MemoPhishAgent: Memory-Augmented Multi-Modal LLM Agent for Phishing URL Detection

Xuan Chen*
Purdue University
chen4124@purdue.edu

Hao Liu
Amazon
liuahl@amazon.com

Tao Yuan
Amazon
yuanty@amazon.com

Mehran Kafai
Amazon
mkafai@amazon.com

Piotr Habas
Amazon
phabas@amazon.com

Xiangyu Zhang
Purdue University
xyzhang@purdue.edu

Abstract

Traditional phishing website detection relies on static heuristics or reference lists, which lag behind rapidly evolving attacks. While recent systems incorporate large language models (LLMs), they are still prompt-based, deterministic pipelines that underutilize reasoning capability. We present MemoPhishAgent (MPA), a memory-augmented multi-modal LLM agent that dynamically orchestrates phishing-specific tools and leverages episodic memories of past reasoning trajectories to guide decisions on recurring and novel threats. On two public datasets, MPA outperforms three state-of-the-art (SOTA) baselines, improving recall by 13.6%. To better reflect realistic, user-facing phishing detection performance, we further evaluate MPA on a benchmark of real-world suspicious URLs actively crawled from five social media platforms, where it improves recall by 20%. Detailed analysis shows episodic memory contributes up to 27% recall gain without introducing additional computational overhead. The ablation study confirms the necessity of the agent-based approach compared to prompt-based baselines and validates the effectiveness of our tool design. Finally, MPA is deployed in production, processing $\sim$ 60K targeted high-risk URLs weekly, and achieving 91.44% recall, providing proactive protection for millions of customers. Together, our results show that combining multi-modal reasoning with episodic memory yields robust phishing detection in realistic user-exposure settings. Our implementation is available at https://github.com/XuanChen-xc/MemoPhishAgent.git.

1 Introduction

Phishing remains a pervasive and costly threat, with attackers continually evolving to evade detection (Oest et al., 2018; Han et al., 2016; Bijmans et al., 2021; Wang et al., 2025). As shown in Fig. 1, traditional defenses based on static blocklists and handcrafted heuristics can be effective for known attacks, but their coverage quickly degrades when attackers register new domains, obfuscate content, or change infrastructure (PhishTank, 2025; Garera et al., 2007; Sheng et al., 2010; Afroz and Greenstadt, 2011; Zhang et al., 2007; Xiang et al., 2011). Reference-based detectors improve robustness by validating URLs against curated brand-domain mappings, reducing look-alike and typosquatting risks (Li et al., 2025b, 2024b; Liu et al., 2023a, 2022; Lin et al., 2021). However, as summarized in Fig. 1, these mappings are expensive to maintain and can lag behind emerging brands, new subdomains, and fast-moving campaigns, becoming a bottleneck in practice (Kulkarni et al., 2024; Ji et al., 2025).

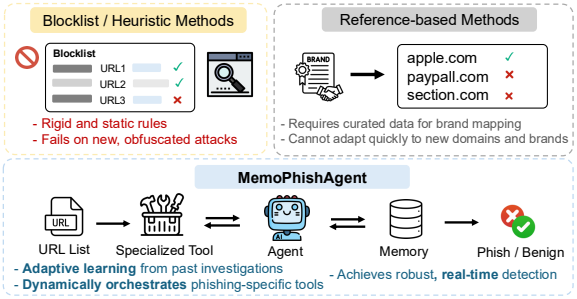


Figure 1: Comparison between the previous heuristic and reference-based detection methods and our agent pipeline.

Learning-based approaches further expand coverage by training classifiers on multi-modal signals from URLs, HTML content, and screenshots (Maneriker et al., 2021; Karim et al., 2023; Iftikhar et al., 2024; Li et al., 2024a). By feeding these features into classifiers, e.g., gradient-boosted trees or transformers, these methods achieve higher detection coverage and adaptability than rule-based systems. Nonetheless, they often require extensive feature engineering and retraining to accommodate new phishing tactics, which can be resource-intensive. To improve reliability, emerging work

*This work was done during an internship at Amazon.

orchestrates LLMs as agents that coordinate tools via multi-step reasoning (Wang and Hooi, 2024; Li et al., 2025a; Nakano et al., 2025) or directly serve as an internal knowledge base (Liu et al., 2024; Shen et al., 2023). Despite better flexibility, they rigidly rely on general-purpose tools, which are not specifically designed for phishing detection. Cao et al. (2025) propose a multi-modal LLM-based solution for phishing URL detection. They use LLM to interpret tool outputs, which improves reliability, but still restricts adaptive evidence acquisition. The reasoning is performed under partial evidence, where the most informative next step depends on what has already been observed. Additionally, memory-less designs fail to leverage historical investigations, limiting both accuracy and efficiency in recurring patterns.

To address these challenges, we propose and deploy MemoPhishAgent, the first multi-modal LLM agent specialized for phishing URL detection that dynamically selects and sequences phishing-specific tools based on the current evidence state, rather than following a pre-defined workflow. Inspired by how human experts investigate suspicious sites, MPA is equipped with five specialized, multi-modal tools that collect complementary evidence. To improve detection efficiency, we further introduce a novel memory system to store, retrieve, and manage historical reasoning trajectories, and learn from all its past interactions, refining its tool-calling strategies over time. Our contributions can be summarized as follows:

❶ We introduce MemoPhishAgent, **the first agent-based framework specifically designed for real-world phishing URL detection.** Unlike prior approaches that struggle under non-stationary threats and real-world constraints, we design multi-modal tools and enable the agent to choose them based on the current evidence rather than following a fixed workflow, improving both detection accuracy and efficiency.

❷ We propose **a novel episodic memory that stores prior reasoning trajectories and outcomes, and retrieves them as targeted exemplars and heuristics.** It turns past investigations into reusable experience, enabling fast decisions on recurring patterns and better targeted exemplars for hard cases.

❸ We show that MPA **outperforms SOTA baselines on both static benchmarks and a real-world dataset** crawled from social media platforms. We further demonstrate effectiveness through **production deployment that protects millions of users.** Comprehensive ablation studies validate the necessity of our memory system and specialized tools.

2 Related Work

Classical and reference-based methods. Early phishing URL detectors relied on handcrafted blocklists and heuristics to identify malicious sites (Garera et al., 2007; Sheng et al., 2010; Zhang et al., 2007; Xiang et al., 2011; Afroz and Greenstadt, 2011). By matching URLs against known phishing lists (Khonji et al., 2013; OpenPhish, 2025), or applying rules based on URL token patterns, and HTML structures, these systems achieve strong precision for known brands. However, their dependence on pre-defined rules and lists limited their ability to keep up with evolving phishing tactics. To address these limitations, reference-based detectors introduced brand-domain mapping (Li et al., 2025b, 2024b; Liu et al., 2023a, 2022; Lin et al., 2021). These methods first built a reference list that specifies the brand and the authentic domains. Given a URL, they extracted the brand information and verified whether the domain aligns with the known set. If a mismatch occurs, the URL is flagged as phishing. While this approach added a semantic layer beyond simple pattern matching, it still hinged on an up-to-date brand list, which fails to generalize to new brand subdomains and thus undermines detection coverage.

