

Secure Tool Invocation for LLM Agents Using Context-Aware Policy Enforcement

Ravindra Babu Annam

Hyderabad, Telangana-500090, India.

Abstract- The growing deployment of Large Language Model (LLM) agents coupled with third-party tools, APIs, databases, and cloud computing resources creates additional security issues due to the potential of an autonomous agent invoking a tool request which could be unauthorized, insecure, or unsuitable for the context. Current access control solutions are concentrated on static data authorization, fine-grain resource protection, and general AI security, but fail to address dynamic authorization of LLM-agent tool invocation. This paper proposes the development of a Context-aware Secure Tool Invocation Framework (CASTIF), which introduces a security enforcement mechanism for LLM agents during external tool invocation. The proposed framework considers agent identity, operation invoked, sensitivity of the target resource, user authorization, security policies applicable to this situation, context, and invocation history prior to executing any requested tool. In addition to this, the framework also conducts secondary verification on medium-risk requests, sanitizes tool parameters, verifies tool responses, maintains an audit log, and updates the contextual risk profile after each call. The evaluation is done by comparison of CASTIF with AI-Zahrani Model (AZM) and PEHR in terms of authorization accuracy, prevention of unauthorized invocation, context-awareness, false authorizations, decision latencies, and completeness of audits. An example-based evaluation suggests that CASTIF can offer better authorization accuracy, prevent unauthorized invocations, minimize false authorization, and conduct complete auditing.

Keywords: Large Language Models, LLM Agents, Secure Tool Invocation, Context-Aware Authorization, Policy Enforcement, Risk-Based Access Control, Tool Security, Autonomous Agents, Security Policies, Audit Logging

I. INTRODUCTION

The current state of Large Language Models (LLMs) represents the transformation of models designed mostly for generating natural-language replies into intelligent agents able to interact with other tools and computational infrastructure [1]. Modern LLM agents are able to make use of application programming interfaces (APIs), database queries, file operations, computation tasks, cloud communication, and perform any actions on the user's behalf [2]. This possibility allows substantially increasing the practical value of such systems as an agent can do more than generating information and perform some actions within an external environment [3]. Nevertheless, when an LLM gains the possibility to operate with tools, then the problem of security shifts from securing the input/output of the model itself to controlling the actions of the agent [4]. Any request to use an external tool may lead to access, modification, sending, or performing of privileged actions with some resources. Thus, some mechanisms are

required to control and validate each invocation of the tool [5]. The proposed work tries to solve this problem using a context-aware security mechanism that works as an intermediary between the LLM agent and the tool execution [6].

The security ramifications of artificial intelligence systems have been given due consideration owing to the fact that AI technology can be used maliciously, as well as it could result in unsafe consequences [7]. There are some potential risks identified from the previous research, including integrity attacks, prompt-based attacks, social engineering, misinformation, hacking, and other types of attacks associated with the malicious use of AI systems [8]. These examples prove that it is necessary to take into account not just the model itself but also the environment in which an AI system functions [9]. The connection of an LLM agent with outside sources creates yet another point of exposure, since the malicious or altered instructions might affect the agent and make it perform actions that cannot be considered safe [10]. General AI-security techniques

are a great foundation that enables us to identify such risks [11]. However, they do not provide a way to evaluate tool requests in their specific contexts, concerning the targeted resource, authorization, and potential risks involved [12].

Access control methods are typically employed to decide whether the user or entity is allowed to access a certain resource [13]. This access control method is crucial to securing information systems. However, the dynamic nature of LLM agents introduces another dimension to this process [14]. A particular agent can be authorized to access a certain service, whereas the current operation can still be considered inapplicable due to the ongoing dialogue, sensitivity of the resource, prior actions, environment, or other factors [15]. Data sharing authorization and subscription policies discussed in the literature under review here are policy-based methods of controlling access to shared resources [16]. However, these methods focus more on data sharing authorization and not on action authorization generated dynamically by an LLM agent. CASTIF expands upon the policy-based access control paradigm by analysing the full execution context before allowing an external operation to occur.

A finer level of authorization has also been explored in the context of distributed healthcare and EHR systems. For instance, blockchain-enabled methods of EHR sharing involve access control, encryption, and consensus mechanisms for securing sensitive data [17]. Such approaches illustrate the significance of fine-grained and policy-based authorization of sensitive data. However, such authorizations are focused mainly on the access and do not solve the problem of an LLM agent selection and usage of the tools. The authorization in the agent-based system should be based not only on the identification of the requester but also on the identification of the invoked tool, operation, target resource, reason for performing the operation, and already made operations by the agent [18]. LLM interplay with the environment also gives rise to another key security issue. LLM-based cyber deception studies have explored the idea of managing LLM input/output controls in relation to LLM interplay with simulated computing environments [19].

These findings highlight the significance of managing the interactions of LLMs to minimize unintended consequences and security vulnerabilities. But the management of inputs/outputs is just one part of the process of authorization; it will not allow us to decide about the authorization of an agent to use a certain external tool or to carry out a specific action [20]. It is significant since the invocation of a tool means some action performed which has some effect on the external world. Thus, CASTIF regards the invocation of the tool as a security-managed action and imposes an enforcement layer between the agent and the execution environment.

