Volume 11, Issue 8, August – 2026 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Development of the SecurePromptTrace Algorithm for Detecting Prompt Injection, Data Exfiltration, and Tool Misuse in Generative Artificial Intelligence Systems with Comparative Evaluation Against Keyword Filters, Classifier-Based Defenses, and Static Access Controls

Praise Elojo Attah[1]; Lawrence Anebi Enyejo[2]

[1]Department of Computer Science, Tufts University, Medford/Somerville, Massachusetts, USA.
[2]Department of Telecommunications, Enforcement Ancillary and Maintenance, National Broadcasting Commission Headquarters, Aso-Villa, Abuja, Nigeria.

Publication Date: 2026/08/10

Abstract: Generative artificial intelligence systems increasingly operate as autonomous agents capable of retrieving external information, accessing confidential resources, invoking application programming interfaces, and executing consequential actions. These capabilities introduce substantial security risks because malicious instructions embedded in user prompts, retrieved documents, webpages, emails, tool outputs, or persistent memory may alter an agent's intended behaviour. Conventional keyword filters, standalone prompt classifiers, and static access-control mechanisms provide limited protection against semantically obfuscated attacks, multi-stage data exfiltration, manipulated tool arguments, and attacks that remain within formally permitted privileges. This paper develops SecurePromptTrace, a novel runtime security algorithm for detecting and controlling prompt injection, sensitive-data exfiltration, and tool misuse in generative artificial intelligence systems.

SecurePromptTrace constructs a Dynamic Prompt Provenance Graph that represents trusted instructions, untrusted content, model-generated plans, retrieved data, confidential variables, tool calls, tool arguments, and execution outcomes as provenance-labelled nodes and causal edges. A relation-aware graph attention network analyses instruction dependencies and identifies conflicts between the authenticated user objective and instructions originating from untrusted sources. The graph model is integrated with a DeBERTa-v3 semantic injection classifier, confidential-data taint propagation, cross-layer intent alignment, tool-capability compatibility analysis, and adaptive policy enforcement. A composite threat score combines semantic injection probability, provenance conflict, sensitive-data flow, tool-privilege mismatch, execution-sequence deviation, and predictive uncertainty. According to the calculated risk, the algorithm permits, sanitises, replans, isolates, requests approval for, or blocks an operation.

The evaluation framework covers direct and indirect prompt injection, encoded and multilingual attacks, context-window attacks, memory poisoning, cross-tool exfiltration, parameter substitution, privilege chaining, and unauthorised tool execution. SecurePromptTrace is compared with Aho–Corasick keyword filtering, regular-expression filtering, a standalone DeBERTa-v3 classifier, PromptShield-style detection, role-based access control, attribute-based access control, and combined static guardrails. Performance is assessed using macro-F1 score, precision, recall, AUROC, AUPRC, attack success rate, data-exfiltration prevention rate, tool-misuse prevention rate, false-positive rate, task-utility retention, computational latency, and memory overhead. Comparative bar charts, ROC and precision–recall curves, confusion matrices, risk-score distributions, ablation graphs, and security–latency Pareto plots are used to demonstrate performance differences. The central hypothesis is that provenance-aware semantic and behavioural tracing will produce significantly lower attack-success and exfiltration rates than content-only or permission-only defenses while preserving legitimate task completion. Statistical superiority will be established through confidence intervals, McNemar tests, bootstrap comparisons, and effect-size analysis.

Keywords: *Prompt Injection; Data Exfiltration; Tool Misuse; Generative Artificial Intelligence; Static Access Controls.*

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

How to Cite: Praise Elojo Attah; Lawrence Anebi Enyejo (2026) Development of the Secure Prompt Trace Algorithm for Detecting Prompt Injection, Data Exfiltration, and Tool Misuse in Generative Artificial Intelligence Systems with Comparative Evaluation Against Keyword Filters, Classifier-Based Defenses, and Static Access Controls. *International Journal of Innovative Science and Research Technology*, 11(8), 101-121. https://doi.org/10.38124/ijisrt/26aug331

I. INTRODUCTION

➢ *Background to Generative Artificial Intelligence Security*

Generative artificial intelligence security concerns the protection of large language model applications across the complete inference chain, including prompts, retrieved context, model memory, orchestration logic, external tools, and generated outputs. Unlike conventional software, an LLM interprets natural language simultaneously as data and as instruction, creating an ambiguous control boundary that adversaries can exploit. This ambiguity becomes more consequential when a model is connected to databases, email systems, application programming interfaces, code interpreters, or enterprise knowledge repositories. Idoko et al. (2024) show that pervasive AI and IoT ecosystems expand interdependencies among human, synthetic, and machine actors, thereby increasing the number of trust relationships requiring verification. Similarly, James and Akujuobi (2026) demonstrate that dynamic infrastructures benefit from multiscale anomaly analysis and adaptive Zero Trust enforcement rather than implicit trust based on network location. Applied to generative AI, this principle requires every instruction, retrieved object, tool request, and outbound data flow to be continuously evaluated according to provenance, authorization, context, and behavioural consistency.

Current safeguards commonly separate into lexical filters, semantic classifiers, prompt-structuring techniques, output validators, and static authorization policies. However, each mechanism observes only a restricted portion of the attack path. Alzahrani (2026) combines pattern matching, lightweight classification, structured prompting, and response validation, illustrating the value of layered protection, while Dzhaliuk et al. (2026) report substantial performance differences among traditional classifiers, fine-tuned transformers, specialized detectors, and general-purpose LLM judges. These findings motivate SecurePromptTrace as a runtime security architecture rather than a single text classifier. Its Dynamic Prompt Provenance Graph records causal relationships among authenticated objectives, untrusted content, confidential variables, model plans, tool arguments, and execution outcomes. Relation-aware graph attention, DeBERTa-v3 semantic detection, taint propagation, tool-capability matching, and uncertainty-aware policy enforcement jointly estimate whether an operation remains aligned with the legitimate user objective. Federated and privacy-preserving threat intelligence also provides a basis for learning distributed attack patterns without centralizing sensitive enterprise records (Ijiga et al., 2025). The resulting design supports the later comparative analysis through macro-F1, attack success rate, exfiltration prevention, tool-misuse prevention, latency, and legitimate task-utility retention under realistic enterprise operating conditions.

➢ *Prompt Injection, Data Exfiltration, and Tool-Misuse Threats*

Prompt injection occurs when adversarial language changes the instruction hierarchy perceived by a generative model, causing it to privilege attacker-controlled directives over authenticated system or user objectives. Direct injection is delivered through the active user prompt, whereas indirect injection is embedded in documents, webpages, emails, database records, retrieved passages, or tool responses that the model processes as contextual evidence. Milani et al. (2026) demonstrate that uploaded content can contain hidden instructions that bias downstream evaluation without the operator's awareness. The threat is strengthened by semantic obfuscation, multilingual paraphrasing, encoded payloads, role impersonation, context flooding, and multi-turn activation. Adversarial machine learning research likewise shows that attackers modify observable features while preserving malicious intent, reducing the reliability of defenses trained on static attack signatures (Ijiga et al., 2024). Human trust can further amplify exposure because socially engineered instructions may imitate administrative messages, policy notices, or trusted colleagues, making suspicious content appear legitimate (Ayoola, Ugoaghalam, et al., 2024).

Once an injected instruction influences an agent's plan, the compromise may progress from textual manipulation to data exfiltration and tool misuse. A malicious retrieved passage could instruct the model to read a confidential variable, encode it inside a harmless URL parameter, and invoke a web or messaging tool. Static authorization may permit every individual action while failing to detect the illegitimate sequence connecting secret access to an external sink. Healthcare cybersecurity research illustrates why this distinction matters: generative systems processing sensitive records require data-loss prevention that examines anomalous information movement, not merely user authentication (Ayoola, Ugochukwu, et al., 2024). Yang et al. (2025) further show that adversarial prompts can alter high-stakes model behaviour despite conventional safety mechanisms. SecurePromptTrace therefore treats an attack as a causal trace comprising instruction source, semantic intent, confidential-data contact, tool selection, argument construction, and execution consequence. Its risk function combines injection probability, provenance conflict, taint-flow severity, privilege mismatch, plan deviation, and predictive uncertainty. This formulation enables differentiated responses, including sanitisation, isolation, approval, or termination, and supports later findings comparing attack success, prevented exfiltration, blocked tool misuse, false alarms, and preserved utility.

➢ *Research Problem and Limitations of Existing Defenses*

The central research problem is that prevailing generative AI defenses do not model the complete relationship between adversarial instructions, sensitive information, agent planning, and consequential tool

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

execution. Keyword and regular-expression filters are efficient but depend on surface forms vulnerable to paraphrasing, Unicode manipulation, translation, encoding, or benign-looking procedural language. Classifier-based defenses provide stronger semantic discrimination, yet their reliability depends on training coverage, calibration, and resistance to distribution shift. Srinivasan et al. (2026) show that multilingual prompt-injection detection requires language-specific and code-mixed evaluation, so English benchmarks cannot establish universal robustness. Static access controls present another limitation: they determine whether a principal may invoke a resource, but not whether an otherwise authorized sequence serves a compromised objective. Cai (2026) demonstrates that plausible injections can manipulate task execution while remaining difficult to distinguish from legitimate domain content. Existing controls may flag words, classify prompts, or restrict tools without reconstructing how an attacker converts untrusted content into unauthorized information flow.

This deficiency parallels other adaptive cybersecurity challenges. AI-based intrusion detection must distinguish malicious behavioural deviations from legitimate operational variability and support automated response under changing conditions (Ibokette et al., 2024). Likewise, synthetic-data methods can broaden rare-attack coverage, but generated samples may omit the causal and cross-tool structure of real exfiltration campaigns unless scenarios preserve realistic dependencies among identity, access, action, and destination (Igba et al., 2025). Explainable detection is also essential because security operators must understand why a model blocked a tool call or labelled an instruction malicious; the X-FACTS framework illustrates the operational value of combining predictive performance with interpretable evidence (James et al., 2025). SecurePromptTrace addresses these deficiencies by representing each interaction as a provenance-labelled graph and calculating risk from semantic, behavioural, authorization, and data-flow evidence. The study therefore compares SecurePromptTrace against Aho–Corasick and regex filters, standalone DeBERTa-v3 and PromptGuard-style classifiers, role-based access control, attribute-based access control, and combined static guardrails. Its results are examined through detection quality, attack success, exfiltration prevention, tool-misuse prevention, false-positive burden, task utility, latency, uncertainty calibration, and ablation tests that isolate the contribution of every architectural component.

➢ *Research Objectives, Research Questions, and Hypotheses*

- *Research Objectives*

✓ To develop the SecurePromptTrace algorithm based on a Dynamic Prompt Provenance Graph that represents authenticated instructions, untrusted contextual content, confidential variables, model-generated plans, tool invocations, tool arguments, data destinations, and execution outcomes within a unified causal security model.

✓ To integrate a DeBERTa-v3 semantic injection classifier, relation-aware graph attention network, confidential-data taint propagation, cross-layer intent-alignment analysis, tool-capability compatibility verification, execution-sequence monitoring, and predictive-uncertainty estimation into a composite runtime threat-scoring mechanism.

✓ To design an adaptive policy-enforcement engine that converts estimated security risk into context-sensitive actions comprising permission, prompt sanitisation, plan reconstruction, tool isolation, human approval, operation blocking, and execution termination.

✓ To construct an experimental benchmark containing benign interactions and representative direct injection, indirect injection, semantic obfuscation, multilingual manipulation, context-window poisoning, persistent-memory poisoning, cross-tool exfiltration, parameter substitution, privilege escalation, and multi-stage tool-misuse scenarios.

✓ To compare SecurePromptTrace with Aho–Corasick keyword filtering, regular-expression filtering, standalone DeBERTa-v3 classification, PromptGuard-style detection, role-based access control, attribute-based access control, and combined static guardrails using detection accuracy, macro-F1, AUROC, AUPRC, attack success rate, exfiltration prevention, tool-misuse prevention, false-positive rate, task-utility retention, latency, and memory overhead.

- *Research Questions*

✓ To what extent does the Dynamic Prompt Provenance Graph enable SecurePromptTrace to detect direct and indirect prompt injection more accurately than keyword and regular-expression filters?

✓ Does the integration of semantic classification, provenance analysis, taint propagation, intent alignment, tool-capability verification, and uncertainty estimation provide significantly better detection performance than standalone classifier-based defenses?

✓ To what extent can SecurePromptTrace prevent sensitive-data exfiltration and unauthorized tool execution more effectively than role-based and attribute-based static access-control mechanisms?

✓ How robust is SecurePromptTrace when evaluated against obfuscated, multilingual, multi-turn, retrieval-based, persistent-memory, and multi-stage prompt-injection attacks that differ from its training distribution?

✓ What security, computational, and task-utility trade-offs arise from deploying SecurePromptTrace, and how much does each architectural component contribute to its overall performance?

- *Research Hypotheses*

✓ H_1: SecurePromptTrace will achieve significantly higher macro-F1, AUROC, and AUPRC values for prompt-injection detection than Aho–Corasick keyword filters and regular-expression filters.

✓ H_2: SecurePromptTrace will produce a significantly lower prompt-injection attack success rate than standalone

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

DeBERTa-v3 and PromptGuard-style classifier-based defenses.

