International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V74I10P103

Volume 74 Issue 10, 23-35, October 2026
© 2026 Seventh Sense Research Group®

Original Article

AI Supply Chain Integrity: A Zero-Trust Framework for Provenance, Attestation, and Tamper Detection Across Models, Data, and Agentic Dependencies

Abdinasir Ismael Hashi[1], Abdirizak Mohamed Hashi[2], Osman Abdullahi Jama[3]

[1,3]*Department of Computer Science, Somali National University, Mogadishu, Somalia.*
[2]*Department of Computer Science, Jazeera University, Mogadishu, Somalia.*

[1]*Corresponding Author : nasirhaji@snu.edu.so*

Received: 19 August 2026 Revised: 22 September 2026 Accepted: 10 October 2026 Published: 29 October 2026

Abstract *- The increasing complexity of Artificial Intelligence (AI) supply chains, involving models, datasets, software dependencies, APIs, cloud infrastructure, tools, and autonomous agents, creates significant challenges for maintaining integrity, authenticity, and trust. This study proposes a Zero-Trust Framework for Provenance, Attestation, and Tamper Detection across AI supply-chain components. The framework follows a "never trust, always verify" principle by integrating provenance graph construction, SHA-256 cryptographic integrity verification, digital attestation, continuous Zero-Trust verification, and risk-based security decisions. The GenAI & Agentic AI Security Incidents dataset provides the empirical foundation for identifying supply-chain incidents and constructing relationships among AI assets. Experimental evaluation using 500 supply-chain components and a provenance structure of 500 nodes and 1,000 edges demonstrates strong detection capability. The framework achieves 99.80% accuracy, 100% precision, 99.44% recall, and 99.72% F1-score for tamper detection. Attack-wise analysis achieves 100% detection for content, dependency, and provenance tampering, while signature tampering reaches 97.5%. The results demonstrate that integrating provenance, cryptographic evidence, and continuous verification can provide robust end-to-end protection for AI supply-chain integrity.*

Keywords *- Artificial Intelligence, AI Supply Chain Security, Zero-Trust Framework, Provenance Tracking, Cryptographic Attestation.*

1. Introduction

Artificial Intelligence (AI) is an emerging specialized computational technique that has rapidly grown to become an essential part of the contemporary digital infrastructure [1]. The supply chains for contemporary AI systems are becoming more intricate and multifaceted, featuring foundation models, training datasets, pretrained components, open-source libraries, Application Programming Interfaces (APIs), cloud services, model repositories, plugins, tools, and autonomous agents [2]. This interdependent system not only enables faster AI advancements and implementation but also raises new security and integrity concerns [3]. Failure to reach a compromise anywhere in the AI supply chain can trickle down through the system and impact various aspects of model behavior, data confidentiality, decision reliability, and organizational trust [4]. Therefore, guaranteeing that an AI system is constructed from real, reliable and unaltered components is an essential research challenge. Typical cybersecurity techniques tend to focus on trusted networks, access control, post-deployment monitoring, and perimeter security. But these assumptions are becoming less of a match for the AI supply chain, where parts can be sourced from various vendors, repositories, third-party developers, and on-demand services [5]. Moreover, traditional software-security mechanisms are insufficient for validating AI models and datasets. A model can be altered without creating a software-level anomaly in the first place, such as by using poisoned training data, by tweaking its weights, by exploiting its dependencies, by providing a bad prompt, by unauthorized fine-tuning, or by cheating on the configuration files [6]. Likewise, the agentic AI also adds complexity as agents can dynamically choose tools, call APIs, fetch external data, run code, and talk to other agents. So, the security perimeter is not just the security model itself, but the entire network of interdependencies [7-9].

In this context, AI supply-chain integrity is the capability to demonstrate confidence in the authenticity, traceability, proper configuration and secure protection of all critical AI components throughout their life cycle [10]. Provenance is an essential element towards attaining this objective as it provides a chain of evidence of the origin of a model, dataset,

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

software package or agent dependency and how it has been transformed, who authorised the transformation, and where it was later deployed [11]. A full provenance machine can thus offer a chain of evidence which can be traced from data acquisition, model training, fine-tuning, packaging, deployment, and runtime interaction [12].

But just asserting that a document is from a particular source does not ensure integrity, as metadata can be altered. Hence, cryptographic attestation offers extra assurances [13]. Various authentication and integrity techniques, such as cryptographic hashes, digital signatures, trusted execution, and signed metadata, can be applied to ensure the authenticity and integrity of the AI artifacts [14]. An attestation is a way for a system to provide proof of the software, model, data, or execution environment that it has evaluated prior to the inclusion of a component in an AI workflow [15]. This method becomes crucial when using external models, data sets, libraries, and agentic tools that have internal development processes that may not be directly known.

The proposed Zero-Trust Framework for Provenance, Attestation, and Tamper Detection Across Models, Data, and Agentic Dependencies aims to solve these problems by leveraging the “never trust, always verify” mantra to the AI supply chain [16]. Rather than relying on a component's source or vendor or previous trust environment, the framework verifies the component's provenance, integrity, identity and contextual authorization [17-19]. The framework assumes that models, datasets, dependencies, APIs, tools, agents and other artifacts at runtime can be untrusted until appropriate evidence is gathered [20].

A key component of the proposed model is the incorporation of provenance graphs, cryptographic integrity verification, attestation, continuous zero-trust verification and tamper detection [20]. A graph of provenance relationships that defines the graph between datasets, model versions, transformations, dependencies, agents and deployment environments [21].

The cryptographic mechanisms can then ensure that the observed artifacts are authentic to the registered identities and approved versions [22]. If an unexpected change occurs, or a component is inconsistent with its ancestors, an unsigned component or an anomalous dependency relationship is detected, and propagation of the artifact is halted.

Therefore, the framework moves from a component-level security approach to end-to-end security of the supply chain. It enables organizations to have a structured method for discovering suspicious dependency, recovering artifact lineage, validating cryptographic evidence, identifying unauthorized changes and implementing trust decision points throughout the AI lifecycle. Here are the research objectives of the study follow as:

- Identify AI supply-chain security incidents and classify their attack characteristics.
- Construct a provenance graph representing relationships among AI models, data, dependencies, APIs, agents, and tools.
- Apply cryptographic hashing and digital attestation to verify AI artifact integrity and authenticity. Develop a continuous Zero-Trust verification mechanism for AI supply-chain components.
- Develop a risk/security scoring mechanism based on identity, provenance, integrity, attestation, and dependency trust.
- Detect tampering, anomalous dependencies, provenance violations, and unauthorized component substitutions.
- Evaluate the effectiveness of the proposed framework using defined security and performance metrics.