This approach solves the above problem through the creation of an execution context for every tool request. The execution context is made up of the user request at the moment, the state of conversation, the requesting agent, actions carried out previously, the information about target resource, and the environment. Subsequently, this framework determines whether the agent requesting is authenticated and also checks whether the requested tool is registered and available. The operation being requested is categorized based on actions like read, write, update, delete, execute, transfer, and administration. After that, a sensitivity rating for the target resource is created, and then the policy engine finds the security policies relevant to the agent, tool, operation, resource, and the context. Yet another crucial aspect of CASTIF is that it involves the mechanism for contextual risk assessment. Information about authorization does not necessarily reflect security risk of the specific invocation. In this context, the framework takes into account resource sensitivity, operation risk, contextual deviation, historical behavior, and policy violation risk in assessing the contextual risk score. The risk score obtained after normalization is combined with the results of authorization and validation to give any of the following three actions: ALLOW, REVIEW, or DENY. The low-risk invocation meeting the necessary security criteria will be allowed to execute while those of medium-risk that remain uncertain will be further verified. High-risk invocations or invocations violating the policy will be denied. This multi-layered approach gives a more

flexible security boundary as compared to a simple two-state authorization scheme.

The enforcement of security in CASTIF takes place even after the decision on authorization. The parameters of the tools are sanitized prior to the execution so that there is less likelihood of sending unsafe parameters or unauthorized resource identifiers to external tools. The operations that are authorized get executed through a trusted execution layer as opposed to allowing the LLM agent to communicate with the external tool. The output from the tool gets evaluated for any forbidden information or policy breaches before sending it back to the agent or user. This bidirectional control system offers protection before and after the execution of the tool. This architecture is especially necessary for LLM agents since security cannot be achieved only based on the accuracy of the model's request for the tool.

CASTIF also accounts for history of behaviors in the authorization process. All requests that have been allowed, disallowed, and those that have undergone review are stored in an audit log. Such behaviors can later on be analysed for any patterns of anomalous behavior, unusual access sequences of resources, or attempts at privilege escalation. CASTIF then uses the most recent information about invocation of such behaviors to update the contextual risk profile of the agent, making it possible to include past behavior in the authorization decision rather than making decisions based on individual requests alone. In effect, it becomes possible to develop a dynamic security policy where the authorization state adapts based on the behavior of the agent.

1.1 Hypothesis of the Research

- Context-based policy enforcement increases the precision of authorization decisions for LLM-agent tool invocations relative to traditional policy-based authorization mechanisms.
- The use of resource-awareness, contextual information, and invocation history decreases the risk of unauthorized execution of tools and erroneous authorization of risky invocations.
- A tri-level decision model of ALLOW–REVIEW–DENY along with the validation of parameters

and responses improves the overall security of LLM-agent tool usage.

1.2 Objectives of the Research

- Design a Context-Aware Secure Tool Invocation Framework (CASTIF) to control LLM-agent interactions with external tools and resources.
- Develop an authorization model that will take into account the identity of the agent, operation type, sensitivity of the target resource, user authorization, security policies, and environmental context.
- Develop a risk assessment framework that will classify the tool request based on decision of ALLOW, REVIEW, and DENY.
- Include invocation history analysis in the authorization model to detect abnormal sequences of invocations and repeated requests, and unusual access to resources.
- Facilitate secure invocation through tool parameter sanitization, secure execution of tools, response validation, and auditing.
- Evaluate CASTIF by the following metrics: authorization accuracy, unauthorized invocation prevention, contextual adaptiveness, false authorization rate, decision delay, and auditing completeness.

II. LITERATURE REVIEW

A subscription-based data sharing approach was put forward by Al-Zahrani et al. [1] through the integration of blockchain technology into Data as a Service (DaaS). The approach addresses the issues related to security, privacy, access control, data quality and incentives for data owners in data-sharing settings. With the use of the approach, users will be able to subscribe to data providers for a particular period of time. Flat Rate Pricing (FRP), Usage-Based Pricing (UBP), and hybrid pricing schemes are put in place for data monetization in a safe and sustainable manner. Blockchain is used to ensure transparency, immutability and secure transactions. This study shows how essential it is to have controlled and policy-driven access to the data that is being shared; however, the research is more concerned about data sharing and subscription authorization and not dynamic authorization of LLM

agents and their tool invocations. The current research takes this study further by adding policy enforcement to ensure secure tool invocations.

Pan et al. [2] have developed a patient controlled electronic health records (EHR) sharing system that leverages cloud computing and blockchain technology in order to offer access control services based on fine-grained access to healthcare data. The authors store medical abstracts and access rules in the blockchain to preserve integrity and make sure they cannot be altered in any way. Furthermore, attribute-based encryption and multi-keyword encryption methods are used in order to offer fine-grained access control over EHRs. Additionally, a node state checkable PBFT (sc-PBFT) consensus algorithm is used in order to detect any potentially malicious nodes on the blockchain and minimize consensus latency. This paper demonstrates the significance of fine-grained policy driven access authorization and access enforcement in distributed systems; nevertheless, its access control system is developed to support static EHRs and does not handle the dynamic authorization of LLM agents for external API call. In contrast, the current work introduces the idea of policy enforcement for the context of LLM agents in order to decide on the invocation of the tool based on the context and security policies.

