

Runtime Policy Firewall: A Zero-Trust Governance Layer for Enterprise Agentic AI

Swapneswar Sundar Ray
Independent Researcher
Randolph, NJ, USA
swapneswar.ray@gmail.com

Abstract—Large Language Model (LLM)-powered agentic systems are rapidly evolving from conversational assistants into autonomous enterprise actors capable of invoking APIs, executing workflows, accessing sensitive data, and making operational decisions. While existing research has advanced reasoning, planning, retrieval, and tool use, comparatively less attention has been paid to runtime governance and policy enforcement for autonomous execution. This paper proposes the Runtime Policy Firewall (RPF), a zero-trust governance layer positioned between AI agents and enterprise tools. Rather than allowing unrestricted tool invocation, each proposed action is evaluated through identity verification, policy enforcement, dynamic risk scoring, evidence validation, approval workflows, and immutable audit logging before execution. The architecture introduces identity-aware policy enforcement, context-sensitive authorization, evidence-backed execution validation, human-in-the-loop approval for high-risk operations, and compliance-oriented observability. An Azure-based reference architecture illustrates a practical enterprise deployment mapping using Azure OpenAI, Microsoft Entra ID, Azure AI Search, Azure Key Vault, Azure API Management, and centralized monitoring. The RPF provides a reusable architectural pattern for trustworthy agentic AI in regulated sectors including banking, healthcare, and government.

Index Terms—agentic AI, zero trust, runtime governance, LLM security, policy enforcement, prompt injection, Azure OpenAI, audit logging

I. INTRODUCTION

Enterprise adoption of generative AI is shifting from passive question answering to autonomous agentic execution. Modern agents can decompose goals, retrieve business context, call tools, update records, send messages, initiate transactions, and coordinate workflows across multiple systems. This creates productivity opportunities, but also introduces a critical control problem: organizations must govern not only what an agent says, but what it does at runtime.

Existing work on LLM applications has investigated retrieval-augmented generation, reasoning-action loops, tool use, software agents, and foundation-model governance [1], [2], [3], [4], [5]. Security research has also documented prompt injection, indirect instruction attacks, data leakage, and tool-abuse risks in LLM-integrated systems [6], [7], [8], [20], [22], [23]. However, enterprise deployment requires a stronger runtime control plane. Traditional application security assumes stable code paths, deterministic authorization checks, and explicit user interfaces. Agentic AI weakens these assumptions because one natural-language request may produce many tool

calls, each with different risk, data exposure, and business impact.

This paper proposes the *Runtime Policy Firewall* (RPF), a mandatory control point between agent orchestration and enterprise tools. The novelty of RPF is not a new access-control mechanism or policy engine, but the elevation of model-generated semantic actions to first-class governable objects with formal mediation semantics, evidence-aware execution constraints, and lifecycle invariants. RPF makes these actions auditable execution tuples evaluated before any enterprise side effect occurs.

The paper asks: RQ1, can semantic action mediation reduce unauthorized execution? RQ2, can evidence-aware governance reduce hallucination-induced side effects? RQ3, can zero-trust runtime mediation preserve compliance with acceptable latency? Operationally, RQ1 is measured by the unauthorized-action execution rate, $N_{\text{unauth,exec}}/N_{\text{unauth}}$; RQ2 by the unsafe side-effect rate for unsupported-evidence requests, $N_{\text{unsupported,exec}}/N_{\text{unsupported}}$; and RQ3 by the policy-compliance rate, $N_{\text{compliant}}/N_{\text{total}}$, together with p50/p95 end-to-end mediation latency from action receipt to an audited terminal decision. The proposed guarantees hold under an explicit trusted computing base and mandatory mediation assumption. The contributions are formal RPF semantics, a runtime evaluation algorithm, and a comparison plus validation methodology.

II. BACKGROUND AND MOTIVATION

Agentic systems commonly combine LLM reasoning with retrieval, memory, planning, and tool invocation. ReAct-style designs interleave reasoning and actions [2]; tool-use approaches teach models to call external functions [3]; retrieval-augmented generation grounds responses in external corpora [1]; and emerging agent frameworks orchestrate multi-step tasks across APIs. These capabilities transform the LLM from a content generator into an operational actor.