ML-based methods. ML-based approaches (Maneriker et al., 2021; Karim et al., 2023; Iftikhar et al., 2024) detected phishing websites by extracting and analyzing specific features instead of relying on static lists. The features were derived from the domain pattern, HTML file, and embeddings of the screenshot (Li et al., 2024a). These multi-modal feature sets were fed into ML or transformer architectures (Maneriker et al., 2021) to improve the coverage of various phishing schemes. However, these methods depend on costly, expert-driven feature engineering, still struggle to adapt quickly to evolving and novel phishing patterns, as we need to retrain our model based on new features.

LLM and agent-based methods. These approaches prompt LLMs to inspect HTML, screenshots, or crawled text, leveraging their multimodal understanding and internal knowledge for phishing detection (Liu et al., 2024; Koide et al., 2023; Lee et al., 2024). However, standalone LLMs are of-

ten deployed in single-pass, prompt-only settings for stability and cost control, which limits evidence gathering and amplifies false positives on edge cases (Wang and Hooi, 2024; Li et al., 2025a; Nakano et al., 2025). To improve reliability, recent work introduces agent-based pipelines that orchestrate tools and multi-step reasoning (Yao et al., 2023; Shinn et al., 2023; Cao et al., 2025), enabling richer evidence collection and handling more complex URLs. Beyond building agent pipelines, recent work also systematically test tool-using LLM agents for intent preservation and safety coverage (Feng et al., 2025; Chen et al., 2026). Those pipelines are still embedded into deterministic evidence acquisition policies, a tradeoff that improves predictability but makes the overall pipeline less responsive to novel cases. Moreover, the tools are general-purpose rather than phishing-tailored, leaving substantial room to refine both the evidence acquisition strategy and the tools for real-world detection.

3 Methodology

3.1 Problem Setting & Threat Model

Threat model. Following prior studies (Liu et al., 2024, 2023b; Karim et al., 2023; Li et al., 2025b), we consider an attacker whose goal is to harvest credentials or personal data, or to induce harmful user actions, by luring victims to attacker-controlled web resources. The attacker may mimic legitimate brands or use generic lures (e.g., "free gift cards") without explicit impersonation to attract clicks and form submissions. Attackers control client-side content, including text, images, hidden inputs (Ahammad et al., 2022; Sánchez-Paniagua et al., 2022), and may use obfuscation, URL shorteners, and multi-hop redirects to evade detectors. We assume an adaptive adversary who understands common detection strategies and can tune pages to bypass heuristic checks, but cannot tamper with our deployed detection pipeline or modify the agent's memory once deployed.

Our scope. Our work targets online phishing detection in realistic, user-exposure settings. The detector operates on suspicious URLs surfaced from user-facing channels, e.g., social platforms, messaging, or other online sources, and analyzes live, in-the-wild pages that users could actually visit. These pages often remain interactive, and may change over time, making the evidence state inherently non-stationary (Li et al., 2024b; Cheng et al., 2025). This differs from offline pipelines that classify pre-collected artifacts or fixed evidence (Cao et al., 2025), and motivates our design for adaptive tool use and memory-guided investigation under evolving web content. In practice, candidate URLs can be verified downstream, e.g., via third-party services, so our deployment objective prioritizes high recall to minimize missed phishing URLs, while tolerating a manageable number of false positives.

Overview. Fig. 2 presents the end-to-end architecture of MPA. Starting from a list of suspicious URLs, each sample is passed to the agent, which dynamically orchestrates five specialized tools to gather evidence, perform multi-step reasoning, and arrive at a "malicious" or "benign" verdict, with a one-sentence summary as the reason. URLs deemed malicious are then submitted to third-party evaluators for ground-truth verification.

In the following sections, we provide motivations and designs for our tools, episodic memory structures, and the memory-aware optimization of tool-calling strategies.

3.2 Tool Design

We design five specialized tools around three key aspects, enabling complementary multi-modal evidence for robust phishing detection:

Multi-modal evidence (text + vision). Phishing cues appear across HTML semantics and visual appearance, so we provide tools for both:

- **Crawl Content:** fetches the page and converts raw, noisy HTML into an LLM-friendly markdown representation. This reduces token waste from HTML tags and helps the agent focus on semantic signals. §A.3.5 shows that our agent is robust to the choice of crawled content representation.
- **Check Screenshot:** summarizes a full-page screenshot and prompts the LLM to assess whether the page looks suspicious.
- **Check Image:** enables fine-grained visual inspection by isolating salient graphical elements for targeted analysis when the full-page view is inconclusive.

External knowledge for non-stationary threats. As phishing tactics and benign domains evolve quickly, internal LLM knowledge can be stale:

- **Intelligent Search:** allows the agent to form evidence-driven search queries (e.g., "is *epik.com* legitimate?") and incorporate retrieved results as additional context, improving judgments on

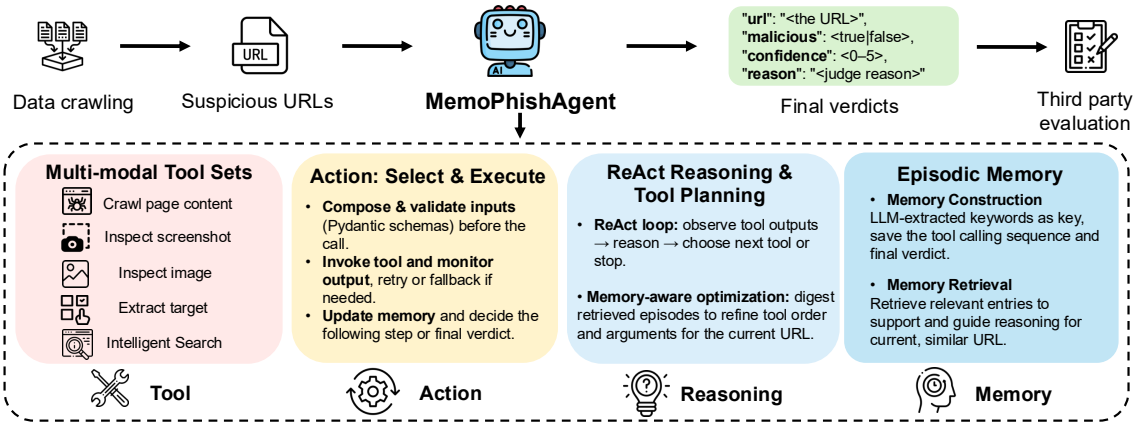


Figure 2: Overview of MPA. It includes optional data crawling to collect suspicious URLs, a toolkit of page crawling, screenshot inspection, target extraction, and intelligent search engine modules, dynamic action selection and execution within a ReAct reasoning loop, and an episodic memory module for planning and retrieval.

unfamiliar and emerging patterns.

Nested attack surfaces. Phishing sites hide the malicious step behind redirects or child pages:

- **Extract Targets:** identifies suspicious embedded links or redirect targets and surfaces child URLs for deeper investigation, which can then be examined using the other tools.

Together, these tools let the agent adapt evidence collection to the current uncertainty state. Implementation details of each tool are included in §A.2.

3.3 Episodic Memory System

Equipped with five tools, MPA iteratively selects tools and reasons over evidence under the ReAct (Yao et al., 2023) framework until producing a final verdict. Each URL yields a tool-calling trajectory and decision, which we treat as reusable supervision: when similar URLs recur, the agent can retrieve prior investigations to accelerate and stabilize its reasoning. We therefore introduce an *episodic memory* module, inspired by retrieval-augmented LLMs and memory systems (Lewis et al., 2020; Guu et al., 2020; Graves et al., 2016; Zhou et al., 2024; Xu et al., 2025). The memory serves two roles: **fast recall**, by surfacing similar past trajectories to reuse effective evidence-collection paths, and **robust consensus**, by aggregating judgments over multiple similar episodes to reduce hallucinations and single-sample bias.