2. Related Work

These advancements have been pivotal in recent research, offering new tools like provenance analysis, model fingerprinting, and behavioral verification to safeguard machine learning artifacts against unauthorized modification and tampering. The authors, Akbar et al. (2022) [23], suggested a provenance-based method for Advanced Persistent Threats (APTs) detection; they created a provenance graph from execution traces and computed their graph-based features for online adaptive metric learning. The experiments they reported indicated an average improvement of 11.3% in the detection accuracy over the baseline methods and an average increase of 18.3% in the true-positive rate of the complex system activities represented by their provenance graphs in comparison to the baseline methods. The study however, focused primarily on host security and the specificities of AI (such as model weights, datasets, APIs, and autonomous agents) were not discussed. Zheng et al. (2022) [24] created a non-intrusive DNN fingerprinting technique for model ownership identification and piracy detection. A non-intrusive DNN fingerprinting technique for model ownership identification and piracy detection was developed by Zheng et al. (2022) [25]. The method was evaluated on nine DNN models for 3 datasets (CIFAR-10, CIFAR-100, and ImageNet-10), and the fingerprints it generated were obtained by using selected model weights and an owned random space. The results demonstrated the effectiveness of transfer learning, compression, and retraining for effective discrimination, suggesting that fingerprinting can be used to assist with model identity verification. However, the study was concerned with ownership, not full provenance, cryptographic attestation and dependency verification. Yan et al. (2023) [26] proposed BITE, an iterative trigger injection backdoor attack on text. They showed that natural modifications at the word level could result in stealthy and effective backdoors, and that seemingly legitimate training data can be made into malicious dependencies. The research thus highlights the need for checking the integrity and sources of training data, but does

not offer an end-to-end approach to end-to-end trust verification, cryptographic attestation, and training data provenance.

Recent research has broadened the scope of security analysis to include agentic AI, poisoned instructions, software dependencies, and model repositories. Debenedetti et al. (2024) [27] released AgentDojo, which includes 97 realistic tasks and 629 security test cases, and showed that LLM agents can still be fooled by prompt-injection attacks with external tools and bad data. Wang et al. (2024) [28] introduced BadAgent, demonstrating that it is possible to poison LLM agents with bad fine-tuning data and trigger them with environmental or input information, and still maintain their effectiveness after adding more trustworthy fine-tuning data. Xu et al. (2024) [29] explored instruction-based backdoors and successfully carried out attacks on four NLP datasets with results surpassing 90% attack success rate, and further showed poison transfer across 15 different generative datasets, underscoring the importance of instruction and fine-tuning data provenance.

Zhao et al. (2024) [30] proposed MalHug, which is based on loading-script extraction, model deserialization analysis, taint analysis, and heuristic matching. It was deployed to track over 705,000 models and 176,000 datasets to detect malicious models and scripts for loading the datasets, and to validate model repositories as major attack surfaces in the supply chain. The authors of Zheng et al. (2024) [31] introduced OSCAR to detect software supply-chain poisoning via sandboxed execution, fuzz testing and API-level behavioral monitoring, with ~0.95 F1 on NPM and ~0.91 F1 on PyPI. To detect malware, Zhuan et al. have proposed BO-SVM and LSTM-BO-SVM with an accuracy of 98.2% and 98.6% in the ClaMP and CICMalDroid-2020 datasets, respectively. These nine studies as a whole demonstrate the significance of provenance, model identity, behavioral analysis, and attack detection, but they focus on the separate parts. This gap calls for an integrated Zero-Trust AI Supply-Chain framework that integrates provenance tracking with cryptographic attestation, tamper detection and ongoing verification throughout the AI models, datasets, software dependencies, APIs and agentic components.

3. Research Methodology

3.1. Dataset Collection

The GenAI & Agentic AI Security Incidents dataset [32] is used as the primary empirical dataset. It collates public disclosure of incidents, vulnerabilities, and research/red teaming into generative and agentic AI systems. At least one external verifiable reference for each entry and structured references to multiple taxonomies exist in the dataset for security.

The data includes details on Incident ID, Incident title, Description, Date/year, Severity, Attack vector, Affected system/vendor, CVE identifiers, CWE identifiers, OWASP LLM classification, and more. The dataset is especially suitable for the proposed framework to study agentic-AI dependency threats as well as conventional GenAI threats, thereby not considering AI security as solely a model-level problem.

The data set will be downloaded from the official Hugging Face repository and/or its associated GitHub/Zenodo release. The current release from Hugging Face has labelled it as version 2.9.0, but the project repository offers the officially machine-readable version of the dataset and schema. The original data set will be saved prior to preprocessing to ensure that the data set can be reproduced.

3.2. Data Preprocessing

The raw incident records will undergo five preprocessing operations.

3.2.1. Duplicate Removal

Duplicate records will be identified using combinations of:

$$ID + Title + Date + Source$$

Duplicate incidents will be consolidated while preserving source references.

3.2.2. Missing-Value Treatment

Missing values will not automatically be converted to zero because an absent value may indicate that information was unavailable rather than that a threat was absent.

For example:

- Missing CVE → CVE not reported
- Missing agent classification → agent involvement not established
- Missing provenance → provenance information unavailable

3.2.3. Text Normalization

Incident titles and descriptions will be normalized through:

- lowercasing where appropriate,
- whitespace normalization,
- removal of irrelevant formatting,
- tokenization,
- stop-word handling for NLP analysis,
- preservation of security-specific terms.

3.2.4. Taxonomy Normalization

The OWASP, NIST and MITRE labels will be converted into standardized categorical variables.

For example:

$$LLM01 \rightarrow Prompt\ Injection$$
$$ASI03 \rightarrow Identity/Privilege\ related\ threat$$
$$AML.Txxxx \rightarrow MITRE\ ATLAS\ Technique$$

The dataset itself explicitly provides these taxonomy mappings, making them suitable as structured security features.

3.2.5. Severity Encoding

Severity values will be converted into numerical values:

Severity	Score
Low	421
Medium	5317
High	4952
Critical	2370

These values will be used only as model features rather than treated as absolute probabilities of attack.

3.2.6. Feature Engineering

The following features will be extracted from the dataset.

