
TelicLens Report¹

Aditya Prasad
Groundless AI

Matt Farr
Groundless AI

Jayson Amati
Groundless AI

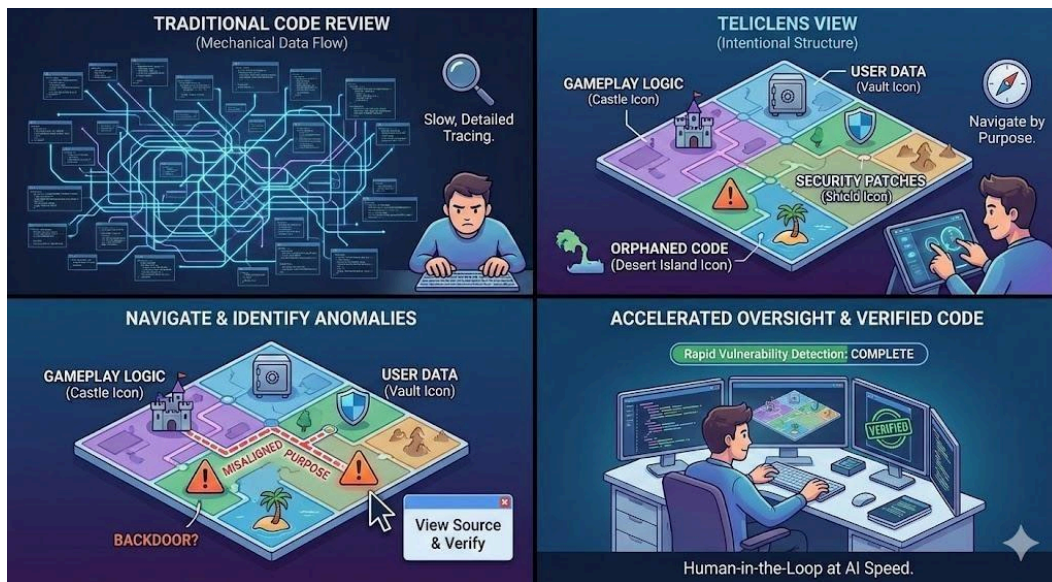
With
Apart Research

Abstract

*Critical infrastructure faces threats emerging faster than humans can manually review security patches. As AI-assisted code generation accelerates defensive responses, we risk deploying unverified code into production systems. **TelicLens** enables human oversight at AI development speeds by visualizing code's intentional structure—showing how components serve system purposes rather than just tracing data flow. Malware exfiltrating user data appears misaligned with stated goals like "create fun gameplay"; inefficient AI-generated code appears as orphaned functionality with no connection to system purposes. Our prototype demonstrates rapid identification of architectural vulnerabilities and redundant implementations. By keeping humans in the loop at AI speed, TelicLens provides essential infrastructure for defensive acceleration.*

Keywords: Critical infrastructure security, AI-assisted patching, human oversight, telic analysis, rapid vulnerability detection, defensive acceleration

¹ Research conducted during the Groundless Alignment Residency Program, Goa, India, 2025.



1. The Speed Problem

Critical infrastructure—power grids, telecommunications, financial systems—needs security patches deployed at the speed threats emerge. Manual line-by-line code review cannot keep pace, especially as AI is increasingly used to assist in patching and maintenance. However, deploying AI-generated patches without verification risks introducing backdoors, vulnerabilities, or incoherent code into systems we all depend on. Many parts of critical infrastructure also rely on open source codebases maintained by volunteers. The security of these codebases is often unknown.

TelicLens keeps humans meaningfully in the loop by making codebases more intuitive. Instead of tracing mechanical data flow (slow, detailed), it visualizes intentional structure—showing what code is *for*, not just what it *does*.

This includes:

- **Misaligned code:** Components whose purposes contradict system goals (e.g., "collect user data" in a "create fun gameplay" system)
- **Vestigial remnants:** Orphaned code with no connection to current purposes
- **Architectural weaknesses:** Single points of failure where critical functionality depends on one component
- **AI over-generation:** Redundant implementations serving identical purposes

Security reviewers can glance at the structure and identify problems that traditional reviews would take hours to perform.