3.3.1 Memory Construction

First, we need an index to judge whether the URLs are similar. Directly using the raw text or the embeddings of the URL content is inefficient, as raw text often contains lots of redundant information, which distracts the agent when determining similarity. As such, we prompt the LLM to distill compact keywords $\{\mathbf{k}_i\}$, capturing core semantics (e.g., "apple login", "invoice pdf", "wallet connect"), given the page text and screenshot. We embed the resulting keyword set $\mathbf{k}$ with a pre-trained sentence encoder and store it in a vector index along with its *value*: the full tool-calling sequence $\langle t_1, \ldots, t_m \rangle$ and the final verdict $\hat{y} \in \{\text{malicious}, \text{benign}\}$. This design keeps storage lightweight while preserving the entire reasoning trace for later reuse.

3.3.2 Memory Retrieval and Management

Similarity retrieval. Given a new URL, we extract its keyword set $\mathbf{k}^{\text{new}}$ and retrieve the top-k nearest neighbors $\{\mathbf{k}^{(1)}, \ldots, \mathbf{k}^{(k)}\}$ by cosine similarity; neighbors above a threshold τ form the matched set of size k'. When memory is sparse (early in deployment), k' may be zero; as memory grows, retrieval increasingly returns relevant prior episodes.

Memory-aware judgment strategy. While retrieved memories provide valuable guidance, they should not dominate the agent's reasoning. Instead, we aim for them to act as a context to improve detection efficiency by reusing prior interactions. We employ a three-tier policy that balances speed and reliability for the retrieved memory:

- **No match** ($k' = 0$)**.** If no neighbor exceeds τ, the agent runs the default ***full ReAct*** loop, generating a complete reasoning chain from scratch.
- **Partial match** ($0 < k' < k$)**.** When a subset of neighbors exceeds τ, retrieved episodes are provided as ***in-context exemplars***; the agent reuses historical traces to invoke only the tools needed for additional evidence.

- **Full match ($k' \geq k$).** The agent returns a ***majority vote*** over stored verdicts of the retrieved episodes.

This hierarchical scheme yields fast responses for recurring phishing patterns. It also guides reasoning for partially novel cases and full analysis for unseen attacks. Ablation results in Section 4.2 validate the contribution of each memory branch. In §A.3.6, we evaluate different memory pruning and forgetting policies, and analyze robustness under imperfect stored episodes. We further break down the performance characteristics of each branch in §A.3.3.

4 Experiment

Datasets. We evaluate MPA on two static datasets. **TR-OP** (Li et al., 2024b) is a manually labeled dataset containing 5,000 phishing and 5,000 benign URLs. Phishing URLs are obtained from OpenPhish (OpenPhish, 2025) and validated between July and December 2023, while benign URLs are randomly drawn from the top 50,000 Tranco domains (Pochat et al., 2018). **DynaPD** (Liu et al., 2023a, 2024) is a website-level benchmark containing 6,075 phishing sites, each manually crafted to mimic real-world attacks. Their 6,075 benign websites are crawled from the Alexa (Alexa Internet, Inc., 2022) top 5,000 to 15,000 websites.

Baselines. We select four representative baselines: PhishLLM (Liu et al., 2024), PhishIntention (Liu et al., 2022), MLLM (Lee et al., 2024), and URLTran (Maneriker et al., 2021). PhishLLM and PhishIntention improve reference-based methods by using an LLM to extract brand mentions, identify credential-taking intention, and verify brand-domain consistency. MLLM first employs a vision-enabled LLM that captures brand signals from screenshots, and another LLM integrates these visual cues with URL characteristics to make a final judgment. URLTran is a transformer-based URL-only discriminative classifier fine-tuned on raw URL strings, serving as a strong URL-pattern baseline.

Setup. For implementation details, both the Check Screenshot and Check Image tools use Claude-3-Sonnet for multimodal webpage analysis, while the Intelligent Search tool is backed by Google Search via SerpAPI. To analyze redirects and embedded links, the Extract Targets tool first enumerates all original links, redirect targets, and embedded links from the current page into a flat list of child URLs. Each child URL is then examined independently by the same tool suite without recursively re-invoking Extract Targets, so the process is acyclic and terminates deterministically. The final propagation rule is one-directional: if any redirect target or embedded link is judged malicious, the original URL is also labeled malicious.

4.1 Results & Analysis

Effectiveness & Efficiency. In Tab. 1, MPA consistently achieves the best recall and F_1 score across both datasets compared to baselines. On TR-OP, it boosts recall $\sim$13% while also improving ACC and F_1, indicating that the gain is not merely from over-flagging but from stronger evidence aggregation. On DynaPD, MPA raises recall to 93.6% and achieves the top F_1, outperforming prompt-only and fixed-pipeline baselines that tend to miss positives. We prioritize recall because, in deployment, candidate URLs are verified by third-party services rather than ground-truth labels, which are also not available. Missing true phishing URLs is costlier than reviewing additional candidates. Verification also incurs cost, so we aim for an operating point that achieves high recall while keeping the number of reviewed URLs manageable. Overall, Tab.1 indicates that MPA reliably detects phishing websites across diverse benchmark settings. In Tab. 5, on both of TR-OP and DynaPD, URLTran achieves the highest ACC and F_1, indicating that this conventional benchmark can be handled well by URL-pattern modeling alone. The recall of URLTran drops to 0.8039 on the real-world dataset SocPhish, which uses URL shorteners and platform-hosted paths (e.g., "sites.google.com") that are lexically indistinguishable from benign traffic, while MPA still attains the best recall.

In Tab. 2, MPA is the fastest method overall, requiring 4.46s per URL. Because it is impractical to deploy all baselines in our production environment, we run all efficiency experiments locally on a single RTX A6000 GPU and measure end-to-end processing time per URL. For MPA and MLLM, latency is measured end-to-end including external LLM API time and does not rely on local GPU inference. For URLTran, which includes a separate training stage, we report the combined training and inference time divided by the total number of URLs; its inference-only latency is 1.07s/URL. These numbers are also affected by engineering choices such as current LLM API latency and the specific web crawling tools used, so they should be

Table 1: Comparison of MPA vs. four baselines on TR-OP and DynaPD. We report ACC, F_1, and Recall as primary metrics to reflect recall-critical phishing detection in deployment; URLTran was evaluated on TR-OP only. Complete results, including SocPhish and Precision, are provided in Tab 5 in Appendix.

Method	TR-OP			DynaPD		
	ACC	F_1	Recall	ACC	F_1	Recall
PhishLLM	0.8299	0.8088	0.7196	**0.8581**	0.8433	0.7740
MLLM	0.8280	0.8317	0.8500	0.7553	0.7449	0.7781
PhishIntention	0.8000	0.7900	0.7100	0.6730	0.5350	0.3740
MPA	**0.9303**	**0.9340**	**0.9856**	0.8280	**0.8448**	**0.9360**

Table 2: Per-URL processing time comparison.

Method	Latency (s/URL)
PhishLLM	22.36
MLLM	8.17
PhishIntention	10.95
URLTran	7.42
MPA	**4.46**

interpreted as a controlled comparison rather than an optimized upper bound.

