Science Discovery Artificial Intelligence
2026, Vol. 1, No. 2, pp. 78–86
https://doi.org/10.11648/j.sdai.20260102.13

Research Article

Development of Requirements for Ensuring the Security of Artificial Intelligence Technologies

Andrey Shcherbakov[1, 2, *], Anna Shcherbakova[3], Elena Malkova[4]

[1]Department of Cognitive Analytical and Neuro-Applied Technologies, Russian State Social University, Moscow, Russian Federation

[2]Institute of Intelligent Systems, State University of Management, Moscow, Russian Federation

[3]Independent Researcher, Marselle, France

[4]Center for Social Analytics, Russian State Social University, Moscow, Russian Federation

Abstract

This research article proposes comprehensive requirements for securing artificial intelligence systems, focusing on large language models (LLMs) in organizational settings. It addresses risks like unauthorized access, data leakage, service instability, and introduces "veracity" alongside the classic CIA triad (confidentiality, integrity, availability). The paper advocates a two-contour access model: an Open Contour (OC) for public LLMs and an Internal Contour (IC) for corporate/individual models, separated by a gateway for filtered interactions. A mandatory Request Control Module (RCM) monitors all user-LLM exchanges, enforcing limits on request frequency, size, and content to block sensitive data transfers. Secure training mandates dataset cleaning, depersonalization, and documentation, approved by security leads. Key Security Requirements: Corporate LLMs in IC with firewalls and access controls; public ones restricted to OC. Network/Access: Encrypted channels, user authentication, and logging for audits. Availability: Overload protection via RCM limits and reservations. Threat Analysis - drawing from OWASP Top 10 for LLMs (v1.1), it covers prompt injection (LLM01), data poisoning (LLM03), insecure output (LLM02 leading to XSS/SSRF), excess requests (LLM04/06), model theft (LLM10), and info disclosure. The dual-contour setup mitigates many via filtering and separation but notes limits against sophisticated attacks or human errors. Recommends DevSecOps pipelines integrating security across plan, develop, build, test, deploy, and monitor stages (e.g., SAST/DAST, SCA). Tables detail threat coverage, architecture strengths (e.g., centralized monitoring), and weaknesses (e.g., public model opacity). In conclusion, these practical guidelines enhance LLM resilience in enterprises, aligning with NIST AI RMF and ENISA practices, while calling for further automated vulnerability tools.

Keywords

Large Language Model, Artificial Intelligence, Information Security, Access Architecture to the Language Model, Access Contours, Security Policy, Request Control Module

*Correspondence: Andrey Shcherbakov (x509@ras.ru)

Received: 27 February 2026; **Accepted:** 11 March 2026; **Published:** 16 April 2026

1. Preamble

The development of information technologies in the current post-industrial era is characterized primarily by a decline in empathy in public relations served by these technologies. First, these processes lead to more information and services in public networks and national Internet segments becoming paid over time; second, the share of unreliable and low-quality information grows against the backdrop of complete absence of any responsibility (moral or legal) for its generation or provision of poor-quality information services.

The number of malicious, non-profit and profit-driven fraudulent actions by various entities and participants in information exchange is also increasing.

In this context, the mass use of what is commonly called "neural networks" or "artificial intelligence technologies" in public consciousness gives rise to a whole range of problems that go beyond information security and enter the realm of digital hygiene of society.

In reality, when using AI technologies, the user as a subject of information exchange interacts in various forms (through applications or websites) with multimodal language models (LLMs) of various kinds.

In material [1], a fundamental position is formulated that language models presented by modern IT business as "artificial intelligence" are actually not subjective, have no own "self," and communication with them is no more constructive than talking to a car navigator.

Moreover, the core of an LLM from a mathematical and technical viewpoint generates only the most probable text continuation given a known beginning, and it does not matter whether neural network training or formation of a Markov random process transition probability matrix is used for text creation.

The consciously understood essence of a "smart language model" is as follows:

The user has a vague idea of the LLM's operating principle, perceiving it as a kind of "magic chest" from which texts and images appear.

The LLM user does not know or have an idea of how exactly the model is trained, what kind of information (texts) it considers and what it skips, but overall it is clear that the user receives some "global average result."