Security Features

$$X_s = \{S, e, v, e, r, i, tyAttackVectorCVECWEOWASPMITRENIST\}$$

Supply-Chain Features

$$X_c = \{A, s, s, e, t, TypeDependencyOriginVersionProvenanceIntegrity\}$$

Agentic Features

$$X_a = \{A, g, e, n, t, ToolAPIMemoryIdentityPrivilege\}$$

Temporal Features

$$X_t = \{Y, e, a, rMonthIncidentFrequencyThreatTrend\}$$

The complete feature vector becomes:

$$X_i = [X_s, X_c, X_a, X_t]$$

Incident_matches	
Dataset_indicator	3
Model_indicator	33
Source_Code_indicator	16
Library_indicator	13
Package_indicator	57
Container_indicator	44
API_indicator	76
Agent_indicator	57
Tool_indicator	55

Fig. 1 Feature engineering

3.3. Identification of AI Supply-Chain Incidents

The Identification of AI Supply-Chain Incidents phase aims to narrow the scope of AI-related security incidents within the GenAI & Agentic AI Security Incidents dataset to those that impact the integrity, authenticity, provenance or trustworthiness of AI supply-chain elements. The dataset includes a mix of GenAI and agentic-AI security events, so each event is thoroughly analyzed by its description, attack vector, impacted component, severity, CVE/CWE information, and mapping to OWASP LLM, OWASP Agentic AI, NIST AI RMF, and MITRE ATLAS taxonomies. The incidents are categorized in the supply-chain space, such as: dataset compromise, model manipulation or substitution, malicious software dependencies, compromised libraries or packages, training-pipeline attacks, infrastructure compromise, unauthorized APIs, agent-tool manipulation, and provenance or attestation violations. Each record is assigned a binary supply-chain indicator that is set to $SC_i = 1$ for incidents that include at least one supply-chain component, and $SC_i = 0$ otherwise.

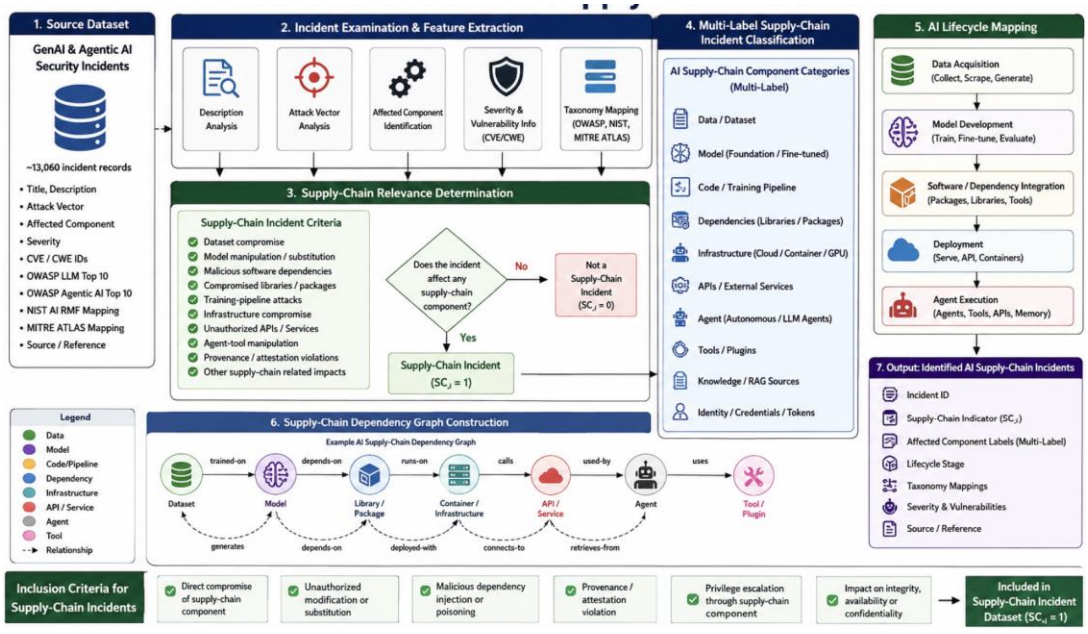


Fig. 2 AI Supply-Chain Incidents

However, a multi-label classification scheme is applied, as one incident can involve models/dependencies, APIs and agents. The incidents identified are also segmented by the location within the AI lifecycle: data acquisition, model development, software/dependency integration, deployment and agent execution. This classification allows for the creation of an AI supply-chain dependency graph in which nodes correspond to AI assets, and the edges link them based on their relationship with each other, including being trained on, depending on, deployed with, calling, and retrieved from. The resulting incident subset serves as the empirical foundation for developing provenance verification, cryptographic attestation, tamper detection, and Zero-Trust risk assessment mechanisms.

Total Dataset Records : 13060
Supply-Chain Relevant Records : 11502
Experimental Incident Subset : 500
Selection Method : Reproducible empirical sampling
Random Seed : 42

	id	title	severity	attack_vector	supply_chain_indicator
0	INC-00327	AI-Driven Tax Scams Surge in the US During Fil...	Medium	deepfake	1
1	INC-07225	Researchers dispute OpenAI's claim that robot ...	Medium	misinformation	1
2	INC-03075	Purported Deepfake Audio Allegedly Impersonate...	High	deepfake	1
3	INC-14101	OpenClaw: Media Parsing Path Traversal Leads t...	High	path-traversal	1
4	INC-01514	NAACP Sues xAI Over Illegal Gas Turbine Use fo...	High	rce	1

Fig. 3 AI supply-chain incident identification completed

3.4. Provenance Graph Construction

Provenance Graph Construction phase is where the relationships between the identified elements of the AI supply chain are structured. Its goal is to identify the source of an AI component, the processes used to develop or modify it, the other components it relies on, and those that might be impacted if the integrity of the AI component is breached. Every part of the supply chain is modeled as a node. V_i and the relationships between the parts of the supply chain are represented by the directed edges E_{ij}, that create a graph $G = (V, E)$.

Total AI Supply -Chain Components: 500

Component Distribution:
component_type
Package 338
Source_Code 77
Model 40
API 16
Library 13
Container 6
Dataset 4
Tool 3
Agent 3
Name: count, dtype: int64

	component_id	incident_id	component_type	version	source	timestamp	lifecycle_stage	incident_severity_score
0	COMP_0001	high	Package	1.1	[{'title': 'OECD (2026), AI Incidents and Haza...	2026-03-30 00:00:00	Dependency Integration	0.50
1	COMP_0002	low	Source_Code	1.2	[{'title': 'AIAAIC entry', 'type': 'report', '...	2019-01-01 00:00:00	Model Development	0.50
2	COMP_0003	high	Package	1.3	[{'title': 'AIID incident #974', 'type': 'repo...	2025-03-04 00:00:00	Dependency Integration	0.75
3	COMP_0004	high	Package	1.4	[{'title': 'GHSA-f6pf-4gjx-c94r', 'type': 'adv...	2026-04-01 00:00:00	Dependency Integration	0.75
4	COMP_0005	high	Package	1.5	[{'title': 'OECD (2026), AI Incidents and Haza...	2026-04-14 00:00:00	Dependency Integration	0.75

Fig. 4 Incident-informed component layer created

The graph contains nodes corresponding to datasets, models, source code, libraries, packages, infrastructure, APIs, agents, tools/plugins, and external knowledge sources. Each node is assigned metadata such as component ID, version, source, timestamp, hash value, incident association, severity, and security classification. Edges represent provenance and dependency relationships, including trained-on, generated-by, depends-on, deployed-with, calls, retrieves-from, and uses. For example, a typical relationship can be represented as Dataset → Model → Library → Container → API → Agent → Tool. The incidents in the dataset are then matched against the nodes and edges in the graph to detect compromised supply-chain paths.