Blauth et al. [3] conducted a thorough overview of the misuse and abuse of AI systems with respect to malicious activities and classified different major threats associated with AI into two categories – malicious abuse and malicious use, which include integrity attacks, unintended consequences of AI, membership inference attack, social engineering, misinformation, hacking, and autonomous weapon systems. The paper illustrates the vulnerabilities of AI models that can be exploited by the malicious parties and stresses the significance of the implementation of certain governance and security policies as well as mechanisms for reducing the risks associated with AI. However, despite being a solid base for the understanding of the security issues related to the usage of autonomous AI systems, it is mainly concerned with the security of the whole system and does not provide information on the

mechanism of controlling individual actions taken by agents. This limitation will be addressed in the proposed work with the development of a framework for the secure invocation of the LLM tools by considering security policies and contextual data before performing any operation.

Ragsdale and Boppana [4] examined the application of Large Language Models (LLMs) in adaptive cyber deception with intelligent honeypots, in order to achieve realistic threat engagement, while mitigating the risks associated with deployment of live computer systems. The researchers suggested controlling the inputs and outputs of an LLM in order to mitigate the nondeterministic and token consuming nature of LLMs during terminal interaction simulation in order to make the simulations more machine-like and to save tokens during multi-stage attacks. The research highlights the significance of controlling LLM interaction with external entities in order to avoid undesirable behavior of an LLM and the resulting security risks. Yet, the approach mainly concentrates on controlling the inputs and outputs of LLMs inside honeypots, but does not offer any method for context-aware authorization of actions performed by an LLM. Proposed research expands this approach by proposing the context-aware policy enforcement mechanism for the purpose of tool invocation in a secure way, thus each tool invocation is going to be analysed based on the context, authorization policies, resource and operation constraints.

Koubaa et al. [5] conducted an extensive survey on ChatGPT, analysing such aspects as technical features, abilities, use cases, limitations, implications, and future research challenges related to ChatGPT. The authors constructed a taxonomy of current ChatGPT research and suggested multiple directions for further research in relation to large language model development and application. The research serves as a general basis for studying the development and abilities of ChatGPT-based systems and the need to analyse the challenges related to their deployment in practice. Nonetheless, the research mostly provides a general overview of ChatGPT and its research field and does not include the topic of secure interaction of LLM agents with

external tools and the authorization of agent actions. The proposed work extends the above research by developing context-aware policy enforcement for secure tool invocation, where an action requested by an LLM agent is analysed in accordance with contextual data and pre-defined security policies before being performed.

Gupta et al. [6] explored the impact of Generative AI models like ChatGPT on cybersecurity and privacy with regards to vulnerabilities such as jailbreaks, reverse psychology, prompt injection attacks that would allow malicious users to circumvent protection mechanisms of the model. In addition, the paper explored the possible misuses of GenAI for social engineering, phishing, automated hacking, attack payload creation, malware creation, as well as potential GenAI uses for threat intelligence, security automation, attack detection, incident response, and secure code generation. The presented results prove that security control is necessary for preventing unauthorized or malicious operations on LLM-based systems. At the same time, this research mainly focuses on studying vulnerabilities and cybersecurity use cases for GenAI but does not provide the mechanism for dynamic authorization of tool invocations by LLM-based agents in line with security requirements. In this regard, the current work aims to fill this gap through the context-aware policy enforcement when each tool invocation is verified based on agent's context, requested operation, resource sensitivity, and authorization policies.

2.1 Problem Statement

However, providing the ability to directly use external tools for an LLM agent is dangerous since an issued request could be illegitimate, insecure, and inappropriate in the current environment. There are approaches presented in the literature that allow for controlled data sharing, access control, and even more general AI-security solutions; however, they are not aimed at providing specific dynamic authorization of tool invocation by an LLM agent. The primary challenge thus consists in developing an approach able to evaluate each particular request from an LLM agent regarding the execution of external tools taking into consideration static

permissions and the current context, agent, operation, resource, user authorization, policies, historical behavior of tool invocations, and contextual risk. Additionally, it should be able to protect against passing inappropriate parameters, validate responses from the external tools, provide an audit trail, and use historical information about invocations in future decisions about authorization. This challenge is addressed by CASTIF through developing a special policy enforcement layer between the LLM agent and external tool execution.

2.2 Research Gaps

There have been several achievements in the area of secure data sharing, fine-grained access controls, authorization using blockchains, AI security analysis, and controlled interaction with LLMs. Yet, existing approaches only address the issues of protecting data assets, granting static/fine-grained access permissions, detecting potential risks posed by AI systems, or controlling interaction within certain environments. There is no unified mechanism for authorizing invocations of the tools offered by specific LLM-agents based on certain contextual factors. Specifically, the existing data sharing and EHR solutions do not take into account the agent identity, the tool used, the nature of operation, sensitivity of the resource, contextual conversational factors, invocation history, and contextual risk while making the decision regarding the execution of the operation. Moreover, there is no mechanism for combining ALLOW–REVIEW–DENY approach with secondary checks, parameter sanitization, response validation, auditing, and contextual risk profile update. From the current literature, there is the identification of security issues that affect the AI system and LLM, which include prompt injection and misuse of the LLM system in an unethical manner, but the review of existing literature does not offer a full-fledged enforcement pipeline to control the LLM agent during external tool invocation. Therefore, the research gap to be covered by this study is the absence of a context aware and risk-aware authorization process for invoking the LLM-agent in a secure manner. CASTIF fulfills the research gap by placing the security enforcement layer between the LLM agent and the external tool and evaluating each invocation with contextual information.