This transformation exposes an authorization gap. In conventional software, developers predefine application flows and policy checks. In agentic workflows, the model may dynamically choose tools, synthesize parameters, decide execution order, and justify actions using retrieved content. Even when the underlying APIs are secure, the agent may create unsafe combinations: querying sensitive records, summarizing them, and sending the summary externally. Therefore, policy must be

evaluated at the semantic action layer, not only at the network or API endpoint layer.

Zero-trust architecture and AI risk-management guidance provide foundations for identity verification, continuous authorization, and lifecycle governance [13], [16], [17], [18]. RPF is inspired by the classical reference monitor concept, but extends mediation from system calls and resource accesses to semantic AI-generated actions [14], [15].

III. THREAT MODEL AND REQUIREMENTS

The RPF assumes that the agent, user, retrieved content, and downstream tools may each contribute risk. The LLM may misunderstand intent, follow malicious instructions in retrieved documents, fabricate evidence, or overgeneralize from examples. A user may be compromised or may attempt to bypass policy through natural language. Tools may expose broad APIs whose native authorization is too coarse for agentic use.

Key threats include:

- **Prompt injection:** malicious user input or retrieved documents instruct the agent to ignore policy, leak secrets, or call unauthorized tools [6], [7], [8].
- **Privilege confusion:** the agent combines service credentials and delegated user authority in ways that exceed least privilege.
- **Unsafe parameter generation:** the tool is permitted, but model-generated arguments create excessive scope, such as exporting all customer records instead of one case.
- **Hallucinated evidence:** the agent claims that a document, policy, or approval supports an action when no reliable evidence exists.
- **Audit gaps:** orchestration frameworks execute tool calls without a durable explanation of who requested the action, why it was allowed, and what evidence was used.

From these threats, six requirements follow. The control layer must be identity-aware, context-sensitive, tool-schema-aware, evidence-aware, risk-adaptive, and audit-complete. It must also fail safely: ambiguous, unsupported, or high-risk actions should be denied, constrained, clarified, or routed for approval.

Assumptions and trust boundary. The trusted computing base (TCB) contains the identity provider, policy store, approval service, audit store, and RPF enforcement code. The LLM, prompt text, retrieved documents, natural-language user intent, and external content are untrusted. The API gateway and execution environment are assumed to enforce that tool calls cannot bypass the RPF; if this mediation assumption fails, the safety proposition does not apply.

IV. FORMAL MODEL

Definition 1 (Runtime Policy Firewall). An RPF is a mandatory mediation function $F : \tau \mapsto D$ placed between an agent orchestrator and enterprise tools. For every proposed action, F evaluates identity, policy, risk, evidence, approval state, and audit obligations before execution.

Definition 2 (Protected Action). A protected action is an execution tuple

$$\tau = \langle u, a, t, x, c, e, p, h \rangle, \quad (1)$$

where u is the user identity, a is the agent or workload identity, t is the tool, x is the typed argument vector, c is context, e is evidence, p is purpose, and h is history.

Definition 3 (Decision). The firewall computes

$$D(\tau) \in \{\text{ALLOW}, \text{DENY}, \text{CLARIFY}, \text{APPROVAL}\}. \quad (2)$$

Execution states are PROPOSED, EVALUATED, APPROVED, EXECUTED, and LOGGED. Every evaluation must terminate in exactly one decision state, and no transition to EXECUTED is valid unless $D(\tau) = \text{ALLOW}$ or an approval token is bound to τ . A risk score is modeled as

$$R(\tau) = f(S_d(t, x), I_b(t, x), E_x(x), Q_e(e), A_h(h), C_c(c)), \quad (3)$$

where S_d is data sensitivity, I_b business impact, E_x external exposure, Q_e evidence quality, A_h historical anomaly, and C_c contextual confidence. In the proposed architecture, f is a policy-defined, configurable scoring function whose monotonicity constraints are set by governance policy rather than learned from model output. The RPF enforces four invariants: (i) no unmediated execution; (ii) least-authority binding across u , a , and p ; (iii) evidence monotonicity, where broader action scope cannot be justified by weaker evidence; and (iv) every evaluation produces an immutable decision audit event, and every executed action additionally produces an immutable execution-result audit event.

