Aleksander Anchimiuk[1]

Politechnika Białostocka

https://doi.org/10.29316/9788368103304_1

INSECURE OUTPUT HANDLING IN LARGE LANGUAGE MODELS

NIEBEZPIECZNE PRZETWARZANIE DANYCH WYJŚCIOWYCH W DUŻYCH MODELACH JĘZYKOWYCH

Abstract

Large Language Models (LLMs) are increasingly deployed across domains such as automation, healthcare, and software development. While existing research predominantly focuses on input-related risks, such as prompt injection and adversarial attacks, the critical challenge of Insecure Output Handling (IOH), as identified by the OWASP Top 10 for LLMs, remains understudied in the academic literature. This paper addresses this gap by providing a formal analysis of IOH as a distinct security threat, in which model-generated outputs introduce vulnerabilities into downstream systems. Through controlled experiments focusing on XSS vulnerabilities, we demonstrate significant variation in model security postures, with baseline exploit rates ranging from 4.2% to 27.6% across different architectures. Crucially, our findings validate OWASP's concerns by showing that standard sanitisation techniques, recommended to mitigate IOH, provide 100% protection against successful XSS exploits in our experimental setup while maintaining negligible performance overhead. Furthermore, we examine how the European Union's AI

[1] student, e-mail: aleksander.anchimiuk.118968@student.pb.edu.pl

Act establishes concrete obligations for addressing such risks, positioning IOH mitigation not just as a technical best practice but as a regulatory imperative. The empirical evidence presented demonstrates that implementing output validation is both technically feasible and increasingly necessary for the deployment of compliant AI systems.

Keywords: AI security, large language models (LLMs), insecure output handling, application security, cross-site scripting (XSS), output sanitisation, EU AI Act, regulatory compliance

Streszczenie

Modele językowe dużej skali (LLM) są coraz częściej wykorzystywane w różnych dziedzinach, takich jak automatyzacja, opieka zdrowotna czy rozwój oprogramowania. Podczas gdy dotychczasowe badania koncentrują się głównie na zagrożeniach związanych z danymi wejściowymi, takich jak wstrzykiwanie promptów czy ataki adwersarialne, kluczowe wyzwanie, jakim jest niebezpieczne przetwarzanie danych wyjściowych (Insecure Output Handling, IOH) – zidentyfikowane przez listę OWASP Top 10 dla LLM – pozostaje słabo zbadane w literaturze naukowej. Niniejszy rozdział wypełnia tę lukę, dostarczając formalną analizę IOH jako odrębnego zagrożenia dla bezpieczeństwa, w ramach której generowane przez modele dane wyjściowe wprowadzają podatności do systemów je wykorzystujących. Na podstawie kontrolowanych eksperymentów dotyczących podatności typu XSS wykazano istotne różnice w poziomie bezpieczeństwa różnych architektur modeli – odsetek udanego wykorzystania podatności w badanej próbie wynosił od 4,2% do 27,6%. Co istotne, wyniki weryfikują obawy OWASP, pokazując, że w przeprowadzonym eksperymencie XSS standardowe techniki sanityzacji – rekomendowane do mitygacji IOH – zapewniły 100% ochrony przed skutecznymi atakami w badanej próbie, przy jednoczesnym minimalnym wpływie na wydajność. Ponadto rozdział analizuje, w jaki sposób Rozporządzenie UE o Sztucznej Inteligencji (AI Act) nakłada konkretne obowiązki w zakresie ograniczania tego typu ryzyk, czyniąc łagodzenie skutków IOH nie tylko najlepszą praktyką techniczną, lecz także wymogiem regulacyjnym. Przedstawione dane empiryczne dowodzą, że wdrożenie walidacji danych wyjściowych jest zarówno technicznie wykonalne, jak i coraz bardziej niezbędne dla zgodnego z przepisami wdrażania systemów AI.