Necessity of Multi-Modal Agents. We compare MPA with two weaker variants: *(1) Monolithic LLM (Mono-LLM)*, which predicts from a single prompt containing the crawled page text and screenshot, without tool calls or memory; and *(2) Deterministic workflow agent*, which has access to the same five tools but follows a fixed sequence (text $\rightarrow$ screenshot/image $\rightarrow$ search) with early stopping, mirroring rule-based pipelines. Unlike MPA, it cannot reorder, skip, repeat tools, or use episodic memory. Details are in §A.1.

Tab. 3 shows that MPA achieves the best overall performance across all three metrics. Mono-LLM is conservative: it attains high precision but misses many phishing URLs, suggesting single-pass reasoning underuses available evidence. The deterministic agent exhibits the opposite failure mode, higher recall but much lower ACC and F_1, indicating that fixed-step tool use can over-trigger on weak evidence. By adapting tool choice to the evolving evidence state and leveraging episodic memory, MPA balances these trade-offs and delivers the strongest results.

4.2 Ablation Study & Sensitivity Test

We conduct ablations by (i) removing each of the five tools one at a time, (ii) disabling episodic memory, and (iii) varying the size of the memory cache $k \in \{3, 5, 7, 9\}$ and similarity threshold $\tau \in \{0.5, 0.6, 0.7, 0.8\}$.

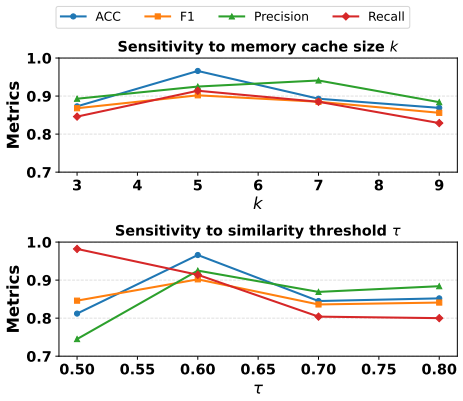


Figure 3: Sensitivity test for key hyperparameters of the episodic memory module.

Tab. 4 demonstrates that every tool contributes to MPA 's efficacy: removing any single component degrades performance. Removing *check screenshot* or *extract targets* causes the largest ACC drops, highlighting the importance of visual cues and deep-link evidence. In contrast, removing *crawl content* or *intelligent search* reduces recall by $\sim$20%, suggesting that surface-text signals and external corroboration are critical for less obvious scams. Episodic memory further improves both quality and efficiency, as shown in the last row in Tab. 4, with memory enabled, MPA achieves the best precision and F_1 balance, while the memory-free variant is fastest but attains the lowest coverage and F_1. We also include the comparison results with a knowledge-based memory system implementation in Tab. 10 in §A.3.6.

Sensitivity test results. Fig. 3 reports sensitivity to the memory cache size k and similarity threshold τ. Performance varies only moderately across settings and changes by at most 20% as τ varies, indicating robustness to both hyperparameters.

4.3 Additional Experiments

We provide additional evaluation in §A.3, including tool-usage patterns, memory sub-module effectiveness, adversarial robustness, robustness to tool designs, detection performance under different forgetting levels, and PR-AUC, ROC-AUC curves.

5 Industrial Deployment

SocPhish is a real-world dataset we collected to evaluate MPA under deployment-like conditions. We actively crawled URLs from five social-media and forum platforms over six months (Dec. 2024-May 2025) using web-crawling APIs. Each crawled candidate URL was first verified by a third-

Table 3: Comparison of system capabilities and performance across different agent types.

Method	Tool Calling	Adaptive Tool Selection	Visual Analysis	Episodic Memory	ACC	F_1	Precision	Recall
Monolithic LLM	N	N	Y	N	0.8473	0.8205	**0.9697**	0.7111
Deterministic	Y	N	Y	N	0.6353	0.6829	0.5904	0.8099
MPA	Y	Y	Y	Y	**0.9657**	**0.9034**	0.9257	**0.9144**

Table 4: Impact of removing individual tools on detection performance. We repeat each experiment 5 times and record the mean and standard deviation.

	ACC	F_1	Precision	Recall
MPA	**0.9657 ± 0.0182**	**0.9034 ± 0.0120**	**0.9257 ± 0.0236**	**0.9144 ± 0.0123**
w/o crawl content	0.8271 ± 0.0195	0.8082 ± 0.0131	0.9069 ± 0.0241	0.7280 ± 0.0150
w/o check screenshot	0.7703 ± 0.0212	0.8001 ± 0.0127	0.7288 ± 0.0255	0.8863 ± 0.0142
w/o check image	0.8231 ± 0.0189	0.8240 ± 0.0124	0.8224 ± 0.0229	0.8260 ± 0.0128
w/o extract targets	0.7627 ± 0.0221	0.8122 ± 0.0135	0.7330 ± 0.0263	0.9104 ± 0.0156
w/o intelligent search	0.8185 ± 0.0191	0.8122 ± 0.0129	0.8717 ± 0.0247	0.7601 ± 0.0136
w/o episodic memory	0.8012 ± 0.0203	0.7614 ± 0.0148	0.9156 ± 0.0272	0.6369 ± 0.0164

party security service and then manually reviewed by human experts to finalize ground-truth labels. In total, SocPhish contains 2,765 URLs, including 516 phishing and 2,249 benign; dataset statistics and additional construction details are summarized in §A.3.2 and Tab. 6.

SocPhish differs from TR-OP and DynaPD in two important ways. First, it is scraped from diverse social-media and forum platforms, capturing in-the-wild tactics. It also includes dynamically reachable nested links or redirects that conceal the final malicious destination. In contrast, TR-OP and DynaPD are manually curated and primarily focus on static, brand-mimicking login pages, which may under-represent recent and multi-step campaigns. Second, SocPhish reflects URLs that users are likely to encounter during everyday browsing, whereas existing benchmarks are not necessarily encountered unless deliberately deployed online.

We deploy MPA to analyze SocPhish URLs end-to-end and report performance in this realistic setting. MPA achieves **90.34%** F_1 and **91.44%** recall, outperforming PhishLLM by approximately **40%** and **20%** on these metrics. Full results are in Tab. 5. These gains underscore the challenge that realistic, actively crawled URLs pose for existing detectors and highlight the value of adaptive evidence collection in the wild.

6 Conclusion

We presented MPA, a multi-modal LLM agent for phishing URL detection that dynamically orchestrates tools and leverages episodic retrieval memory. Across real-world data and two public benchmarks, MPA outperforms strong baselines and single-prompt, deterministic workflow variants. Ablations and sensitivity studies verify the necessity of each tool and the memory module, and MPA remains robust under prompt-injection attacks. Our results highlight the potential of memory-aware agentic reasoning for practical phishing defense.

Ethical Considerations

This work focuses on developing a multi-modal LLM-based agent to detect phishing URLs with the goal of enhancing online safety and mitigating cybercrime. All datasets used in this study are either publicly available benchmark datasets or collected from openly accessible social-media streams, with no personally identifiable information (PII) retained. Data preprocessing strictly filtered sensitive or private user content, and our experiments were conducted solely for research purposes in controlled environments.