Proposition 1 (Completed-Execution Accountability). Every action that successfully completes the RPF-controlled execution path has a durably recorded decision audit and a durably enqueued execution-result audit. Let X_c , V , A_d , and A_x abbreviate COMPLETEDRPFEXECUTION, VALIDAPPROVAL, DECISIONAUDIT, and EXECUTIONAUDIT:

$$\forall \tau : X_c(\tau) \Rightarrow ((D(\tau) = \text{ALLOW} \vee V(\tau)) \wedge A_d(\tau) \wedge A_x(\tau)). \quad (4)$$

Proof sketch. Algorithm 1 invokes EXECUTELEASTPRIVILEGE only after authentication, ambiguity handling, authorization, evidence validation, and risk evaluation. Every branch writes an immutable decision event. The allow branch executes with least privilege and durably enqueues a separate execution-result event before returning. Approved actions follow the same rule after token validation and policy recheck. Therefore completion of the controlled path implies $A_d(\tau)$ and $A_x(\tau)$ under the trusted audit-store assumption. $\square$

Security properties. Under the stated mediation and TCB assumptions, RPF targets: *safety*, unauthorized execution is blocked; *liveness*, every protected action eventually reaches exactly one terminal decision; and *accountability*, each evaluation has an immutable decision event and each execution has an additional immutable result event.

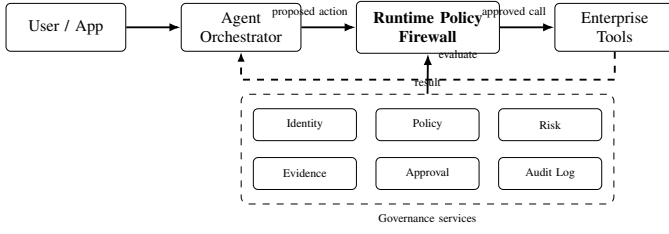


Fig. 1. Runtime Policy Firewall as a mandatory control point for agentic tool invocation.

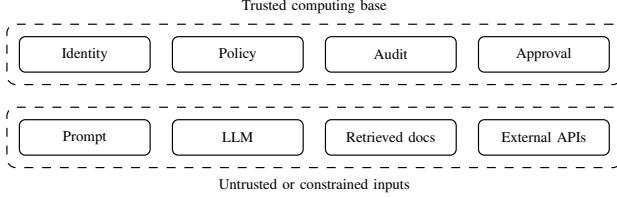


Fig. 2. RPF trusted computing base and trust boundary.

V. RUNTIME POLICY FIREWALL ARCHITECTURE

The RPF is positioned between agent orchestration and enterprise tools, analogous to a network firewall but operating over business actions rather than packets. Its protected object is a structured tool invocation containing subject identity, agent identity, tool name, typed arguments, declared purpose, retrieved evidence, session context, and expected business effect.

Algorithm 1 gives the runtime decision semantics. Clarification is explicit because many unsafe actions are underspecified rather than malicious: “export recent records” must be converted into a bounded request specifying dataset, time range, recipient, legal basis, and business purpose.

Algorithm 1 models the initial mediation decision. Actions returning APPROVAL re-enter the execution path only after a token is bound to τ , validated for expiry, and subjected to policy re-evaluation as shown in Fig. 3. Execution proceeds only after the decision audit has been durably recorded. Execution-result events are written through a durable audit buffer; if the audit subsystem is unavailable before execution, RPF fails closed.

Clarification never authorizes execution: a revised proposal is normalized and fully re-evaluated. Approval tokens are bound to the original action tuple, checked for expiry, and subjected to a policy recheck before execution.

Complexity. Normalization and schema validation are linear in argument size and bounded by tool schemas. Token verification is $O(1)$ with cached keys. Authorization over P applicable indexed policies is $O(P)$, evidence validation over E artifacts is $O(E)$, and risk scoring over R factors is $O(R)$. Thus mediation is $O(P+E+R)$ plus downstream API latency.