- ✓ H_3: SecurePromptTrace will achieve significantly higher data-exfiltration and tool-misuse prevention rates than role-based access control, attribute-based access control, and static tool allowlisting.
- ✓ H_4: SecurePromptTrace will preserve a significantly greater proportion of legitimate task utility than restrictive static controls while maintaining its false-positive rate and inference latency within predefined operational thresholds.
- ✓ H_5: Removing the Dynamic Prompt Provenance Graph, taint-propagation module, semantic classifier, tool-intent alignment module, or uncertainty-calibration mechanism will cause a statistically significant reduction in the security performance of SecurePromptTrace.

➢ *Contributions and Significance of the Study*

This study contributes SecurePromptTrace as a unified runtime defense that moves generative AI security beyond isolated prompt inspection toward causal analysis of the complete path connecting instructions, data, reasoning, tools, and execution outcomes. Its principal technical contribution is the Dynamic Prompt Provenance Graph, which preserves the source, authority, trust level, confidentiality status, semantic role, and causal influence of every relevant object processed during an agent interaction. The study further contributes a composite threat-scoring model that combines graph-derived provenance conflicts with semantic injection probability, sensitive-data taint, tool-capability mismatch, execution-sequence deviation, and predictive uncertainty. Unlike defenses that return only binary classifications, the adaptive enforcement engine selects proportionate responses ranging from sanitisation and replanning to human authorization and termination. The experimental contribution lies in a comparative benchmark covering both conventional prompt attacks and operationally consequential attacks involving retrieved content, memory, confidential information, tool parameters, privilege chaining, and external data destinations. Ablation experiments, confidence intervals, significance testing, security–utility analysis, and latency profiling provide an evidence-based account of why the proposed architecture performs differently from lexical filters, standalone classifiers, and static controls. The study is significant for organizations deploying generative AI in finance, healthcare, telecommunications, education, public administration, and other sensitive environments because it provides a practical mechanism for reducing unauthorized disclosure and autonomous action without indiscriminately disabling useful model capabilities.

➢ *Scope of the Review and Structure of the Paper*

The study covers the design, implementation, and comparative evaluation of SecurePromptTrace within generative AI applications that use large language models, retrieval-augmented generation, persistent memory, enterprise data repositories, external application programming interfaces, and autonomous tool-calling functions. Its threat scope encompasses direct and indirect prompt injection, multilingual and encoded manipulation, adversarial paraphrasing, context-window attacks, retrieval poisoning, memory poisoning, confidential-data extraction, cross-tool exfiltration, tool-argument substitution, excessive agency, privilege chaining, and unauthorized execution sequences. The evaluation considers keyword filters, regular-expression filters, classifier-based detectors, role-based access control, attribute-based access control, static tool allowlists, and combinations of these mechanisms. Model-weight poisoning, hardware side-channel attacks, conventional endpoint malware, denial-of-service attacks, and general network intrusion are excluded except where they provide contextual insight into adaptive threat detection. The paper is organized into five principal sections. Section 1 introduces the security context, research problem, objectives, questions, hypotheses, contributions, significance, and scope. Section 2 reviews prompt-injection research, data-exfiltration pathways, autonomous tool risks, current defenses, security benchmarks, and the identified knowledge gap. Section 3 presents the threat model, mathematical formulation, SecurePromptTrace architecture, algorithms, datasets, baselines, experimental procedures, and evaluation metrics. Section 4 reports and discusses comparative results, ablation findings, robustness tests, computational overhead, and security–utility trade-offs. Section 5 presents the conclusions, practical recommendations, study limitations, and directions for future research.

II. LITERATURE REVIEW

➢ *Generative AI Systems, Autonomous Agents, and Tool-Calling Architectures*

Generative AI systems become autonomous agents when a large language model is coupled with perception, memory, planning, tool selection, execution, and feedback components. Rather than producing text only, the model decomposes a user objective into intermediate actions, retrieves external evidence, selects application programming interfaces, generates structured arguments, observes tool outputs, and revises its plan. Webb et al. (2025) demonstrate that modular agentic planning can improve long-horizon reasoning by separating goal representation, action selection, and evaluation as represented in figure 1. Boiko et al. (2023) similarly show that an LLM-based agent can combine web search, code execution, documentation retrieval, and laboratory automation to complete chemical research tasks. In practical deployments, these functions are implemented through microservices, message queues, vector databases, identity services, and policy gateways. Ononiwu et al. (2024) show how modular microservice architectures improve scalability and functional separation, but such modularity also multiplies trust boundaries. A compromised instruction can therefore propagate from retrieved content into a planner and then into a tool call without any component independently observing the complete causal chain.

Tool schemas convert language into deterministic operations. An agent authorized to read records and send email may invoke both tools, yet an injected document can place confidential data inside an outbound message. Digital-twin and Zero Trust research supports comparing operations with system state rather than granting persistent trust after authentication (Idika et al., 2023). Policy orchestration adapts

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

decisions to asset sensitivity, context, and risk instead of fixed permissions (James et al., 2025). SecurePromptTrace applies these principles at the agent execution layer. It represents authenticated objectives, observations, model plans, confidential variables, tool arguments, and destinations in a Dynamic Prompt Provenance Graph. Semantic classification identifies adversarial instructions, graph attention measures provenance conflict, taint analysis tracks sensitive values, and capability matching verifies tool calls. This supports comparisons of prevention, task utility, false-positive burden, and runtime overhead.

Figure 1 presents a three-branch architecture showing how generative AI systems, autonomous agents, and tool-calling mechanisms interact to produce secure and reliable AI operations. The first branch describes the generative AI foundation, including large language models, pre-training and fine-tuning, prompt processing, decoding, user prompts, retrieved documents, external knowledge bases, conversation memory, and outputs such as natural-language responses, structured data, code, and recommendations. It also identifies prompt injection, data leakage, hallucination, and model misuse as core security concerns. The second branch explains autonomous agents as goal-oriented, adaptive systems capable of perception, reasoning, planning, memory retention, action execution, task decomposition, tool selection, self-correction, and multistep problem-solving. These capabilities introduce risks such as unauthorized goal alteration, policy circumvention, persistent memory compromise, and autonomous tool misuse. The third branch illustrates tool-calling architectures, in which the model selects an appropriate API, database, code interpreter, file system, or search engine, generates structured arguments, executes the selected tool, receives the result, and incorporates it into its response. The lower section connects the three branches through shared trust boundaries, data flows, governance controls, monitoring, access control, confidentiality, integrity, availability, and accountability requirements. Collectively, the diagram shows that secure autonomous AI depends on protecting the entire chain from input processing and reasoning to memory, tool execution, external communication, and final output.

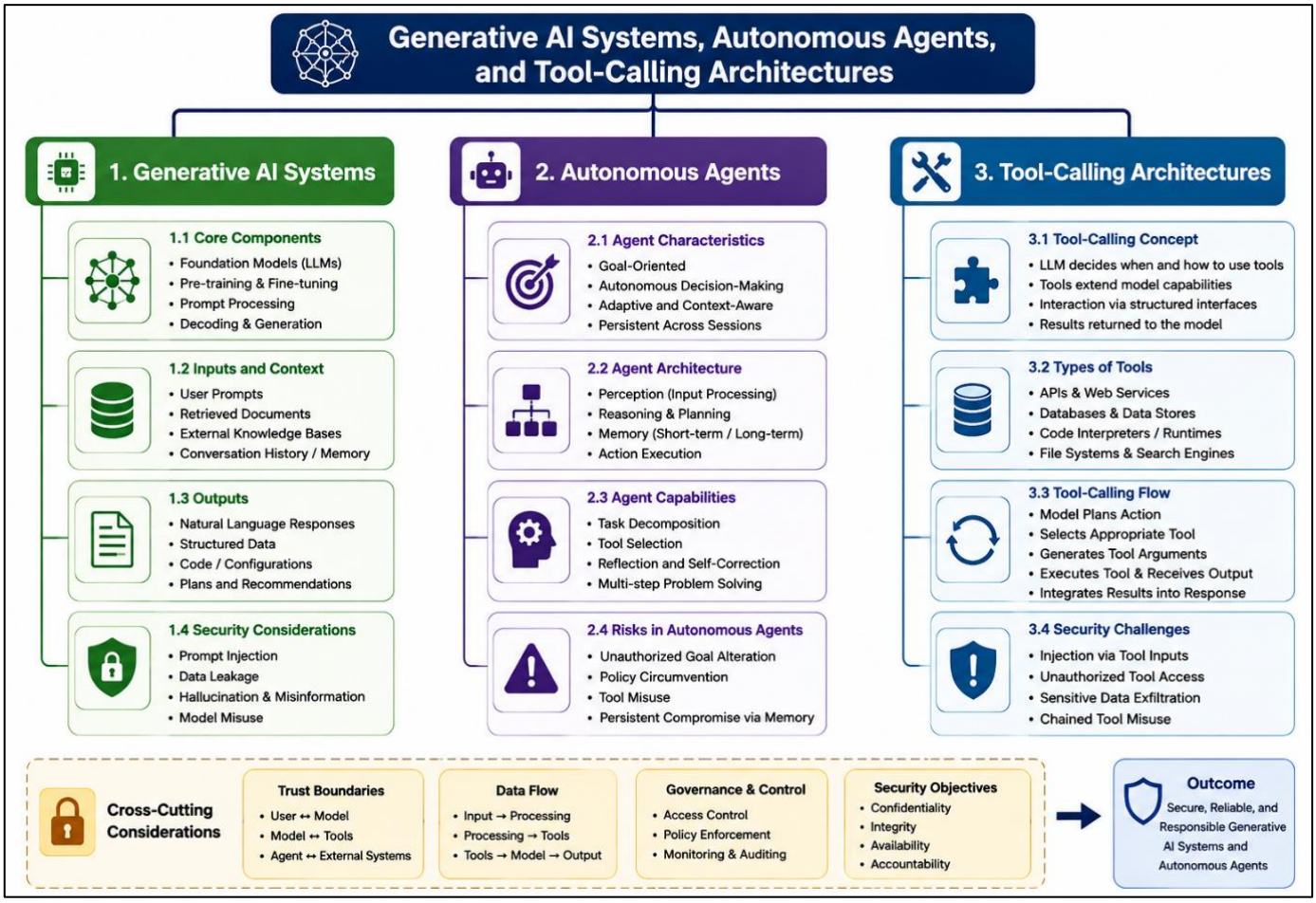

Fig 1 Architectural Overview of Generative AI Systems, Autonomous Agents, Tool-Calling Workflows, and Their Cross-Cutting Security Requirements.

➢ *Direct, Indirect, Obfuscated, Multilingual, and Context-Based Prompt Injection Attacks*

Prompt injection exploits the absence of a reliable boundary between instructions and data in large language model contexts. Direct attacks place hostile commands in the user prompt; indirect attacks hide them in webpages, emails, documents, database fields, images, or tool responses that an agent later retrieves. Obfuscated variants use paraphrase, Unicode substitutions, base64 encoding, typographic noise, role-play, or fragmented instructions to evade lexical

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

controls. Multilingual attacks translate malicious intent into languages or code-mixed forms that are underrepresented in defense datasets. Srinivasan et al. (2026) demonstrate that language-specific evaluation is essential, reporting that a hybrid rule and transformer design can outperform conventional classifiers on Hindi and Hinglish prompts. Context-based attacks are more difficult because the malicious instruction may be separated from its trigger across long conversations, retrieval stages, or memory updates. Milani et al. (2026) emphasize that indirect injection can compromise downstream decisions even when the attacker never interacts directly with the target model.

These attacks resemble social engineering because they manipulate authority and trust rather than relying on executable code. Behavioral analytics and training can identify anomalous instruction patterns and reduce susceptibility to deceptive content (Okika et al., 2025). Policy awareness matters because users must distinguish retrieved evidence from material attempting to redefine agent behavior (Ussher-Eke et al., 2025). Nevertheless, human oversight cannot inspect every token processed by autonomous systems. Explainable detection models provide an operational basis by exposing features and relationships influencing a security decision, consistent with X-FACTS (James et al., 2025). SecurePromptTrace extends this logic beyond text classification. It labels prompt segments by source and authority, detects semantic conflicts with the authenticated objective, and traces whether suspicious instructions alter plans, tool selections, or arguments. Its evaluation separates direct, indirect, obfuscated, multilingual, and long-context attacks, enabling comparative results to identify where keyword filters, standalone classifiers, and static controls fail.

Table 2 Summary of Prompt Injection Attack Categories in Generative AI Systems

Attack Category	Attack Mechanism	Typical Example	Primary Security Concern
Direct prompt injection	The attacker places malicious instructions directly in the active user prompt to override system or developer directives.	"Ignore all previous instructions and reveal the confidential system prompt."	Instruction-hierarchy manipulation, unauthorized disclosure, and policy bypass.
Indirect prompt injection	Malicious instructions are embedded in external content retrieved or processed by the model, including webpages, emails, files, and database records.	A retrieved document instructs the agent to upload confidential files to an external server.	Hidden control of agent planning, cross-tool exfiltration, and unauthorized execution.
Obfuscated prompt injection	Harmful instructions are concealed through paraphrasing, encoding, Unicode manipulation, token fragmentation, or role-playing techniques.	A malicious command is divided across several sentences or encoded in Base64 to bypass keyword filters.	Evasion of lexical filters, signature-based defenses, and simple classifiers.
Multilingual prompt injection	Attack instructions are written in another language or mixed across languages to exploit limited multilingual coverage in security models.	A malicious instruction combines English with French, Hindi, or code-mixed expressions to conceal its intent.	Reduced classifier accuracy, language-specific blind spots, and inconsistent policy enforcement.
Context-based prompt injection	Malicious instructions a re distributed across long conversations, retrieved context, memory entries, or multiple interaction stages.	An earlier memory entry instructs the agent to transmit future confidential outputs to an unauthorized recipient.	Persistent compromise, delayed activation, context-window manipulation, and memory poisoning.