Threat Model

1. **AI-Generated Backdoors:** Malicious functionality from compromised or misaligned AI assistants
 2. **Patch-Introduced Vulnerabilities:** Well-intentioned fixes creating new security holes
 3. **Supply Chain Contamination:** Open-source dependencies maintained by volunteers using LLMs accumulating unverified code
 4. **Single Points of Failure:** Critical components without redundancy, exploitable by adaptive attacks
 5. **AI Hallucinations:** Plausible but incoherent code deployed to production
-

2. Method

TelicLens is designed around the principle that codebases are built functionally, working backwards from an overall objective to implementation. We propose that viewing code in terms of purposes and functionalities (i.e., its teleology) makes it easier for humans to understand and audit.

TelicLens thus renders code in two complementary visual forms to improve human discernment.

Causal View (Traditional): "How does data flow?"

- Bottom-up mechanical analysis
- Nodes = code elements (functions, classes)
- Edges = calls
- Shows execution paths and dependencies

Telic View (Novel): "What is this code *for*?"

- Top-down intentional analysis
- Nodes = functional purposes
- Edges = "serves the purpose of" or "is instrumental for"
- Shows how code contributes to system goals

Generating the Telic View

Telic Lens does the following:

1. **Extracts Stated Purposes:** Identifies top-level goals from documentation ("Secure financial transactions", "Create engaging gameplay")
2. **Decomposes Intentions:** Labels code components by function rather than implementation
3. **Builds Purpose Hierarchy:** Decomposes high-level goals into sub-purposes, then into implementations
4. **Maps to Source:** Every telic node links back to actual code for verification

What Problems Look Like

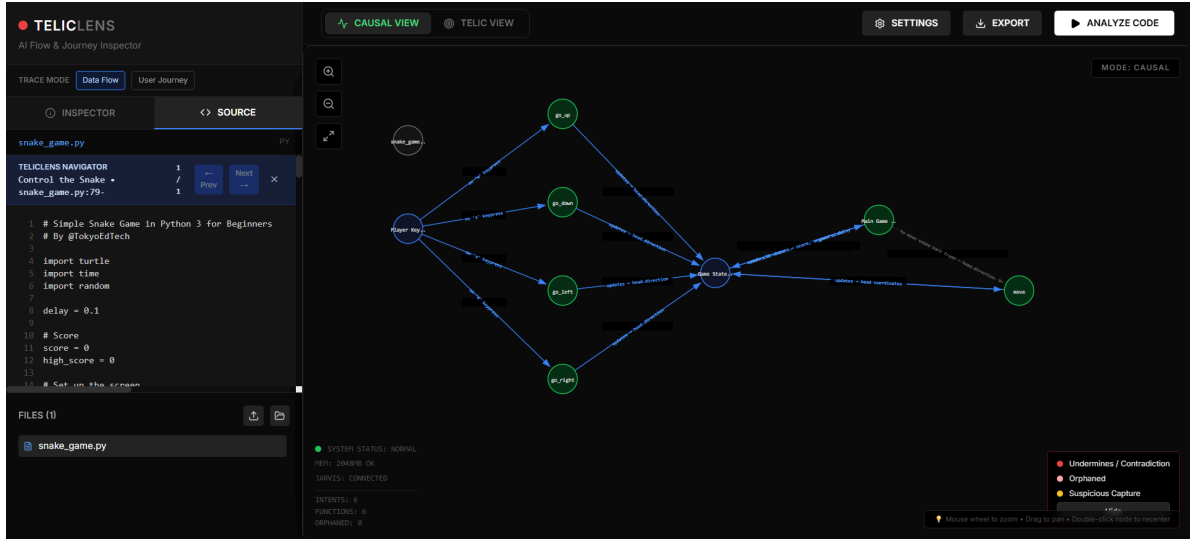
Problem Type	Description	Visual Signal
Misaligned Code	Purpose contradicts system goals	Disconnected/contradictory node
Dead Code	No outgoing edges to system purposes (common in AI over-generation)	Orphaned node in graph
Backdoors	Functionality working against stated goals (e.g., "fun gaming" with "collect browsing data")	Purpose contradiction visible
Architectural Weaknesses	Critical functions depend on single component (e.g., all security needs one time-sync module)	High-centrality node