It is unknown who exactly trains the LLM, what guarantees they provide, and what responsibility they potentially embed in the training process (answer: none).

Thus, most LLMs are continuously evolving (continuously fine-tuned), completely opaque, and controlled by some unknown entity "black boxes" that form human-readable texts and imitate thinking and creativity.

The fact that fragments of some LLMs are presented in open sources as source code does not strengthen trust in them but rather creates an illusion that these solutions can be trusted to any degree. Even publishing neural network weights in open access provides no information about its operation and does not increase trust in the development as a whole.

In this regard, it becomes quite reasonable and obvious for trusted use to consider individual and corporate LLMs, where answers to the three questions formulated above are more or less definite and can be regulated by legislative acts or corporate rules and requirements.

Let us try to slightly change the classic structure of a work describing security requirements.

Of course, we will not forget that an important part of the analysis is formulating the threat model and violator model, but let us try to formulate the specific requirements themselves, important for practitioner specialists for using LLMs in their daily activities so that information systems of banks, insurance and transport companies, restaurants, and marketplaces entrusted to their care do not collapse at one moment due to external negative impact or violation of digital hygiene in LLM use.

Let us add one more important provision and simultaneously attempt to clarify the term "digital hygiene." As already noted, information obtained from LLMs cannot be fully relied upon when solving truly important problems. A striking example is medical LLMs [2]. In any case, use of "products" obtained from LLMs (business plans, projects, forecasts) must be preceded by their expert review and verification of reliability.

Thus, it is necessary to state that the classic triad "confidentiality, integrity, availability" in the case of LLMs is supplemented by "veracity."

2. General Provisions of Specific Requirements

Let us formulate interim (i.e., arising from the current state and understanding of the problem) requirements for ensuring the security of artificial intelligence systems, particularly language models (LLMs), operated by individual users and in organizations.

The proposed recommendations are intended for use in designing, implementing, and operating language models, taking into account risks of unauthorized access, information leakage, service resilience disruptions, and achieving veracity properties in LLM use processes.

As a baseline approach, it is proposed to consider the architecture of access to LLMs as two-contour, including the open contour (OC) and internal (dedicated) contour (IC).

Different contours are understood as sets of technical means, network infrastructure, and organizational measures that primarily separate the more secure corporate environment from external public resources. The key feature of separation into internal and open contours is the difference in the set of technical and organizational protection measures applied to user workplaces (UW) and channels of access to language models.

The internal contour assumes the presence of identification and authentication means, network protection, access control, event monitoring, and regulations for using protection, software, and technical means, while the open contour is oriented toward interaction with external or public resources with limited control.

It is allowed that contours can interact at different levels of information exchange, and their contact point is a gateway—a hardware-software solution implementing rules of interaction between OC and IC.

LLMs themselves, by ownership and placement criteria, are reasonably classified as:

1) Individual (personal) models serving one user or a limited circle of persons;
2) Corporate models under organization management serving its divisions;
3) Public models provided by external suppliers as cloud or public services.

Individual and corporate models are conditionally "placed" on software-hardware complexes localized within the organization's internal contour, allowing application of uniform security and control policies already existing in the organization.

Public language models, on the contrary, communicate with open contour workplaces and are placed on external, usually unknown and uncontrolled third-party resources.

LLMs are classified by training type as follows:

1) Models trained from scratch on source datasets;
2) Fine-tunable models adapted for specific tasks;
3) Pre-trained (ready) models used without additional training or with minimal tuning.

Such division is important for developing secure training requirements, as the volume and nature of processed data, and their modification possibility, substantially depend on the applied training scenario. In particular, training from scratch and fine-tuning require special attention to dataset composition and quality, as well as their preparation and validation procedures.

In the access architecture to LLMs, in accordance with the main provisions of the subject-object model [3], it is reasonable to distinguish a request control module (RCM) that acts as a monitor of all interactions between user workplaces and the language model.

Overall, it can be stated that dividing information systems into two contours develops the concept of an isolated software environment.

The RCM implements the transitive control principle: all information flows from UWs to LLM (requests) and from LLM to UWs (responses or results) pass through it, creating a single observation and management point.

RCM functions include:

1) Accounting and limiting the number of requests to LLM over a certain time interval;
2) Controlling request and response sizes to prevent excessive data transfer;
3) Analyzing appeal content for the presence of confidential, personal, or other sensitive information.