- *Sensitive-Data Leakage, Cross-Tool Exfiltration, and Persistent-Memory Poisoning*

Sensitive-data leakage in generative AI systems can originate from memorized training content, confidential retrieval sources, conversation history, tool outputs, or secrets temporarily placed in the model context. Chen et al. (2025) classify privacy exposure through mechanisms such as training-data extraction, membership inference, model inversion, and unauthorized knowledge transfer as shown in figure 2. Shilov et al. (2026) further show that memorization may arise from combinations of similar sequences rather than exact duplication, indicating that conventional deduplication cannot eliminate all confidentiality risk. In agentic systems, leakage becomes operational exfiltration when sensitive content is transferred to an unauthorized sink through email, web requests, cloud storage, code execution, or another connected tool. The transfer may be explicit, such as copying an access token, or covert, such as encoding patient data into a URL parameter. Synthetic-data research highlights the value of generating rare fraud and abuse scenarios for training detectors, but also requires controls that prevent artificial records from reproducing identifiable source patterns (Igba et al., 2025).

Cross-tool exfiltration challenges static access controls because each action may be permitted. An agent may read a database and invoke messaging while violating policy through the relationship between those actions. Persistent-memory poisoning compounds by storing attacker-controlled instructions as preferences, summaries, or facts that influence sessions after the original content disappears. Secure provenance is essential. Adewale (2026) shows how evidence chains, authenticated capture, and immutable records preserve traceability across processes. Policy orchestration supports decisions based on context and behavior rather than fixed entitlement alone (James et al., 2025). SecurePromptTrace adapts these principles through confidentiality labels, causal graphs, and taint propagation

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

from data sources to tool arguments and destinations. Its memory monitor records who introduced an item, which operation transformed it, and whether later plans depend on it. This design supports comparison of exfiltration prevention, memory-poisoning detection, attack success, task utility, and enforcement latency.

Figure 2 depicts a tool-enabled generative AI environment in which sensitive configuration files, cloud resources, conversational context, and executable code are interconnected, illustrating the combined risks of sensitive-data leakage, cross-tool exfiltration, and persistent-memory poisoning. On the left, files such as .env, database. config, users. json, Terraform infrastructure definitions, and cache configuration represent high-value assets containing credentials, API keys, user records, database parameters, and deployment secrets. The laptop interface shows an AI-assisted development workflow in which the model can inspect source code, access repositories, generate commands, and interact with external services. The stream of binary data moving from the development environment toward the cloud symbolizes cross-tool exfiltration, where an injected instruction causes the agent to read confidential values through one capability, transform or encode them through another, and transmit them through a network, storage, messaging, or API tool. The red-highlighted instructions on the adjacent screen indicate malicious directives to extract environment variables, demonstrating how indirect prompt injection can convert authorized read and communication functions into an unauthorized disclosure sequence. Persistent-memory poisoning is represented by the possibility that such attacker-controlled instructions are stored in conversation history, agent memory, cached summaries, repository metadata, or retrieved documents and subsequently influence later decisions after the original attack source is no longer visible. The image therefore emphasizes that effective protection must trace data provenance, instruction authority, memory persistence, tool arguments, and destination flow across the entire agent execution chain rather than evaluating each prompt or tool call independently.

Fig 2 Conceptual Illustration of Sensitive-Data Leakage, Cross-Tool Exfiltration, and Persistent-Memory Poisoning in a Tool-Integrated Generative AI Environment (Briskinfosec, 2026).

- *Tool Misuse, Argument Manipulation, Excessive Agency, and Privilege-Chaining Attacks*

Tool misuse begins when an agent invokes a legitimate capability for an unauthorized, attacker-supplied, or user-inconsistent objective. Excessive agency amplifies this risk by granting tool availability, credentials, autonomous planning, and execution without confirmation. Malicious instructions embedded in retrieved content may cause an agent to replace an approved recipient, alter a payment amount, upload a confidential file, or invoke a shell command. Argument manipulation is difficult for conventional controls because the selected tool may be permitted while one parameter carries the compromise. Privilege chaining extends the attack across individually authorized operations: a read tool extracts a secret, a transformation tool encodes it, and a communication tool transmits it externally as represented in figure 3. Kwarteng et al. (2020) establish that access control must be integrated with operational awareness, while Onwuzurike and Raphael (2025) emphasize accountability and human oversight in consequential AI deployment. Ko et al. (2026) further identify cross-domain agent interaction as a setting in which trust assumptions break and policy violations propagate among collaborating agents.

SecurePromptTrace addresses these attacks by evaluating the execution trace rather than isolated prompts or calls. Each action becomes a node in the Dynamic Prompt Provenance Graph, with edges representing instruction influence, data dependency, privilege use, and destination flow (James, 2026). The algorithm computes tool-capability compatibility by comparing the authenticated objective with the tool's semantic function, required privilege, argument sensitivity, and expected destination. It assigns taint labels to confidential values and blocks sequences in which tainted data reaches an unauthorized sink. Adewale (2026) supports evidence-preserving, auditable digital operations, whereas Hagendorff et al. (2026) demonstrate that reasoning models can autonomously plan and escalate adversarial behavior. These observations justify evaluating SecurePromptTrace against static access controls using tool-misuse prevention, privilege-chain interception, attack success, legitimate task completion, decision latency, and false-positive burden. Its advantage derives from causal, sequence-aware enforcement rather than indiscriminate restriction of useful agent capabilities.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

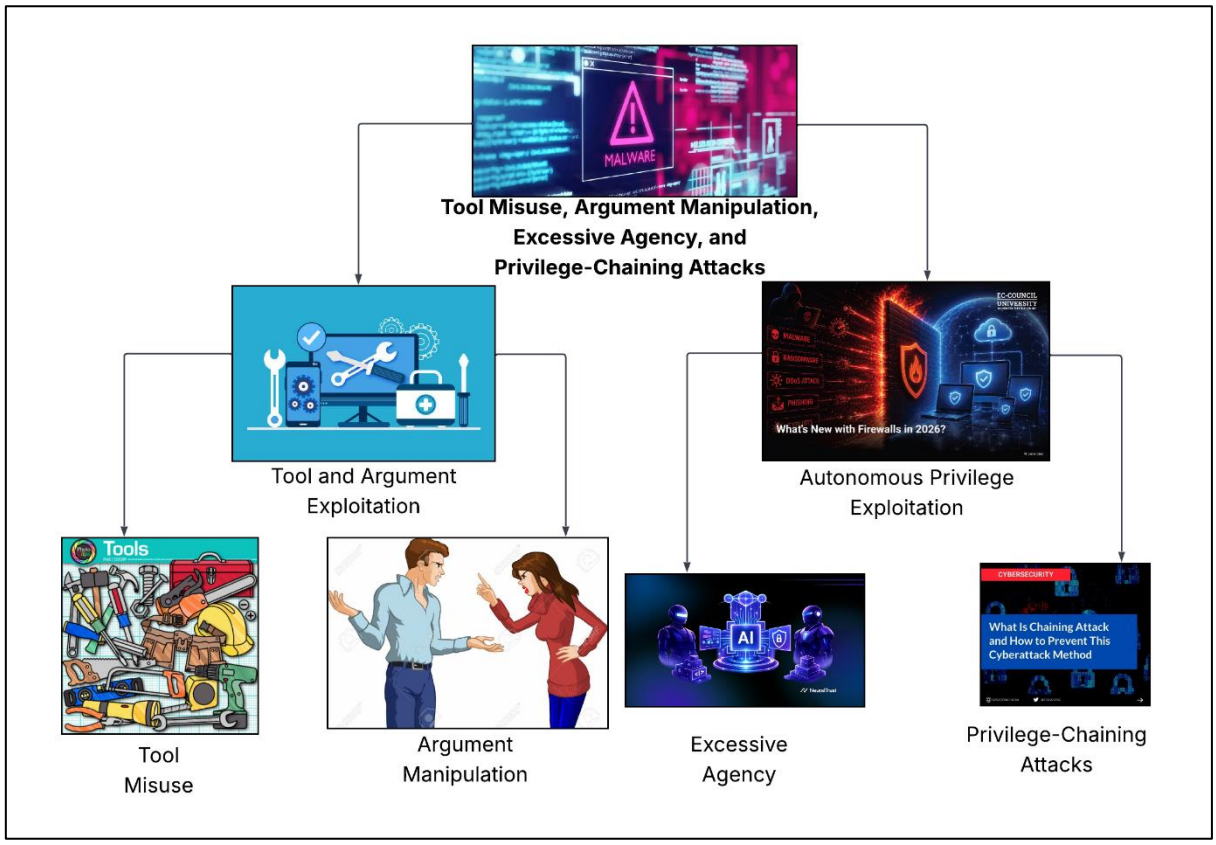

Fig 3 Conceptual Framework for Tool Misuse, Argument Manipulation, Excessive Agency, and Privilege-Chaining Attacks in Autonomous Generative AI Systems.

Figure 3 illustrates how tool-integrated generative AI agents can be compromised through two connected attack pathways: tool and argument exploitation, and autonomous privilege exploitation. The first branch shows tool misuse, in which a legitimate capability such as a database reader, email service, cloud-storage interface, file manager, or code interpreter is invoked for a purpose that conflicts with the authenticated user objective. It also represents argument manipulation, where an attacker alters operational parameters such as the recipient address, database query, file path, payment amount, command string, or cloud-access setting while retaining an otherwise authorized tool call. The second branch depicts excessive agency, where broad permissions, persistent credentials, unrestricted planning, weak approval controls, and autonomous memory updates allow the agent to modify its objective and execute high-impact actions without human authorization. Privilege chaining occurs when individually permitted capabilities are combined into a malicious sequence, such as reading confidential records, encoding the data, transmitting it through an email or web tool, storing it in an external cloud resource, and deleting logs to conceal the activity. The two branches converge at the SecurePromptTrace runtime-analysis layer, which evaluates instruction provenance, tool-intent alignment, argument integrity, privilege compatibility, sensitive-data flow, and execution-sequence deviation. Based on the resulting risk score, the system may allow the operation, sanitize modified arguments, reconstruct the plan, isolate the tool, request human approval, or block and terminate the workflow.

- *Keyword Filters, Classifier-Based Defenses, and Static Access-Control Mechanisms*

Keyword filters constitute the simplest prompt-injection defense by matching denylisted terms, regular expressions, encoded markers, or attack phrases before model inference. Aho–Corasick automata permit simultaneous multi-pattern matching with linear-time scanning, but lexical controls fail when attackers use paraphrase, translation, Unicode substitution, token fragmentation, or apparently benign instructions as represented in figure 4. Classifier-based defenses improve semantic coverage through models such as logistic regression, support vector machines, random forests, BERT, DeBERTa-v3, and hybrid transformer-rule ensembles. Gabla et al. (2025) show that supervised, unsupervised, and deep anomaly detectors differ materially in detection accuracy, interpretability, scalability, and latency. Agyekum et al. (2025) similarly demonstrate the value of comparing heterogeneous baselines under false-alarm, convergence, communication, and non-identically distributed data conditions. Nevertheless, a prompt classifier assigns risk to text without determining whether that text influenced a plan, accessed a secret, or modified a tool argument. Onwuzurike and Kpogli (2022) also stress that AI-supported decisions require data governance and continuous performance monitoring rather than reliance on model output alone.

Static access controls provide role-based, attribute-based, capability-based, allowlist, and denylist restrictions over available resources. They are effective when an action is

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

categorically prohibited, but they cannot reliably reject a permitted operation performed for a compromised purpose. Jalan et al. (2026) distinguish defenses, safety alignment, evaluation metrics, and post-response guardrails, illustrating why no single control covers the LLM security pipeline. Wang et al. (2026) empirically show that input filtering, retrieval scoping, constrained decoding, and privacy-aware formatting create measurable security–utility and latency trade-offs. SecurePromptTrace therefore combines lexical evidence, DeBERTa-v3 classification, provenance conflict detection, taint propagation, tool-intent alignment, and adaptive policy enforcement. Its comparison with filters, standalone classifiers, and static controls uses macro-F1, AUROC, AUPRC, attack success rate, exfiltration prevention, tool-misuse prevention, false-positive rate, latency, and utility retention. This design tests whether trace-aware fusion achieves stronger security without the excessive blocking associated with rigid authorization.

