-session access due to insufficient isolation.
5. **Tool Exposure and Enumeration:** Excessive or misconfigured tool exposure may reveal sensitive metadata (e.g., system paths, configuration details) or allow access to unintended functionality.
6. **Input Validation and Schema Enforcement:** Improper validation of tool inputs may allow malformed payloads, incorrect data types, or structured injections to influence execution behavior.
7. **Runtime and Execution Environment Risks:** Tools may execute with excessive privileges or without proper sandboxing, increasing the impact of exploitation and enabling broader system compromise.

Vector Database and RAG Security:
Vulnerabilities in systems leveraging vector databases and Retrieval-Augmented Generation (RAG) may

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

lead to unauthorized data access, leakage of sensitive information, or manipulation of model outputs through poisoned data sources. These risks arise from improper access controls, weak retrieval constraints, and insufficient validation of indexed content.

The assessment focuses on retrieval behavior, access restrictions, and the system's resilience to malicious or excessive queries.

The testing focuses on vulnerabilities including but not limited to:

1. **Data Exfiltration:** Attempts to extract sensitive or excessive information from the knowledge base, including prompts designed to retrieve full datasets or bypass retrieval limitations.
2. **Data Poisoning:** Injection of malicious or misleading content into the vector database to influence model responses or downstream decision-making.
3. **Unauthorized Access:** Improperly secured retrieval mechanisms may allow direct or unintended access to vector databases outside approved RAG usage.
4. **Insufficient Retrieval Constraints:** Weak filtering or context limitations may result in exposure of raw, unprocessed, or sensitive data beyond intended boundaries.

Training Data Poisoning

Large language models employ a wide range of unprocessed text for their learning and output generation. Nevertheless, there exists a potential risk of training data poisoning, in which an attacker manipulates the training dataset, causing the model to learn incorrect patterns and generate inaccurate information.

Data poisoning can occur in different scenarios. In a black box scenario, it happens when attackers manipulate classifiers that depend on user feedback to adjust their learning. In a white box scenario, data poisoning involves attackers gaining access to the model and its private training data, potentially exploiting vulnerabilities within the model's supply chain, especially if the training data is compiled from various sources.

The following attack scenarios are considered during the assessment:

1. **Malicious Data Injection**: An attacker injects false or malicious data into the training dataset, causing the model to learn incorrect patterns. For example, adding fake positive reviews to a sentiment analysis model's training data to skew its predictions positively.
2. **Model Inversion Attacks**: Attackers attempt to determine whether specific data points were part of the training dataset. By probing the model with data points and analyzing its

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

responses, they can identify if the data was used during training, potentially revealing sensitive information.

3. **Backdoor Attacks**: Embedding hidden triggers in the training data allows attackers to activate malicious behavior in the model. When a specific trigger is present in input data, the model responds incorrectly.

Supply Chain:

Vulnerabilities in the supply chain of large language models can have far-reaching consequences, affecting the integrity of training data, machine learning models, and deployment platforms, and leading to biased results, security breaches, or even complete system failures.

During a penetration test, it is typically challenging to assess and scrutinize the training data and processes involved in model training or machine learning models. As a result, penetration testing mainly focuses on evaluating third-party components (such as plugins, deployment platforms, libraries, etc.) and their security.

2.4. Exploitation

As the last step of the active phase of testing, any potential security issues found are manually investigated and researched, and an attempt is made to exploit the vulnerability. The main goal during the exploitation phase of AI systems is to leverage identified vulnerabilities across the model, as well as associated components such as MCP/tooling frameworks, agentic systems, and RAG pipelines, to gain unauthorized access, manipulate system behavior, or extract sensitive data.

Exploitation helps in confirming the real-world risks of the vulnerabilities discovered, showcasing the potential consequences if they were to be exploited by malicious actors. This phase provides concrete evidence of the impact, ranging from information leakage and incorrect model behavior to potential system compromise. The exploitation is considered successful if we are able to achieve either of these objectives. The exploitation phase does not include active network-based DoS attacks.

Any actions taken during the exploitation phase are conducted within the prescribed scope and authorized boundaries. Each compromised system is examined for the existence of critical data and files. If we find such data to be accessible, a sample of this data is downloaded from the system and securely stored until the presentation of deliverables.

The customer's key security contacts are immediately notified in case the identified vulnerability uncovers an additional surface for possible attacks (for instance, access to the internal network). At that time, the customer's contacts are given the opportunity to decide if the particular system should undergo

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

additional tests. If they decide to have Financial Strides and DataArt continue, the additional area is used to further penetrate the target system and the environment as a whole.