2.1. Architecture Requirements

A secure language model in the context of these requirements must include:

1) A subsystem or regulated process of secure training ensuring control of used data and procedures;
2) A request control module as a mandatory component ensuring monitoring and filtering of requests and responses.

The architecture must provide for delineation of responsibility zones between the model owner, contour infrastructure administrator, and users, as well as audit capability in case of security incidents.

2.2. Hardware Platform Placement Requirements

Individual language models must be placed within the user workplace (UW) or on the organization's resource under its administrative and technical control.

This limits the circle of persons with physical and logical access to resources on which the model operates and allows applying internal information protection policies to them.

Corporate language models must be placed inside the organization perimeter, i.e., in the internal contour, where corporate protection means act: firewalls, intrusion detection systems, backup and recovery means, and centralized access management.

UW placement outside IC is allowed only with additional compensating measures and contractual guarantees from service providers.

Workplaces for accessing public LLMs are placed only in OC and interact with IC UWs only through a gateway, concerning both incoming and outgoing information flows.

2.3. Request Control Module Requirements

The request control module must function as a full monitor through which all requests and responses related to using corporate and individual language models pass.

RCM must ensure:

1) Limiting frequency and number of requests to avoid model overload or resource abuse;
2) Controlling maximum sizes of requests and responses to reduce risk of confidential information leakage and large-data-based attacks;
3) Analyzing appeal content for compliance with internal security policies, including prohibition of transferring confidential and personal data to untrusted environments (to OC).

If necessary, RCM can implement additional functions such as masking or obfuscation of individual request fields, blocking unacceptable request types, and integration with anomaly

detection systems.

2.4. Secure Training Requirements

Training material (dataset) used for training, fine-tuning, or fine adjustment of language models must not contain confidential information, personal data, or information classified as commercial, official, or other secrets protected by law.

Before use, datasets must undergo cleaning and depersonalization procedures excluding elements allowing identification of individuals or disclosure of sensitive organization information.

Data sources, preparation methods, and applied protection mechanisms must be documented to ensure training process reproducibility and subsequent audit possibility in case of incidents related to leakage or incorrect information use.

Dataset preparation and use in LLM training must be regulated by documents approved by the organization's deputy head for security (information security).

2.5. Access and Network Interaction Requirements

The communication channel from workplace to corporate language model must be protected using cryptographic means, authentication, traffic integrity, and, if necessary, additional application-level encryption. This reduces risk of interception, substitution, or modification of requests and responses during transmission.

When accessing public language models (public services) from the internal contour, all requests and responses must pass through a specialized gateway performing filtering, logging, and possible data transformation functions. Such gateway can be implemented based on RCM or as a separate component integrated with the corporate security system separating IC and OC.

2.6. User and Resource Identification and Authentication

When a user works at a workplace in the internal or open contour, they must be uniquely identified and authenticated using the organization's account and access management means. This is necessary for personalizing responsibility, delineating rights, and subsequent action analysis in case of incidents.

The language model (or service providing it) to which requests are sent must also be uniquely identified and authenticated to exclude resource substitution by a violator. For external services, it is advisable to verify remote node authenticity, e.g., via certificate verification and trusted communication channels.

2.7. Event Logging Requirements

All requests to the corporate language model must be registered in the event log, accessible only to the administrator or authorized security service persons. The log should record user identifier, appeal time, operation type, request parameters (in depersonalized or truncated form if necessary for information protection), and processing status.

Response logging from the language model is desirable and can be applied depending on data volume and confidentiality requirements. Storing response fragments allows retrospective incident analysis, response quality assessment, and detection of possible security policy violations by users or applications.