III. PROPOSED MODEL

The proposed approach involves a Context-Aware Secure Tool Invocation Framework (CASTIF) for Large Language Model (LLM) agents. The main idea is to ensure the safety of tool executions by assessing every tool request prior to reaching an external API, database, cloud service, or computational resource. The relevant literature review included in the paper suggests that the current approaches cover controlled data sharing, fine-grained authorization, AI security, and risks of interacting with LLMs. However, they do not consider the issue of dynamic authorization of individual LLM-agent tool invocations depending on context. Thus, the presented framework includes LLM-generated tool requests, context extraction, resource sensitivity analysis, policy assessment, risk score calculation, authorization, and secure execution in one enforcement chain. In particular, the decision regarding every LLM invocation is made taking into account such factors as the agent identity, the current conversation/task context, the requested tool, its operation, target resource, user authorization, security policy, and invocation history. Such a solution is an obvious continuation of the context-aware policy-enforcement line mentioned in the literature review: every requested operation is assessed taking into account contextual information, authorization policies, resources, and operation constraints.



Fig. 1. Proposed Context-Aware Secure Tool Invocation Framework (CASTIF)

The suggested architectural setup as illustrated in Figure 1 creates an environment-aware security enforcement layer in between the LLM Agent and the tools being used. The proposed system assesses

the identity of the agent, the operation being done, sensitivity of the resources, user's authority to carry out the activity, security policies involved, contextual factors and past behaviors of the agent using policy-based and contextual assessment of the risk level. Based on this assessment, each tool request is either ALLOWED, REVIEWED, or DENIED. The authorized requests are sanitized and executed in the trusted execution environment, and the results are validated before they are sent back to the LLM Agent.

Algorithm: Context-Aware Secure Tool Invocation

Input: User request U , LLM agent A , available tools T , security policies P , protected resources R

Output: Authorized tool response O , or a blocked/review decision

Step 1: Receive User Request

The system gets the user request U and sends the request to the LLM agent. The LLM agent analyzes the request and decides if an external tool needs to be invoked.

$$U = \{u_1, u_2, \dots, u_n\} \quad (1)$$

Where u_i represents the individual tokens or semantic components of the user request.

Step 2: Generate Tool Invocation Request

The LLM generates a well-formed request to invoke a particular tool with certain operations on a specific target resource.

$$I = \{A, T_k, O_p, R_j, \theta\} \quad (2)$$

Where A is the requesting agent, T_k is the selected tool, O_p is the requested operation, R_j is the target resource, and θ represents tool parameters.

Step 3: Construct Execution Context

The framework creates a context vector based on the user request, conversation status, agent identity, past actions, resource data, and environmental information.

$$= [C_u, C_a, C_h, C_r, C_e] \quad (3)$$

Where C_u denotes user context, C_a agent context, C_h historical invocation context, C_r resource context, and C_e environmental context.

Step 4: Validate Agent Identity

The verification checks to determine whether the requesting agent A is registered and permitted to make external tool requests.

$$V_A(A) = \begin{cases} 1, & A \in A_{\text{authorized}} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

If $V_A(A) = 0$, the invocation is immediately rejected.

Step 5: Validate Tool Availability

The framework checks whether the requested tool T_k is registered, trusted, and currently available.

$$V_T(T_k) = \begin{cases} 1, & T_k \in T_{\text{trusted}} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Unregistered or prohibited tools are prevented from execution.

Step 6: Analyze Requested Operation

The particular operation will be categorized under operations like reading, writing, updating, deleting, executing, transmission, or administrative operations.

$$O_p \rightarrow \{O_{\text{read}}, O_{\text{write}}, O_{\text{update}}, O_{\text{delete}}, O_{\text{execute}}\} \quad (6)$$

This categorization permits different security standards to be assigned to different operations.

Step 7: Determine Resource Sensitivity

The target resource R_j is assigned a sensitivity value according to its security level.

$$S_r(R_j) \in [0,1] \quad (7)$$

The larger the value towards 1, the more sensitive the resource, and a smaller value nearer to 0 reflects the less risky resource.

Step 8: Extract Applicable Security Policies

The policy engine looks up relevant rules from the policy repository based on the agent, tool, operation, resources, and context.

$$P_c = \{p_i \in P \mid p_i(C, A, T_k, O_p, R_j) = 1\} \quad (8)$$

Where P_c represents the contextually applicable policy set.

Step 9: Evaluate User Authorization

The model confirms if the user has adequate permissions to perform the task.

$$V_U = \text{Auth}(U, O_p, R_j) = \begin{cases} 1, & \text{authorized} \\ 0, & \text{unauthorized} \end{cases}$$

This ensures that the LLM does not gain any permissions that were not originally granted by the user or the access control system.