Słowa kluczowe: bezpieczeństwo AI, modele językowe dużej skali (LLM), niebezpieczne przetwarzanie danych wyjściowych (Insecure Output Handling, IOH), bezpieczeństwo aplikacji, cross-site scripting (XSS), sanityzacja danych wyjściowych, Rozporządzenie UE o Sztucznej Inteligencji (AI Act), zgodność regulacyjna

Introduction

Large Language Models (LLMs) have become essential components of modern digital ecosystems, supporting chatbots, virtual assistants, code generation, automated testing, and database query automation (Sajadi et al., 2025). Security research on LLMs has predominantly focused on input-oriented threats, including prompt injection, adversarial manipulation, and training data poisoning (He et al., 2023; Gupta et al., 2017). However, outputs generated by LLMs can also serve as vectors for malicious activity. This phenomenon, which this paper defines as Insecure Output Handling (IOH), occurs when downstream systems consume model outputs without proper safeguards, such as sanitisation, schema validation, or parameterisation (Siddiq et al., 2023; Alwan et al., 2017; Hong et al., 2024; Luo et al., 2024).

The OWASP GenAI Top 10 2025 list identifies Improper Output Handling as a critical risk, highlighting that unsafe consumption of LLM outputs can lead to exploits, including XSS, SQL Injection, and remote code execution (OWASP GenAI Security Project, 2025). Building on this, the paper contributes by (1) formalising IOH as a distinct LLM security risk, (2) providing an empirical case study for XSS with mitigation analysis, and (3) situating IOH within the AI Act's regulatory framework, highlighting its technical and compliance relevance.

Empirical studies further demonstrate that LLM-generated code and HTML may propagate vulnerabilities if handled insecurely (He et al., 2023; Alwan et al., 2017; Hong et al., 2024). Despite these findings, systematic analyses of IOH remain limited. From a regulatory perspective, the European Union's Artificial Intelligence Act (AI Act, 2024) mandates risk management, robustness, technical documentation, and cybersecurity for high-risk AI systems (European Union, 2024 (AI Act)). While IOH is not named explicitly, its mitigation directly supports compliance with these provisions.

By synthesising perspectives from classical application security, emerging AI risk classifications, and regulatory policy, this work advances the discourse on AI security engineering. It provides actionable insights for researchers, practitioners, and policymakers.

Threat Definition: Insecure Output Handling (IOH)

Concept and Definition

Insecure Output Handling (IOH) refers to security risks arising when outputs generated by a Large Language Model (LLM) are integrated into downstream systems without sufficient safeguards, such as sanitisation, schema enforcement, or context-aware filtering. Unlike adversarial prompt injection, where an attacker manipulates the model's input, IOH exploits the fact that model-generated text or code may contain unsafe patterns itself. These outputs, when executed, displayed, or queried by other software components, can unintentionally enable attacks such as script injection, query manipulation, or even remote execution of commands. Prior research on code generation and AI-assisted programming confirms that LLMs often replicate insecure practices present in their training data, making them prone to propagate vulnerabilities by default (Siddiq et al., 2023; Alwan et al., 2017; Hong et al., 2024; Luo et al., 2024).

Adversarial Assumptions and Scope

To analyse IOH systematically, we consider the following assumptions:

- adversary model: the attacker does not require direct access to model internals (weights, training set). Instead, the attack surface is the downstream system that consumes raw LLM output,
- capabilities: by crafting natural prompts or leveraging ordinary user inputs, an attacker can steer the model toward producing code, queries, or markup that contain exploitable patterns,
- impact scope: any environment where LLM outputs are automatically processed – such as web rendering engines, database interfaces, automation pipelines, or monitoring tools – is within the risk boundary.

The threat of Insecure Output Handling (IOH) materialises when vulnerable downstream components process unsanitized LLM output. The following scenarios exemplify how this core vulnerability can lead to classic software exploits:

- Cross-Site Scripting (XSS): LLMs tasked with generating HTML, Markdown, or other content for web interfaces can inadvertently produce

payloads containing executable JavaScript. For instance, an LLM might be prompted to create a user-friendly message and output a string like <script>alert(“XSS”) </script>. If this output is directly embedded into a web page without proper encoding or sanitisation, it will execute in the user’s browser, leading to a client-side attack (Gupta et al., 2017; Siddiq et al., 2023). This scenario highlights IOH as the critical failure point where the application trusts and renders the model’s raw output.