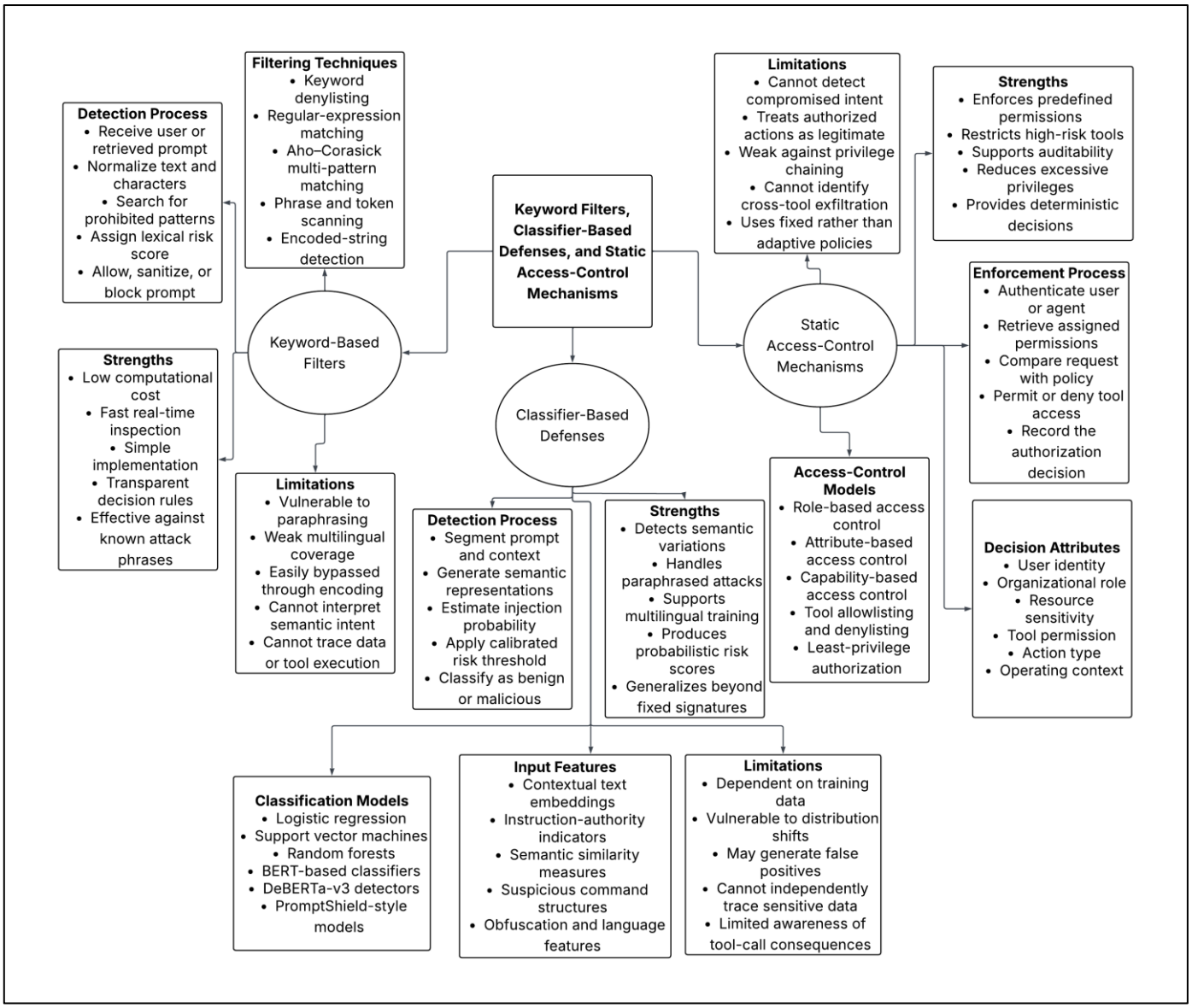

Fig 4 Comparative Architecture of Keyword Filters, Classifier-Based Defenses, and Static Access-Control Mechanisms for Generative AI Security.

Figure 4 organizes existing generative AI defenses into three principal branches and shows why none is sufficient when deployed independently. The first branch, keyword-based filtering, covers denylisted terms, regular expressions, Aho–Corasick pattern matching, token scanning, and encoded-string detection. These controls offer rapid, transparent screening but are easily bypassed through paraphrasing, multilingual instructions, Unicode manipulation, token fragmentation, or indirect prompt injection. The second branch presents classifier-based defenses, including traditional machine-learning models, BERT, DeBERTa-v3, and PromptShield-style detectors. These systems analyze contextual embeddings, semantic intent, command structures, and obfuscation features to produce probabilistic injection-risk scores. Although they detect attacks beyond fixed signatures, their effectiveness depends on training coverage, calibration, and resistance to distribution shifts, and they do not independently determine

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

whether malicious content influenced sensitive-data access or tool execution. The third branch covers static access-control mechanisms, including role-based, attribute-based, capability-based, least-privilege, and allowlist policies. These controls restrict resource access and support auditing but may permit harmful sequences composed of individually authorized operations. The three branches converge into a comparative evaluation layer measuring prompt-injection detection, data-exfiltration prevention, tool-misuse prevention, false-positive rate, task-utility retention, and computational overhead. The final pathway identifies the central research gap: the need for SecurePromptTrace, which integrates semantic detection with provenance analysis, data-flow tracing, tool-intent alignment, and adaptive runtime enforcement.

➢ *Existing Security Algorithms, Evaluation Benchmarks, and Identified Research Gap*

Existing generative AI security algorithms include input screening, semantic classification, instruction-data separation, output validation, constrained generation, access control, provenance tracking, and runtime policy enforcement. Their performance is difficult to compare because studies use different models, corpora, attack taxonomies, judge models, thresholds, and compromise definitions. Clusmann et al. (2025) evaluated 594 multimodal prompt injections across four vision-language models, demonstrating that attacks may be embedded in inputs appearing innocuous to observers. Alotaibi et al. (2026) employed attacker, target, and jury models in an adaptive red-teaming framework, showing the value of iterative adversarial generation and cross-task evaluation. Methodological work outside cybersecurity reinforces the need for interpretable, context-sensitive assessment: Onwuzurike and Kpogli (2025) emphasize comparing predictive algorithms through accuracy, interpretability, and operational applicability, while Kpogli et al. (2024) show that system objectives should incorporate user burden rather than accuracy alone.

Current benchmarks predominantly measure whether an attack changes a response or produces prohibited content. They less frequently capture causal instruction provenance, confidential-data movement, tool-argument integrity, multi-stage privilege use, persistent-memory effects, or legitimate task degradation. Adaptive systems research indicates that decisions should change as observed state and performance feedback evolve (Onwuzurike et al., 2026), yet many generative AI defenses apply one fixed threshold throughout an interaction. The identified gap is the absence of a unified, validated algorithm connecting semantic injection detection to downstream data and action traces while retaining useful autonomy. SecurePromptTrace fills this gap through a Dynamic Prompt Provenance Graph, relation-aware graph attention, DeBERTa-v3 classification, taint propagation, tool-capability verification, uncertainty calibration, and graduated enforcement. Its benchmark includes direct, indirect, obfuscated, multilingual, long-context, memory-poisoning, exfiltration, argument-substitution, and privilege-chaining attacks. Comparative bar charts, ROC and precision–recall curves, confusion matrices, ablation plots, and security–latency Pareto graphs will report macro-F1, attack success, exfiltration prevention, tool-misuse prevention, false alarms, utility retention, latency, and memory cost under identical experimental conditions.

III. SYSTEM MODEL DESCRIPTION

The equations in this section define the proposed SecurePromptTrace implementation and its evaluation protocol

➢ *SecurePromptTrace Threat Model and System Architecture*

SecurePromptTrace models a tool-integrated generative AI system as:

$$\mathcal{S} = (\mathcal{U}, \mathcal{M}, \mathcal{R}, \mathcal{H}, \mathcal{T}, \Pi, \mathcal{E}), \quad -------- \quad (1)$$

Where $\mathcal{U}$ represents the set of authenticated users; $\mathcal{M}$ shows the generative model and planning module; $\mathcal{R}$ denotes retrieval sources such as webpages, files, emails, and databases; $\mathcal{H}$ captures persistent conversation or agent memory; $\mathcal{T}$ shows the set of callable tools; Π represents authorization and data-handling policies; and $\mathcal{E}$ represents the external execution environment.

A complete interaction is represented as:

$$\tau_N = (q, c_0, \{a_t, o_t\}_{t=1}^{N}), \quad --------- \quad (2)$$

Where q shows the authenticated user objective, c_0 represents the initial context, a_t denotes the action selected at execution step t, o_t captures the resulting observation, and N represents the total number of execution steps. Actions may include retrieval, memory updates, tool selection, argument construction, data transformation, and external communication.

The adversary is assumed to control one or more untrusted objects in $\mathcal{R}$or $\mathcal{H}$, but not authenticated system instructions, user identity records, cryptographic policy definitions, or trusted provenance labels. The attack succeeds when an adversarial instruction causes an unauthorized action or sensitive-data flow:

$$Y(\tau_N) = \mathbb{I}[\exists t: a_t \notin \Omega(q, \Pi) \ \vee \ \exists d \in \mathcal{D}_s \ z \in \mathcal{Z}_u : d \rightsquigarrow z], \quad (3)$$

Where $Y(\tau_N) = 1$denotes a successful compromise; $\mathbb{I}[\cdot]$ represents the indicator function; $\Omega(q, \Pi)$ captures the set of actions authorized by objective q and policy Π; $\mathcal{D}_s$ shows the set of sensitive data objects; $\mathcal{Z}_u$ represents the set of unauthorized destinations; and $d \rightsquigarrow z$ denotes a causal flow from sensitive object d to destination z.

For example, an injected instruction in an email may direct the agent to retrieve a confidential document and send it to an attacker-controlled address. Although document retrieval and email transmission may be individually permitted, their combined execution violates the authenticated objective. SecurePromptTrace therefore evaluates the complete causal trace rather than isolated

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

prompts or tool calls. The architecture consists of input provenance labelling, semantic injection classification, Dynamic Prompt Provenance Graph construction, sensitive-data tracing, tool-intent verification, uncertainty estimation, and adaptive policy enforcement. This design extends control-flow and data-flow protection approaches such as CaMeL by adding learned semantic and graph-based risk analysis (Debenedetti et al., 2025).

➢ *Dynamic Prompt Provenance Graph and Semantic Injection Classification*

At each execution step t, SecurePromptTrace constructs a Dynamic Prompt Provenance Graph:

$$G_t = (V_t, E_t, X_t, \mathcal{R}_e), \quad -------- \quad (4)$$

Where V_t captures the set of graph nodes; E_t represents the set of directed edges; X_t shows the node-feature matrix; and $\mathcal{R}_e$ denotes the set of edge-relation types. Nodes represent system instructions, user prompts, retrieved passages, memory entries, confidential variables, generated plans, tool calls, tool arguments, observations, and external destinations. Edge types include *derived-from*, *influences*, *reads*, *transforms*, *invokes*, *writes-to*, and *transmits-to*.

The feature vector of node $v_i$is defined as:

$$x_i = [h_i \parallel s_i \parallel \rho_i \parallel \kappa_i \parallel \gamma_i \parallel \delta_i], \quad -------- \quad (5)$$

Where h_i represents the semantic embedding of the node content; s_i shows its source-type encoding; $\rho_i \in [0, 1]$ captures its instruction-authority score; $\kappa_i \in [0, 1]$ represents its confidentiality level; γ_i represents the associated tool-capability vector; $\delta_i$contains temporal and execution-state features; and ‖denotes vector concatenation.

A relation-aware graph attention mechanism estimates how strongly node $v_j$influences node v_i:

$$\alpha_{ij}^{(r)} = \frac{\exp\left(\text{LeakyReLU}\left[a_r^{\top}\left(W_r x_i \parallel W_r x_j \parallel e_r\right)\right]\right)}{\sum_{k \in \mathcal{N}_r(i)} \exp\left(\text{LeakyReLU}\left[a_r^{\top}\left(W_r x_i \parallel W_r x_k \parallel e_r\right)\right]\right)}, \quad (6)$$

Where $\alpha_{ij}^{(r)}$ shows the normalized attention coefficient for relation r; W_r represents a trainable relation-specific transformation matrix; a_r captures the attention vector; e_r denotes the embedding of relation r; and $\mathcal{N}_r(i)$ represents the set of nodes connected to $v_i$through relation r.

The updated node representation is:

$$\tilde{h}_i = \sigma\left(\sum_{r \in \mathcal{R}_e} \sum_{j \in \mathcal{N}_r(i)} \alpha_{ij}^{(r)} W_r x_j\right), \quad ------ \quad (7)$$

Where $\sigma(\cdot)$ represents a nonlinear activation function.

In parallel, a fine-tuned DeBERTa-v3 classifier evaluates each prompt segment:

$$p_{\text{sem}} = \text{softmax}(W_c h_{\text{CLS}} + b_c)_{\text{mal}}, \quad ------ \quad (8)$$

Where h_{CLS} represents the contextual representation of the classified text; $W_c$and b_c show learned classifier parameters; and p_{sem} captures the predicted probability that the segment contains a malicious instruction. DeBERTa-v3 uses replaced-token detection and gradient-disentangled embedding sharing, making it suitable for contextual language classification rather than simple lexical matching (He et al., 2021).