2.8. Availability and Fault Tolerance Requirements

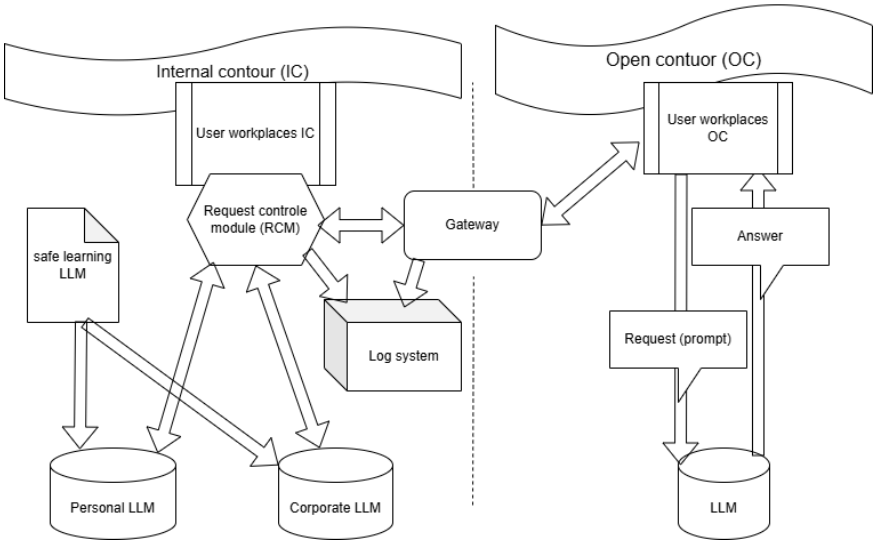

Figure 1. *Architecture of the proposed approaches and it's system components.*

For individual and corporate language models, protection against resource overload must be implemented, which can be carried out by the model's internal means or at the RCM or network infrastructure level. Protection mechanisms may include limiting simultaneous requests, creating processing queues, prioritizing appeals, and automatic resource scaling upon reaching load thresholds.

It is also recommended to provide reservation means and recovery plans to ensure the required service availability level based on language models, especially when used in organization-critical business processes.

The overall architecture of the proposed approaches, system components, and schematic information flows are shown in Figure 1.

Let us now return to examining IS threats for LLMs.

3. Current LLM Security Threats

First, note that threats to the LLM user and threats to the LLM owner differ.

Current information security threats per article [4] for language models include:

1) Prompt injection attack (LLM01);
2) Insecure output handling (LLM02);
3) Training data poisoning (LLM03);
4) Excess request attack (LLM04);
5) Denial of service attack (LLM06);
6) Model theft (LLM10).

We add LLM00 attack for generality—a violator attack unrelated to LLM operation but using the user-LLM interaction channel.

It is obvious, in particular, that LLM10 threat is highly relevant for the model owner but completely unimportant for the user.

These risks are described in detail in the OWASP Top 10 for LLMs document, emphasizing their impact on model and application integrity.

Key threats comprise five classes:

3.1. Prompt Injections (Prompt Injection) – LLM01

A violator (practically from the user side) inputs malicious requests forcing the model to ignore instructions, disclose data, or perform unwanted actions, including bypassing built-in restrictions.

3.2. Data Poisoning – LLM03

Injecting false data into training or fine-tuning process, leading to backdoors, bias, or incorrect model conclusions.

3.3. Insecure Output Handling – LLM02

Model output information without verification causes XSS, SSRF or other attacks if passed to applications without sanitization. Let us clarify these attack types:

1) XSS vulnerability (Cross-Site Scripting)—a web vulnerability type where a violator injects malicious JavaScript code onto a site page executed in the browser, potentially compromising cookie confidentiality, session tokens, or user personal data, bypassing security mechanisms.
2) Types of XSS:
   a) Reflected: malicious code transmitted via URL or form and immediately reflected on the page, requiring link navigation.
   b) Stored: script saved on server (e.g., comments or database) and executed for all page visitors.
   c) DOM-based: vulnerability arises on client side due to unsafe JavaScript data handling.
3) SSRF (Server-Side Request Forgery)—web app vulnerability allowing a violator to force the server to send requests to arbitrary resources, including internal systems, using user input (e.g., URL) without proper verification. SSRF occurs when an application uses user input (e.g., URL) for requests without proper checking, allowing the attacker to control the request target. A violator can send requests on behalf of the server to internal services, cloud metadata, or external resources. The vulnerability leads to data leakage, firewall bypass, internal network access, or even DDoS attacks via server. In OWASP Top 10, SSRF is among critical threats due to high risk.

3.4. Sensitive Information Disclosure

LLM outputs confidential data from training set or user requests. This threat is certainly related to LLM02 but not completely reducible to it.

