**Comprehensive Research Report on LLM Vulnerability Assessment Tools**

# Introduction

The increasing adoption of Large Language Models (LLMs) in real-world applications raises significant concerns about security, safety, and ethical use. As these models are exposed to the public via APIs and chat interfaces, they become potential targets for adversarial attacks. This research report evaluates the leading tools developed to assess, audit, and fortify LLMs against vulnerabilities such as prompt injection, data leakage, jailbreaks, hallucination, and bias.  
  
This report covers seven prominent tools: **Garak, Gandalf, Rebuff, Llama Guard, PromptBench, AdvPromptGuard, and OpenPromptEval**. Each is analyzed based on purpose, inputs, outputs, usage, threat coverage, and practical applicability.

## 1. Garak (NVIDIA)

### Overview

Garak is a versatile red-teaming toolkit developed by NVIDIA for LLM vulnerability scanning. It simulates real-world adversarial inputs to evaluate LLMs against threats such as prompt injection, model inversion, and XSS.

**Key Capabilities**

Supports 15+ attack probes including:

* Prompt Injection
* Cross-Site Scripting (XSS)
* Jailbreak Prompts
* Data Leakage
* Malware Generation
* Model Inversion

### Inputs

* Model Type (OpenAI, HuggingFace, etc.),
* API Keys (e.g., OPENAI\_API\_KEY),
* Probe list (e.g., promptinject, xss, dan), Optional YAML config file.

### Outputs

* Protection metrics (HTML, CSV),
* Matrix of success/failure per probe,
* Detailed probe-response logs.

### How to Use

* Clone the repo, install dependencies,
* set API keys,
* choose model/probe via CLI (e.g., `garak model openai:gpt-4 probes xss`).

### Importance

Covers a broad set of vulnerabilities, supports confidence comparison, ideal for red-teaming.

### Strengths

* High configurability,
* multi-vector attack simulation,
* visual reporting.

### Limitations

* Batch-based,
* requires configuration knowledge.

## 2. Gandalf (Protect AI)

### Overview

Gandalf is a specialized tool designed to detect prompt injection vulnerabilities. It sends adversarial inputs to an LLM and checks if the payload is executed or blocked.

### Key Capabilities

* Sends structured malicious inputs (payloads) to model endpoints
* Detects:
  + Prompt injection success
  + Prompt leakage
  + Model behavior (execute/block)

### Inputs

* LLM API endpoint,
* YAML configuration file.

### Outputs

* JSON logs (prompt → response → status),
* optional HTML summaries,
* flagged injection attempts.

### How to Use

* Install via pip,
* define endpoint in YAML,
* run `gandalf config config.yaml`.

### Importance

Focused on prompt injection, lightweight for CI/CD.

### Strengths

* Quick deployment,
* clear logs,
* easy configuration.

### Limitations

* Limited to prompt injection detection.

## 3. Rebuff

### Overview

Rebuff acts as a real-time firewall for LLMs, filtering malicious inputs and outputs in live deployments.

### Key Capabilities

* Real-time scanning of:
  + User inputs for malicious patterns
  + Model outputs for sensitive or policy-violating content
* Detects:
  + Jailbreak attempts
  + Personally Identifiable Information (PII) exposure
  + Leakage of internal prompts

### Inputs

* Wrapped LLM endpoint,
* YAML-based policy configuration.

### Outputs

* Real-time alert logs,
* policy violation traces,
* jailbreak attempt logs.

### How to Use

* Deploy as proxy middleware,
* define YAML rules,
* monitor logs via API or dashboard.

### Importance

Ideal for production environments to intercept threats on the fly.

### Strengths

* Real-time filtering, flexible policy control.

### Limitations

* Requires infrastructure integration, operational maintenance.

## 4. Llama Guard (Meta)

### Overview

Llama Guard is a moderation classifier developed by Meta that flags harmful, biased, or unsafe inputs and outputs.

### Key Capabilities

* Classifies content into categories such as:
  + **Violence**
  + **Hate Speech**
  + **Misinformation**
* Flags unsafe content before or after LLM execution

### Inputs

* Prompt or output text string.

### Outputs

* Safety label (Safe/Unsafe),
* Risk category (Violence, Hate, etc.),
* JSON output.

### How to Use

* Load via Hugging Face,
* use `predict(text)` via API.

### Importance

Supports moderation and fairness reviews.

### Strengths

* Simple integration,
* multi-risk categorization.

### Limitations

* No traceability or logging features.

## 5. PromptBench

### Overview

PromptBench provides standardized prompts to test LLMs for jailbreak vulnerability and compares their resistance.

### Key Capabilities

* Uses a curated set of adversarial prompts
* Measures:
  + Jailbreak success rates
  + Response severity
  + Model-wise resistance

### Inputs

* LLM model name or API endpoint,
* standardized test prompt set.

### Outputs

* Jailbreak success rates,
* flagged response categories,
* visual graphs.

### How to Use

* Clone repo, run `python run\_benchmark.py `

### Importance

Enables benchmarking and comparative scoring.

### Strengths

* Easy visualization,
* reproducible tests.

### Limitations

* Focuses only on jailbreak detection.

## 6. AdvPromptGuard

### Overview

AdvPromptGuard simulates stealthy and obfuscated prompt injections to evaluate LLM filter evasion.

### Key Capabilities

* Generates stealthy prompt injections
* Evaluates:
  + Filter evasion
  + Attack success
  + Defense bypass effectiveness

### Inputs

* Prompt dataset,
* target model or endpoint.

### Outputs

* Injection logs,
* success rates,
* prompt-response trace comparison.

### How to Use

* Install via pip,
* run scans on prompt set and LLM endpoint.

### Importance

Specialized for adversarial red teaming.

### Strengths

* Tests model robustness against filtered bypass attacks.

### Limitations

* Advanced configuration may require expertise.

## 7. OpenPromptEval

### Overview

OpenPromptEval evaluates LLMs on bias, fairness, hallucination, and explainability using curated prompts.

### Key Capabilities

* Tests models for:
  + **Unintended bias**
  + **Hallucinated outputs**
  + **Toxicity and hate speech**
* Used in compliance testing and AI ethics auditing

### Inputs

* Task (bias, hallucination, etc.),
* LLM API or model.

### Outputs

* Bias graphs,
* fairness scores,
* toxicity distributions.

### How to Use

* Clone repo,
* run `python eval.py --task bias --model mistral`.

### Importance

Supports ethics auditing and compliance.

### Strengths

* Rich explainability metrics,
* visual output.

### Limitations

* Not designed for runtime defense.

## ****Conclusion****

LLM security is multi-faceted, and no single tool can cover all angles. The following hybrid strategy is recommended:

* **Use Garak** for full-spectrum scanning and red teaming.
* **Integrate Gandalf** into CI/CD pipelines for regular prompt injection testing.
* **Deploy Rebuff** in production environments for real-time input/output threat filtering.
* **Use Llama Guard and OpenPromptEval** to enforce ethical safety, fairness, and bias management.
* **Leverage PromptBench** for quantitative evaluation and benchmarking of new models.
* **Utilize AdvPromptGuard** for advanced threat simulations.

This modular toolchain enables organizations and researchers to achieve defense-in-depth for LLMs across all threat vectors—prompt-based, behavioral, ethical, and systemic.