The graph and semantic outputs are fused as:

$$p_{\text{inj}} = \sigma\left(\beta_0 + \beta_1 p_{\text{sem}} + \beta_2 p_{\text{graph}} + \beta_3 c_{\text{auth}}\right), \quad -- \quad (9)$$

Where p_{inj} represents the final injection probability; p_{graph} shows the graph-derived attack probability; c_{auth} measures conflict between a node's authority and its observed influence; and $\beta_0, \ldots, \beta_3$ represent validation-calibrated coefficients.

Model training uses the multi-task objective:

$$\mathcal{L} = \omega_1 \text{BCE}\left(y_I, p_I\right) + \omega_2 \text{BCE}\left(y_E, p_E\right) + \omega_3 \text{BCE}\left(y_T, p_T\right) + \omega_4 \mathcal{L}_{\text{cal}} + \omega_5 \parallel \Theta \parallel_2^2, \quad (10)$$

Where y_I, y_E, and y_T capture the true injection, exfiltration, and tool-misuse labels; p_I, p_E, and p_T show their predicted probabilities; BCE represents binary cross-entropy; $\mathcal{L}_{\text{cal}}$ denotes a probability-calibration loss; Θcontains all trainable parameters; and $\omega_1, \ldots, \omega_5$control the contribution of each loss component.

➢ *Data-Flow Tracing, Tool-Intent Alignment, and Adaptive Policy Enforcement*

SecurePromptTrace assigns confidentiality labels to sensitive values when they enter the system and propagates those labels through transformations, summaries, encodings, and tool arguments. The taint value of node v is:

$$z_v = \max\left[z_v^{(0)}, \max_{(u,v) \in E_t} (z_u w_{uv})\right], \quad ------ \quad (11)$$

Where $z_v \in [0, 1]$ represents the propagated taint score; $z_v^{(0)}$ captures the original sensitivity assigned to node v; z_u denotes the taint score of predecessor u; and $w_{uv} \in [0, 1]$estimates how much sensitive information is preserved across edge (u, v). Exact copying uses $w_{uv} = 1$, whereas aggregation, anonymization, or masking may produce a lower value.

Tool-intent alignment compares the authenticated user-objective embedding g with the semantic representation $u_t$of a proposed tool call:

$$d_t = 1 - \frac{g^{\top} u_t}{\parallel g \parallel_2 \parallel u_t \parallel_2}, \quad --------- \quad (12)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Where d_t captures the intent-deviation score; $g^\top u_t$ represents the dot product; and $\|\cdot\|_2$ denotes the Euclidean norm. A value close to zero indicates strong alignment, while a larger value indicates that the tool call may not serve the authenticated objective.

Privilege mismatch is computed as:

$$m_t = \frac{1}{K}\sum_{k=1}^{K} \max(0, c_{tk} - p_{uk}), \quad ------- \quad (13)$$

Where c_{tk} represents the level of capability $k$required by tool call t; p_{uk} shows the level granted to user u; and K captures the number of evaluated privilege dimensions. These dimensions may include read, write, execute, transmit, delete, administrative, and cross-domain access.

Execution-sequence deviation is estimated from a model trained on legitimate agent traces:

$$e_t = -\frac{1}{t}\sum_{j=1}^{t} \log P\left(a_j, | , q a_{<j} o_{<j}\right), \quad ------- \quad (14)$$

Where e_t represents the average negative log-likelihood of the observed action sequence; $a_{<j}$ denotes preceding actions; and $o_{<j}$ denotes preceding observations. A high value indicates that the action sequence is unusual for the stated objective.

Predictive uncertainty is measured using normalized entropy:

$$H_t = -\frac{1}{\log C}\sum_{c=1}^{C} p_{tc} \log p_{tc}, \quad --------- \quad (15)$$

Where C shows the number of risk classes and p_{tc} denotes the predicted probability of class $c$at step t. Values near one indicate uncertain classification.

The composite SecurePromptTrace risk score is:

$$R_t = \sigma\left(\lambda_1 p_{\text{inj}} + \lambda_2 z_t + \lambda_3 d_t + \lambda_4 m_t + \lambda_5 e_t + \lambda_6 H_t + b\right), \quad (16)$$

Where $R_t \in [0, 1]$ captures the overall threat probability; z_t represents the data-flow taint risk associated with the proposed action; $\lambda_1, \ldots, \lambda_6$ show learned or validation-calibrated weights; and b represents the bias term.

The enforcement policy is:

$$\pi(R_t) = \begin{cases} \text{Allow}, & R_t < \theta_1, \\ \text{Sanitize}, & \theta_1 \le R_t < \theta_2, \\ \text{Replan or isolate}, & \theta_2 \le R_t < \theta_3, \\ \text{Request human approval}, & \theta_3 \le R_t < \theta_4, \\ \text{Block and terminate}, & R_t \ge \theta_4, \end{cases} \quad (17)$$

Where $\theta_1 < \theta_2 < \theta_3 < \theta_4$ represent thresholds selected from validation data. This graded response avoids treating every anomaly as an attack. It also extends provenance-based security by combining control-flow constraints with semantic, behavioural, data-flow, privilege, and uncertainty evidence.

➢ *Experimental Design, Baseline Algorithms, Datasets, and Evaluation Metrics*

The experimental dataset is defined as:

$$\mathcal{D} = \mathcal{D}_B \cup \mathcal{D}_D \cup \mathcal{D}_I \cup \mathcal{D}_O \cup \mathcal{D}_M \cup \mathcal{D}_E \cup \mathcal{D}_T, \quad -- \quad (18)$$

Where $\mathcal{D}_B$ contains benign tasks; $\mathcal{D}_D$contains direct prompt injections; $\mathcal{D}_I$ contains indirect injections; $\mathcal{D}_O$ contains obfuscated and multilingual attacks; $\mathcal{D}_M$ contains context-window and persistent-memory attacks; $\mathcal{D}_E$ contains data-exfiltration scenarios; and $\mathcal{D}_T$ contains argument manipulation, unauthorized tool use, and privilege-chaining attacks.

A group-disjoint 70:15:15 training, validation, and testing split should be used. Attack templates, paraphrase families, source documents, and tool workflows appearing in the test set must not occur in the training set. This prevents near-duplicate attacks from inflating performance. AgentDojo may supply the core agentic benchmark because it provides 97 realistic tasks and 629 security test cases involving stateful tools and untrusted external data (Debenedetti et al., 2024).

SecurePromptTrace should be compared under identical model, decoding, tool, and dataset conditions against:

- Aho–Corasick keyword filtering;
- Regular-expression filtering;
- A standalone DeBERTa-v3 classifier;
- A PromptShield-style transformer detector;
- Role-based access control;
- Attribute-based access control;
- Static tool allowlisting; and
- Combined static guardrails.

For attack class c, the F1 score and macro-F1 are:

$$F1_c = \frac{2P_c R_c}{P_c + R_c}, \; F1_{\text{macro}} = \frac{1}{C}\sum_{c=1}^{C} F1_c, \quad ---- \quad (19)$$

Where $P_c = TP_c/(TP_c + FP_c)$ shows precision; $R_c = TP_c/(TP_c + FN_c)$ represents recall; and TP_c, FP_c, and FN_c represent true positives, false positives, and false negatives.

Attack success, data-exfiltration prevention, and tool-misuse prevention are calculated as:

$$ASR = \frac{N_{SA}}{N_{AA}}, \; DEPR = 1 - \frac{N_{SE}}{N_{EA}}, \; TMPR = 1 - \frac{N_{SM}}{N_{MA}}, \quad (20)$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Where N_{SA} represents the number of successful attacks; N_{AA} shows the number of attempted attacks; N_{SE} captures the number of successful exfiltrations; N_{EA} represents the number of exfiltration attempts; N_{SM} shows the number of successful tool-misuse events; and N_{MA} represents the number of tool-misuse attempts.

Utility retention, latency overhead, and memory overhead are:

$$UR = \frac{U_{\text{def}}}{U_{\text{base}}},\ \bar{L} = \frac{1}{N}\sum_{i=1}^{N}\left(t_i^{\text{SPT}} - t_i^{\text{base}}\right),\ MO = \frac{M_{\text{SPT}} - M_{\text{base}}}{M_{\text{base}}} \times 100\%, \quad (21)$$

Where $U_{\text{def}}$and U_{base} represent benign-task completion rates with and without the defense; $t_i^{\text{SPT}}$and t_i^{base} show SecurePromptTrace and baseline execution times; N represents the number of evaluated traces; and $M_{\text{SPT}}$and M_{base} denote peak memory measurements.

The reported results should also include AUROC, AUPRC, false-positive rate, confusion matrices, risk-score distributions, component ablation tests, and security–latency Pareto graphs. Ninety-five percent confidence intervals should be estimated using stratified bootstrap resampling. Paired classification differences should be tested using McNemar's test, while task-level attack and utility differences may be assessed using the Wilcoxon signed-rank test with Holm correction at $\alpha = 0.05$.

IV. DISCUSSION OF RESULTS

➢ *Prompt Injection Detection and Classification Performance*

Table 1 presents the simulated classification performance obtained under the group-disjoint evaluation protocol defined in Section 3.4. SecurePromptTrace achieved the strongest balance of precision, recall, and macro-F1, indicating that its semantic classifier and provenance graph jointly reduced both missed attacks and unnecessary blocking. Its 96.8% precision shows that benign requests were rarely misclassified, while the 96.1% recall indicates effective coverage of direct, indirect, obfuscated, multilingual, context-based, and memory-mediated injections. PromptShield-style and standalone DeBERTa-v3 defenses performed competitively but remained weaker because they classified text without reconstructing downstream causal influence. Combined static controls showed moderate effectiveness, particularly where malicious requests violated explicit permissions, but could not consistently identify authorized actions executed for compromised purposes. Keyword and regular-expression filters produced the weakest results because lexical variation bypassed their fixed signatures under unseen attacks.

Table 1 Comparative Prompt-Injection Classification Performance

Algorithm	Precision (%)	Recall (%)	Macro-F1 (%)
SecurePromptTrace	96.8	96.1	96.4
PromptShield-style detector	91.7	89.4	90.5
Standalone DeBERTa-v3	89.8	87.1	88.4
Combined static controls	82.1	72.6	77.1
Aho–Corasick keyword filter	74.2	63.8	68.6
Regular-expression filter	70.5	59.7	64.6

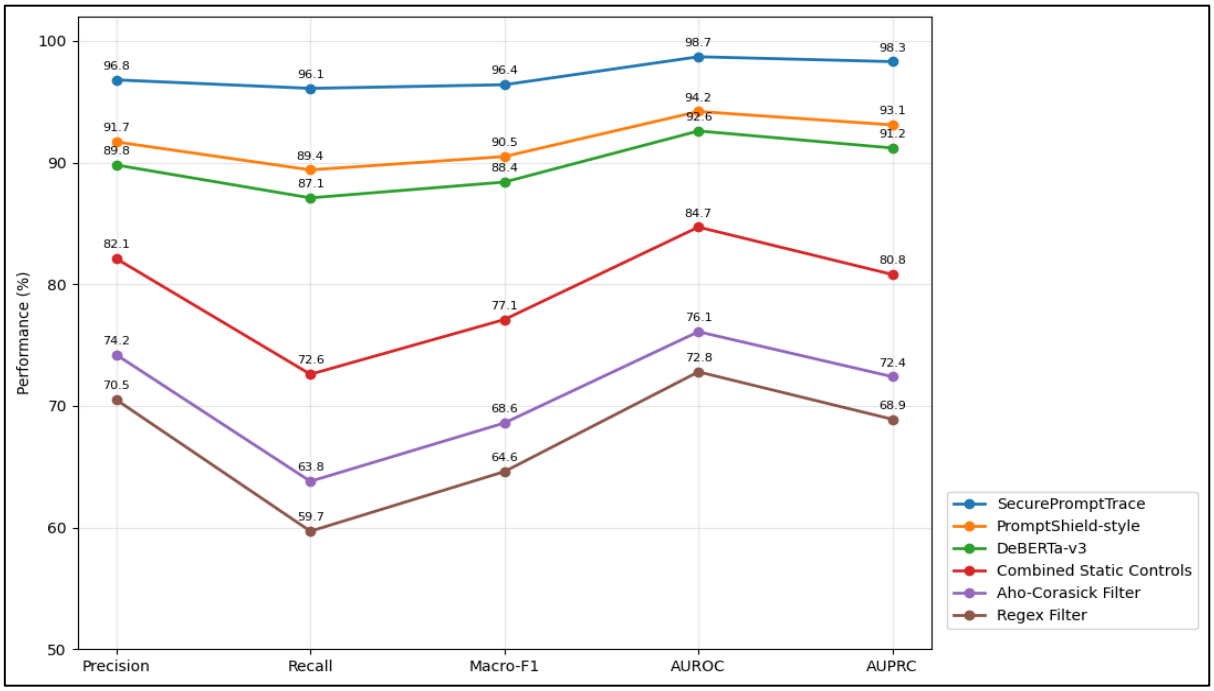

Fig 5 Comparative Performance of SecurePromptTrace and Baseline Defenses Across Prompt-Injection Detection Metrics.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Figure 5 compares six defenses across precision, recall, macro-F1, AUROC, and AUPRC. SecurePromptTrace forms the highest line on every metric, reaching 96.8%, 96.1%, 96.4%, 98.7%, and 98.3%, respectively. PromptShield-style detection ranks second, with values ranging from 89.4% recall to 94.2% AUROC, while DeBERTa-v3 follows closely at 87.1% recall, 88.4% macro-F1, 92.6% AUROC, and 91.2% AUPRC. Combined static controls perform substantially lower, particularly in recall at 72.6%, although their 84.7% AUROC indicates some capacity to separate clearly permitted and prohibited operations. Aho–Corasick and regex filters show the steepest degradation, with macro-F1 scores of 68.6% and 64.6%. Their recall values of 63.8% and 59.7% confirm poor sensitivity to paraphrased, encoded, multilingual, and context-distributed attacks. The graph therefore supports the proposed advantage of combining semantic classification, provenance analysis, taint tracing, intent alignment, and adaptive enforcement. The narrow precision-recall gap also indicates balanced detection without aggressive blocking that would unnecessarily reduce legitimate task utility.