- SQL Injection (SQLi): The widespread use of natural-language-to-SQL systems presents a significant IOH risk. An LLM might generate a SQL query constructed via string concatenation based on a user’s request. Without mandatory output parameterisation, a maliciously crafted prompt can induce the model to create a query containing a tautology (e.g., ‘OR ‘1’=’1’) or a UNION-based payload. If this generated string is executed directly against the database, it can lead to unauthorised data access, manipulation, or exfiltration (Alwan et al., 2017; Hong et al., 2024; Luo et al., 2024).
- Extended Attack Vectors:
  - Remote Code Execution (RCE): LLMs generating shell commands, CI/CD pipeline configurations, or code snippets can produce output that, when executed automatically by a downstream interpreter or shell, results in unauthorised command execution on the host system.
  - Server-Side Request Forgery (SSRF): An LLM suggesting URLs or API call structures might be manipulated into outputting an internal endpoint (e.g., http://169.254.169.254/). If a downstream component executes this request without validation, it could access sensitive metadata or internal services.
  - Configuration and Log Injection: LLM outputs embedded into configuration files, log entries, or monitoring dashboards can poison these systems. Subsequent parsing of these poisoned entries by administrative tools can lead to data corruption, false alerts, or even command execution, depending on the parsing logic.

Characteristics of the IOH Risk

IOH is distinctive because it fuses classical software security concerns with probabilistic AI behaviour:

- Unpredictability of outputs: LLM responses are stochastic, making the presence of unsafe patterns difficult to predict or detect in advance (Siddiq et al., 2023).
- Dependency on integration context: The risk severity is shaped less by the LLM itself than by how its output is consumed (e.g., web rendering, SQL execution, shell interpretation).
- Effectiveness of lightweight mitigations: Empirical studies show that simple measures – such as HTML sanitisation, strict schema validation, and query parameterisation – dramatically reduce exploitability without degrading system performance (Alwan, Younis, 2017; Hong et al., 2024; Luo et al., 2024).

This framing positions IOH as a distinct class of AI-enabled vulnerability. It demonstrates the necessity of handling model outputs with the same rigour as that traditionally applied to untrusted user inputs, thereby bridging the gap between AI-specific risk categories and established secure coding practices.

Methodology

To empirically assess the risks of Insecure Output Handling (IOH) for Cross-Site Scripting (XSS) in LLM-powered applications, we designed and implemented a controlled, automated experimentation pipeline. The study focused on the security implications of model-generated HTML outputs, simulating a realistic scenario where an LLM assists in generating user interface components.

Experimental methodology employs a black-box empirical approach to evaluate Insecure Output Handling (IOH) risks in operational contexts. The framework treats LLMs as opaque systems, focusing on observable input-output behaviours rather than internal mechanisms. We adopt a universal validation paradigm where all model outputs undergo security screening, reflecting zero-trust principles in production AI systems. The design incorporates comparative

analysis across different model architectures to identify consistent security patterns versus model-specific vulnerabilities.

Experimental Design and Model Selection

To quantitatively assess IOH risks, we employed a comparative framework analysing two architecturally distinct open-weight models:

- DeepSeek-R1 1.5B (DeepSeek, 2025): Representing the emerging class of efficient, reasoning-optimised models.
- Llama 3 8B (Meta, 2024 (Llama 3): A state-of-the-art general-purpose model.

This selection enables the investigation of how model scale and specialisation affect security properties and output safety.

Both selected models were deployed and interacted with locally using Ollama (version 0.12.3), a widely adopted framework for running LLMs. This ensured a controlled and consistent inference environment across all experiments. All prompts were submitted to, and responses collected from, the models via the Ollama API, simulating a typical developer workflow for integrating open-weight LLMs into applications.

Threat Modelling and Test Corpus

We constructed a stratified prompt corpus (N=500) designed to probe different aspects of IOH vulnerability:

- Tier 1: Innocuous Tasks (25%) – Baseline HTML generation tasks
- Tier 2: Security-Ambiguous Tasks (25%) – Requests implying interactivity
- Tier 3: Explicitly Risky Tasks (50%) – Direct requests for executable code

This distribution reflects real-world usage patterns while enabling granular analysis of risk factors.