VI. POLICY, RISK, AND EVIDENCE CONTROLS

Policy enforcement combines role-based, attribute-based, and purpose-based constraints. OPA/Rego evaluates declara-

Algorithm 1 Runtime Policy Firewall Evaluation

Require: Proposed action q , policy set Π , threshold vector θ

- 1: $\tau \leftarrow \text{NORMALIZE}(q)$
- 2: **if** $\neg \text{AUTHENTICATE}(\tau.u, \tau.a, \tau.c)$ **then**
- 3: $\text{AUDITDECISION}(\tau, \text{DENY}, \text{authentication})$
- 4: **return** DENY
- 5: **end if**
- 6: **if** $\text{AMBIGUOUS}(\tau.t, \tau.x, \tau.p)$ **then**
- 7: $\text{AUDITDECISION}(\tau, \text{CLARIFY}, \text{ambiguous scope})$
- 8: **return** CLARIFY
- 9: **end if**
- 10: **if** $\neg \text{AUTHORIZE}(\tau, \Pi)$ **then**
- 11: $\text{AUDITDECISION}(\tau, \text{DENY}, \text{authorization})$
- 12: **return** DENY
- 13: **end if**
- 14: $v \leftarrow \text{VALIDATEEVIDENCE}(\tau.e, \tau.t, \tau.x)$
- 15: **if** $\neg v$ **then**
- 16: $\text{AUDITDECISION}(\tau, \text{DENY}, \text{invalid evidence})$
- 17: **return** DENY
- 18: **end if**
- 19: $r \leftarrow R(\tau)$
- 20: **if** $r \geq \theta_{\text{approval}}$ **then**
- 21: $\text{AUDITDECISION}(\tau, \text{APPROVAL}, r)$
- 22: **return** APPROVAL
- 23: **end if**
- 24: **if** $\neg \text{AUDITDECISION}(\tau, \text{ALLOW}, r)$ **then**
- 25: **return** DENY
- 26: **end if**
- 27: $y \leftarrow \text{EXECUTELEASTPRIVILEGE}(\tau)$
- 28: $\text{PERSISTEXECUTIONAUDIT}(\tau, y)$
- 29: **return** ALLOW

TABLE I
COMPARISON WITH DIRECT ENFORCEMENT SYSTEMS

System	Enforced object	Native strength	Gap or RPF role
OPA/Rego [10]	Structured policy input	General policy-as-code	RPF policy backend; no agent lifecycle
Cedar [11]	Principal/action/resource	Verified authorization	RPF authorization; no evidence semantics
Agent guardrails [12]	Model action/content	Adaptive safety checks	Often model-dependent; not full mediation
Sandbox/capability	Process/resource/effect	Isolation, least privilege	Constrains effects, not semantic justification
RPF	Semantic action tuple	Evidence, risk, approval, audit	Composes engines and effect controls

tive policy over structured input, while Cedar provides fine-grained principal–action–resource authorization [10], [11]. Either can serve as an RPF policy engine, but neither alone defines evidence sufficiency, clarification, approval-token lifecycle, or semantic action auditing. Agent guardrails such as GuardAgent inspect agent actions [12], while sandbox and capability systems constrain reachable resources and effects. RT-BAS addresses prompt-injection and privacy-leakage defenses, whereas AgentDojo provides an environment for evaluating attacks and defenses against tool-using agents [20], [24]. RPF composes these mechanisms at a mandatory semantic-action mediation point.

Evidence validation remains distinctive: a natural-language rationale is insufficient. The RPF requires machine-verifiable artifacts such as source identifiers, retrieval timestamps, access labels, policy references, approval tickets, or database records.

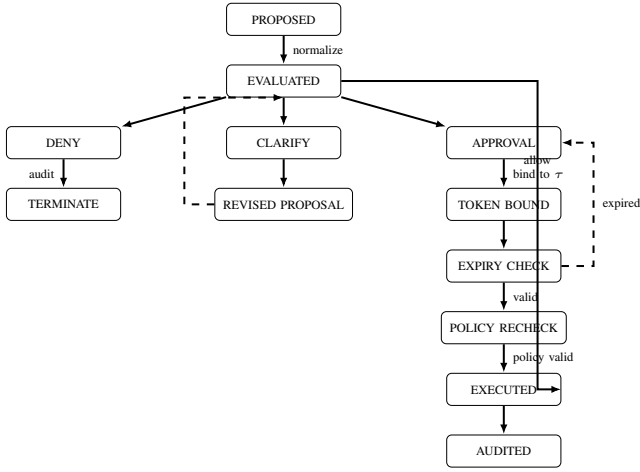


Fig. 3. RPF execution state machine with clarification re-evaluation and approval-token validation.