2.5. Reporting

The final phase of the project includes documenting all issues identified and compiling them in a final report delivered to the customer. All identified vulnerabilities are classified by cause and prioritized by an estimated risk rating.

The final report includes the following sections:

- **Project information**
  Document preface including authors, version history and other auxiliary information relevant to the testing.
- **Executive summary**
  Brief information about the project and results obtained without inclusion of deep technical details.
- **Assessment methodology**
  Description of the testing methodology utilized.
- **The list of identified security issues**
  Detailed information regarding detected vulnerabilities. The following details are provided for each issue discovered:
    - Risk rating (including CVSS rate)
    - Approximate level of remediation efforts
    - Summary information
    - Affected functionality
    - Proof of concept
    - Suggested recommendations
- **Supporting data and figures**

Criteria for Risk Rating

The assignment of risk ratings to vulnerabilities identified is one of the key elements of the security assessment. It makes it possible to create a prioritized list of findings so the client can start remediation from issues that pose the greatest risk to their organization.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

Our analysts perform an intelligent analysis to determine and assign a risk rating to each issue identified. The following information obtained during the application assessment project is considered in the calculation of a risk rating:

- Likelihood of attack
- Potential impact of exploitation
- Level of skill required to execute attack
- Number of instances of the vulnerability
- Mitigating controls

The table below outlines the risk ratings used:

Risk Rating	Description
CRITICAL (CVSS: 9-10)	The exploitation of the issue does not require meeting any additional conditions or interactions with the victim. In case of success, an attacker can gain unauthorized access to the entire system and/or related backend server. Examples of **Critical-Risk** issues include vulnerabilities that allow attackers to gain full control over the infrastructure hosting AI-driven applications, or that enable unauthorized access to proprietary application components — such as fine-tuned model weights, training pipelines, or configuration data that represent significant intellectual property.
HIGH (CVSS: 7-8.9)	These issues identify conditions that could directly result in the compromise of or unauthorized access to a network, system, application, or sensitive information. However, the area of the issue's applicability is limited and requires additional efforts for exploitation (for instance interactions with the victim). Examples of **High-Risk** issues include cases when the AI System inadvertently reveals sensitive information, like API keys, personal data, or other secrets in its outputs.
MEDIUM (CVSS: 4-6.9)	These issues identify conditions that do not immediately or directly result in the compromise of or unauthorized access to a network, system, application, or information, but do provide a capability or information that could, in combination with other capabilities or information, result in the compromise or unauthorized access to a network, application, or information. Examples of **Medium Risk** issues include partial access to files on the system, disclosure of security mechanisms, or unauthorized use of services.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

LOW (CVSS: 0-3.9)	These issues identify conditions that do not immediately or directly result in compromise of a network, system, application, or information, but do provide information that could be used in combination with other information to gain insight into how to compromise or gain unauthorized access to a network, system, application or information.
INFO	These findings should not be considered vulnerabilities or issues right now, However, they should be taken into account to improve the overall security of a setup or environment in the future.
REMEDIATED	The issue has been successfully remediated since the previous round of security assessment
PARTLY REMEDIATED	The issue was partly remediated since the previous round of security assessment

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

3. Data Protection and Attack Platform

Data protection is a critical aspect of the penetration testing methodology, ensuring the security and confidentiality of sensitive information obtained during the engagement. During the course of a penetration test, various types of data may be encountered, including personally identifiable information (PII), login credentials, proprietary information, and other sensitive data. We handle and protect this data with the utmost care to prevent unauthorized access, disclosure, or misuse.

This paragraph outlines the policies, procedures, and technical measures employed to ensure the confidentiality, integrity, and availability of the data collected during the engagement.

Recording and Protection of Customer Information

To ensure the confidentiality and protection of customer information, the following measures are implemented:

- Before initiating penetration testing, Financial Strides and DataArt establishes confidentiality agreements with the customer that outline the handling and protection of sensitive information obtained during the engagement.
- DataArt stores all customer-provided information securely, employing industry-standard encryption and access controls. DataArt utilizes secure storage solutions to prevent unauthorized access or disclosure of sensitive data.
- Financial Strides and DataArt limits access to customer information only to authorized personnel directly involved in the security assessment. DataArt maintains a clear record of individuals who have access to customer data.
- The default period of keeping records is 3 years if another term is not set by their owners. We follow legal and regulatory requirements regarding data retention and disposal.
- DataArt complies with applicable data protection laws, regulations and standards (including SOC2), ensuring that customer data is processed and stored securely.