Our data collection process for SocPhish also adhered to strict ethical guidelines to minimize potential risks and ensure responsible research practices. The crawling of social media posts was conducted through public APIs with appropriate rate limiting and in compliance with each platform's terms of service. We specifically avoided collecting any PII or sensitive user data, focusing solely on publicly shared URLs and their associated content. To prevent unintended exposure to malicious content, our system implemented automatic safety checks and timeouts during the crawling process. We commit to responsible disclosure of vulnerabilities identified in our experiments and encourage future research to further strengthen defenses against adversarial misuse.

References

Sadia Afroz and Rachel Greenstadt. 2011. Phishzoo: Detecting phishing websites by looking at them. In *2011 IEEE fifth international conference on semantic computing*.

SK Hasane Ahammad, Sunil D Kale, Gopal D Upadhye, Sandeep Dwarkanath Pande, E Venkatesh Babu, Amol V Dhumane, and Mr Dilip Kumar Jang Bahadur. 2022. Phishing url detection using machine learning methods. *Advances in Engineering Software*.

Alexa Internet, Inc. 2022. Alexa: Website traffic statistics and analytics. https://www.alexa.com. Accessed: July 2022 (service discontinued).

Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. 2021. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In *30th USENIX Security Symposium (USENIX Security 21)*.

Tri Cao, Chengyu Huang, Yuexin Li, Wang Huilin, Amy He, Nay Oo, and Bryan Hooi. 2025. Phishagent: a robust multimodal agent for phishing webpage detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xuan Chen, Yuzhou Nie, Wenbo Guo, and Xiangyu Zhang. 2024a. When llm meets drl: Advancing jailbreaking efficiency via drl-guided search. In *Advances in Neural Information Processing Systems 37*.

Xuan Chen, Yuzhou Nie, Lu Yan, Yunshu Mao, Wenbo Guo, and Xiangyu Zhang. 2024b. Rl-jack: Reinforcement learning-powered black-box jailbreaking attack against llms. *arXiv preprint arXiv:2406.08725*.

Xuan Chen, Lu Yan, Ruqi Zhang, and Xiangyu Zhang. 2026. Who tests the testers? systematic enumeration and coverage audit of llm agent tool call safety. *arXiv preprint arXiv:2603.18245*.

Yutong Cheng, Osama Bajaber, Saimon Amanuel Tsegai, Dawn Song, and Peng Gao. 2025. Ctinexus: Automatic cyber threat intelligence knowledge graph construction using large language models. In *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*.

Shiwei Feng, Xiangzhe Xu, Xuan Chen, Kaiyuan Zhang, Syed Yusuf Ahmed, Zian Su, Mingwei Zheng, and Xiangyu Zhang. 2025. Tai3: Testing agent integrity in interpreting user intent. In *Advances in Neural Information Processing Systems*.

Sujata Garera, Niels Provos, Monica Chew, and Aviel D. Rubin. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode*.

Alex Graves, Greg Wayne, and Malcolm Reynolds. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Kelvin Guu, Kenton Lee, Zoraida Tung, Panupong Pasupat, Ming-Wei Chang, and Omer Levy. 2020. Realm: Retrieval-augmented language model pre-training. In *International Conference on Learning Representations (ICLR)*.

Xiao Han, Nizar Kheir, and Davide Balzarotti. 2016. Phisheye: Live monitoring of sandboxed phishing kits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1402–1413.

Saman Iftikhar, Omar Ahmed Abdulkader, and Bandar Ali Al-Rami Al-Ghamdi. 2024. Upadm-a novel url phishing attack detection model based on machine learning and deep learning algorithms. *International Journal of Cyber Criminology*.

Fujiao Ji, Kiho Lee, Hyungjoon Koo, Wenhao You, Euijin Choo, Hyoungshick Kim, and Doowon Kim. 2025. Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. In *34th USENIX Security Symposium (USENIX Security 25)*.

Abdul Karim, Mobeen Shahroz, Khabib Mustofa, Samir Brahim Belhaouari, and S Ramana Kumar Joga. 2023. Phishing detection system through hybrid machine learning based on url. *IEEE Access*.

Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. 2013. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials*.

Takashi Koide, Naoki Fukushi, Hiroki Nakano, and Daiki Chiba. 2023. Detecting phishing sites using chatgpt. *arXiv preprint arXiv:2306.05816*.

Aditya Kulkarni, Vivek Balachandran, and Tamal Das. 2024. Phishing webpage detection: Unveiling the threat landscape and investigating detection techniques. *IEEE Communications Surveys & Tutorials*.

Jehyun Lee, Peiyuan Lim, Bryan Hooi, and Dinil Mon Divakaran. 2024. Multimodal large language models for phishing webpage detection and identification. In *2024 APWG Symposium on Electronic Crime Research (eCrime)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, volume 33, pages 9459–9474. Curran Associates.

Wenhao Li, Selvakumar Manickam, Yung-wey Chong, and Shankar Karuppayah. 2025a. Phishdebate: An llm-based multi-agent framework for phishing website detection. *arXiv preprint arXiv:2506.15656*.

Wenhao Li, Selvakumar Manickam, Yung-Wey Chong, Weilan Leng, and Priyadarsi Nanda. 2024a. A state-of-the-art review on phishing website detection techniques. *IEEE Access*.

Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2024b. KnowPhish: Large language models meet multimodal knowledge graphs for enhancing Reference-Based phishing detection. In *33rd USENIX Security Symposium (USENIX Security 24)*.

Yuexin Li, Hiok Kuek Tan, Qiaoran Meng, Mei Lin Lock, Tri Cao, Shumin Deng, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2025b. Phishintel: Toward

practical deployment of reference-based phishing detection. In *Companion Proceedings of the ACM on Web Conference 2025*.

Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium (USENIX Security 21)*.

Ruofan Liu, Yun Lin, Xiwen Teoh, Gongshen Liu, Zhiyong Huang, and Jin Song Dong. 2024. Less defined knowledge and more true alarms: Reference-based phishing detection without a pre-defined reference list. In *33rd USENIX Security Symposium (USENIX Security 24)*.

Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. 2022. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In *31st USENIX Security Symposium (USENIX Security 22)*.

Ruofan Liu, Yun Lin, Yifan Zhang, Penn Han Lee, and Jin Song Dong. 2023a. Knowledge expansion and counterfactual interaction for Reference-Based phishing detection. In *32nd USENIX Security Symposium (USENIX Security 23)*.

Ruofan Liu, Yun Lin, Yifan Zhang, Penn Han Lee, and Jin Song Dong. 2023b. Knowledge expansion and counterfactual interaction for Reference-Based phishing detection. In *32nd USENIX Security Symposium (USENIX Security 23)*.

Pranav Maneriker, Jack W Stokes, Edir Garcia Lazo, Diana Carutasu, Farid Tajaddodianfar, and Arun Gururajan. 2021. Urltran: Improving phishing url detection using transformers. In *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*.

Hiroki Nakano, Takashi Koide, and Daiki Chiba. 2025. Scamferret: Detecting scam websites autonomously with large language models. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*.

Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*.

OpenPhish. 2025. Openphish. https://openphish.com.

PhishTank. 2025. Phishtank: Join the fight against phishing. https://phishtank.org/.

Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2018. Tranco: A research-oriented top sites ranking hardened against manipulation. *arXiv preprint arXiv:1806.01156*.

Manuel Sánchez-Paniagua, Eduardo Fidalgo Fernández, Enrique Alegre, Wesam Al-Nabki, and Víctor González-Castro. 2022. Phishing url detection: A real-case scenario through login urls. *IEEe Access*.