User Workflow

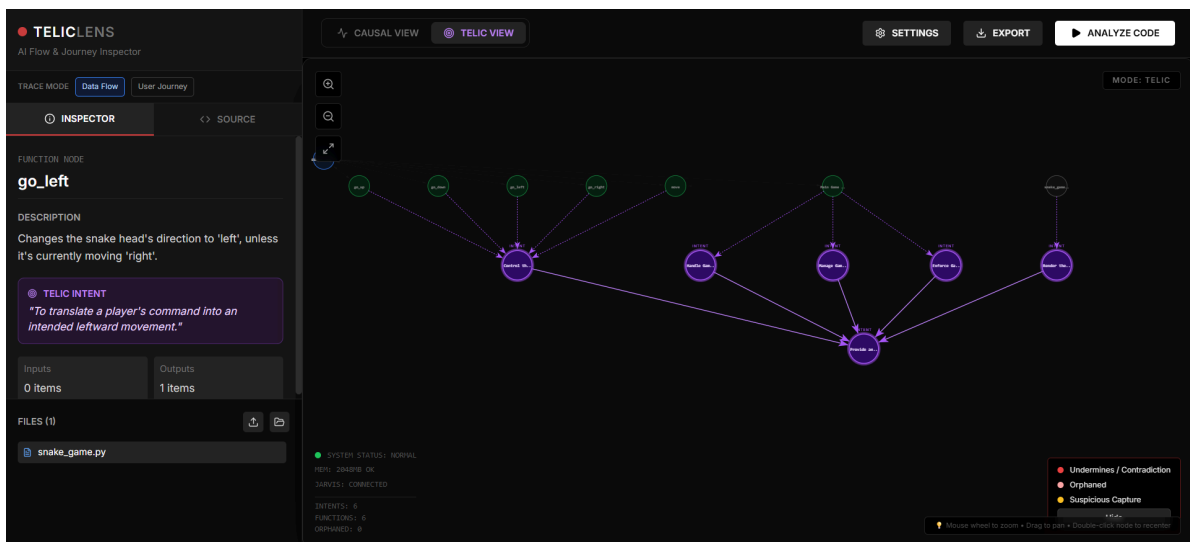
1. **Upload** codebase (30-60 seconds processing)
2. **Visually scan** telic graph for structural anomalies
3. **Investigate** suspicious nodes
4. **Verify** via causal view and traditional tools
5. **Report** findings for detailed review

3. Results

We built a basic functional prototype demonstrating rapid telic analysis on codebases of up to 2000 lines.



Causal view



Telic view

Capabilities: Interactive navigation, code tracing, view toggling, source code editing, automated identification of basic problems

Current Limitations

- Single codebases only (may need to work across multiple interacting systems)
- Only first glance human validation
- Telic nodes don't reliably align with file/function boundaries

4. Discussion

Speed Matching for Infrastructure Defense

Critical infrastructure protection is increasingly a race between adaptive attacks and defensive responses. TelicLens accelerates **human-supervision and oversight**:

The Core Innovation: Making Misalignment Visible

AI code optimizing for plausibility rather than purpose shows characteristic patterns:

1. **Purpose Contradiction**: Security code establishing external connections
2. **Orphaned Functionality**: Implementation with no path to system purposes
3. **Over-Specification**: Multiple components serving identical purposes
4. **Instrumental Incoherence**: Component serving incompatible purposes together

These patterns are hard to detect mechanically but immediately apparent in telic representation.