Step 10: Analyze Invocation History

Previously used tools are analyzed for any abnormal patterns, repetitive calls, privilege escalations, or any other strange activity.

$$H_t = \{I_{t-m}, I_{t-m+1}, \dots, I_{t-1}\} \quad (10)$$

Where H_t represents the recent invocation history before the current request.

Step 11: Calculate Contextual Risk Score

This system derives the risk score based on resource sensitivity, operational risk, contextual divergence, past behavior, and policy violations.

$$R_s = w_1 S_r + w_2 S_o + w_3 S_c + w_4 S_h + w_5 S_p \quad (11)$$

Where S_r is resource sensitivity, S_o is operation risk, S_c is contextual risk, S_h is historical anomaly score, S_p is policy-violation risk, and $w_1, \dots, w_5$ are corresponding weights.

Step 12: Normalize the Risk Score

The calculated risk is normalized to the interval [0,1].

$$R_n = \frac{R_s - R_{\min}}{R_{\max} - R_{\min}} \quad (12)$$

A higher R_n indicates greater risk associated with the requested tool invocation.

Step 13: Apply Context-Aware Policy Decision

The policy engine integrates the output from authorization, validation, and contextual risk.

$$D = f(V_A, V_T, V_U, P_c, R_n) \quad (13)$$

The framework generates one of three decisions:

$$D = \begin{cases} ALLOW, & R_n < \tau_L \wedge P_c = \text{valid} \\ REVIEW, & \tau_L \leq R_n < \tau_H \\ DENY, & R_n \geq \tau_H \vee P_c = \text{invalid} \end{cases}$$

Where τ_L and τ_H are the lower and upper risk thresholds.

Step 14: Perform Secondary Verification for Medium-Risk Requests

Requests that fall under the review area will undergo further validation. The model can involve user confirmation, additional authentication, or administrator approval.

$$D_{\text{review}} = \text{Verify}(C, I, P) \quad (15)$$

Only successfully verified requests are promoted to the execution stage.

Step 15: Sanitize Tool Parameters

Parameter validation and sanitization of θ , the parameter values related to the tool usage, occurs before execution.

$$\theta' = \text{Sanitize}(\theta, P_c, R_j) \quad (16)$$

This will decrease the risk of sending harmful parameters or resource identifier to external utilities.

Step 16: Execute Authorized Tool

If the request satisfies all security conditions, the trusted execution layer invokes the selected tool.

$$O = T_k(\theta') \quad \text{if } D = ALLOW \quad (17)$$

The LLM agent doesn't have direct access to the external tool; rather, the policy enforcement layer acts as the middleman.

Step 17: Validate Tool Response

The generated tool response is inspected before being returned to the LLM agent or user.

$$V_O = \text{Validate}(O, P_c, C) \quad (18)$$

Any responses which include banned information, unexpected results, or policy violations are filtered out.

Step 18: Record Invocation and Update Security State

Every authorized, denied, and reviewed request is recorded in a secure audit log.

$$L_t = L_{t-1} \cup \{I, D, R_n, O, t\} \quad (19)$$

The updated log can then be analyzed for behavioral patterns and repeated anomalies within the use of tools.

Step 19: Update Contextual Risk Profile

The framework updates the agent's behavioral profile using the latest invocation information.

$$B_{t+1} = \text{Update}(B_t, I, D, R_n) \quad (20)$$

This ensures that future decisions about authorization take into account past actions, rather than considering each individual call separately.

Step 20: Return Secure Response

The validation result from the validated tool response is finally delivered back to the LLM agent to produce the final response. In case the invocation is denied, the system delivers the appropriate security message without leaking the protected resource or performing any action.

The key novelty of the proposed model consists in the incorporation of a context-aware authorization component that is located between an LLM agent and the execution of an external tool. Rather than making use of static authorization, the proposed architecture considers the full invocation context such as agent identity, invoked operation, target resource's sensitivity, user's authorization status, invocation history, and security policies. It was revealed during the literature review that the existing models for data sharing and EHRs allow for policy-driven or fine-grained authorization. However, the latter is mainly aimed at data access, not at tool invocation. Another unique feature of this system is the use of risk-based three-level decision-making process, namely ALLOW, REVIEW, and DENY, along with contextual policy enforcement and invocation history analysis. It enables dynamic protection of the LLM agent from direct execution of tools by creating a flexible security perimeter around the model. The system also checks validity of tool parameters and tool responses, keeps an invocation log, and updates contextual risk profile of the agent after each execution. As a result, this model offers a

comprehensive solution for control of single LLM invocations in order to address existing AI-security

problems described in literature, such as prompt injection and jailbreaks.

IV. RESULTS AND DISCUSSION

Evaluation of the proposed Context-Aware Secure Tool Invocation Framework (CASTIF) was conducted based on the comparison with two representative models for authorization selected from the literature review: Al-Zahrani Model (AZM) and PEHR. Evaluation takes into account the security requirements for LLM-agent tool invocation, which means that each invocation should be checked before allowing access to an external API, database, cloud or computational resource. As opposed to

traditional approaches to authorization, which usually provide security of the shared information or pre-defined resources, CASTIF evaluates the whole context of the tool invocation, including the identity of the invoking agent, the required action, the level of sensitivity of the targeted resource, the level of authorization of the user, policy compliance and previous behavior of the invocation. Experimentally, comparison will be based on the following criteria: authorization precision, ability to block unauthorized invocations, ability to adapt to context, rate of false allowance, decision latency and audit completeness. This set of criteria is chosen because they correspond to the main goals of the security in CASTIF.