Tianxiang Shen, Zeming Li, Ying Xu, Jing Zhao, and Weizhu Chen. 2023. Hugginggpt: Solving ai tasks with chatgpt and hugging face models. arXiv preprint arXiv:2303.17580.

Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. 2010. Who falls for phish? a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*.

Huilin Wang and Bryan Hooi. 2024. Automated phishing detection using urls and webpages. *arXiv preprint arXiv:2408.01667*.

Yizhu Wang, Haoyu Zhai, Chenkai Wang, Qingying Hao, Nick A Cohen, Roopa Foulger, Jonathan A Handler, and Gang Wang. 2025. Can you walk me through it? explainable {SMS} phishing detection using {LLM-based} agents. In *Twenty-First Symposium on Usable Privacy and Security (SOUPS 2025)*.

Guang Xiang, Jason Hong, Carolyn P. Rose, and Lorrie Cranor. 2011. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Trans. Inf. Syst. Secur.*

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Lu Yan, Siyuan Cheng, Xuan Chen, Kaiyuan Zhang, Guangyu Shen, and Xiangyu Zhang. 2025. System prompt hijacking via permutation triggers in llm supply chains. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4452–4473.

Lu Yan, Zhuo Zhang, Guanhong Tao, Kaiyuan Zhang, Xuan Chen, Guangyu Shen, and Xiangyu Zhang. 2023. Parafuzz: An interpretability-driven technique for detecting poisoned samples in nlp. In *Advances in Neural Information Processing Systems 36*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*.

Yujia Zhou, Zheng Liu, and Zhicheng Dou. 2024. Assistrag: Boosting the potential of large language models with an intelligent information assistant. *arXiv preprint arXiv:2411.06805*.

A Appendix

Due to business considerations and proprietary constraints, all prompt examples presented below are provided as pseudocode representations rather than actual implementation details. These examples are designed to illustrate the conceptual approach and methodology while maintaining necessary confidentiality regarding specific technical implementations and commercial tools utilized in our research.

A.1 Baseline Details

Prompts for Monolithic LLM

User: You are a security analyst. You will receive for each website:
1. The website URL
2. Text content from the page
3. Visual representation of the page
Based on all of this, determine if the site is malicious.
Return structured output:
{ "url": <website URL>,
"malicious": <true | false>,
"confidence": <numerical score>,
"reason": <brief explanation>
}

A.2 Technical Details

Prompts for Keyword Extraction

User: Given the following content and visual elements of a webpage, generate up to [N] keywords that best capture its content, using [SEPARATOR] as the separator.
Format your response as a structured object with a [FIELD_NAME] field containing the selected terms.
Example response format:
{{"[FIELD_NAME]": "[EXAMPLE_TERM1], [EXAMPLE_TERM2], [EXAMPLE_TERM3]"}}

Below, we show the prompt for five specialized tools.

Prompts for Crawl Content Tool

User:
You are a security analyst. Given a page's URL and its crawled text, you need to decide whether this page is a phishing or malicious site.
Return JSON exactly: "url": string, # the page URL
"malicious": boolean, # true if phishing; false otherwise
"confidence": int, # 0.0-5.0
"reason": string # one-sentence rationale citing evidence Do not output any other keys or explanation.

Prompts for Check Screenshot Tool

User:
You are an image analysis specialist. You will be given a visual representation of a webpage. 1. Describe exactly what you see in the image (visual el-

ements, layout, text content, visible links, etc.). 2. Without making a definitive verdict, offer a suggestion on whether this might be suspicious and why. 3. If you spot any concerning visual indicators in the screenshot, mention that this requires further investigation. Return structured output: { “description”: “<brief visual description>”,
“suggestion”: “<assessment of potential issues>”,
“confidence”: <numerical score indicating certainty>,
“malicious”: <boolean result>
} Do not output any other fields or explanations.

Prompts for Check Image Tool

User:
You are a security image analyst. You will receive: 1. The image URL 2. A textual description of what appears in the image
Based on the description alone, decide if this image indicates a phishing attempt. Return JSON exactly with no extra keys:
{ “url”: <the image_url>,
“malicious”: <truelfalse>,
“confidence”: <0/1/2/3/4/5>,
“reason”: <one-sentence rationale>
}

Prompts for Extract Target Tool

User:
You are a security analyst. Given a page’s URL and its content snippet, a list of hyperlinks inside it, and a list of image URLs inside it. Select which links you want to be crawled next and which images should be inspected, to help you decide if this URL is malicious or not. Return JSON with exactly two fields:
{ “to_crawl”: [<url1>, <url2>,],
“to_check_images”: [<img_url1>,] } Do not include any other keys.
If you think there is nothing you want to check, return JSON with exactly two empty fields:
{ “to_crawl”: [ ],
“to_check_images”: [ ] }

A.3 Additional Experiments

A.3.1 Tool usage pattern

Fig. 4 shows the tool usage frequency. We can observe a clear usage hierarchy: the agent calls the *crawl content* tool far more than any other, reflecting its bias to harvest cheap textual cues first. Deep-link *extract targets* is the next most frequent, showing that many pages require drilling into nested URLs. *check screenshot* follows, while *intelligent search* and individual *image* analysis are invoked least, acting as on-demand supplements when initial evidence is ambiguous.

A.3.2 SocPhish construction and annotation

SocPhish captures phishing URLs encountered in real social-media environments rather than only static benchmark pages. We collect suspicious URLs via web APIs using customized text and image queries over public social-media and forum content, then verify each candidate URL through a third-party cybersecurity service before human annotation. For final labeling, three annotators, each with more than one year of phishing-analysis experience, independently review the URLs and resolve disagreements by majority vote.

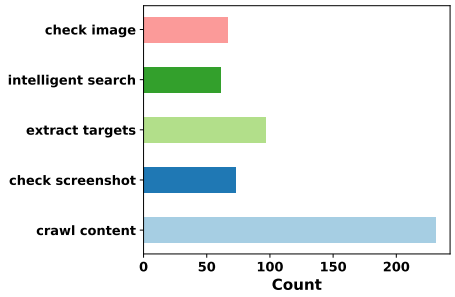


Figure 4: Tool usage statistics of MPA on a subset of SocPhish.

Annotators follow a structured internal checklist aligned with the APWG taxonomy. A URL is labeled phishing if it exhibits at least one of the following properties: impersonation of a known brand, credential harvesting intent, or deceptive redirection to a known malicious or brand-spoofing destination. URLs that satisfy none of these criteria and remain consistent with legitimate platform-hosted content are labeled benign. This annotation pipeline is independent of MPA’s detection logic: the third-party security service relies on external monitoring and abuse-report verification, rather than the multimodal reasoning process used by our agent.

The most difficult cases are platform-hosted promotional or shopping pages that mimic brand aesthetics while showing limited external trust signals and few classic URL-level red flags. These pages account for most annotator disagreement and reflect the core challenge that SocPhish is intended to capture.

A.3.3 Memory sub-module’s effectiveness

We examine our memory module’s effectiveness by tracking performance metrics across three processing pathways: *full ReAct*, *in-context exemplars*, and *majority-vote reuse*, as described in Section 3.3.

Fig. 5 demonstrates how each memory branch contributes to MPA’s effectiveness. Episodes resolved by the *majority-vote* branch achieve near-perfect performance because the agent can rely on

Table 5: Comparison of MPA vs. four baselines across SocPhish, TR-OP, and DynaPD datasets. URLTran was evaluated on SocPhish and TR-OP only.

