**Case Study**

**Malware Analyzer — Interactive Node Simulation**

* **Executive Summary**

This case study evaluates the Malware Analyzer interactive node simulation (https://shivam139420.github.io/shivam/). The simulation demonstrates how different malware types (virus, worm, trojan) propagate in a network of nodes. The objective was to assess educational effectiveness, usability, technical design, and provide actionable recommendations to improve accuracy, scalability, and instructional value.

* **Background**

Malware propagation is a critical topic in cybersecurity education. Simulations that model infection spread help students and practitioners visualize dynamics, test defensive measures, and learn trade-offs between attack vectors and countermeasures. The Malware Analyzer project provides an interactive, browser-based environment for exploring these concepts in a safe, educational context.

* **Objectives**

The case study aims to:

• Evaluate the simulation for pedagogical clarity and technical correctness.

• Identify usability strengths and weaknesses.

• Recommend features and optimizations to increase educational value and performance.

* **Methodology**

The evaluation consisted of: (1) hands-on interaction with the web simulation, (2) inspecting available UI elements and logs, (3) comparing observable behaviors against simplified epidemic models (SIR-style), and (4) proposing practical improvements.

* **Observations & Key Findings**

1. Clarity and Educational Value

• The simulation allows users to start infections manually, observe propagation, and view logs — which supports experiential learning.

• Missing explicit legend/definition for node states (infected, susceptible, recovered) can confuse new learners.

2. Technical Design & Accuracy

• Propagation behavior shows sensible per-link infection attempts. However, the simulation does not document the underlying probabilistic model (e.g., infection probability per tick, recovery rules), limiting reproducibility of experiments.

3. Usability & UI

• Interactive controls (start/stop/reset/add node) are clear, but tooltips and parameter inputs (infection rate, delay) would improve exploratory experimentation.

4. Performance & Scalability

• Works well on small topologies. For larger graphs (>200 nodes) canvas rendering and algorithmic complexity may become bottlenecks.

* **Recommendations**

Short-term (quick wins):

• Add a visible legend explaining node colors and states.

• Expose core parameters (infection probability, tick duration, recovery chance) as UI controls and display their current values.

• Add simple charts (infections over time) and exportable CSV for analysis.

Medium-term:

• Implement common epidemic models (SIS/SIR) as selectable modes and document assumptions for each model.

• Add countermeasures (patching, firewalls, quarantining) so learners can test defenses.

• Improve accessibility: color-blind friendly palettes and keyboard navigation.

Long-term / Technical:

• Optimize rendering with WebGL or virtualized canvas layers and replace O(n^2) neighbor checks with efficient adjacency lists.

• Add server-side experiment logging to save runs for reproducible research and classroom assignments.

* **Sample Classroom Experiment**

Objective: Compare propagation speed for a 'worm' vs a 'trojan' on a 50-node random graph.

Steps:

1. Create a 50-node random topology (preferably with average degree 3).

2. Set tick duration = 500ms, infection probability = 0.2 for worm, and synchronous single-target infection for trojan.

3. Seed a single infected node and run three trials per malware type.

Expected metrics to record: time-to-50%-infection, peak infection count, and total infected after 5 minutes.

* **Conclusion**

The Malware Analyzer simulation is a strong foundation for hands-on cybersecurity education. With modest additions—parameter controls, model documentation, and visual analytics—it can become a powerful classroom tool. Longer-term improvements in scalability and experiment reproducibility will broaden its suitability for research and larger teaching cohorts.

* **Appendix: Suggested UI Controls & Parameters**

• Infection Probability (per-contact) — float 0.0–1.0  
• Tick Duration — milliseconds  
• Recovery Probability (per-tick) — float 0.0–1.0  
• Node Immunity / Patch Status — boolean per node  
• Network Topology Presets — random, scale-free, grid, small-world  
• Export CSV / PDF report — include time series and trial metadata