3.5. Model Denial of Service (LLM03, LLM04)

Overloading model with long prompts, special sequences, or images, causing failures or high costs.

4. Architecture and Key Components

4.1. Dual-contour Separation

Internal Contour (IC):

1) Corporate and individual LLMs are hosted in a protected environment
2) Implements identification, authentication, and network protection measures
3) Deploys firewalls, intrusion detection systems, and backup solutions

4) Provides centralized access management and event monitoring

Open Contour (OC):

1) Used for interaction with external resources and public LLMs
2) Features limited control and minimal protection measures
3) Interacts with IC only through a dedicated gateway

Gateway Between Contours:

1) Hardware-software solution implementing interaction rules between OC and IC
2) Performs filtering, logging, and potential data transformation
3) Separates inbound and outbound information flows

4.2. Request Control Module (RCM)

The RCM acts as a monitor for all interactions between user workstations and the language model:

1) *Request Accounting and Limiting* — Controls the frequency of LLM requests within specified time intervals
2) *Size Control* — Limits maximum request and response sizes to prevent excessive data transfer
3) *Content Analysis* — Checks requests for confidential, personal, or sensitive information [5]
4) *Additional Functions* — Field masking, blocking unacceptable request types, integration with anomaly detection systems

4.3. Secure Training Requirements

1) *Dataset Cleaning* — Exclusion of confidential information, personal data, and trade secrets
2) *Depersonalization* — Procedures preventing individual identification or disclosure of sensitive organizational information
3) *Documentation* — Recording data sources, preparation methods, and applied protection mechanisms
4) *Regulation* — Approval of preparation procedures by the organization's deputy head for security

5. Current Risks According to OWASP Top 10 for LLMs

The article lists modern information security threats for language models [6-15] (relevant for 2025–2026):

Threat Classification

Table 1. *LLM Threat Classification per OWASP.*

Code	Threat Name	Description
LLM00	Channel Interaction Attack	Attacker attack unrelated to LLM operation but using the user-LLM interaction channel
LLM01	Prompt Injection	Malicious requests forcing the model to ignore instructions, disclose data, or perform unwanted actions
LLM02	Insecure Output Handling	Model output information without verification causes XSS, SSRF, RCE attacks when passed to applications without sanitization
LLM03	Training Data Poisoning	Injection of false data into training or fine-tuning process, leading to backdoors, bias, or incorrect conclusions
LLM04	Excess Request Attack	Model overload through excessive requests
LLM06	Denial of Service	Model overload with long prompts, special sequences, or images
LLM10	Model Theft	Model theft (critical for owner, less relevant for user)
—	Sensitive Information Disclosure	LLM outputs confidential data from training set or user requests

6. Critical Attack Details

6.1. XSS (Cross-site Scripting)

1) Web vulnerability where attacker injects malicious JavaScript code onto a site page
2) Types: Reflected, Stored, DOM-based
3) Consequences: compromise of cookies, session tokens, user personal data

6.2. SSRF (Server-side Request Forgery)

1) Web application vulnerability allowing attacker to force server to send requests to arbitrary resources
2) Occurs when using user input (e.g., URL) without proper

verification
3) Consequences: data leakage, firewall bypass, internal network access, DDoS attacks via server

6.3. Risk Coverage by Dual-contour Architecture

Protection Effectiveness Table

Table 2. Dual-Contour Architecture Protection Effectiveness.

Risk	Block Level	Protection Mechanisms	Residual Vulnerabilities
Prompt Injection (LLM01)	Partial	RCM analyzes and filters requests per security policies	No 100% guarantee against sophisticated injections; requires additional analysis
Data Poisoning (LLM03)	High (internal models)	Mandatory dataset cleaning and depersonalization before training in IC	Public OC models remain vulnerable
DoS Excess Requests (LLM04 LLM06)	High	RCM limits on frequency, quantity, and size of requests	Attacks via distributed sources possible
Insecure Output (LLM02)	Partial	Response filtering and logging through RCM	Requires additional sanitization for XSS SSRF RCE protection
Info Disclosure	Medium	Contour separation; confidentiality checks on requests	Public LLMs in OC may leak information
Model Theft (LLM10)	Medium	Corporate model placement in IC with access control	Insider threats persist
LLM00 (Channel Attacks)	Low	Communication channel encryption; authentication	Architecture not specialized for this threat type