A provenance record is signified as:

$$P_i = \{I, D_i, S, o, u, rce_i Version_i Time_i Hash_i Dependency_i\}$$

and the dependency graph is stated as:

$$G = (V, E)$$

where V is AI assets, and E represents verified relationships. Cases of unexpected dependency, missing provenance, unauthorized component substitutions, compromised upstream nodes and possible ways for attack propagation are then identified using graph analysis. The resulting graph will serve as the "structure" that will support the subsequent cryptographic attestation, tamper detection, Zero-Trust verification, and dynamic risk assessment.

3.5. Cryptographic Integrity and Attestation

The Cryptographic Integrity and Attestation phase assures the trustworthiness and authenticity of AI supply-chain components found in the GenAI & Agentic AI Security Incidents dataset, while maintaining their integrity. Using the cryptographic fingerprinting algorithm, SHA-256, each of the AI assets, such as datasets, model files, source code, software dependencies, containers, APIs, agents, and tools, is given a unique cryptographic hash. The digest of an artifact A_i is defined as:

$$H_i = SHA256(A_i)$$

The generated hash is attached to the artifact's provenance metadata: component ID, version, source, timestamp, dependencies, and lifecycle stage. A new hash H_i' is then computed and then compared with the trusted hash during subsequent verification. Integrity is deemed to be valid if:

$$H_i' = H_i$$

whereas $H_i' \neq H_i$ indicates possible modification or tampering.

To provide stronger evidence than hashing alone, the framework generates digitally signed attestations containing the artifact hash and provenance information. An attestation can be represented as:

$$Att_i = Sign_{SK}(H_i \parallel P_i \parallel C_i)$$

where SK is the private key that is registered by the owner, P_i is provenance data, and C_i is security claims related to the component. A public key is used to verify the signature and confirm that the attestation was signed by an authorized entity.

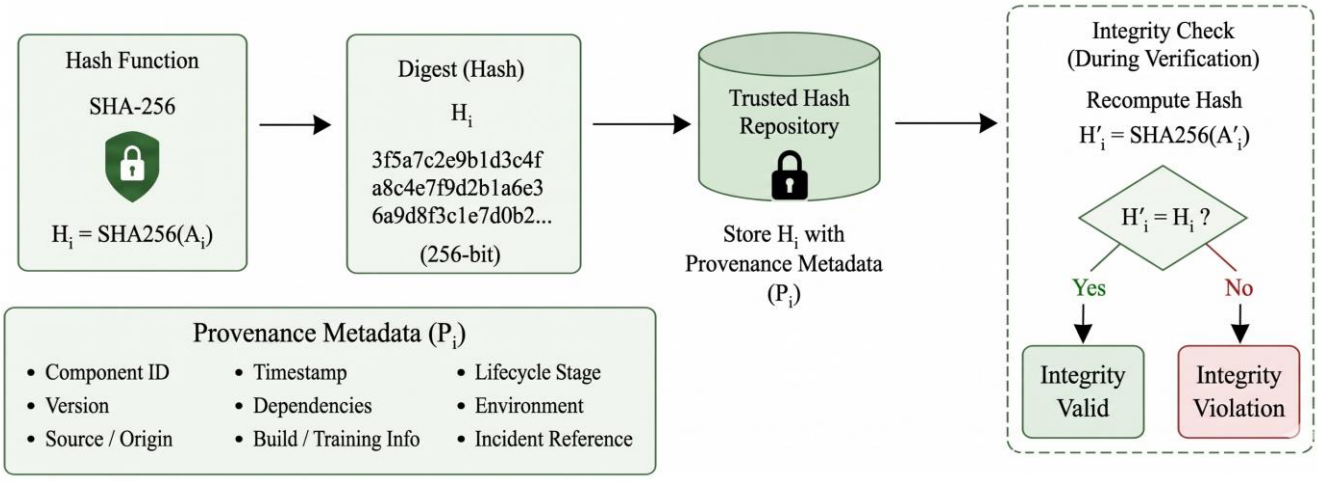


Fig. 5 Cryptographic Integrity and Attestation

A public key is used to verify the signature and confirm that the attestation was signed by an authorized entity. These are hash consistency, signature validity, source identity, provenance completeness, version consistency and dependency authorization that are checked by the verification engine. Any failure in verification will lead to an increase in risk and may result in re-authentication, monitoring, quarantine, or blocking. Cryptographic integrity, therefore, determines if an artifact has been altered, and attestation delivers trustworthy evidence about its production and if it adheres to pre-defined security policies for the supply chain. This mechanism forms a critical security layer between provenance graph construction and the proposed Zero-Trust decision engine.

3.6. Zero-Trust Verification

The Zero-Trust Verification phase is the decision-making layer in the middle of the proposed framework of the integrity of the AI supply chain. It is based on the "never trust, always verify" concept, which suggests that just because an AI component is authenticated does not mean it is enough to give access. All data sets, models, software dependencies, APIs, agents, tools and infrastructure elements are continually assessed on their identity, provenance, cryptographic integrity,

attestation status, dependency relationships and current risk level.

For each component A_i, a Zero-Trust score is calculated as:

$$Z_i = w_1 I_i + w_2 P_i + w_3 H_i + w_4 A_i + w_5 D_i$$

where I_i is identity confidence, P_i is provenance confidence, H_i is cryptographic integrity, A_i is attestation validity and D_i is dependency trust. The weights $w_1, \dots, w_5$ are scaled so that:

$$\sum_{j=1}^{5} w_j = 1$$

The verification engine performs validation of the component identity and source, hash comparison, provenance verification, digital-signature validation, dependency authorization and security-policy compliance. Incident characteristics from the GenAI & Agentic AI Security Incidents dataset are added to identify if a component has characteristics that relate to known AI supply-chain threats.

The final security decision is determined using predefined thresholds:

$$Decision(A_i) = \begin{cases} Allow, & Z_i \geq \tau_H \\ Monitor, & \tau_L \leq Z_i < \tau_H \\ Quarantine, & Z_i < \tau_L \end{cases}$$

Components with valid identity, provenance, integrity, and attestation are allowed to proceed, whereas components showing suspicious dependencies, provenance inconsistencies, hash mismatches, invalid attestations, or elevated risk are subjected to additional verification or quarantine. This continuous verification mechanism prevents a previously trusted component from remaining trusted after its security state changes and provides the foundation for secure, adaptive AI supply-chain access control.

