DDoS & Resilience Testing for AI Applications

Engagement use cases — controlled testing with the CaosBlitz platform June 2026

AI applications change the economics of denial-of-service. A traditional web app needs high request volume to exhaust connections or CPU; an AI application is the opposite — each request can occupy a GPU or Tokens Budget for seconds, consume large volumes of metered tokens, and fan out into multiple downstream calls. A small number of crafted requests can therefore degrade availability or inflate cost. The scenarios below are controlled and authorized, and target that cost asymmetry rather than raw throughput.

Test scenarios

1. **Inference endpoint saturation.** Ramp concurrent requests to find the GPU/inference ceiling — where queue depth grows, time-to-first-token spikes, and users time out — a breaking point reached at far lower request rates than a traditional web tier.
2. **Denial of Wallet (economic DoS).** Moderate-volume, cost-maximizing requests that drive the model-API or autoscaling-GPU bill upward without taking the service down, validating spend caps, budget alerts, and cost circuit-breakers.
3. **Input amplification.** Maximum-size prompts and oversized documents that multiply prefill compute and memory, testing context-size limits and worker stability (out-of-memory) under large payloads.
4. **Output & streaming exhaustion.** Concurrent maximum-length, slow generations that both consume compute and hold streaming connections open — exhausting gateway sockets and connection pools — while validating output caps and timeouts.
5. **RAG pipeline fan-out.** Queries that expand into embedding, vector search, reranking, and multiple LLM calls; measures the amplification factor and saturates the vector database and semantic cache (cache-miss flooding).
6. **Agentic / tool-loop exhaustion.** Requests that induce long agent loops and repeated tool calls, consuming compute and downstream third-party quota, while validating iteration caps, timeouts, and loop detection.
7. **Autoscaling & cold-start abuse.** Bursts from an idle or scaled-to-zero state, plus oscillating “yo-yo” load, to measure cold-start penalty, requests dropped during scale-up, and cost behavior under scaling churn.
8. **Rate-limit & quota fairness.** Tests of whether throttling is cost/token-aware rather than naive request-count, plus per-user quota enforcement and resistance to distributed or multi-account evasion.

What you receive

Each engagement reports the application’s true capacity ceiling under realistic AI load, the effectiveness of cost-aware rate limiting and quotas, input and output guardrails, autoscaling and cost circuit-breaker behavior, graceful-degradation handling (load shedding, 429s, fallback models), and — most often the critical gap — whether monitoring detects anomalous token consumption, cost spikes, and queue growth rather than only classic traffic volume.