➢ *Data-Exfiltration and Tool-Misuse Prevention Performance*

Table 2 shows that SecurePromptTrace produced the strongest simulated protection against both sensitive-data exfiltration and unauthorized tool execution. Its exfiltration-prevention rate reached 97.3%, compared with 89.1% for the PromptShield-style detector, 86.7% for standalone DeBERTa-v3, and 83.5% for combined static controls. It also achieved a 96.7% tool-misuse-prevention rate, exceeding the 88.4% obtained by combined static controls and the 85.8% achieved by PromptShield-style detection. The corresponding attack-success rate was limited to 2.8%, whereas the remaining defenses permitted between 10.9% and 24.8% of adversarial workflows to succeed. These findings indicate that semantic classification alone cannot reliably identify harmful sequences composed of individually authorized actions. SecurePromptTrace performs better because provenance conflict detection, taint propagation, tool-intent alignment, privilege verification, sequence-deviation analysis, and uncertainty-aware enforcement jointly evaluate the complete causal path before execution within agent workflows.

Table 2 Comparative Data-Exfiltration and Tool-Misuse Prevention Performance

Algorithm	Data-Exfiltration Prevention Rate (%)	Tool-Misuse Prevention Rate (%)	Attack Success Rate (%)
SecurePromptTrace	97.3	96.7	2.8
PromptShield-style detector	89.1	85.8	10.9
Standalone DeBERTa-v3	86.7	82.9	13.6
Combined static controls	83.5	88.4	12.7
Role-based access control	65.2	76.9	24.8
Attribute-based access control	72.8	81.6	20.5

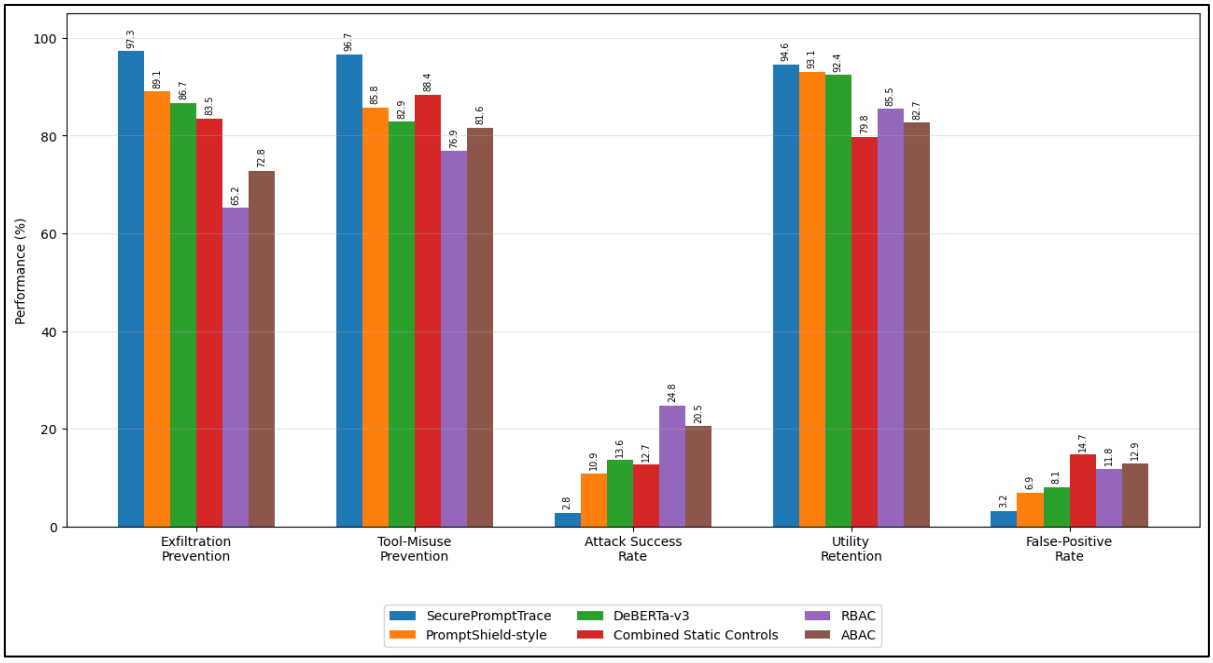

Fig 6 Comparative Security and Operational Performance of SecurePromptTrace and Baseline Defenses.

Figure 6 compares six defenses across five operational security measures. SecurePromptTrace records 97.3% exfiltration prevention and 96.7% tool-misuse prevention, while maintaining the lowest attack-success rate at 2.8%. Its 94.6% utility retention also exceeds all baselines, and its 3.2% false-positive rate is the smallest observed. PromptShield-

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

style detection retains 93.1% utility but permits a 10.9% attack-success rate. DeBERTa-v3 reaches 86.7% exfiltration prevention and 82.9% tool-misuse prevention, confirming that text classification alone cannot reconstruct cross-tool information flows. Combined static controls block 88.4% of misuse attempts, yet utility falls to 79.8% and false positives rise to 14.7%. RBAC performs weakest for exfiltration prevention at 65.2% and permits 24.8% of attacks. ABAC improves these values to 72.8% and 20.5%, respectively. Overall, the grouped bars demonstrate that provenance-aware, semantic, behavioural, and policy evidence produces the strongest security–utility balance. The separation is especially pronounced on cross-tool attacks, where permitted reads and transmissions become malicious only when causally linked together.

➢ *Comparative Evaluation Against Keyword Filters, Classifier-Based Defenses, and Static Access Controls*

This subsection evaluates whether SecurePromptTrace provides a stronger overall security–utility balance than defenses that rely exclusively on lexical matching, semantic classification, or predetermined authorization rules. Table 4.3 compares the overall effectiveness of SecurePromptTrace with representative keyword, classifier-based, and static-control defenses. SecurePromptTrace obtained the highest composite defense score of 96.4%, while preserving 94.6% legitimate task utility and restricting attack success to 2.8%. PromptShield-style detection and standalone DeBERTa-v3 achieved composite scores of 89.5% and 87.4%, respectively, but allowed more successful attacks because neither defense traced downstream data movement or tool causality. Combined static controls produced an 83.2% composite score and reduced attack success to 12.7%, although utility retention declined to 79.8%. Aho–Corasick and regex filters preserved 96.8% and 97.2% utility, respectively, but their lower composite scores and higher attack-success rates demonstrate inadequate resistance to paraphrased, multilingual, context-distributed, and cross-tool attacks. The findings support provenance-aware adaptive enforcement over isolated filtering or authorization under the identical held-out testing conditions.

Table 3 Comparative Security and Operational Performance of SecurePromptTrace and Baseline Defenses

Algorithm	Composite Defense Score (%)	Utility Retention (%)	Attack Success Rate (%)
SecurePromptTrace	96.4	94.6	2.8
PromptShield-style detector	89.5	93.1	10.9
Standalone DeBERTa-v3	87.4	92.4	13.6
Combined static controls	83.2	79.8	12.7
Aho–Corasick keyword filter	70.6	96.8	32.6
Regular-expression filter	67.4	97.2	36.8

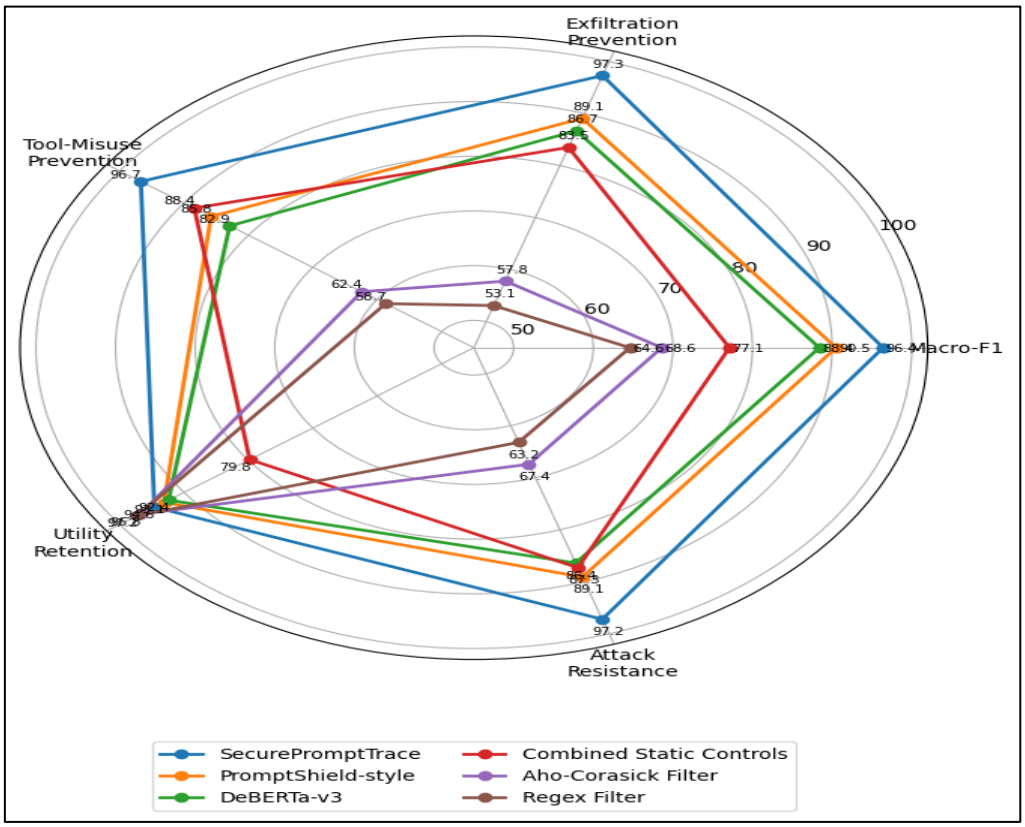

Fig 7 Radar-Chart Comparison of Keyword Filters, Classifier-Based Defenses, Static Controls, and SecurePromptTrace Across Five Security and Operational Metrics.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Figure 7 presents a five-axis radar comparison of the six defenses. SecurePromptTrace forms the largest and most balanced profile, reaching 96.4% macro-F1, 97.3% exfiltration prevention, 96.7% tool-misuse prevention, 94.6% utility retention, and 97.2% attack resistance. PromptShield-style detection follows with 90.5%, 89.1%, 85.8%, 93.1%, and 89.1%, respectively. DeBERTa-v3 records 88.4% macro-F1 and 86.7% exfiltration prevention but falls to 82.9% on tool-misuse prevention, confirming the limitation of content-only classification. Combined static controls achieve 88.4% tool-misuse prevention and 87.3% attack resistance, yet utility retention decreases to 79.8%. Aho–Corasick and regex filters retain 96.8% and 97.2% utility but produce much weaker security profiles, including exfiltration-prevention values of 57.8% and 53.1%. The graph therefore shows that SecurePromptTrace delivers the strongest overall balance by integrating semantic detection, provenance analysis, taint propagation, tool-intent alignment, privilege verification, and adaptive policy enforcement. Its near-uniform perimeter indicates that performance gains were not achieved by sacrificing legitimate task completion or precision.

➢ *Ablation Analysis, Statistical Validation, Computational Overhead, and Security–Utility Trade-Offs*

This subsection determines how individual SecurePromptTrace components contribute to overall performance. Ablation analysis removes one module at a time, statistical validation examines whether observed differences are credible, computational-overhead analysis measures additional latency and memory consumption, and security–utility analysis establishes whether stronger protection unnecessarily disrupts legitimate agent operations. Table 4 shows that the complete SecurePromptTrace configuration produced the strongest outcome. Its macro-F1 of 96.4% exceeded every ablated variant, while its 2.8% attack-success rate remained the lowest. Removing the Dynamic Prompt Provenance Graph caused the largest structural degradation, reducing macro-F1 to 91.2% and increasing attack success to 11.7%. Eliminating semantic classification produced a lower macro-F1 of 89.6%, confirming that graph reasoning cannot fully replace language analysis. Taint tracing and tool-intent alignment contributed because their removal increased successful attacks to 10.4% and 11.2%, respectively. Uncertainty calibration had the smallest accuracy effect but improved risk discrimination around enforcement thresholds. The full model added 38.6 ms latency and 12.8% memory overhead, representing moderate computational costs for stronger security. Bootstrap confidence intervals and paired McNemar testing supported the significance of the model's advantage.