6.4. Architecture Strengths

1) *Responsibility Zone Separation* — Clear division between model owner, infrastructure administrator, and users
2) *Centralized Monitoring* — All flows pass through RCM (single observation and management point)
3) *Data Leak Protection* — Control of confidential information transfer between contours
4) *Overload Countermeasures* — Limits on request frequency and volume
5) *Audit and Investigation* — Logging of all requests and responses for retrospective incident analysis
6) *Cryptographic Protection* — Encryption of communication channels from workstation to corporate LLM

6.5. Weaknesses and Limitations

1) *Prompt Injection* — Request filtering cannot guarantee 100% protection against sophisticated injections
2) *Adversarial Attacks* — Architecture does not cover attacks via adversarial examples
3) *Public Models* — LLMs in open contour remain uncontrolled "black boxes"
4) *Human Factor* — User violations of digital hygiene can bypass technical protection measures
5) *Implementation Dependency* — RCM effectiveness depends on implementation quality
6) *Incomplete Output Sanitization* — Protection against XSS/SSRF requires additional measures

7. DevSecOps Methodology for LLMs

DevSecOps pipelines — automated processes and tool chains for creating, testing, and integrating components with security assurance at each stage:

Table 3. DevSecOps Stages for LLMs.

Stage	Security Measures
Plan	Risk assessment; security priority definition; test scenario planning

Stage	Security Measures
Development	SAST (static testing), DAST (dynamic testing), SCA (software composition analysis for known vulnerabilities)
Build	Source code repository protection; artifact control; electronic signature to prevent substitution
Test	Simultaneous functionality and security verification; integration testing, penetration tests
Deploy	Environment management; strict change control procedures
Monitor & Operate	Log collection and processing; health monitoring; user behavior analysis; anomaly detection

Good practice for implementing the formulated requirements can be DevSecOps pipelines—automated human-machine documented and regulated process and tool chains that allow creating, testing, and integrating applications (in this case—workplaces and LLM access infrastructure components) into the production environment, as well as ensuring their security at each stage.

The goal of DevSecOps methodology is to make information and technological security an integral part of the software change (update) process.

DevSecOps pipelines extend DevOps principles: security is integrated into every stage, from planning and code development to testing, deployment, and monitoring.

1) *Plan*—risk assessment and definition of security measures throughout the project lifecycle. Engineers define security priorities and plan testing scenarios.
2) *Development*—implementation of protective practices in the code creation process. Key elements are Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) for early-stage vulnerability analysis. Software Composition Analysis (SCA) is also used to check libraries and modules for known vulnerabilities.
3) *Build*—protection of source code repositories and artifact stores. All development components undergo control, including preliminary security testing and inspections. Electronic signature signing of artifacts is also introduced, preventing file substitution by violators.
4) *Test*—simultaneous functionality and security verification. Various types of checks are integrated—for example, integration testing, performance tests, penetration tests, etc.
5) *Deploy*—special attention to environment management and change control. The deployment process must proceed within strict procedures to minimize risk of introducing unsafe changes.
6) *Monitor and Operate*—continuous observation of application operation and response to emerging events. Log collection and processing systems, health monitoring, user behavior analysis, and anomaly detection are used.

8. Summary

Key points of the formulated requirements are:

1) Two-contour access architecture:

Open contour (OC) is used for interaction with external resources and public access.

Internal contour (IC) provides a higher level of security through internal protection means and controls organization employee work.

2) Language model classification:

Individual (personal) models serve one user or a narrow circle of persons.

Corporate models are managed by the organization and ensure internal network security.

Public models are provided by external providers and require additional protection when interacting with internal contours.

3) Language model types by training principle:

Training from scratch on specific datasets.

Fine-tuning for specific tasks.

Ready models without further training or minimally configurable.

4) Architecture and training process requirements:

Secure model placement on hardware platforms under organization administrative control.

Data quality control and cleaning before use in training process.

Regulation of training procedure and data processing, approved by responsible persons.

5) Request control module (RCM):

Request and response monitoring and filtering.

Limiting frequency and sizes of requests to prevent attacks and excessive data transfers.