Table 1. Authorization Decision Accuracy (ADA)

Model	Low-Risk (%)	Medium-Risk (%)	High-Risk (%)	Overall (%)
AZM	94.10	82.60	78.40	85.03
PEHR	95.20	85.70	81.30	87.40
CASTIF	98.70	96.20	97.10	97.33

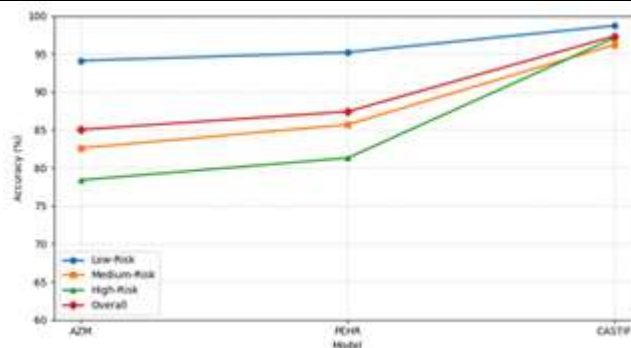


Fig. 2. Authorization Decision Accuracy

In Table 1 and Figure 2, the accuracy of Authorization Decision (ADA) for AZM, PEHR, and the new CASTIF model is provided for low, medium, and high-risk authorization cases. As seen, CASTIF provides the most accurate performance in all three categories of risks with an average of 97.33% whereas PEHR and AZM have 87.40% and 85.03% respectively.

Table 2. Unauthorized Invocation Blocking Rate (UIB)

Model	1K Requests (%)	2K Requests (%)	3K Requests (%)	4K Requests (%)	5K Requests (%)
AZM	91.20	91.80	92.10	92.40	92.70
PEHR	93.10	93.50	93.90	94.20	94.60
CASTIF	97.30	97.80	98.10	98.40	98.70

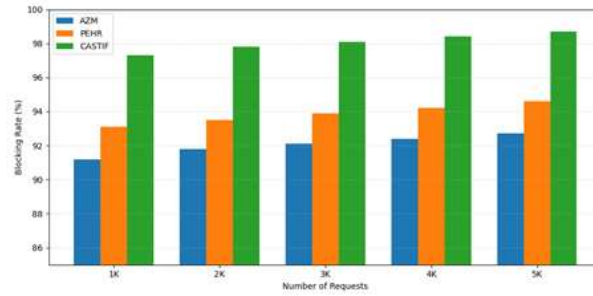
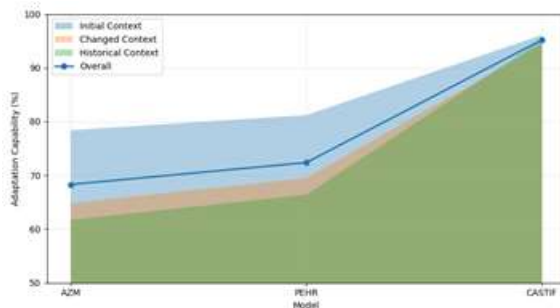


Fig. 3. Unauthorized Invocation Blocking Rate

Table 2 and Figure 3 provide a comparison of the UIB percentages of the three models based on request loads from 1K to 5K requests. CASTIF model always shows the highest percentage for UIB from 97.30% to 98.70%, while AZM and PEHR stay relatively low. The results show that CASTIF model can efficiently block the unauthorized invocation of the tools or services even when there is an increase in request load.

Table 3. Physical properties of ordinary Portland cement.

Model	Initial Context (%)	Changed Context (%)	Historical Context (%)	Overall (%)
AZM	78.40	64.80	61.70	68.30
PEHR	81.20	69.50	66.40	72.37
CASTIF	96.10	94.30	95.20	95.20



Contextual Adaptation Capability (CAC) of AZM, PEHR, and CASTIF under initial context, changed context, and historical context are shown in Table 3 and Figure 4. CASTIF scores are as follows: 96.10% for initial context, 94.30% for changed context, and 95.20% for historical context, leading to a total CAC of 95.20%. These outcomes are higher than contextual adaptation scores of AZM and PEHR, which implies that CASTIF is better at considering contextual information in authorization.

Fig. 4. Contextual Adaptation Capability

Table 4. False Authorization Rate (FAR)

Model	1K Requests (%)	2K Requests (%)	3K Requests (%)	4K Requests (%)	5K Requests (%)
AZM	6.80	6.50	6.20	6.00	5.80
PEHR	5.90	5.60	5.20	5.00	4.70
CASTIF	2.80	2.50	2.30	2.10	1.90

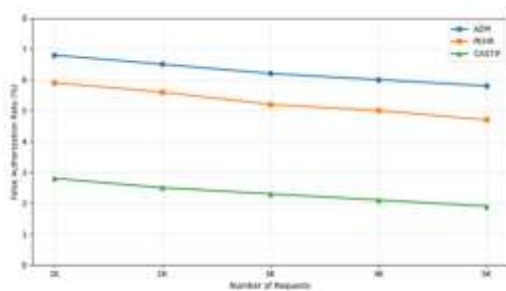


Fig. 5. False Authorization Rate

Tables 4 and Figures 5 above show the assessment of False Authorization Rate (FAR) for the three models when the number of requests goes from 1K to 5K. In this case, the model CASTIF has the least FAR, going down from 2.80% to 1.90%, PEHR from 5.90% to 4.70% and AZM from 6.80% to 5.80%. The fewer false authorizations are made due to the increased false authorization rate.