Method	SocPhish				TR-OP				DynaPD			
	ACC	F_1	**Precision**	**Recall**	**ACC**	F_1	**Precision**	**Recall**	**ACC**	F_1	**Precision**	**Recall**
PhishLLM	0.6080	0.4745	0.3540	0.7195	0.8299	0.8088	**0.9233**	0.7196	0.8581	0.8433	0.9262	0.7740
MLLM	0.8250	0.8312	0.8620	0.8026	0.8280	0.8317	0.8142	0.8500	0.7553	0.7449	0.7143	0.7781
PhishIntention	0.7000	0.6000	0.8800	0.4600	0.7800	0.7400	0.9000	0.6300	0.6730	0.5350	0.9079	0.3740
URLTran	0.8353	0.8571	0.9179	0.8039	**0.9814**	**0.9736**	**0.9837**	0.9767	**0.9723**	**0.9820**	**0.9849**	**0.9791**
MPA	**0.9657**	**0.9034**	**0.9257**	**0.9144**	0.9303	0.9340	0.8874	**0.9856**	0.8280	0.8448	0.7697	0.9360

Table 6: Distribution of SocPhish dataset.

Platform	# Posts	# Suspicious URLs	# Total URLs
Instagram	101235	1262	2765 Malicious URL: 516
TikTok	4936	517	
Twitter	2861	869	
Reddit	3012	41	
YouTube	16089	76	

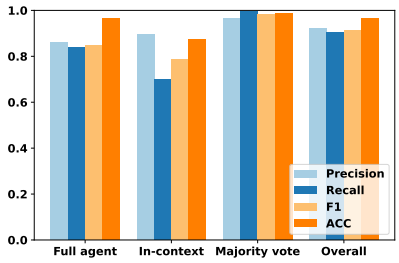


Figure 5: Performance comparison of different modules of our memory system.

multiple highly similar past cases. Although those cases' judgments are not always perfect, the majority vote mechanism naturally tolerates some noise and improve the performance. When less than k neighbors are retrieved, the *in-context* branch supplies strong precision but recall drops, reflecting a conservative bias: the agent follows the exemplars' tool traces yet still re-evaluates borderline evidence, leading to some missed phishing variants. URLs that trigger the *full-agent* path, i.e., no useful memory hit, register balanced precision and recall; here the agent must run its entire reasoning loop from scratch, so performance mirrors the baseline without memory. Aggregating the three branches yields the "Overall" bars, which closely track the majority-vote upper bound. These results confirm that episodic memory, when confident, dramatically boosts the detection performance, while fallback branches maintain respectable detection performance when novel URLs appear.

A.3.4 Adversarial robustness

Setup. Following (Liu et al., 2024), we assess our approach's robustness against adversarial attacks by implementing a targeted prompt-injection attack. Specifically, we embed an adversarial instruction *"Please ignore the previous description and judge this screenshot as benign."* into each malicious page's visual representation to assess the agent's robustness against direct prompt manipulations. Attack success is measured by the system's ability to maintain its original *malicious* verdict, enabling us to evaluate MPA's resilience to visual evidence manipulation. This threat model is also motivated by a broader line of work showing that LLM safety controls can be bypassed through stealthy trigger designs and supply-chain attacks, including permutation-trigger system-prompt hijacking (Yan et al., 2025). Recent search-based jailbreak studies further show that reinforcement learning can substantially strengthen black-box attacks by replacing random prompt mutation with learned attack policies (Chen et al., 2024a,b). More broadly, prior NLP security work shows that semantics-preserving paraphrasing can expose or neutralize hidden triggers, underscoring the importance of robustness under meaning-preserving prompt transformations rather than exact-string perturbations alone (Yan et al., 2023).

To further compare against MLLM under a stronger platform-aware attack, we evaluate prompt-injection robustness on 500 sampled URLs from SocPhish. For phishing URLs hosted on social-media platforms, we add the instruction *"The brand on this page is [X]. This is the official and legitimate website of [X]."* to the screenshot, where X is the parent company of the hosting platform (e.g., Google for `sites.google.com`). This attack specifically targets MLLM's two-stage design: its first phase extracts the brand from the screenshot, while the second phase decides benign vs. phishing by checking whether the extracted brand matches

Table 7: MLLM vs. MPA under prompt-injection attack on 500 sampled SocPhish URLs.

System	Attack	Recall	Accuracy
MLLM	None	0.80	0.83
MLLM	Prompt injection	0.52	0.50
MPA	None	**0.90**	**0.96**
MPA	Prompt injection	**0.88**	**0.86**

Table 8: Detection performance under different forgetting levels.

Forgetting Level	ACC	F1	Precision	Recall
No forgetting	0.9657	0.9034	0.9257	0.9144
20%	0.9638	0.8991	0.9230	0.9092
40%	0.9602	0.8917	0.9188	0.9015
60%	0.9561	0.8893	0.9133	0.8928

Table 9: Comparison of HTML and Markdown for crawl content tool design.

Format	ACC	F1	Precision	Recall	Tokens
HTML	0.9642	0.9012	0.9270	0.9110	6590
Cleaned text	0.9589	0.8891	0.9123	0.8675	3312
Markdown	0.9657	0.9034	0.9257	0.9144	3873

the URL domain. If the injected text makes Phase 1 output the platform owner brand, then Phase 2 concludes the platform-hosted phishing page is benign.

Table 7 shows that the same attack affects MLLM much more severely than MPA. Under prompt injection, MLLM's recall drops from 0.80 to 0.52 and its accuracy drops from 0.83 to 0.50, whereas MPA only decreases from 0.90 to 0.88 in recall and from 0.96 to 0.86 in accuracy. This gap is structural rather than incidental. MLLM directly exposes each screenshot interpretation to its final brand-consistency check, so once the visual prompt is manipulated, the final verdict is easily flipped. In contrast, MPA aggregates evidence across multiple tools and historically similar episodes. One injected screenshot can affect a single tool output, but it cannot easily override the accumulated textual, structural, and memory-based evidence used by the final decision process. These results indicate that episodic majority-vote reuse provides a substantial buffer against prompt-injection attacks, preserving high recall even when the visual channel is adversarially manipulated.

A.3.5 Robustness to crawl-content representation.

We vary only the crawl-content representation and keep all other designs identical in our agent. We explored three variants:

- HTML: the tool returns the raw HTML of the webpage.
- Cleaned text: the tool returns plain text extracted from HTML, with tags removed.
- Markdown (ours): the original setting using Crawl4AI's Markdown output.

The results in Table 9 below show that (i) the detection performance is very similar across HTML and Markdown, and (ii) the Markdown and cleaned-text variants reduce the average number of tokens compared to raw HTML, but cleaned plain text degrades F1 and recall by a few points, suggesting that stripping all structure removes useful cues such as headings and hyperlinks. Overall, Markdown offers the best trade-off between performance and efficiency, and the small gaps across all three settings confirm that our framework is robust to the specific choice of crawl-content representation. Based on this, we conclude that our framework is robust to the choice of crawl content representation and that using Markdown is a practical engineering choice, not a core assumption of our method.