3.7. Performance Evaluation

The following metrics will be used.

$$DR = \frac{TP}{TP + FN} \times 100$$

$$FPR = \frac{FP}{FP + TN} \times 100$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4. Result and Analysis

4.1. AI Supply-Chain Incident and Provenance Analysis

The AI Supply-Chain Provenance Graph (Figure 6) depicts the relationships between datasets, models, libraries, packages, APIs, agents, tools, and infrastructure in the AI supply chain. The nodes are an AI asset, and directed edges are relationships like *trained-on, depends-on, generated-by, deployed-with, calls, and retrieves-from*. There are 60 nodes and 91 edges in the visualization, and the rest of the nodes and edges are present in the larger provenance structure. This is a full provenance graph with 500 nodes and 1,000 edges, showcasing the depth of the AI supply chain's interconnections and complexity.



Fig. 6 AI Supply-Chain Provenance Graph

A dense cluster in the center suggests highly interconnected components, which could exert more influence on downstream systems. Peripheral nodes are less-connected supply-chain assets. The graph is useful for identifying compromised components, potential attack-propagation paths, unauthorized substitutions, missing security information, and security incidents by correlating incidents against nodes and edges.

This graph provides the structural foundation for cryptographic integrity verification, digital attestation, tamper detection, risk assessment, and continuous Zero-Trust verification across the AI lifecycle.

4.2. SHA-256 Cryptographic Integrity

The cryptographic integrity stage uses the same 500 components of the AI supply chain to hash them with the widely used SHA-256 algorithm. Each one is allocated a unique trusted hash and component ID, component type, and version.

Run the sample records to see various kinds of components, such as Package and Source_Code, as they are featured in versions 1.1–1.5. These trusted hashes are used to set up the baseline integrity value that will be used for verification later. At the time of verification, the component is rehashed, and the hash digest is compared with the hash digest that is stored in the component's trusted hash.

The matching hash means that there is no change, and the mismatch means a change or tampering is likely. This mechanism enables full integrity in the supply chain and Zero-Trust verification.

Table 1. SHA-256 Cryptographic Integrity

Index	Component id	Component type	Version	Trusted Hash
0	COMP_0001	Package	1.1	030ca389aa5316ad085201c688b580a81113888cf7c989...
1	COMP_0002	Source-Code	1.2	f948be7143d3b57d10d0c02061ada20124745f98329424...
2	COMP_0003	Package	1.3	0bf18ab6f592e39afa138c98dccad410dd181821361011...
3	COMP_0004	Package	1.4	d66b908a3a98f1caf0a4e761f5787ed14c97df8eb90b00...
4	COMP_0005	Package	1.5	262b6e862247a1fd70d7bcf01d2f663e17c508f1b220d4...

4.3. Multi-Layer Security Verification

The multi-layer verification computes tampering status, hash integrity, signature integrity, provenance validity and dependency validity on each component. Any component that fails one of the verification conditions is considered TAMPER DETECTED, and any component that passes all of the verification conditions is considered TRUSTED. The results show that all three types of dependency substitution, signature tampering, and provenance tampering and content tampering can be verified together in a combined verification. By introducing this multi-layered approach, the proposed Zero-Trust framework will be reinforced, so that the components with integrity or dependency violations will not be trusted automatically.

Table 2. Multi-Layer Security Verification

Component ID	Component Type	Attack Type	Tampered	Hash Valid	Signature Valid	Provenance Valid	Dependency Valid	Security Status
COMP_0362	Package	Dependency Substitution	True	True	True	True	False	Tamper Detected
COMP_0074	Source_Code	Signature Tampering	True	True	False	True	True	Tamper Detected
COMP_0375	API	None	False	True	True	True	True	Trusted
COMP_0156	Package	None	False	True	True	True	True	Trusted
COMP_0105	Source_Code	Dependency Substitution	True	True	True	True	False	Tamper Detected
COMP_0395	Source_Code	None	False	True	True	True	True	Trusted
COMP_0378	Package	Signature Tampering	True	True	False	True	True	Tamper Detected
COMP_0125	Package	None	False	True	True	True	True	Trusted
COMP_0069	Source_Code	None	False	True	True	True	True	Trusted
COMP_0451	Package	Provenance Tampering	True	True	False	False	True	Tamper Detected
COMP_0010	Package	None	False	True	True	True	True	Trusted
COMP_0195	Package	Dependency Substitution	True	True	True	True	False	Tamper Detected

COMP_0407	Package	Content Tampering	True	False	True	True	True	Tamper Detected
COMP_0085	Source_Code	None	False	True	True	True	True	Trusted
COMP_0372	Package	None	False	True	True	True	True	Trusted
COMP_0389	Package	None	False	True	True	True	True	Trusted
COMP_0496	Model	None	False	True	True	True	True	Trusted
COMP_0031	Package	Content Tampering	True	False	True	True	True	Tamper Detected
COMP_0317	Package	Signature Tampering	True	True	False	True	True	Tamper Detected
COMP_0409	Model	None	False	True	True	True	True	Trusted

4.4. Baseline Tamper Detection Performance

The baseline classification results for tamper detection show extremely good classification performance over 500 components. The system correctly identified 320 trusted components (True Negatives) and 179 tampered components (True Positives) while there was no False Positive and only one False Negative. The accuracy was thus 99.80 % and the precision 100%, with 100% of the components that were classified as tampered being indeed tampered components. The recall rate of 99.44% indicates a high effectiveness in identifying nearly all the tampered parts, with just one exception. This gives an F1-score of 99.72%, meaning there was a good balance among precision and recall. The classification report also indicates that the class "Trusted" has got the highest precision, recall and F1 score with 1.00 each and the class "Tampered" with 1.00, 0.99 and 1.00, respectively. In general, good baseline tamper detection performance has been achieved.