TABLE II
PROTOTYPE POLICY AND TEST METHODOLOGY

Policy rule	Tested threat	Expected	Evidence	Threshold
Identity/tool allowlist	Privilege confusion	Deny	Valid claims/schema	Hard rule
Injection isolation	Prompt injection	Deny	Cannot override deny	Hard rule
Scope completeness	Overbroad parameters	Clarify	Scoped resource ID	Hard rule
Evidence provenance	Unsupported claim	Deny	At least one doc-* ID	Hard rule
External-send policy	Data exfiltration	Deny	Evidence insufficient	Hard rule
Risk aggregation	Sensitive allowed action	Approval	Valid doc-* ID	$R \geq 75$
Normal read/update	Benign action	Allow	Valid doc-* ID	$R < 75$

If semantic support is weak, the firewall narrows scope, masks fields, escalates, or denies.

VII. REFERENCE ARCHITECTURE AND VALIDATION PLAN

The Azure reference architecture maps agents to Azure OpenAI, identities to Microsoft Entra ID, retrieval to Azure AI Search, secrets to Key Vault, tools to Azure API Management, and telemetry to Azure Monitor/Sentinel. The key requirement is that all generated tool calls traverse the RPF before API Management invokes the target tool with least-privilege credentials.

Methodology and artifacts. Actions use three typed schemas: `read_ticket(id, scope)`, `update_ticket(id, scope)`, and `send_email(external, scope)`. Hard-deny/clarify rules precede risk scoring. The score assigns 50 points for sensitive terms, 50 for injection terms, 25 for external destinations, and 75 for explicitly high-impact actions; valid evidence requires nonempty doc-* identifiers, and $R \geq 75$ requires approval. Requests are generated deterministically and labeled from Table II. Baseline A uses API authorization only; Baseline B disables evidence validation; the ablation disables risk scoring. A sanitized package supplies code, cases, configurations, evidence, labels, logs, versions, timing protocol, and evaluation script. Tests use CPython 3.13 on x86-64 Linux, single process, without network or persistent storage.

TABLE III
DETERMINISTIC CONSTRUCTION CHECKS (NOT SECURITY VALIDATION)

Metric	Prototype result	Interpretation
RQ1 unauthorized execution	0/50 (0%)	Expected by rule construction
RQ2 unsafe side effects	0/15 (0%)	Expected by rule construction
RQ3 policy compliance	100/100 (100%)	Closed deterministic replay
In-process policy latency	avg. 0.002 ms; p95 0.005 ms	One local run; excludes I/O
Decision audit coverage	100/100 events	All terminal decisions logged
Baseline A: API only	50/50 adversarial allowed	No semantic controls
Baseline B: no evidence	15/15 unsafe allowed	Evidence control removed
No-risk ablation	10/10 high-risk allowed	Full RPF: 10 approvals
RQ3 end-to-end latency	Not yet measured	Future Azure benchmark

A minimal deterministic Python prototype implements Algorithm 1 and replays 50 benign and 50 adversarial requests. Table III remains a construction check, not statistical robustness evidence. Baseline B allowed all 15 unsupported-evidence requests that full RPF denied. On 10 additional valid-evidence, high-impact probes, full RPF returned APPROVAL for all 10, whereas removing risk scoring allowed all 10 to execute. The reported timing is strictly *in-process policy evaluation latency*; it excludes network, audit persistence, approval waiting, and downstream execution.

RQ3 is therefore only partially evaluated in the current prototype: policy compliance is measured, while end-to-end latency remains for the planned Azure evaluation.

Limitations. The closed replay tests conformance, not generalization, and policy misconfiguration can cause both false positives and false negatives. Approval tokens require secure binding, expiry, revocation, and replay prevention; policy or evidence changes between decision and execution create TOCTOU risk. Concurrent actions may race or produce unsafe aggregate effects. Audit-store outage requires fail-closed behavior but may reduce availability, while logs create privacy, retention, and access-control obligations. Approval fatigue, denial-of-service against mediation or approval queues, ambiguous evidence, adaptive attacks, and compromised TCB components remain unresolved. Future evaluation will run baselines and ablations on independently authored tasks, report confidence intervals and utility/security tradeoffs, test concurrency and failure injection, and measure end-to-end Azure overhead.