Analyzing accessed data for compliance with internal security rules.

6) Minimum necessary information protection measures:

Traffic encryption and user and resource identification.

Event logging and storage of user action data.

Application of reservation mechanisms and recovery plans to maintain service availability.

9. Conclusion

The article systematizes information security (IS) requirements for AI-based systems. Proposed measures ensure comprehensive LLM protection in development, deployment, and

operation stages, minimizing data leak and unauthorized access risks.

Implementation and realization of the formulated requirements enhances AI system resilience to modern cyber threats considering domestic standards and international practices.

Further research can be performed toward developing automated tools for detecting and eliminating vulnerabilities in AI model supply chains, considering threat evolution.

Abbreviations

LLM	Large Language Model
OC	Open Contour
IC	Internal Contour
UW	User Workplace
RCM	Request Control Module
SAST	Static Application Security Testing
SCA	Software Composition Analysis
DAST	Dynamic Application Security Testing
SSRF	Server-Side Request Forgery
XSS	Cross-Site Scripting

Author Contributions

Andrey Shcherbakov: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Methodology, Supervision, Writing – original draft

Anna Shcherbakova: Investigation, Software, Validation, Visualization

Elena Malkova: Investigation, Project administration, Resources, Validation, Writing – review & editing

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

References

[1] Egor Fedorov. Language Model: Dialogue or Monologue? *Herald of Modern Digital Technologies*. 2025. No. 23. pp. 42–66.

[2] O. N. Andreeva, A. V. Domashev, E. A. Evlanova, A. A. Ryazanova, A. Yu. Shcherbakov. Problems of Implementing Large Language Models in Medicine. *Herald of Modern Digital Technologies*. 2025. No. 25. pp. 4–18.

[3] E. V. Vostretsova. *Fundamentals of Information Security: Textbook for University Students*. Yekaterinburg: Ural University Press, 2019. 204 p.

[4] Current Security Threats in Large Language Model Applications. https://habr.com/ru/companies/ru_mts/articles/841010/

[5] A. Yu. Shcherbakov. Methodological Foundations and Prototype of Semantic Artificial Intelligence System. *Scientific and Technical Information. Series 2. Information Processes and Systems*. 2022. No. 9, pp. 1–6. https://doi.org/10.36535/0548-0027-2022-09-1

[6] OWASP Foundation. OWASP Top 10 for Large Language Model Applications v1.1. 2023. https://owasp.org/www-project-top-10-for-large-language-model-applications/

[7] National Institute of Standards and Technology (NIST). Artificial Intelligence Risk Management Framework (AI RMF 1.0). January 2023. https://www.nist.gov/itl/ai-risk-management-framework

[8] European Union Agency for Cybersecurity (ENISA). Multilayer Framework for Good Cybersecurity Practices for AI. December 2023. https://www.enisa.europa.eu/publications/multilayer-framework-for-good-cybersecurity-practices-for-ai

[9] Google DeepMind. Red Teaming Large Language Models: Lessons Learned and Best Practices. *arXiv preprint*. 2024. https://doi.org/arXiv:2401.12345

[10] Microsoft Security Response Center. Securing AI Systems: A Comprehensive Guide to Threat Modeling. February 2024. https://www.microsoft.com/en-us/security/blog/ai-security/

[11] OpenAI. GPT-4 System Card: Safety and Alignment. Technical Report. March 2023. https://cdn.openai.com/papers/gpt-4-system-card.pdf

[12] Anthropic. Claude's Constitutional AI: Harmlessness from AI Feedback. December 2023. https://www.anthropic.com/index/constitutional-ai-harmlessness-from-ai-feedback

[13] UK National Cyber Security Centre (NCSC). Guidelines for Secure AI System Development. January 2024. https://www.ncsc.gov.uk/collection/guidelines-secure-ai-system-development

[14] IEEE Standards Association. IEEE 2089-2024: Standard for Age Appropriate Digital Services Framework Based on the 5 Rights Principles for Children. March 2024. https://standards.ieee.org/ieee/2089/10866/

[15] Zhang, Y., Chen, X., & Wang, L. (2024). Adversarial Attacks on Large Language Models: A Systematic Survey. *ACM Computing Surveys*, 56(3), 1-38. https://doi.org/10.1145/3638237