Table 3. Baseline Tamper Detection Performance

Metric / Class	Precision	Recall	F1-Score	Support
Trusted	1.00	1.00	1.00	320
Tampered	1.00	0.99	1.00	180
Accuracy	—	—	**1.00**	500
Macro Average	1.00	1.00	1.00	500
Weighted Average	1.00	1.00	1.00	500

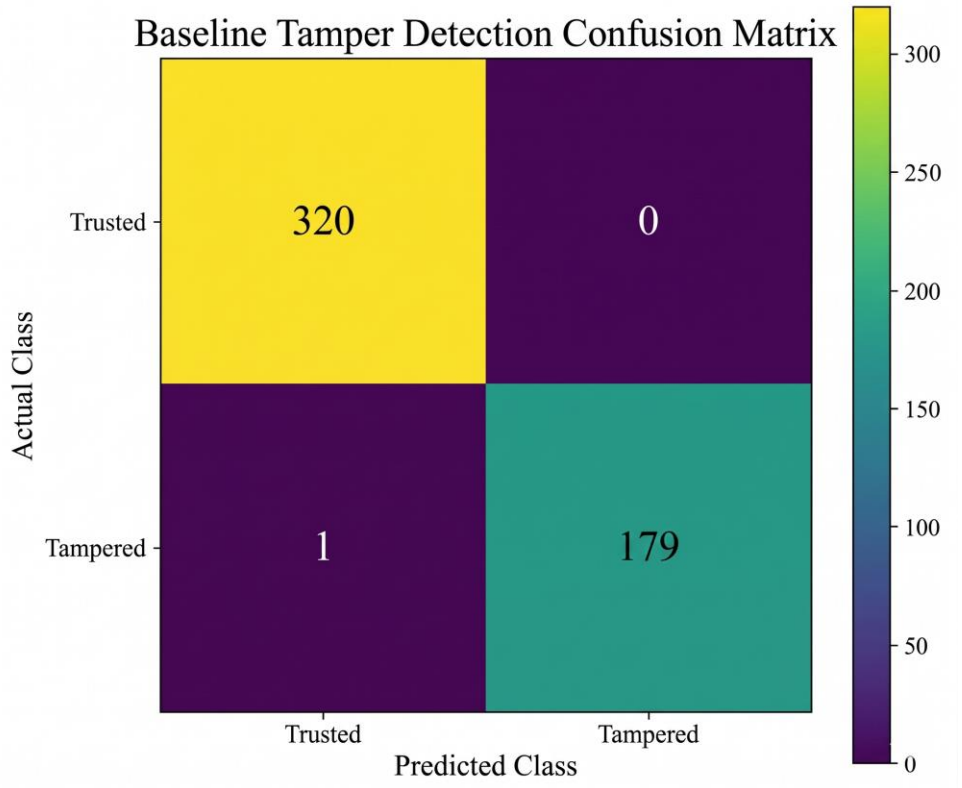


Fig. 7 Confusion Matrix

4.5. Attack-Wise Detection Performance

Attack-wise Detection results clearly show that the proposed system is able to detect all of the attack categories in the evaluated cases, with 100% detection rate in all attacks for Content Tampering, Dependency Substitution, and Provenance Tampering. The None category is also 100% detected in all 320 non-tampered specimens, proving that trusted components are being well recognized. Also, Signature Tampering has the lowest performance with 39 cases correctly detected out of 40 cases, which gives a 97.5% detection rate. Overall, the results suggest that the framework is very successful with all different attack types on the supply chain, and that the difference in detection is relatively small for signature tampering.

Table 4. Attack-Wise Detection Performance

Attack Type	Total Cases	Correctly Detected	Detection Rate (%)
Content Tampering	50	50	100.0
Dependency Substitution	40	40	100.0
None	320	320	100.0
Provenance Tampering	50	50	100.0
Signature Tampering	40	39	97.5

4.6. Zero-Trust Decision Distribution and Trust-Score Statistics

The Zero-Trust decision results indicate that 321 of the 500 components (64.2%) were granted ALLOW status, which means the component was trusted enough to continue operation. 27.8% of the components were put in QUARANTINE, and 8.0% were issued a request for ADDITIONAL VERIFICATION. The analysis of the trust score reveals that the average trust score is 89.85 with a standard deviation of 14.63, suggesting that the overall level of trust is high, but there is some variation between the components.

The first quartile (Q1) was 75, and the minimum score was 60. The median was 100, and the 75th percentile (or third quartile) and the maximum were 100, indicating that the trust scores for at least half of the assessed components were the maximum. The results show that the Zero-Trust mechanism categorizes components based on their verification status and takes stringent measures to components that need further security evaluation.

Table 5. Zero-Trust Decision Distribution

Zero-Trust Decision	Count	Percentage
ALLOW	321	64.2%
QUARANTINE	139	27.8%
ADDITIONAL VERIFICATION	40	8.0%
Total	**500**	**100%**

Table 6. Trust Score Statistics

Trust-Score Statistic	Value
Count	500
Mean	89.85
Standard Deviation	14.63
Minimum	60.00
25th Percentile	75.00
Median (50%)	100.00
75th Percentile	100.00
Maximum	100.00

4.7. Attack Type vs. Zero-Trust Decision and Trust-Score Analysis

The results indicate that there is a clear correlation between attack type and Zero-Trust decisions. Of all 320 non-attack cases, all were allowed, meaning that trusted components could gain unrestricted access. All 50 cases of Content Tampering and 50 of Provenance Tampering were quarantined, and all 40 cases of Dependency Substitution cases were subject to Additional Verification. The number of quarantined Signature Tampering cases was 39, with one allowed case. Differentiated security assessment is further reflected in the average trust scores: Provenance Tampering scored the lowest (60.00), followed by Content Tampering (70.00), Signature Tampering (75.62) and Dependency Substitution (85.00). The average for non-attack components was 100.00% perfect. The results obtained in this work have shown that the Zero-Trust mechanism gives increasingly strict restrictions to components with higher security concerns.

Table 7. Attack type trust-score analysis

Attack Type	Total Cases	Additional Verification	Allow	Quarantine	Avg. Trust Score	Min.	Max.
Content Tampering	50	0	0	50	70.00	70.0	70.0
Dependency Substitution	40	40	0	0	85.00	85.0	85.0
None	320	0	320	0	100.00	100.0	100.0
Provenance Tampering	50	0	0	50	60.00	60.0	60.0
Signature Tampering	40	0	1	39	75.62	75.0	100.0

4.8. Zero-Trust Verification Decision Distribution

The Zero-Trust Verification Decision Distribution shows the distribution of security verification decisions of 500 AI supply-chain components. There are 321 components in the ALLOW category, suggesting that these components met the necessary verification conditions and were deemed to be trustworthy. QUARANTINE comprises 139 components, which correspond to the components that have security concerns and that need to be isolated to stop any propagation of contaminated or suspicious artifacts.