Multi-Layer Defence Evaluation

Model output underwent systematic testing across a defence-in-depth matrix to evaluate protection effectiveness. The tested defence layers included:

- L1: Server-side Sanitization
  - Implementation: Bleach (allow-list based HTML sanitiser)
  - Scope: Universal
  - Purpose: Primary server-side filtering of malicious HTML patterns
- L2: Client-side Sanitization
  - Implementation: DOMPurify + Content Security Policy (CSP)
  - Scope: Universal
  - Purpose: Secondary client-side sanitisation and policy enforcement

This layered approach allowed for the assessment of individual and combined defence effectiveness against LLM-generated XSS vectors.

Analytical Metrics Framework

Primary Risk Assessment

- Baseline Exploit Rate (BER):

$$\text{BER} = \frac{\text{Number of Successfully Executed Outputs}}{\text{Total Outputs}} \times 100$$

Measures the inherent risk of a model generating exploitable outputs without any safeguards.

- Pattern Prevalence (PP):

$$\text{PP} = \frac{\text{Outputs with Detected XSS Patterns}}{\text{Total Outputs}} \times 100\%$$

Quantifies how often a model produces code that appears dangerous based on pattern matching.

- Exploitation Efficiency (EE):

$$\text{EE} = \frac{\text{Successfully Executed Outputs}}{\text{Outputs with Detected Patterns}} \times 100\%$$

Measures the quality and functional effectiveness of the detected attack vectors.

Defence Efficacy Metrics

- Exploit Neutralisation Rate (ENR):

$$\text{ENR} = \frac{\text{Exploits Blocked}}{\text{Total Exploits}} \times 100\%$$

The percentage of successfully executed attacks prevented by a defence mechanism.

- Pattern Reduction Rate (PRR):

$$\text{PRR} = \frac{\text{Patterns Removed}}{\text{Total Patterns}} \times 100\%$$

Measures the effectiveness of sanitisation in removing potentially dangerous code patterns.

- Cross-Model Defence Consistency: Evaluates whether a defence mechanism performs equally well across different model architectures.

Experimental Protocol

The sequential testing protocol ensured controlled comparisons:

- Baseline Characterisation: Establish inherent risk profiles for each model,
- Defence Isolation Testing: Measure individual layer effectiveness,
- Combined Defence Analysis: Assess synergistic effects of multi-layer protection,
- Robustness Validation: Test edge cases and adversarial variations.

This comprehensive framework enables not only the quantification of vulnerability but also a deeper investigation into the relationship between model architecture, training methodology, and security properties in generative AI systems.

Experimental Artefacts and Reproducibility

To ensure full transparency and reproducibility of our findings, the complete experimental pipeline, including prompt templates, sanitisation implementations, evaluation scripts, and raw results, is publicly available under an open-source license. The repository can be accessed at: GitHub Repository: https://github.com/AAnchimiuk/IOH-xss-experiment. This includes:

- Python source code of experimental pipeline,
- Raw results for pipeline execution for both models,
- Raw files of computed metrics for both models.

Results

Comparative Model Vulnerability Analysis

The experimental results reveal significant differences in security posture between the two evaluated models. DeepSeek-R1 exhibited a Baseline Exploit Rate (BER) of 4.2% (21/500), whereas Llama 3 8B demonstrated a substantially higher BER of 27.6% (138/500). This 6.6-fold difference in inherent vulnerability highlights the profound impact of model architecture and training methodology on security properties.

Pattern Detection vs. Actual Exploitation

A critical finding emerges from the disparity between pattern detection and successful exploitation, quantified by the Exploitation Efficiency (EE) metric:

- DeepSeek-R1 showed extensive Pattern Prevalence (PP) of 87.8% but a very low EE of 4.8%. This indicates the model generates a high volume of low-quality, non-functional, or easily mitigated attack vectors.
- Llama 3 demonstrated a lower PP (74.2%) but a significantly higher EE of 37.2%, indicating it produces fewer but more potent and readily exploitable vulnerabilities.

The scale of this “false positive” problem in static analysis is stark: 95.2% of DeepSeek’s and 62.8% of Llama 3’s flagged patterns were non-exploitable,

revealing a critical limitation of relying solely on pattern detection for security assessment.