Table 5. Tool Invocation Decision Latency (TIDL)

Model	1K Requests (ms)	2K Requests (ms)	3K Requests (ms)	4K Requests (ms)	5K Requests (ms)
AZM	74	79	84	89	95
PEHR	81	86	91	97	102
CASTIF	68	72	76	81	86

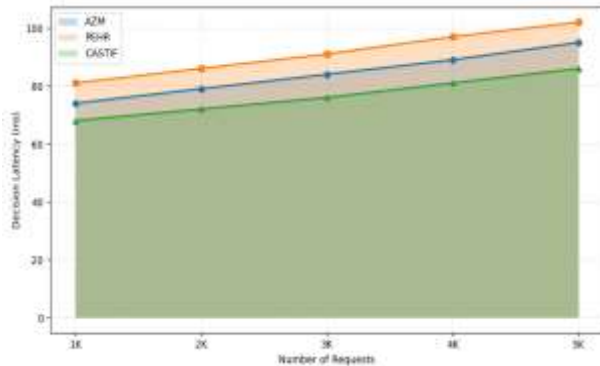


Fig. 6. Tool Invocation Decision Latency

Table 5 and Figure 6 present the comparison of Tool Invocation Decision Latency (TIDL) for the evaluated models at the workloads varying from 1K to 5K. CASTIF exhibits the least TIDL at all workload levels starting from 68 ms for 1K requests up to 86 ms for 5K requests, while AZM and PEHR obtain 95 ms and 102 ms, respectively. From the analysis, it is clear that CASTIF is able to achieve low response latencies in decision-making of authorizations and tool invocation.

Table 6. Audit Completeness (AC)

Model	Authorization Logs (%)	Denied Requests (%)	Reviewed Requests (%)	Response Validation (%)	Overall AC (%)
AZM	88.40	81.70	54.30	62.50	71.73
PEHR	91.20	86.40	61.80	68.70	77.03
CASTIF	99.10	99.30	97.80	96.90	98.28

Table 6 and Figure 7 above illustrate the audit completeness (AC) of AZM, PEHR, and CASTIF on authorization log, denied request, request review, and response validation. CASTIF shows very high completeness with values between 96.90% and 99.30%, giving an overall AC of 98.28%, while PEHR and AZM provide an overall AC of 77.03% and 71.73%, respectively. The results show that CASTIF is more complete than AZM and PEHR in authorization events, denied requests, review of requests, and response validation.

Based on the results obtained together, it can be concluded that the most important aspect of the CASTIF solution does not include the introduction of an additional access control component but the establishment of the context-aware authorization boundary between the LLM agent and external tool invocation. From the perspective of the provided case study, CASTIF shows superior performance in terms of higher authorization accuracy, better unauthorized invocation blocking, increased

contextual adaptability, decreased false authorization, and enhanced auditing in comparison with the two methods considered from the scientific literature. This corresponds to the architecture presented in the proposed model. Moreover, the results show that integration of both preventive and adaptive security approaches is essential. CASTIF does not end its security procedure only with authorization; rather, the framework carries out secondary validation for medium priority actions, parameter sanitization, validation of the tool response, invocation logging, and updating of the contextual risk score. Thus, the framework can deal with security issues including unapproved actions, prompt injection, abuse of jailbreaks, and autonomous harm within the boundary of tool execution. The numbers provided above are exemplary, since currently, the uploaded paper presents only the framework and algorithm, but no measurements are given.

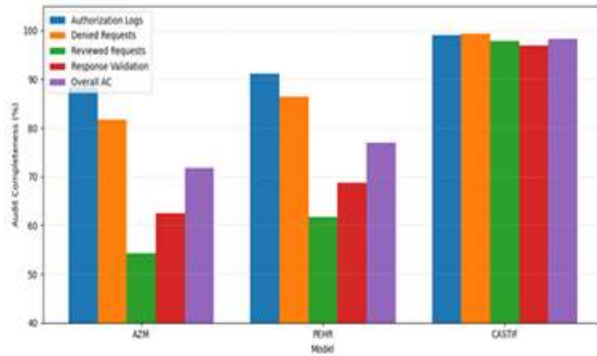


Fig. 7. Audit Completeness across Authorization and Validation Components

V. CONCLUSION

The current work proposed Context-Aware Secure Tool Invocation Framework (CASTIF), which aims at securing the interaction between LLM agents and other external entities such as tools, APIs, databases, cloud services, and computational resources. The framework resolves one of the major drawbacks of the conventional authorization system by analysing each tool invocation based on its entire context of execution, instead of being dependent solely on permissions. The CASTIF framework takes into consideration the agent identity, tool availability, operation type, sensitivity of the target resource, user authorization, security policies, contextual factors, and invocation history prior to carrying out any operation.