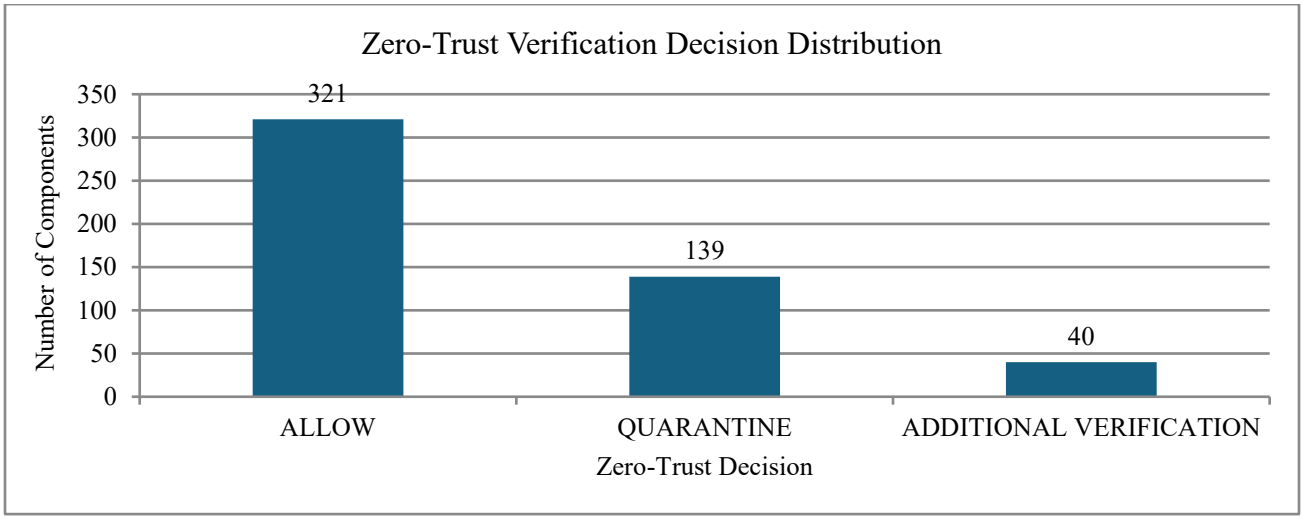


Fig. 8 Zero-trust decision distribution

A further 40 components were assigned to ADDITIONAL VERIFICATION, indicating that their security status required deeper assessment before a final trust decision. The overall distribution will show that the Zero-Trust mechanism is not zero-trust for all components, but rather, it distinguishes between access decisions based on verification results. The figure emphasizes the framework's function in implementing flexible security measures on elements of the AI supply chain.

The multi-layer tamper detection and Zero-Trust framework exhibit virtually identical performance in terms of all measured metrics, as shown in the Performance Comparison of Security Frameworks. The overall classification is fairly accurate (99.8% of accuracy) in both frameworks. Both methods achieve 100% precision, meaning that the classified security violations are very accurate - and no false-positive classifications are observed. The recall is of about 99.4%, meaning almost all of the parts that have been tampered with will be identified, with very few likely to be missed. F1 score is about 99.7%, indicating good precision and recall. Near overlapping bars show that the combination of several verification mechanisms and the ZT decision-making is highly effective at detection. Overall, the results demonstrate the robustness and reliability of both security approaches for AI supply-chain integrity verification.

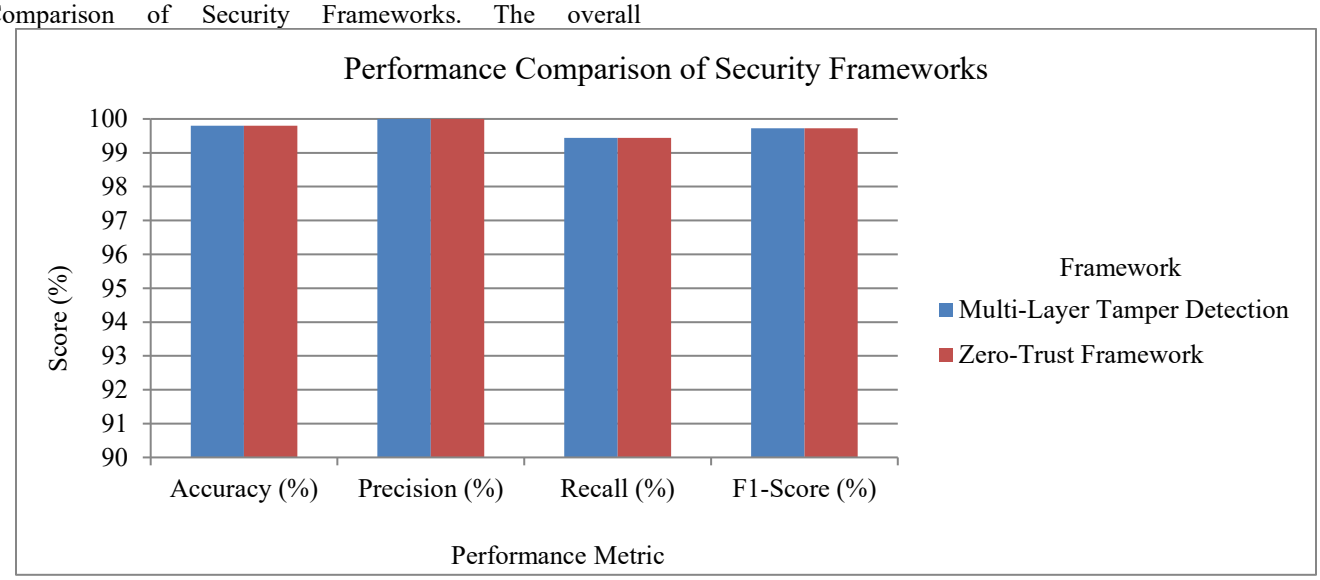


Fig. 9 Performance comparison visualization

5. Conclusion

This study demonstrates that securing modern AI systems requires an end-to-end supply-chain approach rather than isolated model-level or network-level protection. The proposed Zero-Trust Framework for Provenance, Attestation and Tamper Detection enables the establishment of trustworthy connections between datasets, models, source code, packages, APIs, agents, tools, and infrastructure through a combination of provenance graphs, SHA-256 cryptographic hashing, digital attestation, multi-layer verification, and continuous Zero-Trust decision making. The lineage of components was successfully represented as a structured graph with 500 nodes and 1,000 edges to trace the component's lineage and find potential compromised paths, and this was done using a provenance analysis. The integrated verification process was effective as shown by experimental results of 500 components with 99.80% accuracy, 100% precision, 99.44% recall and 99.72% F1-score in tamper detection. The detection rates for attack types were: 100% for content tampering, 100% for dependency substitution and 100% for provenance tampering, and 97.5% for signature tampering. Zero-Trust also categorized the components into 321 Allowed, 139 Quarantined and 40 Additional verifications, showing Adaptive security decisions. In conclusion, the results suggest that a multi-modal approach of provenance-based, cryptographic, and periodic checks can significantly enhance the integrity of AI supply chains, thereby mitigating the likelihood of tampering, malicious actors, and provenance violations.