Vector Distribution and Attack Patterns

Analysis of attack vectors reveals distinct model-specific tendencies:
DeepSeek-R1 Vector Profile:

- Primary Mechanism: High-volume, low sophistication,
- Script Tags: 924 instances (56.9% of total vectors),
- Event Handlers: 560 on_attr patterns,
- Advanced Vectors: Minimal (7 iframe srcdoc, two JavaScript protocols).

Llama 3 Vector Profile:

- Primary Mechanism: Balanced, high sophistication,
- Script Tags: 287 instances (41.6%),
- Advanced Vectors: Significant numbers of iframe srcdoc (28), meta refresh (27), and javascript: protocols (20),
- Attack Quality: Demonstrated higher capability for multi-vector attacks.

Defence Mechanism Efficacy

The implemented sanitisation defences demonstrated exceptional performance:

- Exploit Neutralisation Rate (ENR): both server-side (Bleach) and client-side (DOMPurify + CSP) sanitisation achieved 100% ENR, eliminating all 159 successful exploits,
- Pattern Reduction Rate (PRR): sanitisation reduced XSS patterns by 76.1% for DeepSeek-R1 and 88.4% for Llama 3,
- Functional Integrity: benign content maintained full functionality post-sanitisation,
- Cross-Model Defence Consistency: both defence layers provided identical 100% protection regardless of model architecture or vulnerability profile.

Risk Assessment Summary

Table 1. Summary of results from experimental metrics

Metric	DeepSeek-R1:1.5B	Llama 3:8B
Baseline Exploit Rate (BER)	4.2%	27.6%
Pattern Prevalence (PP)	87.8%	74.2%
Exploitation Efficiency (EE)	4.8%	37.2%
Residual Risk (Post-Defence)	0%	0%
Primary Risk Characteristic	Volume of low-quality vectors	Quality of exploitable vectors

Source: own study.

The results demonstrate that while Llama 3 generates more immediately dangerous outputs, both models produce vulnerabilities that necessitate comprehensive output sanitisation. The 100% effectiveness of standard web security measures against LLM-generated XSS provides compelling evidence for integrating these established practices into AI-augmented systems (Table 1).

Discussion

Technical Perspective

This study demonstrates that Insecure Output Handling (IOH) bridges two domains that have traditionally been treated in isolation: adversarial risks in Large Language Models (LLMs) and well-known web and software vulnerabilities. Our empirical investigation focused specifically on Cross-Site Scripting (XSS), providing systematic, quantitative evidence that even benign prompts, without adversarial manipulation, can yield exploitable HTML/JavaScript outputs when passed directly to downstream applications.

The XSS scenario reflects long-standing security concerns: generated HTML and Markdown outputs often contain inline JavaScript or unsafe attributes, reproducing vectors that have been extensively documented in XSS research (Gupta et al., 2017). Our experiments confirm that "off-the-shelf" LLMs do not incorporate security considerations into their generative logic by default (Sajadi et al., 2025; He et al., 2023). A significant contribution of this work is the quantification of the Exploit Success Rate (ESR) for XSS under both baseline

and secured configurations. This empirical evidence shows that lightweight safeguards – such as HTML sanitisation and restrictive Content Security Policies – can significantly reduce or even eliminate exploitability.

While our experimental validation centred on XSS, the IOH paradigm extends to other critical vulnerabilities. The SQL injection (SQLi) scenario, drawn from prior literature (Alwan, Younis, 2017; Hong et al., 2024), illustrates this clearly. LLMs frequently generate queries via unsafe string concatenation, a practice identified as insecure in both classical surveys (Alwan et al., 2017) and more recent evaluations of NL-to-SQL systems (Hong et al., 2024). Similarly, threats like Remote Code Execution (RCE) or Server-Side Request Forgery (SSRF) represent high-risk IOH vectors where generated code or suggested URLs could compromise systems if executed without validation.

However, the technical perspective also highlights significant limitations and nuances revealed by our data.

First, mitigations are highly context dependent. Our experiments confirmed the total effectiveness of HTML sanitisation against XSS, but this defence is specific to that vector. A Content Security Policy (CSP) may block inline JavaScript but not malicious data URLs; parameterisation eliminates SQLi but does nothing against insecure shell script generation. Therefore, the choice of output sanitisation must be precisely tailored to the integration context and the potential attack vectors it enables.

