Pre-Proposal: AutoSE-Guard

1. Introduction

In the rapidly evolving landscape of software development, the imperative for building secure systems from inception has never been more critical. As software permeates every facet of modern life, from critical infrastructure to personal devices, the consequences of security vulnerabilities can range from significant financial losses and reputational damage to severe safety and privacy breaches. Despite this, security is often treated as an afterthought, primarily addressed during later stages of the Software Development Lifecycle (SDLC), such as coding and testing. This reactive approach, commonly referred to as 'shift-right' security, leads to exponentially higher costs for remediation, increased project delays, and a persistent cycle of vulnerability management rather than proactive prevention.

The root of many critical security flaws can often be traced back to the earliest phases of software development, particularly the requirements engineering stage. Ambiguous, incomplete, or missing security requirements lay a weak foundation upon which the entire system is built, allowing vulnerabilities to propagate silently through design, implementation, and deployment. Current security tools, predominantly focused on code-level analysis (e.g., SAST, DAST), are ill-equipped to identify these fundamental issues at their origin. Manual processes like threat modeling, while valuable, are time-consuming, resource-intensive, and struggle to scale across complex, agile projects. This creates a significant gap in the ability of software engineering teams to effectively identify and mitigate security risks at the most impactful and cost-efficient point in the development process.

This pre-proposal introduces **AutoSE-Guard**, an innovative, AI-driven framework designed to revolutionize security risk assessment by focusing specifically on the requirements engineering phase. Leveraging advanced Natural Language Processing (NLP) and machine learning techniques, AutoSE-Guard aims to automatically analyze requirements documents, identify potential security risks, classify requirements based on their clarity and completeness regarding security, and provide actionable insights. By shifting security left to the very beginning of the SDLC, AutoSE-Guard seeks to enable the development of inherently more secure software, reduce development costs, accelerate time-to-market, and foster a truly secure-by-design development paradigm.

2. Problem Statement

Despite the critical importance of security in modern software, current practices and tools largely fail to address security risks effectively at the earliest and most impactful stage:

requirements engineering. The prevailing paradigm of security assessment is heavily skewed towards post-implementation analysis, primarily through code-level scanning (SAST, DAST) and penetration testing. While these methods are essential, they are inherently reactive, identifying vulnerabilities long after they have been embedded into the system's design and code. This late detection leads to:

1. **Exorbitant Remediation Costs:** The cost of fixing a security flaw escalates dramatically with each subsequent SDLC phase. A vulnerability originating in requirements, if discovered during testing or, worse, in production, can be hundreds or thousands of times more expensive to rectify than if caught at its inception.
2. **Propagation of Flaws:** Ambiguous, incomplete, or missing security requirements at the outset can propagate as fundamental design flaws throughout the entire system architecture and implementation. These foundational weaknesses are difficult to patch and can lead to systemic vulnerabilities that compromise the entire application.
3. **Lack of Automation for Early Stages:** Unlike code analysis, there is a significant dearth of automated, intelligent tools specifically designed to identify and assess security risks directly from natural language requirements documents. Existing early-stage security activities, such as threat modeling, are predominantly manual, time-consuming, and heavily reliant on expert knowledge, making them unscalable for the rapid pace of modern agile and DevOps environments.
4. **Inadequate Contextual Understanding:** Code-centric tools lack the broader context of business logic, user roles, and system interactions that are defined in requirements. They cannot effectively identify security risks arising from insecure design choices or misinterpretations of security needs at a conceptual level.

Consequently, software engineering teams lack a proactive, automated mechanism to ensure that security is built into the software from its very foundation. This gap results in a reactive security posture, increased development costs, delayed releases, and ultimately, less secure software products. There is a clear and urgent need for an adaptive framework that can automatically assess and score security risks within requirements documents, enabling early intervention and fostering a truly secure-by-design development paradigm.

3. Related Work

The landscape of software security and requirements engineering has seen significant advancements, particularly with the integration of Artificial Intelligence (AI) and Natural Language Processing (NLP). However, a comprehensive review of recent literature (2020-2025) reveals a distinct gap in automated frameworks specifically designed for **security risk scoring within the requirements engineering phase**.

Existing research often falls into several categories:

- **AI/NLP for General Requirements Analysis:** Studies explore the use of AI and NLP to improve the quality of requirements, identify ambiguities, or automate categorization. For instance, Batool et al. (2025) and Ahmad et al. (2025) investigate NLP and ML for automated categorization and fuzzy engineering of security requirements. Similarly, Budake et al. (2023) and Sadovykh et al. (2023) discuss AI techniques for general requirements analysis and security requirements analysis, respectively. Cheng et al. (2024) provide a systematic review of generative AI in requirements engineering. While these works demonstrate the potential of AI in understanding requirements, they do not propose integrated frameworks for *security risk scoring* based on these analyses.
- **Security Risk Assessment Frameworks (Broader Scope):** Many frameworks exist for security risk assessment, but they typically operate at a higher level (e.g., organizational, system-wide) or focus on later SDLC stages. Examples include frameworks for critical infrastructure security (Ali et al., 2024) or general cybersecurity frameworks (NIST, ISO). While these are crucial for overall security posture, they lack the granularity and automation needed for real-time risk assessment directly from requirements documents.
- **AI in Cybersecurity (Post-Requirements):** A substantial body of work focuses on AI applications in cybersecurity, such as threat detection, vulnerability management, and security testing, but predominantly at the code or operational level. Mamidi (2024) discusses future trends in AI-driven cybersecurity, and Jacob (2025) analyzes AI-driven security controls in platform engineering. AI-driven security testing (Thinksys Inc., 2025) and compliance automation (Comet, 2023; Akitra, 2024) are also active areas. These efforts are vital but occur *after* the requirements have been finalized and implemented.
- **Automated Security Policy/Requirement Extraction:** Some research has explored automating the extraction of security policies or requirements from natural language. While foundational (e.g., Tao Xie, 2012), more recent works like those on automated categorization (Batool et al., 2025) and security requirements classification using NLP transformers (IEEE, 2022) contribute to this area. However, these often focus on identification or classification rather than a comprehensive risk scoring framework.
- **AI in Project Management/SDLC (General Risk):** AI-driven decision support systems are used in agile project management to enhance risk mitigation and resource allocation (Almalki, 2025; Mohapatra et al., 2025). While valuable for project-level risks, they do not delve into the technical or security risks inherent in the requirements content itself.

The Gap:

Crucially, the literature lacks an integrated, automated framework that combines advanced NLP techniques to analyze natural language requirements documents for security risks, classifies these risks (e.g., unclear, incomplete, conflict), and generates an adaptive, real-

time security risk score specifically for the requirements phase. Existing solutions either focus on general requirements analysis without a security-specific risk scoring component, operate at later SDLC stages, or rely on manual processes for early-stage security assessment. AutoSE-Guard aims to fill this critical void by providing a proactive, automated, and intelligent solution for identifying and mitigating security risks at the earliest and most impactful point in the software development lifecycle.

Research Questions

Based on the identified problem and gap, this research will address the following questions:

RQ1: How effectively can an AI-driven framework, leveraging Natural Language Processing, automatically identify, classify (unclear, incomplete, conflict, clear), and score security risks within natural language software requirements documents?

RQ2: To what extent can the early detection and mitigation of security risks, facilitated by the proposed AutoSE-Guard framework, lead to a measurable reduction in the cost and effort of security vulnerability remediation in subsequent Software Development Lifecycle phases?

References

1. Batool, R., Naseer, A., Maqbool, A., & Kayani, M. (2025). Automated categorization of software security requirements: an NLP and ML based approach. *Requirements Engineering*.
2. Ahmad, S., Arif, M., Ansari, M., & Nazim, M. (2025). AI-driven fuzzy requirement engineering for cybersecurity: integrating linguistic decision models. *Iran Journal of Computer Science*.
3. Ehrlich, M., Lukas, G., & Trsek, H. (2024). Requirements Analysis for the Evaluation of Automated Security Risk Assessments. *2024 IEEE 20th International Conference on Software Engineering and Formal Methods (SEFM)*.
4. Ali, S. M., Razzaque, A., Yousaf, M., & Shan, R. U. (2024). An automated compliance framework for critical infrastructure security through Artificial Intelligence. *IEEE Access*.
5. Cheng, H., Husen, J. H., Lu, Y., & Racharak, T. (2024). Generative AI for requirements engineering: A systematic literature review. *arXiv preprint arXiv:2409.06741*.
6. Mamidi, S. R. (2024). Future Trends in AI Driven Cyber Security. *IRE Journal*.
7. Jacob, J. (2025). Qualitative analysis of security-aware platform engineering: Integrating AI-driven security controls in surveillance device lifecycle management. *World Journal of Advanced Research and Reviews*.

8. Sadovykh, A., Yakovlev, K., & Naumchev, A. (2023). Natural language processing with machine learning for security requirements analysis: practical approaches. In *Software Engineering: From Requirements to Deployment* (pp. 23-45). Springer.
9. Nature. (2025). AI-driven cybersecurity framework for software development based on artificial neural network and interpretive structural modeling. *Nature Scientific Reports*.
10. ScienceDirect. (2025). Beyond domain dependency in security requirements identification. *Information and Software Technology*.
11. King Fahd University of Petroleum and Minerals (KFUPM). (202X). *Automated Analysis of Security Requirements Using Machine Learning and Natural Language Processing*. (Research Project).
12. IREB. (2024). AI Assistants in Requirements Engineering | Part 1. *RE-Magazine*.
13. NetImpact Strategies. (2025). AI-Powered Requirements Engineering for Federal IT Modernization. *NetImpact Strategies Insights*.
14. Comet. (2023). NLP techniques used for compliance checks. *Comet Blog*.
15. DLABI. (2024). The Role of Natural Language Processing in Automating Cybersecurity Audit. *Journal of Cybersecurity Technology and Applied Management*.
16. Akitra Blog. (2024). The Role of Natural Language Processing in Compliance Document Management. *Medium*.
17. VerityAI. (2025). AI Security Vulnerabilities in NLP Systems: Language Model Security. *VerityAI Blog*.
18. Hansch, G. (2020). *Automating security risk and requirements management for cyber-physical systems*. University of Goettingen Dissertation.
19. Siddique, I. (2023). Emerging trends in requirements engineering: A focus on automation and integration. *European Journal of Advances in Engineering and Technology*.
20. IGI Global. (202X). AI-Driven Threat Modeling: Enhancing Risk Assessment in Software Projects. *IGI Global Chapter*.
21. Springer. (202X). Artificial intelligence for system security assurance: A systematic literature review. *Journal of Computer Virology and Hacking Techniques*.
22. Xie, T. (2012). Automated Extraction of Security Policies from Natural-Language Software Documents. *Proceedings of the 20th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*.
23. IEEE. (2022). Security requirements classification into groups using nlp transformers. *IEEE Xplore*.