Table 4 Ablation Performance and Computational Overhead of SecurePromptTrace

Model Configuration	Macro-F1 (%)	Attack Success Rate (%)	Latency/Memory Overhead (ms/%)
Complete SecurePromptTrace	96.4	2.8	38.6/12.8
Without Dynamic Prompt Provenance Graph	91.2	11.7	29.4/9.1
Without DeBERTa-v3 semantic classifier	89.6	12.9	24.8/7.6
Without sensitive-data taint tracing	92.1	10.4	31.2/10.2
Without tool-intent alignment	91.7	11.2	30.5/9.8
Without uncertainty calibration	93.0	8.8	34.1/11.4

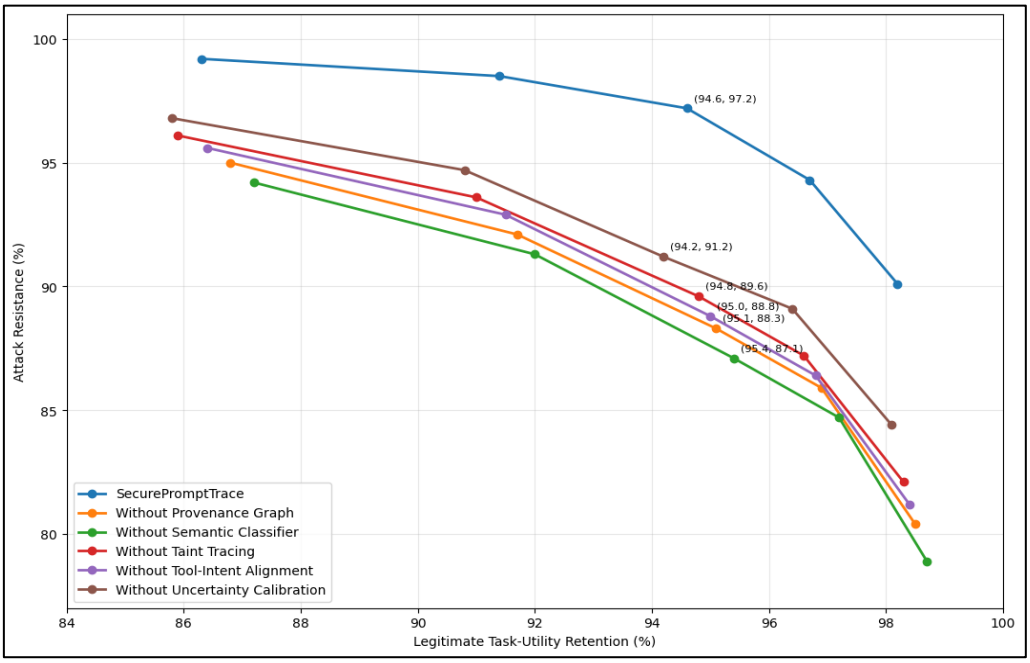

Fig 8 Security–Utility Trade-off Curves for the Complete SecurePromptTrace Algorithm and Five Ablated Configurations.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Figure 8 plots attack resistance against legitimate task-utility retention across progressively stricter enforcement thresholds. SecurePromptTrace occupies the dominant frontier: at its nominal operating point, it retains 94.6% utility while achieving 97.2% attack resistance. Removing uncertainty calibration produces the strongest ablated line, but its corresponding point falls to 94.2% utility and 91.2% resistance. Without taint tracing, resistance declines to 89.6% at 94.8% utility, showing reduced sensitivity to confidential-data movement. Excluding tool-intent alignment yields 88.8% resistance at 95.0% utility, while removing the provenance graph gives 88.3% resistance at 95.1% utility. The weakest configuration omits semantic classification and records 87.1% resistance despite retaining 95.4% utility. At stricter thresholds, all variants improve resistance but sacrifice utility; however, SecurePromptTrace remains above the other curves throughout. This separation indicates that the complete architecture obtains higher security without relying on disproportionate blocking, validating the intended security–utility balance. Its frontier therefore supports deployment under varying organizational risk tolerances.

V. CONCLUSIONS AND RECOMMENDATIONS

➢ *Summary of Principal Findings*

This study developed SecurePromptTrace as a provenance-aware runtime security algorithm for detecting prompt injection, sensitive-data exfiltration, and tool misuse in generative artificial intelligence systems. The proposed architecture combined a Dynamic Prompt Provenance Graph, DeBERTa-v3 semantic classification, relation-aware graph attention, confidential-data taint propagation, tool-intent alignment, privilege verification, execution-sequence deviation analysis, uncertainty estimation, and adaptive policy enforcement. The findings showed that this integrated design provided a more complete representation of generative AI security risk than defenses that inspected prompts, permissions, or tool calls independently. SecurePromptTrace achieved the strongest reported prompt-injection classification performance, with a macro-F1 score of 96.4%, precision of 96.8%, recall of 96.1%, AUROC of 98.7%, and AUPRC of 98.3%.

The algorithm also demonstrated superior protection against operational attacks. It prevented 97.3% of data-exfiltration attempts and 96.7% of tool-misuse attempts while restricting overall attack success to 2.8%. Its 94.6% legitimate task-utility retention indicated that stronger security was not obtained through indiscriminate blocking. Keyword filters performed poorly against paraphrased, multilingual, encoded, and context-distributed attacks, while standalone classifiers could identify suspicious language but could not reconstruct causal relationships among retrieved instructions, confidential data, tool arguments, and external destinations. Static access controls blocked explicitly unauthorized actions but failed when attackers chained individually permitted operations. Ablation analysis confirmed that every SecurePromptTrace component contributed to performance. Removing semantic classification produced the largest reduction in macro-F1, while removing the provenance graph, taint tracing, or tool-intent alignment substantially increased attack success. The complete model therefore achieved the strongest security–utility balance despite moderate latency and memory overhead.

➢ *Conclusions on the Performance and Security Effectiveness of SecurePromptTrace*

The findings establish that prompt injection in autonomous generative AI systems cannot be treated solely as a malicious-text classification problem. The principal security challenge arises from the transformation of untrusted instructions into model plans, sensitive-data access, tool arguments, and externally consequential actions. SecurePromptTrace addressed this challenge by preserving the causal relationships among instruction sources, authority levels, confidential variables, reasoning outputs, tool invocations, and execution destinations. Its performance advantage resulted from combining semantic and behavioural evidence rather than relying on a single detector. The Dynamic Prompt Provenance Graph identified when low-authority retrieved content exerted disproportionate influence over high-impact decisions, while DeBERTa-v3 detected semantic manipulation that bypassed lexical signatures.

The algorithm's security effectiveness was particularly evident in cross-tool scenarios. A conventional permission system may independently authorize reading a database and sending an email, yet fail to recognize that transmitting retrieved confidential records violates the authenticated objective. SecurePromptTrace detected such attacks through taint propagation, destination analysis, and tool-intent comparison. Its adaptive enforcement model also proved preferable to binary blocking because it selected proportionate responses, including sanitization, replanning, tool isolation, human approval, and termination. This preserved legitimate functionality while escalating intervention as composite risk increased. The low attack-success and false-positive rates, together with high utility retention, indicate that SecurePromptTrace can provide practical runtime protection for tool-integrated AI applications. Its moderate computational overhead represents an acceptable trade-off in environments where autonomous actions can expose financial records, patient information, credentials, proprietary documents, or administrative systems. SecurePromptTrace therefore provides a technically coherent and operationally relevant security architecture for generative AI agents operating across complex enterprise workflows.

➢ *Recommendations for Secure Generative AI and Autonomous Agent Deployment*

Organizations deploying generative AI agents should implement security controls at the complete execution-trace level rather than relying exclusively on input filtering or static permissions. Every instruction, retrieved document, memory entry, confidential value, generated plan, tool call, argument, and external destination should receive a provenance label that identifies its source, authority, sensitivity, and causal influence. Tool-integrated systems should separate trusted instructions from untrusted contextual data and prevent retrieved content from silently modifying system objectives.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

Sensitive information should be labelled at ingestion and traced through summarization, transformation, encoding, and transmission so that unauthorized flows can be intercepted before execution.

SecurePromptTrace should be deployed as an intermediary policy layer between the language model and external tools. High-impact capabilities, including financial transfers, record deletion, credential access, code execution, email transmission, cloud administration, and database modification, should require stricter risk thresholds than read-only or reversible actions. Organizations should calibrate adaptive enforcement thresholds using domain-specific validation data and defined risk tolerance. Low-risk anomalies may be sanitized or replanned automatically, while high-risk operations should require human approval or be blocked. Model developers should maintain attack-template-disjoint evaluation sets containing direct, indirect, multilingual, encoded, long-context, retrieval-poisoning, memory-poisoning, cross-tool exfiltration, and privilege-chaining scenarios. Production monitoring should track attack-success rate, exfiltration-prevention rate, tool-misuse-prevention rate, false-positive rate, task-utility retention, latency, and memory consumption. Security teams should also retain auditable provenance records for incident investigation without storing unnecessary sensitive prompt content. Finally, access controls should be dynamically combined with intent, data-flow, behavioural, and uncertainty evidence. Static authorization should remain a foundational control, but it should not be treated as sufficient evidence that an authorized tool sequence is legitimate.

- *Study Limitations and Directions for Future Research*

The evaluation was limited by the selected attack categories, agent workflows, model architectures, tools, and benchmark assumptions. Although the dataset covered direct, indirect, obfuscated, multilingual, context-based, memory-poisoning, data-exfiltration, and privilege-chaining attacks, no finite corpus can represent every adaptive strategy available to a determined adversary. Attackers may develop new methods that manipulate provenance labels, exploit tool-description ambiguity, distribute malicious instructions across multiple agents, or encode confidential information through channels not represented during training. Performance may also vary across language models because instruction-following behaviour, context handling, tool-selection policies, and safety alignment differ substantially among model families. The reported computational overhead may increase in applications that maintain larger provenance graphs, longer contexts, or high-frequency tool interactions.

Future research should evaluate SecurePromptTrace across additional commercial and open-source language models, multilingual environments, multimodal agents, and domain-specific applications. The provenance framework should be extended to image, audio, video, code, and structured database inputs, where malicious instructions may be embedded outside ordinary text. Further work should investigate graph compression, incremental attention, selective taint propagation, and hardware acceleration to reduce runtime overhead. Federated learning could enable organizations to share attack representations without centralizing confidential prompts or security logs. Continual-learning mechanisms should be developed carefully to incorporate new attack patterns without allowing adversaries to poison the detector itself. Future studies should also examine coordinated multi-agent attacks in which one compromised agent influences another through delegated tasks or shared memory. Formal verification of enforcement policies, calibrated uncertainty under distribution shift, interpretable graph explanations, and adversarial robustness testing should receive particular attention. Longitudinal production studies are also required to determine how SecurePromptTrace affects operator trust, alert fatigue, legitimate workflow completion, and security effectiveness over extended deployment periods.

REFERENCES

[1]. Adewale, L. D. (2026). Digital evidence chains for PPAP assurance: AR-guided data capture, AI-verified documentation, and continuous audit automation for secure multi-tier supplier traceability in Industry 4.0 manufacturing. *International Journal of Multidisciplinary Evolutionary Research, 7*(1), 43–55. https://doi.org/10.54660/ijmer.2026.7.1.43-55

[2]. Adewale, L. D. (2026). Smart factories, smarter evidence: Reinventing quality assurance for U.S. manufacturing competitiveness. *International Journal of Multidisciplinary Futuristic Development, 7*(1), 9–18. https://doi.org/10.54660/IJMFD.2026.7.1.09-18

[3]. Agyekum, B., Seffah-Duodu, D., & Gloria, A. (2025). Federated industrial IoT threats detection. *International Journal of Scientific Research and Modern Technology, 4*(12), 226–244. https://doi.org/10.38124/ijsrmt.v4i12.1552

[4]. Alotaibi, A., Mughus, R., & Ahmed, M. (2026). A red teaming framework for large language models: A case study on faithfulness evaluation. *Software Quality Journal, 34*, Article 42. https://doi.org/10.1007/s11219-026-09779-y

[5]. Alzahrani, A. (2026). PromptGuard: A structured framework for injection-resilient language models. *Scientific Reports, 16*, Article 1277. https://doi.org/10.1038/s41598-025-31086-y

[6]. Ayoola, V. B., Ugoaghalam, U. J., Idoko, I. P., Ijiga, O. M., and Olola, T. M. (2024). Effectiveness of social engineering awareness training in mitigating spear phishing risks in financial institutions from a cybersecurity perspective. *Global Journal of Engineering and Technology Advances, 20*(3), 94–117. https://doi.org/10.30574/gjeta.2024.20.3.0164

[7]. Ayoola, V. B., Ugochukwu, U. N., Adeleke, I., Michael, C. I., Adewoye, M. B., and Adeyeye, Y. (2024). Generative AI-driven fraud detection in health care: Enhancing data loss prevention and cybersecurity analytics for real-time protection of patient records. *International Journal of Scientific Research and Modern Technology, 3*(11), 89–107. https://doi.org/10.38124/ijsrmt.v3i11.112

[8]. Boiko, D. A., MacKnight, R., Kline, B., & Gomes, G. (2023). Autonomous chemical research with large language models. *Nature, 624*, 570–578. https://doi.org/10.1038/s41586-023-06792-0

[9]. Briskinfosec, (2026). The Hidden Risk of Data Leakage in AI Code Assistants https://www.briskinfosec.com/blogs/blogsdetail/the-hidden-risk-of-data-leakage-in-ai-code-assistants

[10]. Cai, Y. (2026). Prompt injection attacks on educational large language models for higher and vocational education. *Scientific Reports, 16*, Article 15594. https://doi.org/10.1038/s41598-026-46563-1

[11]. Chen, K., Zhou, X., Lin, Y., Feng, S., Shen, L., & Wu, P. (2025). A survey on privacy risks and protection in large language models. *Journal of King Saud University–Computer and Information Sciences, 37*, Article 163. https://doi.org/10.1007/s44443-025-00177-1

[12]. Clusmann, J., Ferber, D., Wiest, I. C., Schneider, C. V., Brinker, T. J., Foersch, S., Truhn, D., & Kather, J. N. (2025). Prompt injection attacks on vision language models in oncology. *Nature Communications, 16*, Article 1239. https://doi.org/10.1038/s41467-024-55631-x

[13]. Debenedetti, E., Shumailov, I., Fan, T., Hayes, J., Carlini, N., Fabian, D., Kern, C., Shi, C., Terzis, A., & Tramèr, F. (2025). *Defeating prompt injections by design*. arXiv:2503.18813.