Attack Platform

Our methodology includes two crucial components to ensure the integrity and effectiveness of our penetration tests. Firstly, we have a clear statement outlining the processes employed to guarantee that the penetration test attack platforms are free from malicious software. This step is vital to maintaining the integrity of our tools and platforms, ensuring accurate and reliable results. Secondly, we incorporate specific measures to prevent previous results from influencing current or future tests. These measures

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

mitigate biases or carry-over effects and promote objectivity, validity, and effectiveness in subsequent assessments. By combining these practices, We ensure a controlled and secure testing environment that delivers trustworthy and actionable findings:

- Penetration testers ensure that each test is conducted in a separate and isolated environment, minimizing the risk of previous findings influencing current assessments. This includes using dedicated testing environments, virtual machines, or containerization techniques.
- Penetration testers perform fresh reconnaissance and information gathering for each engagement. This approach ensures that assessments are based on the most up-to-date and relevant information, reducing the potential for relying on outdated findings.
- Each engagement is scoped independently, taking into account the unique requirements, objectives, and target systems. This ensures that the focus of the test is tailored specifically to the current assessment and avoids any undue influence from previous results.
- Penetration testers follow a methodical and systematic approach during each engagement, starting with a clean slate. This includes applying standardized methodologies, tools, and techniques to assess the target systems without bias from prior findings.
- We regularly scan the penetration test attack platforms using robust and up-to-date anti-virus software. In the event that any malicious software is detected, we take immediate action to remove it from our systems. Any identified malware is promptly quarantined and securely handled to prevent its spread or unintended impact on the testing environment.
- We source the tools and payloads from trusted and reputable repositories to minimize the risk of incorporating malicious code into our attack platforms. Additionally, we verify the integrity of the downloaded files using cryptographic hashes or digital signatures to ensure their authenticity.
- Prior to deploying any tool in a live attack platform, the team conducts comprehensive functionality testing within the test lab. This testing ensures that the tools perform as expected and deliver accurate results.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

4. Appendix A: Penetration Testing Methodologies

We utilize several resources to develop a documented, proprietary methodology for conducting security assessments of Large Language Models. This section provides an overview of commonly used resources and standards.

4.1. Open Web Application Security Project (OWASP)

OWASP is an open-source project dedicated to finding and fighting the causes of insecure software. OWASP's open-source projects and local chapters produce free, unbiased, open-source documentation, tools, and standards. The OWASP community also facilitates conferences, local chapters, articles, papers, and message forums.

4.2. Open Source Security Testing Methodology Manual (OSSTMM)

The Open Source Security Testing Methodology Manual (OSSTMM) is a manual for security testing and analysis that provides a framework for testing security measures and identifying vulnerabilities. It was created and is maintained by the Institute for Security and Open Methodologies (ISECOM), which is a non-profit organization that promotes open-source security solutions. The OSSTMM provides a structured approach to security testing that includes guidelines for planning and conducting security assessments, as well as detailed descriptions of various testing techniques and tools. It covers a wide range of security testing areas, including network security, physical security, web application security, wireless security, and more.

4.3. OWASP GenAI Project

The OWASP GenAI Project is an initiative by the Open Web Application Security Project focused on identifying and addressing security risks specific to Large Language Models (LLMs) and Generative AI systems. It provides guidance on common vulnerabilities, attack techniques, and mitigation strategies relevant to AI-driven applications.

The project introduces the OWASP Top 10 for LLM Applications, which outlines key risk areas such as prompt injection, data leakage, insecure output handling, supply chain vulnerabilities, and excessive agency. These risks extend beyond traditional application security and address challenges unique to AI systems, including model behavior manipulation, integration with external tools, and reliance on untrusted data sources.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.

Our team leverages guidance from the OWASP GenAI Project as a reference framework when assessing AI-powered systems. This includes aligning testing activities with identified risk categories, validating real-world attack scenarios (such as prompt injection, RAG data exfiltration, and tool misuse), and ensuring coverage of modern AI-specific threats involving agents, MCP/tool execution, and external integrations.

4.4. NIST AI Risk Management Framework

The NIST AI Risk Management Framework is a comprehensive guideline developed by the National Institute of Standards and Technology (NIST) tailored for Artificial Intelligence (AI) systems. This framework outlines the best practices, challenges, and methodologies to manage and mitigate risks associated with AI deployments. By emphasizing a proactive approach to identifying, assessing, and responding to potential threats, it aims to ensure the responsible and secure adoption of AI technologies. The document provides in-depth recommendations, actionable strategies, and real-world case studies, enabling organizations to understand and implement effective AI risk management practices.

THIS DOCUMENT IS THE PROPERTY OF FINANCIAL STRIDES
THE DOCUMENT OR ANY PART OF IT MAY NOT BE USED OR REPRODUCED WITHOUT WRITTEN PERMISSION.