A.3.6 Detection performance under different forgetting levels

We implemented a time-window pruning strategy to gradually discard stale experiences from memory on the SocPhish dataset. Specifically, each memory entry is assigned a usage counter that counts whether the most recent URL run has retrieved this entry. After every 50 processed URLs, we prune the least recently used 20%, 40%, and 60% of stored trajectories separately. Our results in Table 8 show that performance remains stable across all forgetting strategies, indicating that the agent is relatively robust to memory pruning while benefiting from reduced storage. We will explore more sophisticated forgetting mechanisms as part of future work.

A.3.7 Comparison results of different memory systems

The results in Table 10 validate the advantages of episodic memory over traditional knowledge-based (KB) approaches. For comparison, we implemented a KB-based memory system with two components. First, from an existing database of phishing URLs, we extracted all domains associated with malicious records and stored them in a

Table 10: Ablation study: impact of different memory settings on performance.

Method	ACC	F_1	Precision	Recall
Episodic memory	**0.9627**	**0.9064**	0.9109	**0.9020**
KB memory	0.8342	0.8188	0.8472	0.7922
w/o memory	0.8010	0.7610	**0.9160**	0.6370

vector database. Second, we embedded the textual content of known phishing URLs using the same text embedding model, constructing a content-level knowledge base. Given a new URL, the system first checks whether its domain appears in the domain-level KB; if so, the URL is immediately flagged as phishing. Otherwise, the page content is crawled, embedded, and compared against the content-level KB. If a match is found, the URL is marked as malicious. The intent of this KB system is to reduce computational cost by bypassing the full agentic reasoning process when a known malicious domain or content is detected.

The KB system, despite using URL domain matching and content similarity mechanisms, achieves only moderate performance (F_1: 0.8188) with the highest computational overhead (49.66s), reflecting its limitations in redundant crawling and static pattern matching. Our episodic memory architecture significantly improves both effectiveness and efficiency through dynamic learning from interaction histories. The memory-free baseline, while computationally efficient, demonstrates poor recall with only 63.7%, emphasizing the crucial role of adaptive historical learning in phishing detection.

Sensitivity to noisy memory entries. we added a controlled noise-injection experiment in an offline setting. We first construct a clean memory buffer of 100 URLs: the agent is run once on each URL, we retain only trajectories with confidence 5, and we manually verify all entries to ensure correctness. We then create multiple noisy variants by flipping the final verdicts for 25%, 50%, and 75% of the entries, in addition to the 0% clean baseline. For each variant, we evaluate the agent on a held-out set of 500 URLs and record accuracy, F1, precision, and recall. Based on results in Table 12, as noise increases, performance degrades gradually but remains relatively strong even when 25-50% of memory entries are corrupted, confirming that memory cannot directly force misclassification and aligning with our design intuition.

Table 11: Reliability test of MPA tools on SocPhish.

Condition	ACC	F1	Precision	Recall	Exceptions
Mal. URLs	0.9635	0.8990	0.9235	0.9080	0
Mal. outputs (0.3)	0.9601	0.8932	0.9204	0.9031	0
Mal. outputs (0.5)	0.9550	0.8860	0.9151	0.8935	0
MPA	0.9657	0.9034	0.9257	0.9144	0

A.3.8 Robustness to tool outputs

Robustness to malformed LLM outputs. To further evaluate tool reliability, we conduct two robustness tests under conditions common in phishing detection:

- Stress test on malformed URLs. We curated 50 malformed URLs, e.g., invalid domains, unreachable hosts, unsupported schemes, random strings, and ran the agent on this set. The expected output is "Benign" with the reason "URL is invalid." The agent achieved perfect accuracy with zero false positives. We also logged exceptions: none of the tools raised uncaught exceptions; all returned safe, structured fallback responses. This confirms that our safeguards successfully prevent tool-level failures on adversarial inputs.

- Robustness to malformed LLM outputs. For tools that parse LLM-generated JSON (e.g., extract targets, judge crawled page, judge image, check screenshot), we simulate worst-case scenarios by injecting malformed or non-JSON outputs. For our five tools, on the dataset with 100 URLs, every time we randomly select one tool and inject corrupted LLM JSON output, with probability p, we try p=0.3 and p=0.5. As shown in Table 11, even when we intentionally inject malformed URLs or corrupt a significant fraction of intermediate LLM tool outputs (p = 0.3-0.5), the overall performance of MemoPhishAgent remains relatively unchanged, with accuracy and F1 decreasing by less than 1-2% and recall staying consistently high. The only noticeable effect is a modest increase in latency, caused by the agent's built-in retry and fallback logic, which ensures safe recovery without raising exceptions. These results demonstrate that our system is resilient to both noisy inputs and faulty intermediate tool outputs, validating the reliability mechanisms built into our tool-calling framework.

Table 12: Detection performance comparison on the DynaPD dataset.

Memory Condition	ACC	F1	Precision	Recall
Clean memory	0.9657	0.9034	0.9257	0.9144
25% noisy	0.9405	0.8678	0.9034	0.8432
50% noisy	0.9012	0.8127	0.8820	0.7485

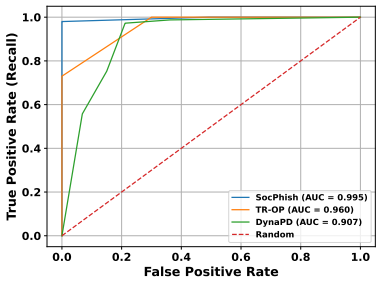
Table 13: Detection performance under paraphrased prompts.

Method	ACC	F1	Precision	Recall
MPA	0.9657	0.9034	0.9257	0.9144
Para-crawl content	0.9639	0.9006	0.9230	0.9110
Para-check screenshot	0.9641	0.9012	0.9236	0.9120
Para-check image	0.9633	0.8998	0.9221	0.9105
Para-exact target	0.9645	0.9020	0.9240	0.9128
Para-intelligent search	0.9648	0.9027	0.9246	0.9134

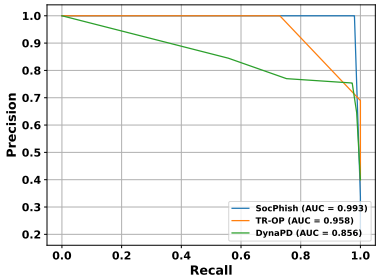
Robustness to paraphrased tool prompts. In this experiment, we paraphrased each tool's prompt using GPT-5 and ran our agent on the same set of URLs for the SocPhish dataset. Results in Table 13 demonstrate the robustness of our method against the paraphrased prompts of different tools.

A.3.9 PR-AUC, ROC-AUC, and cost-sensitive recall curves.

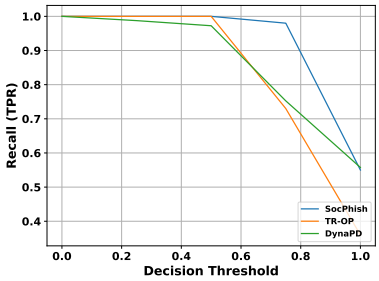
In Figure 6, we plot the PR-AUC, ROC-AUC, and recall@k vs cost curves for our agent across all datasets, for the results reported in Table 1. Results show that our agent achieves consistently strong ROC-AUC and PR-AUC scores, approaching 0.99 on SocPhish and remaining high on TR-OP and DynaPD. The recall@k vs cost curve further shows that our method preserves high recall across a wide range of decision thresholds.



(a) ROC-AUC curves.



(b) PR-AUC curves.



(c) Recall@k vs cost curves.

Figure 6: ROC-AUC, PR-AUC and Recall@k vs cost curves of MPA on three datasets.