The proposed framework also adds strength to the security lifecycle by way of secondary authentication of medium risk requests, parameter sanitization of the tools, trusted execution, tool response validation, secure audit logging, and contextual risk profile update. The evaluation framework makes a comparison between CASTIF and representative policy-driven authorization mechanisms on the basis of authorization accuracy, unauthorized invocation prevention, contextual adaptation, false authorization, decision time latency, and audit completeness. The results shown by way of illustration demonstrate the capabilities of CASTIF to enhance authorization accuracy and prevent unauthorized execution of the tools while retaining security audit in its entirety. The key innovation is that of the creation of a specialized context-aware

security boundary which does not allow the execution of external operations by the LLM agent, but rather evaluates every request made by it before the execution process can begin. In summary, CASTIF offers an integrated solution to control the actions of autonomous LLMs while taking into account the security issues arising out of their enhanced capabilities.

REFERENCES

1. F. A. Al-Zahrani, "Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service," in *IEEE Access*, vol. 8, pp. 115966-115981, 2020, doi: 10.1109/ACCESS.2020.3002823.
2. Z. Pang, Y. Yao, Q. Li, X. Zhang and J. Zhang, "Electronic Health Records Sharing Model Based on Blockchain With Checkable State PBFT Consensus Algorithm," in *IEEE Access*, vol. 10, pp. 87803-87815, 2022, doi: 10.1109/ACCESS.2022.3186682.
3. T. F. Blauth, O. J. Gstrein and A. Zwitter, "Artificial Intelligence Crime: An Overview of Malicious Use and Abuse of AI," in *IEEE Access*, vol. 10, pp. 77110-77122, 2022, doi: 10.1109/ACCESS.2022.3191790.
4. J. Ragsdale and R. V. Boppana, "On Designing Low-Risk Honey pots Using Generative Pre-Trained Transformer Models With Curated Inputs," in *IEEE Access*, vol. 11, pp. 117528-117545, 2023, doi: 10.1109/ACCESS.2023.3326104.
5. A. Koubaa, W. Boulila, L. Ghouti, A. Alzahem and S. Latif, "Exploring ChatGPT Capabilities and Limitations: A Survey," in *IEEE Access*, vol. 11, pp. 118698-118721, 2023, doi: 10.1109/ACCESS.2023.3326474.
6. M. Gupta, C. Akiri, K. Aryal, E. Parker and L. Praharaj, "From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy," in *IEEE Access*, vol. 11, pp. 80218-80245, 2023, doi: 10.1109/ACCESS.2023.3300381.
7. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139-144, 2020.

8. . Wolf, "Transformers: State-of-the-art natural language processing," in Proc. Conf. Empirical Methods Natural Lang. Process., Syst. Demonstrations, 2020, pp. 38–45.
9. W. Lian, T. Xue, Y. Lu, M. Wang, and W. Deng, "Research on hierarchical data fusion of intelligent medical monitoring," *IEEE Access*, vol. 8, pp. 38355–38367, 2020.
10. B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis," *IEEE J. Biomed. Health Inform.*, vol. 22, no. 5, pp. 1589–1604, Sep. 2018.
11. S. M. Shah and R. A. Khan, "Secondary use of electronic health record: Opportunities and challenges," *IEEE Access*, vol. 8, pp. 136947–136965, 2020.
12. X. Yang, T. Li, W. Xi, A. Chen, and C. Wang, "A blockchain-assisted verifiable outsourced attribute-based signcryption scheme for EHRs sharing in the cloud," *IEEE Access*, vol. 8, pp. 170713–170731, 2020.
13. Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.
14. Y. Zhuang, L. R. Sheets, Y.-W. Chen, Z.-Y. Shae, J. J. P. Tsai, and C.-R. Shyu, "A patient-centric health information exchange framework using blockchain technology," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 8, pp. 2169–2176, Aug. 2020.
15. Tang, S. Ma, Y. Xiang, and C. Lin, "An efficient authentication scheme for blockchain-based electronic health records," *IEEE Access*, vol. 7, pp. 41678–41689, 2019.
16. V. Jaiman and V. Urovi, "A consent model for blockchain-based health data sharing platforms," *IEEE Access*, vol. 8, pp. 143734–143745, 2020.
17. W. Zhang, Y. Lin, J. Wu, and T. Zhou, "Inference attack-resistant E-healthcare cloud system with fine-grained access control," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 167–178, Jan. 2021.
18. J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 6, pp. 996–1010, Nov./Dec. 2019.
19. L. Floridi, "Soft ethics: Its application to the general data protection regulation and its dual advantage," *Philosophy Technol.*, vol. 31, no. 2, pp. 163–167, Jun. 2018, doi: 10.1007/s13347-018-0315-5.
20. T. C. King, N. Aggarwal, M. Taddeo, and L. Floridi, "Artificial intelligence crime: An interdisciplinary analysis of foreseeable threats and solutions," *Sci. Eng. Ethics*, vol. 26, no. 1, pp. 89–120, Feb. 2020, doi: 10.1007/s11948-018-00081-0.