Second, our results underscore that vulnerability detection and successful exploitation are distinct problems. We observed a high volume of detected XSS patterns (e.g. 87.8% for DeepSeek-R1) but a much lower rate of successful execution (4.2% for the same model). This "pattern-to-exploit gap" means that static analysis alone is an insufficient metric for risk assessment, as it generates numerous false positives and fails to account for the functional quality and context of the attack vector.

Finally, the probabilistic nature of LLMs means that a model's security posture cannot be guaranteed by testing a finite set of prompts. A seemingly safe model, or one that generates mostly low-quality, non-functional vectors, could still produce a high-quality, exploitable output in a previously untested scenario. This inherent unpredictability, coupled with the high stakes of a single successful

exploit, necessitates continuous runtime monitoring and defence-in-depth rather than relying on one-off audits or the assumption of general model ‹safety›.

Regulatory Perspective

From a regulatory standpoint, IOH raises direct concerns under the European Union's Artificial Intelligence Act (European Union, 2024 (AI Act)). Although IOH is not named explicitly, the Act requires high-risk AI systems to comply with several obligations that IOH directly undermines. Our empirically demonstrated XSS risks provide a concrete example of how IOH can violate these provisions:

- risk management (Art. 9): our ESR-based evaluation for XSS offers a structured way to quantify IOH risk and integrate it into risk management pipelines. Similar to penetration testing in classical software security, IOH testing can become a verifiable compliance artefact,
- robustness and accuracy (Art. 15): systems that propagate XSS through generated outputs cannot be considered robust. By contrast, applying sanitisation fulfils the robustness expectation, as demonstrated by our results,
- transparency and documentation (Art. 13): maintaining records of how outputs are sanitised, validated, and monitored supports compliance with documentation requirements. Such logs can also serve as evidence during regulatory audits,
- human oversight (Art. 14): oversight cannot be limited to inputs; it must also extend to outputs. Unsafe outputs must be either blocked automatically or flagged for human review, aligning with the principle of "meaningful human control".

These obligations resonate with guidance from the OWASP GenAI Security Project (2025) (OWASP GenAI Security Project, 2025), which explicitly identifies Improper Output Handling as a top-tier risk for generative AI. IOH thus becomes not just a best-practice issue but a compliance necessity: failure to mitigate it could result in enforcement action, liability for damages, or loss of certification to deploy high-risk AI systems in the EU.

Broader Implications

The implications of IOH extend beyond immediate technical controls and regulatory compliance, encompassing questions of trust, liability, and systemic risk.

First, the sectoral impact is profound. In healthcare, insecure outputs embedded in electronic health records or clinical decision support tools could compromise patient safety and violate GDPR. In finance, LLM-generated code or queries vulnerable to injection attacks could distort fraud detection pipelines or risk assessments, raising compliance concerns under MiFID II or Basel III. In critical infrastructure, IOH could surface in CI/CD pipelines, where generated configuration scripts introduce vulnerabilities into industrial control systems.

Second, IOH introduces significant liability and governance challenges. Under the AI Act, non-compliance may lead to substantial fines (European Union, 2024 (AI Act)). But even outside formal regulation, organisations risk civil liability if insecure outputs lead to breaches. For instance, if an LLM-generated SQL injection (as identified in prior works (Alwan et al., 2017; Hong et al., 2024)) compromises a hospital's patient database, legal disputes could emerge over whether responsibility lies with the AI provider, the integrator, or the deploying institution.

Finally, IOH points toward an evolving security paradigm. Historically, input validation became a non-negotiable baseline in secure system design. Our results suggest that output validation may soon be regarded as equally essential for AI-enabled systems. Extending this into practice could mean that future development pipelines treat every LLM response as "untrusted input," subject to the same rigour of sanitisation and validation as external user data.

