
def/acc Hackathon

Patchwork Report¹

Zaynab Batool Reza
Saarland University
AI Safety Saarland

Najia Gul
Saarland University
AI Safety Saarland

Rafay Rashed
RPTU
AI Safety Saarland

With
Apart Research

Abstract

AI-enabled cyber threats are scaling faster than traditional security tools can adapt. Modern developer workflows with distributed teams, high code velocity, growing dependency chains, and widespread use of AI coding assistants, have dramatically expanded the attack surface. Patchwork is an automated defensive system designed to bridge this growing gap by bringing enterprise-grade, AI-powered security scanning and patching directly into developer and consumer environments.

Our system performs multi-domain security analysis on GitHub repositories using a suite of AI-driven scanners for vulnerabilities such as secret leakage, SQL injection, cryptographic misuse, access control weaknesses, LLM prompt injection, insecure AI API usage, and more. Unlike static analyzers, Patchwork generates context-aware patches, deduplicates redundant findings, constructs red-team attack scenarios, and automatically creates GitHub pull requests with fixes. The tool integrates a fully modern web interface and a FastAPI backend, enabling real-time scanning and visualized results.

Our results show that large language models can meaningfully improve developer-side defensive security by catching complex vulnerabilities and also by highlighting emerging AI-enabled risks. We demonstrate that automated patch generation is feasible,

¹ Research conducted at the Apart Research Hackathon, 2025

reduces remediation time, and enables scalable, user-accessible defensive acceleration.

Keywords: AI security, defensive AI, automated patching, code vulnerability detection, LLM threat analysis, red-teaming

1. Introduction

Agentic-AI systems have substantially increased the speed and scale at which cyberattacks can be orchestrated. The rise of autonomous agents has shifted the threat landscape: instead of slow, manual exploitation, modern AI systems can generate, mutate, and deploy attacks at machine speed. This has enabled large-scale attacks on consumer devices, developer environments, and open-source ecosystems. With the 2025 revelations by Anthropic demonstrating multi-agent, multi-step autonomous exploitation chains, the need for consumer-level AI-enabled defensive tools has never been higher.

However, consumer and developer environments present difficult constraints. Existing security tools are largely rule-based, static, or signature-driven—effective for known patterns but insufficient for AI-enabled threats or complex, cross-file vulnerabilities. Even when tools identify issues, **the real pain point is remediation**: applying patches manually is slow, error-prone, and often beyond the user’s expertise. Current scanners stop at detection; they do not provide context-aware fixes, generate patches, or streamline integration into the codebase.

Our project aims to democratize cybersecurity by providing everyday developers and consumers with a lightweight defensive system that not only detects vulnerabilities but also **suggests patches, generates ready-to-review pull requests, and creates red-team attack scenarios**. Patchwork integrates traditional static analysis with LLM-based reasoning, producing an adaptive, multi-layer system inspired by Wazuh-like principles but optimized for consumer hardware and modern AI attack surfaces.

We explore the following research question:

Can AI systems autonomously detect, reason about, and patch multi-domain vulnerabilities in software repositories, including AI-enabled attack surfaces, at a level useful for real-world defensive acceleration?

We hypothesize that:

1. LLMs can detect a broader class of vulnerabilities than traditional static analyzers, especially contextual ones.
2. Automated patch generation via LLMs is feasible when grounded in the project’s codebase.

3. Combining scanning + patching + red-teaming in one pipeline substantially improves defensive tooling for developers.

Our work contributes a practical, reproducible, high-utility defensive system that targets the asymmetry between rapidly advancing AI-enabled attacks and slow manual remediation.

2. Methods

We developed **Patchwork**, a two-part system:

1. **A FastAPI backend orchestrating scanning workflows**, LLM calls, patch generation, deduplication, and GitHub integration.
2. **A Next.js frontend** providing a fully interactive UI with real-time scanning updates, patch previews, and red-team visualizations.

Security Scanning Architecture

We built seven parallelized scanners:

- Secret scanning
- SQL injection detection
- Authentication analysis
- Cryptographic misuse detection
- URL handling & SSRF
- Access control analysis
- General vulnerabilities (XSS, CSRF, command injection, deserialization, etc.)
- AI security: prompt injection, insecure AI API usage, model misuse, adversarial input exposure

Each scanner uses OpenAI GPT-based reasoning on code snippets and produces structured findings.