VIII. CONCLUSION

Agentic AI changes enterprise security from controlling generated text to controlling autonomous action. The Runtime Policy Firewall addresses this shift by placing a zero-trust governance layer between agents and tools. Through identity-aware policy, dynamic risk scoring, evidence validation, approval workflows, and immutable audit logging, the RPF enables organizations to adopt agentic AI while preserving operational control. The Azure reference architecture illustrates how

the pattern can be implemented using existing cloud identity, API, retrieval, key-management, and monitoring services. For regulated industries, the RPF offers a reusable foundation for trustworthy, accountable, and compliant enterprise AI execution.

REFERENCES

- [1] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proc. NeurIPS*, 2020.
- [2] S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” in *Proc. ICLR*, 2023.
- [3] T. Schick et al., “Toolformer: Language models can teach themselves to use tools,” in *Proc. NeurIPS*, 2023.
- [4] J. S. Park et al., “Generative agents: Interactive simulacra of human behavior,” in *Proc. UIST*, 2023.
- [5] J. Achiam et al., “GPT-4 technical report,” *arXiv:2303.08774*, 2023.
- [6] K. Greshake et al., “More than you’ve asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models,” in *Proc. AISec*, 2023.
- [7] Y. Liu et al., “Prompt injection attack against LLM-integrated applications,” *arXiv:2306.05499*, 2023.
- [8] OWASP Foundation, *OWASP Top 10 for Large Language Model Applications 2025*, 2025.
- [9] NVIDIA, “garak: LLM vulnerability scanner,” project documentation, 2025.
- [10] Open Policy Agent, “Policy language: Rego,” project documentation, 2026.
- [11] J. W. Cutler et al., “Cedar: A new language for expressive, fast, safe, and analyzable authorization,” *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, pp. 670–697, 2024.
- [12] Z. Xiang et al., “GuardAgent: Safeguard LLM agents via knowledge-enabled reasoning,” in *Proc. ICML*, 2025.
- [13] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, *Zero Trust Architecture*, NIST Special Publication 800-207, 2020.
- [14] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” *Proc. IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.
- [15] D. F. Ferraiolo and D. R. Kuhn, “Role-based access controls,” in *Proc. 15th National Computer Security Conference*, 1992.
- [16] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, 2023.
- [17] International Organization for Standardization, *ISO/IEC 42001:2023 Artificial Intelligence Management System*, 2023.
- [18] European Union, *Regulation (EU) 2024/1689 Artificial Intelligence Act*, 2024.
- [19] National Institute of Standards and Technology, *Generative AI Profile for the AI Risk Management Framework*, 2025.
- [20] P. Y. Zhong, S. Chen, R. Wang, M. McCall, B. L. Titzer, H. Miller, and P. B. Gibbons, “RTBAS: Defending LLM agents against prompt injection and privacy leakage,” *arXiv:2502.08966*, 2025.
- [21] M. A. Ferrag, N. Tihanyi, D. Hamouda, L. Maglaras, A. Lakas, and M. Debbah, “From prompt injections to protocol exploits: Threats in LLM-powered AI agents workflows,” *arXiv:2506.23260*, 2025.
- [22] A. Chhabra, S. Datta, S. K. Nahin, and P. Mohapatra, “Agentic AI security: Threats, defenses, evaluation, and open challenges,” *arXiv:2510.23883*, 2025.
- [23] Y. Ling, S. Yu, Z. Chen, and C. Fang, “Toward secure LLM agents: Threat surfaces, attacks, defenses, and evaluation,” *arXiv:2606.10749*, 2026.
- [24] E. Debenedetti et al., “AgentDojo: A dynamic environment to evaluate attacks and defenses for LLM agents,” *arXiv preprint*, 2024.
- [25] M. Bhatt et al., “CyberSecEval 2: A wide-ranging cybersecurity evaluation suite for large language models,” *arXiv preprint*, 2024.
- [26] R. Fang, R. Bindu, A. Gupta, Q. Zhan, and D. Kang, “LLM agents can autonomously hack websites,” *arXiv preprint*, 2024.
- [27] R. Fang, R. Bindu, A. Gupta, and D. Kang, “Teams of LLM agents can exploit zero-day vulnerabilities,” *arXiv:2406.01637*, 2024.
- [28] M. Feffer, A. Sinha, W. H. Deng, Z. C. Lipton, and H. Heidari, “Red-teaming for generative AI: Silver bullet or security theater?” *arXiv:2401.15897*, 2024.