Future Directions

- Increase user-friendliness and accessibility of visualization
- Multi-system analysis across distributed architectures
- Benchmarking against traditional security tools on known vulnerabilities

Broader Impact

TelicLens demonstrates a paradigm for defensive AI: **augmenting rather than replacing human judgment**. As AI becomes more capable, the risk isn't AI doing things wrong—it's humans losing the ability to tell what AI is doing at all.

Tools that restore, accelerate and democratize human legibility and oversight are essential infrastructure for ensuring a resilient society and securing the benefits of AI progress. The watchtowers we need aren't fully automated systems blindly scanning for threats. They're human-AI partnerships where machines make things visible and humans make things right.

5. Conclusion

The defensive acceleration challenge requires tools operating at the intersection of human cognition and machine speed. TelicLens demonstrates that intentional (telic) analysis provides this intersection for code security.

By making purposes explicit and misalignment visible, we enable security engineers to maintain meaningful oversight even as patching accelerates to machine speeds. This isn't just a technical capability—it's a prerequisite for trustworthy critical infrastructure in an AI-accelerated world.

References

1. Delima, M. (2025, October 23). Smarter Threats Need Smarter Defenses: AI, APIs, and the Reality for Critical Infrastructure Security. *Security Boulevard*.
<https://securityboulevard.com/2025/10/smarter-threats-need-smarter-defenses-ai-apis-and-the-reality-for-critical-infrastructure-security/>
2. The Hacker News. (2025, November). When Attacks Come Faster Than Patches: Why 2026 Will be the Year of Machine-Speed Security.
<https://thehackernews.com/2025/11/when-attacks-come-faster-than-patches.html>
3. U.S. Department of Homeland Security. (2024, April 29). DHS Publishes Guidelines and Report to Secure Critical Infrastructure and Weapons of Mass Destruction from AI-Related Threats.
<https://www.dhs.gov/archive/news/2024/04/29/dhs-publishes-guidelines-and-report-secure-critical-infrastructure-and-weapons-mass>
4. GitHub. (2025, August 11). Securing the supply chain at scale: Starting with 71 important open source projects. *The GitHub Blog*.
<https://github.blog/open-source/maintainers/securing-the-supply-chain-at-scale-starting-with-71-important-open-source-projects/>
5. Open Source Security Foundation. (2025, January 23). Predictions for Open Source Security in 2025: AI, State Actors, and Supply Chains.
<https://openssf.org/blog/2025/01/23/predictions-for-open-source-security-in-2025-ai-state-actors-and-supply-chains/>
6. Spracklen, J., et al. (2025, April 14). AI Code Tools Widely Hallucinate Packages. *Dark Reading*.
<https://www.darkreading.com/application-security/ai-code-tools-widely-hallucinate-packages>
7. Trend Micro. The Mirage of AI Programming: Hallucinations and Code Integrity.
<https://www.trendmicro.com/vinfo/us/security/news/vulnerabilities-and-exploits/the-mirage-of-ai-programming-hallucinations-and-code-integrity>
8. Center for Security and Emerging Technology. (2024, October 1). Securing Critical Infrastructure in the Age of AI. Georgetown University.
<https://cset.georgetown.edu/publication/securing-critical-infrastructure-in-the-age-of-ai/>

Appendix: Security Considerations

Attack Vectors

- **Adversarial Code:** Sophisticated AI generating plausible telic structures masking malicious functionality
- **Model Manipulation:** Exploiting LLM weaknesses to mischaracterize harmful code
- **False Security:** Users over-relying on Daily Glance, skipping additional verification

Appropriate Use: Rapid assessment, architectural review, preliminary screening in multi-layer security **Inappropriate Use:** Sole security validation, automated deployment gating, legal/compliance certification alone

Critical Infrastructure Requirements

- Multi-layer verification (TelicLens + static analysis + human review + formal methods where possible)
- Mandatory human validation of flagged components
- Audit trails with confidence scores
- Regular tool validation on known vulnerabilities
- Graceful degradation to manual review

Known Limitations: Sophisticated adversarial AI can likely fool telic analysis; human validation necessary; purpose ambiguity in some legitimate code; LLM updates may change interpretations unpredictably