Patch Generation

After deduplication, each unique finding is sent through a patch-generation module:

- Generates unified diff patch
- Ensures minimal, context-aware changes
- Validates patch using unidiff

Red-Team Scenario Generation

For every detected vulnerability, the system:

- Generates exploit payloads
- Describes attacker objectives
- Outlines impact
- Suggests mitigations

Project Context Analysis

We built an LLM-based classifier that labels the repository domain (e.g., ecommerce, streaming, fintech) to provide domain-specific recommendations. This is first done at a file level, and summaries of each are combined into a unified project-level context block.

Frontend

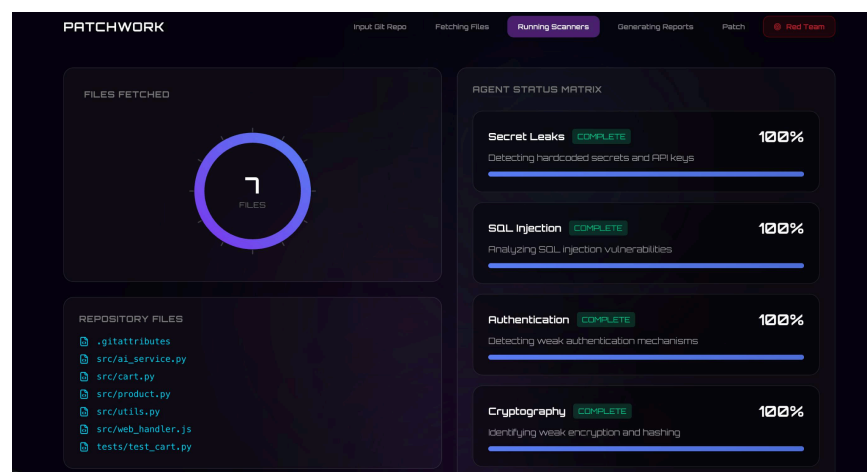
A modern glassmorphism UI with:

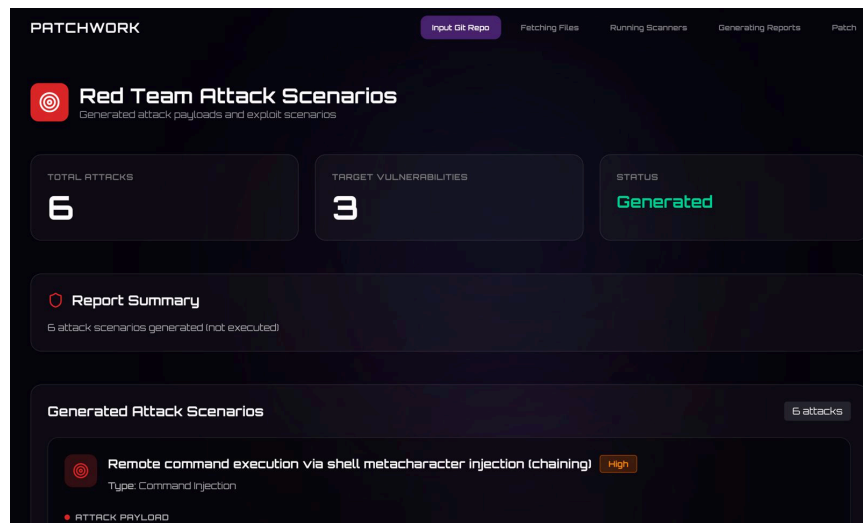
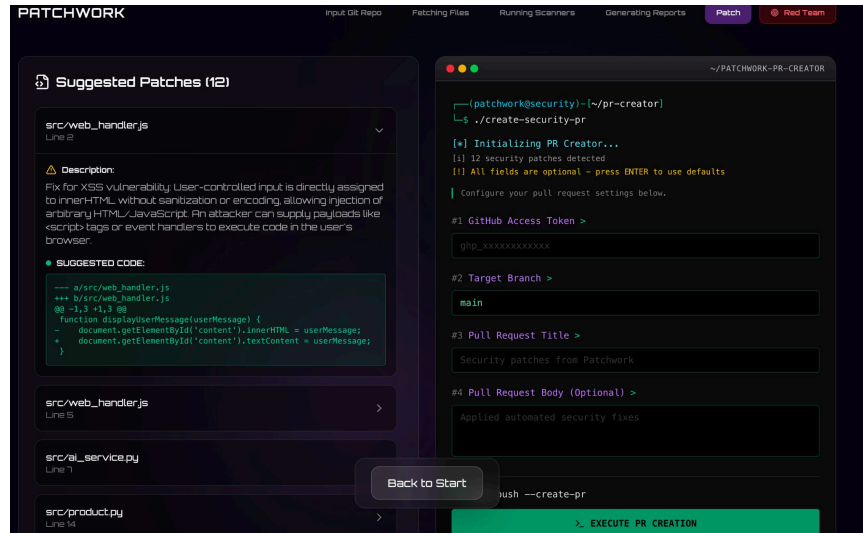
- Live progress tracking
- Interactive vulnerability cards
- Terminal-style PR creation
- PDF export
- Red-team visualization