Conclusion

This study introduces and empirically grounds the concept of Insecure Output Handling (IOH) as a critical class of risks in LLM-based systems. Through a controlled experimental framework focused on Cross-Site Scripting (XSS), this work provides quantitative evidence that:

- LLMs inherently generate insecure outputs, with the nature and severity of the risk being highly model-dependent. Our results showed a dramatic 6.6-fold difference in the baseline exploit rate between the two tested models (DeepSeek-R1 at 4.2% vs Llama 3 at 27.6%), underscoring that architectural and training choices directly impact security posture.
- IOH poses immediate and practical security threats when outputs are consumed without validation. The successful execution of generated XSS vectors in a simulated production environment confirms that classical web vulnerabilities are readily reintroduced through LLM outputs.
- Conventional, lightweight defences are highly effective and essential. Our central finding is that standard security measures – specifically HTML sanitisation and Content Security Policies (CSPs) – achieved 100% effectiveness in neutralising all successful XSS exploits across both models. This provides a strong, evidence-based argument for integrating these established practices into AI-integrated systems.
- The mitigation of IOH is not merely a technical best practice but a cornerstone of regulatory compliance. Our empirical framework for quantifying IOH risk directly supports obligations under the EU AI Act concerning risk management, robustness, and documentation, transforming a technical vulnerability into a manageable compliance requirement.

Overall, this study positions IOH as a critical convergence point between AI safety research and established application security. By demonstrating that a probabilistic, generative component can be secured using deterministic, classical defences, we bridge a crucial gap in the engineering of trustworthy AI systems.

Future work must expand upon this experimental foundation. The immediate priority is to extend rigorous empirical testing to other critical IOH vectors, such as SQL Injection, remote code execution, and server-side request forgery, to build a comprehensive, data-driven understanding of the IOH threat landscape. Furthermore, proactive approaches – such as fine-tuning models on secure coding corpora, developing standardised output validation libraries, and exploring runtime monitoring solutions – present vital pathways for building more secure and resilient generative AI applications.

References

1. Alwan, Z. S., Younis, M. F. (2017). Detection and prevention of SQL injection attack: a survey. *International Journal of Computer Science and Mobile Computing*, *6*(8), 5-17.
2. Gupta, S., Gupta, B. B. (2017). Cross-site scripting (XSS) attacks and defence mechanisms: classification and state of the art. *International Journal of System Assurance Engineering and Management*, *8*, 512-530. https://doi.org/10.1007/s13198-015-0376-0
3. He, J., Vechev, M. (2023). Large Language Models for Code: Security Hardening and Adversarial Testing. Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), 1865-1879. https://doi.org/10.1145/3576915.3623175
4. Hong, Z., Yuan, Z., Chen, H., Zhang, Q., Huang, F., Huang, X. (2024). Knowledge-to-SQL: Enhancing SQL Generation with Data Expert LLM. Findings of the Association for Computational Linguistics: ACL 2024, 10997-11008. https://doi.org/10.18653/v1/2024.findings-acl.653
5. Luo, Y., Lin, Z., Zhang, Y., Sun, J., Lin, C., Xu, C., Su, X., Shen, Y., Guo, J., Gong, Y. (2024). Ensuring Safe and High-Quality Outputs: A Guideline Library Approach for Language Models. *Proceedings of NAACL-HLT 2024,* 1152-1197. https://doi.org/10.18653/v1/2024.naacl-long.65
6. Sajadi, A., Le, B., Nguyen, A., Damevski, K., Chatterjee, P. (2025). Do LLMs consider security? An empirical study on responses to programming questions. *Empirical Software Engineering*. https://doi.org/10.1007/s10664-025-10658-6
7. Siddiq, M. L., Santos, J. C. (2023). Generate and Pray: Using SALLMs to Evaluate the Security of LLM Generated Code. *IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. https://doi.org/10.1145/3691621.3694934

Netography

1. DeepSeek. (2025). *DeepSeek-R1: A Series of Efficient Reasoning Models*. Retrieved from: https://github.com/deepseek-ai/DeepSeek-R1 (access: 06.11.2025).
2. European Union (2024). *Artificial Intelligence Act*. *Official Journal of the European Union*.

3. Meta AI. (2024). *The Llama 3 Herd of Models*. Retrieved from: https://github.com/meta-llama/llama3 (access: 06.11.2025).
4. OWASP GenAI Security Project. (2025). *LLM052025 – Improper Output Handling*. Retrieved from: https://genai.owasp.org/llmrisk/llm052025-improper-output-handling/ (access: 06.11.2025).