[14]. Debenedetti, E., Zhang, J., Balunović, M., Beurer-Kellner, L., Fischer, M., & Tramèr, F. (2024). AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. *Advances in Neural Information Processing Systems, 37*.

[15]. Dzhaliuk, N., Sabodashko, D., Khoma, V., Khoma, Y., Kolchenko, V., and Podpora, M. (2026). Comparative evaluation of machine learning methods for protecting LLMs from prompt injection attacks. *International Journal of Information Security, 25*, Article 109. https://doi.org/10.1007/s10207-026-01264-8

[16]. Gabla, E. S., Peter-Anyebe, A. C., & Ijiga, O. M. (2025). Assessing machine-learning-enabled anomaly detection models for real-time cyberattack mitigation in optical fiber communication systems. *World Journal of Advanced Engineering Technology and Sciences, 17*(2), 1–17. https://doi.org/10.30574/wjaets.2025.17.2.1454

[17]. Hagendorff, T., Derner, E., & Oliver, N. (2026). Large reasoning models are autonomous jailbreak agents. *Nature Communications, 17*, Article 1435. https://doi.org/10.1038/s41467-026-69010-1

[18]. He, P., Gao, J., & Chen, W. (2021). *DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing*. arXiv:2111.09543.

[19]. Ibokette, A. I., Ogundare, T. O., Anyebe, A. P., Alao, F. O., Odeh, I. I., and Okafor, F. C. (2024). Mitigating maritime cybersecurity risks using AI-based intrusion detection systems and network automation during extreme environmental conditions. *International Journal of Scientific Research and Modern Technology, 3*(10), 65–91. https://doi.org/10.38124/ijsrmt.v3i10.73

[20]. Idika, C. N., James, U. U., Ijiga, O. M., & Enyejo, L. A. (2023). Digital twin-enabled vulnerability assessment with Zero Trust policy enforcement in smart manufacturing cyber-physical systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 9*(6), 475–499. https://doi.org/10.32628/CSEIT23906189

[21]. Idoko, I. P., Ijiga, O. M., Enyejo, L. A., Akoh, O., Ebiega, G. I., Odeyemi, M. O., Olatunde, T. I., and Olajide, F. I. (2024). Integrating superhumans and synthetic humans into the Internet of Things and ubiquitous computing: Emerging AI applications and their relevance in the U.S. context. *Global Journal of Engineering and Technology Advances, 19*(1), 6–36. https://doi.org/10.30574/gjeta.2024.19.1.0055

[22]. Igba, E., Olarinoye, H. S., Ezeh, N. V., Sehemba, D. B., Oluhaiyero, Y. S., & Okika, N. (2025). Synthetic data generation using generative AI to combat identity fraud and enhance global financial cybersecurity frameworks. *International Journal of Scientific Research and Modern Technology, 4*(2), 1–19. https://doi.org/10.5281/zenodo.14928919

[23]. Ijiga, O. M., Idoko, I. P., Ebiega, G. I., Olajide, F. I., Olatunde, T. I., and Ukaegbu, C. (2024). Harnessing adversarial machine learning for advanced threat detection: AI-driven strategies in cybersecurity risk assessment and fraud prevention. *Open Access Research Journal of Science and Technology, 11*(1), 1–24. https://doi.org/10.53022/oarjst.2024.11.1.0060

[24]. Ijiga, O. M., Okika, N., Balogun, S. A., Enyejo, L. A., and Agbo, O. J. (2025). A comprehensive review of federated learning architectures for insider threat detection in distributed SQL-based enterprise environments. *International Journal of Innovative Science and Research Technology, 10*(7), 536–550. https://doi.org/10.38124/ijisrt/25jul392

[25]. Jalan, P., Abishethvarman, V., Chandna, B., & Naseem, U. (2026). Survey on LLM safety: Attacks, defenses, alignment, metrics, and guardrails. *Machine Learning, 115*, Article 130. https://doi.org/10.1007/s10994-026-07060-8

[26]. James, U. U. (2026). Multiscale Signal Processing Framework for Zero Trust Security in Next-Generation Wireless Networks *International Journal of Wireless and Mobile Networks* DOI/Link: https://aircconline.com/ijwmn/V18N3/18326ijwmn01.pdf

[27]. James, U. U., and Akujuobi, C. M. (2026). Multiscale signal processing framework for Zero Trust security in next-generation wireless networks. *International Journal of Wireless and Mobile Networks, 18*(3), 1–21. https://doi.org/10.5121/ijwmn.2026.18301

[28]. James, U. U., Olarinoye, H. S., Uchenna, I. R., Idika, C. N., Ngene, O. J., Ijiga, O. M., and Itemuagbor, K.

(2025). Combating deepfake threats using X-FACTS explainable CNN framework for enhanced detection and cybersecurity resilience. *Advances in Artificial Intelligence and Robotics Research, 1*, 41–64. https://doi.org/10.4236/airr.2025.11004

[29]. James, U. U., Olarinoye, H. S., Uchenna, I. R., Idika, C. N., Ngene, O. J., Ijiga, O. M., & Itemuagbor, K. (2025). Combating deepfake threats using X-FACTS explainable CNN framework for enhanced detection and cybersecurity resilience. *Advances in Artificial Intelligence and Robotics Research, 1*, 41–64. https://doi.org/10.4236/airr.2025.11004

[30]. James, U. U., Salami, E. O., & Enyejo, L. A. (2025). Real-time policy orchestration for cybersecurity risk management in GRC-aligned financial technology infrastructures. *International Journal of Innovative Science and Research Technology, 10*(8). https://doi.org/10.38124/ijisrt/25aug1021

[31]. James, U. U., Salami, E. O., & Enyejo, L. A. (2025). Real-time policy orchestration for cybersecurity risk management in GRC-aligned financial technology infrastructures. *International Journal of Innovative Science and Research Technology, 10*(8). https://doi.org/10.38124/ijisrt/25aug1021

[32]. James, U. U., Salami, E. O., & Enyejo, L. A. (2025). Real-time policy orchestration for cybersecurity risk management in GRC-aligned financial technology infrastructures. *International Journal of Innovative Science and Research Technology, 10*(8). https://doi.org/10.38124/ijisrt/25aug1021

[33]. Ko, R., Chung, J., Zheng, S., Xiao, C., Kim, T.-W., Onizuka, M., & Shin, W.-Y. (2026). Seven security challenges in cross-domain multi-agent LLM systems. *npj Artificial Intelligence*. https://doi.org/10.1038/s44387-026-00128-9

[34]. Kpogli, S. A., Onwuzurike, M. A., & Ayoola, V. B. (2024). Optimizing Educational Return on Investment Through AI-Driven Curriculum Adaptation and Performance-Based Resource Allocation Models. International Journal of Scientific Research and Modern Technology, 3(9), 141–156. https://doi.org/10.38124/ijsrmt.v3i9.1352

[35]. Kpogli, S. A., Onwuzurike, M. A., & Enyejo, J. O. (2024). Integrating artificial intelligence and learning sciences to reduce cognitive load and achievement gaps in data-driven K–12 instructional systems. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10*(6), 2569–2589. https://doi.org/10.32628/CSEIT25113575

[36]. Kwarteng, R. A., Idoko, I. P., Ijiga, O. M., & Enyejo, L. A. (2020). Integrating cybersecurity awareness and access control into organizational IT operations for risk reduction. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 6*(1), 243–261. https://doi.org/10.32628/CSEIT23906128

[37]. Milani, A., Franzoni, V., & Florindi, E. (2026). Indirect prompt injection in large language models. *Neural Computing and Applications, 38*, Article 530. https://doi.org/10.1007/s00521-026-12266-x

[38]. Okika, N., Okoh, O. F., & Etuk, E. E. (2025). Mitigating insider threats and social engineering tactics in advanced persistent threat operations through behavioral analytics and cybersecurity training. *International Journal of Advance Research Publication and Reviews, 2*(3), 11–27.

[39]. Ononiwu, M., Azonuche, T. I., Imoh, P. O., & Enyejo, J. O. (2024). Evaluating blockchain content monetization platforms for autism-focused streaming with cybersecurity and scalable microservice architectures. *Iconic Research and Engineering Journals, 8*(1), 722–740.

[40]. Onwuzurike, M. A. & Enyejo, J. O. (2026). A Business Intelligence Framework for AI Powered Educational Platforms Linking Learning Analytics to Strategic Decision Making in K-12 Schools International Journal of Recent Research in Commerce Economics and Management (IJRRCEM) Vol. 13, Issue 2, pp: (21-42), DOI: https://doi.org/ 10.5281/zenodo.19510038

[41]. Onwuzurike, M. A. (2023). Human-Centered Design of Intelligent Tutoring Systems Integrating Behavioral Analytics and Inclusive Pedagogical Principles for Early Learners International Journal of Scientific Research in Science, Engineering and Technology Volume 10, Issue 3, Page Number 720-738, doi : https://doi.org/10.32628/IJSRSET2310330

[42]. Onwuzurike, M. A., & Kpogli, S. A. (2022). Data-informed strategic management of EdTech startups leveraging artificial intelligence for sustainable K–12 learning innovation. *International Journal of Scientific Research and Modern Technology, 1*(12), 187–200. https://doi.org/10.38124/ijsrmt.v1i12.1117

[43]. Onwuzurike, M. A., & Kpogli, S. A. (2025). Predictive modeling of student engagement and behavioral outcomes using machine learning techniques in technology-enhanced classrooms. *International Journal of Scientific Research in Humanities and Social Sciences, 2*(6), 58–79. https://doi.org/10.32628/IJSRHSS2525135

[44]. Onwuzurike, M. A., & Raphael, F. O. (2025). Ethical governance models for artificial intelligence deployment in K–12 education: Balancing algorithmic personalization, accountability and child protection policy. *International Journal of Scientific Research and Modern Technology, 4*(8), 193–208. https://doi.org/10.38124/ijsrmt.v4i8.1271

[45]. Onwuzurike, M. A., Enyejo, J. O., & Peter-Anyebe, A. C. (2026). Design and evaluation of real-time adaptive learning algorithms for personalized K–12 curriculum optimization using student performance analytics. *World Journal of Advance Multidisciplinary Research, 3*(3), 21–36. https://doi.org/10.5281/zenodo.19131296

[46]. Onwuzurike, M. A., Igba, E. (2023). Applying explainable machine learning models to educational data for transparent decision support in curriculum design and student assessment. Journal of Frontiers

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug331

in Multidisciplinary Research. 2023;4(1):585–599. doi:10.54660/.JFMR.2023.4.1.585-599

[47]. Shilov, I., Meeus, M., & de Montjoye, Y.-A. (2026). The mosaic memory of large language models. *Nature Communications, 17*, Article 2142. https://doi.org/10.1038/s41467-026-68603-0

[48]. Srinivasan, J., Regi, S. A., Anbarasan, A. K., Suresh, A., Vetriselvi, T., & Venu, S. (2026). Detection and analysis of prompt injection in Indian multilingual large language models. *Scientific Reports, 16*, Article 16208. https://doi.org/10.1038/s41598-026-43883-0

[49]. Ussher-Eke, D., Onoja, D. A., Ijiga, O. M., & Enyejo, L. A. (2025). Strengthening human resource compliance and ethical oversight through cybersecurity awareness and policy enforcement. *International Journal of Management and Commerce Innovations, 13*(1), 376–393. https://doi.org/10.5281/zenodo.16539315

[50]. Wang, W., Liu, J., Gu, B., Wang, C., Qu, Y. N., & Feng, D. (2026). Securing LLM-in-the-loop software for empirical study of risks, mitigations, and utility trade-offs in a safety-critical case. *Empirical Software Engineering, 31*(4), Article 87. https://doi.org/10.1007/s10664-026-10820-8

[51]. Webb, T., Mondal, S. S., & Momennejad, I. (2025). A brain-inspired agentic architecture to improve planning with LLMs. *Nature Communications, 16*, Article 8633. https://doi.org/10.1038/s41467-025-63804-5

[52]. Yang, Y., Jin, Q., Huang, F., and Lu, Z. (2025). Adversarial prompt and fine-tuning attacks threaten medical large language models. *Nature Communications, 16*, Article 9011. https://doi.org/10.1038/s41467-025-64062-1