All components are documented, reproducible, and available on GitHub.

3. Results

The screenshots below demonstrate the system's ability to detect vulnerabilities across multiple files, generate context-aware patch suggestions, and produce red-team attack scenarios based on discovered weaknesses. These outputs collectively illustrate the pipeline's end-to-end functionality—from scanning to patching to simulated attack generation.





4. Discussion and Conclusion

Patchwork demonstrates that LLM-based security scanning and patching is both technically feasible and practically valuable for defensive acceleration. It bridges the gap between static analysis and real-world AI-enabled attack surfaces by integrating traditional vulnerability patterns with modern AI threats.

Key conclusions:

- LLMs can effectively identify non-trivial vulnerabilities requiring contextual understanding.
- Automated patch generation significantly reduces developer remediation time.
- Red-team scenario generation provides an educational and defensive layer missing in traditional tools.
- The system meaningfully addresses the offense-defense asymmetry by enabling fast, scalable defensive actions.

This project strengthens consumer-level defensive security and contributes a practical step toward accessible, AI-enabled cybersecurity.

5. References

“Overview,” Langchain. [Online]. Available: <https://docs.langchain.com/oss/python/langchain/overview> (accessed Nov. 23, 2025).

W. Le, D. G. Kim, K. E. Smith, and M. P. G. LeMay, “GenProg: A generic method for automatic software repair,” presented at the 18th ACM SIGSOFT Int. Symp. Foundations Softw. Eng., Szeged, Hungary, Nov. 2011. [Online]. Available: <https://roars.dev/pubs/le2011genprog.pdf>

J. Routh and H. C. Zhang, “The Edge of Agency: Defending Against the Risks of Agentic AI,” Akamai, Nov. 21, 2023. [Online]. Available: <https://www.akamai.com/blog/security/edge-of-agency-defending-against-risks-a-gentic-ai> (accessed Nov. 23, 2025).

Snyk, “Snyk and Continue Partner to Embed AI-Powered Security into Every Step of the Developer Workflow,” *Snyk Blog*, Nov. 18, 2025. [Online]. Available: <https://snyk.io/blog/snyk-continue-partner-integration/>

6. Appendix: Security Considerations

This system provides automated vulnerability detection, patch suggestion, and red-team attack generation, but several security limitations remain:

1. **LLM Misclassification Risk:** The model may miss deeply embedded or complex vulnerabilities, especially those requiring full program-flow understanding.
2. **False Positives:** Static text-based scanning can occasionally over-report issues without ground-truth runtime context.
3. **Limited Runtime Validation:** Generated red-team attacks are not yet executed in a sandbox, so exploitability cannot be verified.

Future Improvements

1. **Add sandbox execution** for running generated attacks and collecting runtime behavior and exploit success metrics.
2. **Introduce more specialized scanners** for more accurate detection.
3. **Integrate external threat feeds and CVE sources** to ensure up-to-date scanning of libraries and newly emerging vulnerabilities.
4. **Enable continuous monitoring of repositories** so that future commits are automatically analyzed for new vulnerabilities.