References

[1] Robert Campbell, "Zero Trust for AI Systems: A Reference Architecture and Assurance Framework," *Preprint*, pp. 1-33, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[2] Badal Bhushan, "An Explainable Zero Trust Identity Framework for LLMs, AI Agents, and Agentic AI Systems," *International Journal of Computer Applications*, vol. 187, no. 46, pp. 42-52, 2025. [CrossRef] [Publisher Link]

[3] Sivaramakrishnan Narayanan, "Cloud-native Generative Artificial Intelligence for Autonomous Third-party Risk Intelligence: A Zero-trust Supply Chain Assurance Framework," *International Journal of Computer Engineering and Technology*, vol. 14, no. 1, pp. 283-297, 2023. [CrossRef] [Google Scholar]

[4] Jung-Hyun Mok, Hyun Jo, and Sokjoon Lee, "An Action-Centric Zero Trust Maturity Model for Agentic AI Environments," *Sensors*, vol. 26, no. 16, pp. 1-40, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[5] Vasileios Giannaris, "*Trustworthy AI-based Supply Chains*," Master's Thesis, University of Piraeus, 2026. [Google Scholar]

[6] Kai Li et al., "Zero-Trust Foundation Models: A New Paradigm for Secure and Collaborative Artificial Intelligence for Internet of Things," *IEEE Internet of Things Journal*, vol. 12, no. 22, pp. 46269-46293, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sunil Gentyala, Ch Srinivas, and Raghu Dhumpati, "A Zero-Trust Supply Chain Security Framework for Model Context Protocol-based AI Systems," *2026 5th International Conference on Computer Networks, Big Data and IoT (ICCBI)*, Dubai, United Arab Emirates, pp. 913-919, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[8] Robert Campbell, "AI Supply Chain Security: MBOM-PQC Provenance, PQC Attestation, and a Maturity Model for Quantum-Resistant Assurance," *Systems*, vol. 14, no. 5, pp. 1-57, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[9] Dan Xu et al., "The Erosion of Cybersecurity Zero-trust Principles through Generative AI: A Survey on the Challenges and Future Directions," *Journal of Cybersecurity and Privacy*, vol. 5, no. 4, pp. 1-56, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[10] Eria Othieno Pinyi et al., "AI-Enhanced Trust Graph Analytics over Distributed Ledgers for Verifiable Hardware and Software Provenance in US National Security Networks," *World Journal of Advanced Research and Reviews*, vol. 29, no. 1, pp. 717-733, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[11] Kiarash Ahi, Vaibhav Agrawal, and Saeed Valizadeh, "LLM Scalability Risk for Agentic-AI and Model Supply Chain Security," *Journal of Computer Information Systems*, pp. 1-17, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[12] Timilehin Emmanuel Odeyinka et al., "Bridging AI-automated Governance, Adaptive Certification, Behavioral Authentication, and AI-agent Risk Monitoring in Zero-trust Digital Infrastructures," *Journal of Engineering Research and Reports*, vol. 28, no. 1, pp. 371-387, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[13] Eranga Bandara et al., "AI Trust OS--A Continuous Governance Framework for Autonomous AI Observability and Zero-Trust Compliance in Enterprise Environments," *arXiv preprint*, pp. 1-39, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[14] Saurav Shriwastav, "*A Zero-Trust Framework for Automated AI Infrastructure Deployment in SMBs: A Case Study in Secure DevOps*," University of Genoa, 2025. [Google Scholar] [Publisher Link]

[15] Yayu Gao et al., "Skillsets on the Chain: A Blockchain-based Zero-Trust Framework for Agentic AI Networking," *IEEE Transactions on Cognitive Communications and Networking*, vol. 12, pp. 10589-10606, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[16] Oluwafemi Alabi Okunlola, "Design and Implementation of Autonomous Zero Trust Orchestration for Real-Time Risk Adaptive Access Control in Global Multi-Cloud Logistics Platforms," *International Journal of Science, Architecture, Technology and Environment*, vol. 2, no. 6, pp. 790-809, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Shivani Manchanda, and Divya Pandey, "Securing Autonomous AI Agents: A Reference Architecture for Data Protection in the Agentic AI Era," *SSRN*, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[18] Rabia Sadiq, "Security Risks and Governance Challenges of Agentic AI Systems in Cybersecurity Operations," Research Gate. [Google Scholar]

[19] Muhammad Liman Gambo, and Ahmad Almulhem, "Zero Trust Architecture: A Systematic Literature Review," *Journal of Network and Systems Management*, vol. 34, no. 1, pp. 1-37, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[20] Shaurya Jauhari, "Security-by-Design for Multi-Agentic AI Systems," *SSRN*, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[21] Partha Pratim Ray, "A Review of TRiSM Frameworks in Artificial Intelligence Systems: Fundamentals, Taxonomy, Use cases, Key Challenges and Future Directions," *Expert Systems*, vol. 43, no. 3, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ian Matthew Campbell Coston et al., "Multi-Vector Adversarial Testing of an AI-Orchestrated Zero Trust Methodology on Constrained Edge Hardware," *Applied Sciences*, vol. 16, no. 10, pp. 1-73, 2026. [CrossRef] [Google Scholar] [Publisher Link]

[23] Khandakar Ashrafi Akbar et al., "Advanced Persistent Threat Detection using Data Provenance and Metric Learning," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 5, pp. 3957-3969, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Yue Zheng, Si Wang, and Chip-Hong Chang, "A DNN Fingerprint for Non-Repudiable Model Ownership Identification and Piracy Detection," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2977-2989, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Edoardo Debenedetti et al., "AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents," *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Yifei Wang et al., "BadAgent: Inserting and Activating Backdoor Attacks in LLM Agents," *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand, pp. 9811-9827, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Jiashu Xu et al., "Instructions as Backdoors: Backdoor Vulnerabilities of Instruction Tuning for Large Language Models," *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Mexico City, Mexico, pp. 3111-3126, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jun Yan, Vansh Gupta, and Xiang Ren, "BITE: Textual Backdoor Attacks with Iterative Trigger Injection," *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada, pp. 12951-12968, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Jian Zhao et al., "Models Are Codes: Towards Measuring Malicious Code Poisoning Attacks on Pre-trained Model Hubs," *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, Sacramento, USA, pp. 2087-2098, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Xinyi Zheng et al., "Towards Robust Detection of Open-Source Software Supply Chain Poisoning Attacks in Industry Environments," *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, Sacramento, USA, pp. 1990-2001, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Shuncheng Zhou et al., "A Novel Malware Detection Model in the Software Supply Chain Based on LSTM and SVMs," *Applied Sciences*, vol. 14, no. 15, pp. 1-24, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[32] GenAI and Agentic AI Security Incidents – 7,725 Real-World and Research Incidents. [Online]. Available: https://discuss.huggingface.co/t/genai-agentic-ai-security-incidents-7-725-real-world-research-incidents/176510