24. Almalki, S. S. (2025). AI-Driven Decision Support Systems in Agile Software Project Management: Enhancing Risk Mitigation and Resource Allocation. *Systems*, 13(3), 208.
25. Mohapatra, H., Pramanik, S., & Mishra, S. R. (2025). Revolutionizing Software Development: The Transformative Influence of Machine Learning Integrated SDLC Model. In *Revolutionizing Software Development through Intelligent Systems* (pp. 3-14). Cham: Springer.

4. Methodology

This research proposes a novel methodology for automated security risk assessment at the requirements stage, leveraging Natural Language Processing (NLP) and machine learning. The AutoSE-Guard framework will operate through the following key steps:

4.1. Requirements Document Collection

The initial step involves collecting software requirements documents. These documents, typically in natural language, can include user stories, use cases, functional specifications, non-functional requirements, and other textual artifacts that define the system's intended behavior and constraints. The framework will be designed to integrate with common requirements management platforms (e.g., Jira, Confluence) or accept direct document uploads.

4.2. Natural Language Processing (NLP) and Feature Extraction

Once collected, the requirements documents will undergo extensive NLP processing. This involves:

- **Text Preprocessing:** Cleaning the text (e.g., tokenization, stop-word removal, stemming/lemmatization) to prepare it for analysis.
- **Security Keyword and Pattern Identification:** Utilizing predefined lexicons of security-related terms (e.g., authentication, authorization, encryption, data privacy, access control, vulnerability) and patterns (e.g., phrases indicating missing security controls, ambiguous statements) to identify potential security concerns within individual requirements.
- **Feature Engineering:** Extracting relevant features from the processed text that are indicative of security risks or their absence. This may include n-grams, part-of-speech tags, dependency parsing, and embedding vectors (e.g., Word2Vec, BERT embeddings) to capture semantic meaning.

4.3. Requirements Classification and Risk Scoring

The extracted features will then feed into a machine learning model for classification and risk scoring. The model will classify each requirement based on its security posture into one of the following categories:

- **Unclear:** Requirements that are vague, ambiguous, or lack sufficient detail regarding security aspects, making their implementation or verification difficult.
- **Incomplete:** Requirements that omit critical security considerations or fail to cover all necessary security controls for a given functionality or data type.
- **Conflict:** Requirements that contradict other security requirements or established security policies, potentially leading to vulnerabilities or system instability.
- **Clear:** Requirements that are well-defined, complete, consistent, and explicitly address security considerations.

For requirements identified as unclear, incomplete, or conflicting, the framework will provide specific feedback highlighting the problematic areas. For **clear requirements**, the system will then proceed to suggest relevant security controls or best practices, potentially referencing standards like **ISO/IEC 27001 (Information Security Management)**, **ISO/IEC 27002 (Code of Practice for Information Security Controls)**, or **OWASP Top 10** guidelines. This suggestion will aim to further strengthen the security posture of well-defined requirements by aligning them with recognized industry standards.

Simultaneously, an **AI-based adaptive risk scoring mechanism** will assign a quantitative risk score to each requirement and, aggregated, to the entire set of requirements. This score will be based on the classification results, the severity of identified security issues, the criticality of the requirement, and potentially historical project data. The model will be designed to learn and adapt over time, improving its accuracy with more data.

4.4. Reporting and Visualization

The final stage involves generating comprehensive reports and visualizations. These will include:

- **Detailed Risk Reports:** Listing identified security risks, their classification, assigned risk scores, and specific problematic phrases or sections within the requirements.
- **Actionable Recommendations:** Providing concrete suggestions for refining unclear, incomplete, or conflicting requirements, and proposing ISO-aligned controls for clear requirements.
- **Risk Trend Visualization:** Displaying the overall security risk level of the requirements set over time, allowing teams to track progress and identify trends.

This methodology ensures a proactive, automated, and intelligent approach to integrating security directly into the requirements engineering process, significantly enhancing the

overall security posture of software projects from their inception.

5. AutoSE-Guard Framework Diagram

The following diagram illustrates the key steps and flow of the AutoSE-Guard framework for security risk assessment at the requirements stage: