Supplementary Methods: Seed Hijacking of LLM Sampling and Quantum Random Number Defense

Ziyang You[1], Xiaoke Yang[2], Zhanling Fan[3], Feng Guo[1], Xiaogen Zhou[1*], Xuxing Lu[1,4*]

[1]School of Electronic, Electrical and Physics, Fujian University of Technology, Fuzhou, 350118, China.
[2]School of Humanities, Fujian University of Technology, Fuzhou, 350118, China.
[3]Department of Investigation, Fujian Police College, Fuzhou, 350007, China.
[4]Institute of Applied Physics and Materials Engineering, University of Macau, Macau, 999078, China.

*Corresponding author(s). E-mail(s): xiaogenzhou@fjut.edu.cn; xuxinglu@um.edu.mo;

1 Methods

Attack implementation: SeedHijack operates in three stages. First, the attacker recovers the internal state of the Mersenne Twister (MT19937) pseudorandom number generator (PRNG) by observing 624 consecutive 32-bit outputs, sufficient to reconstruct the full $624 \times 32 = 19{,}968$-bit state vector. In our supply-chain scenario, the attacker directly instruments the PRNG module via Python `ctypes` to expose the internal state array. Second, given the recovered state, all subsequent PRNG outputs are predicted deterministically using the MT19937 recurrence relation. Third, at each generation step, the attack intercepts the sampling call: it receives the softmax distribution P from the model, computes the cumulative distribution function (CDF) $F(t) = \sum_{i=1}^{t} p_i$, identifies the target token t^*, and returns a crafted value $u^* = F(t^* - 1) + p_{t^*}/2$, which falls strictly within $[F(t^* - 1), F(t^*))$. The inverse-transform sampling mechanism then deterministically selects t^*. The attack modifies only the uniform random number generation function, a single function call in the pipeline, leaving tokenizer, model forward pass, and post-processing untouched. In

all experiments, the SeedHijack attack is continuously enabled during inference, consistent with a supply-chain compromise that persistently manipulates the PRNG module.

QRNG defense deployment: The QRNG600 PCIe card generates true random numbers by measuring quantum vacuum fluctuations in an optical homodyne detection system. The security of this source rests on the Heisenberg uncertainty principle applied to the measured quadrature of the vacuum state, which guarantees intrinsic quantum indeterminacy independent of device calibration. The raw quantum signal undergoes Toeplitz-matrix extraction to remove classical noise correlations, yielding output certified against the National Institute of Standards and Technology (NIST) SP 800-22 statistical test suite. The card provides sustained 600 Mbps throughput via PCIe 3.0 interface.

We implemented a pre-buffered architecture: 50 million uniform $[0, 1)$ float64 values are generated offline from the QRNG hardware and stored on disk (400 MB). At runtime, memory-mapped access provides sequential quantum random values with only 7.7 MB added to the process memory footprint. During inference, a circular index pointer advances through the buffer, providing one quantum random value per token generation step. Buffer exhaustion triggers asynchronous replenishment without interrupting inference. This architecture decouples random number generation latency from the inference critical path, limiting overhead to a single sequential memory read per token (+0.6% median latency relative to on-demand MT19937 computation).

2 Theoretical Analysis

We first establish the deterministic guarantee of SeedHijack by analyzing the determinism of the MT19937 pseudorandom number generator (PRNG). The MT19937 PRNG holds a 19,968-bit internal state (624 words of 32 bits), with a linear recurrence structure over $\mathbb{F}_2$ and a characteristic polynomial of degree 19,937, giving a period of $2^{19937} - 1$. Observing just 624 consecutive 32-bit outputs yields 19,968 bits of constraint, which is sufficient to uniquely determine the full internal state—the system is overdetermined by 31 bits, and the invertible tempering transformation ensures no information loss during state recovery. Once the state is known, all future PRNG outputs become fully deterministic and predictable.

For token injection, standard inverse-transform sampling selects a token t by comparing a uniform random value $u \sim \text{Uniform}[0, 1)$ against the cumulative distribution function (CDF):

$$F(t) = \sum_{i=1}^{t} p_i, \quad F(0) = 0, \quad F(t-1) \leq u < F(t). \tag{1}$$

By replacing the PRNG output with a crafted value $u^* = F(t^* - 1) + p_{t^*}/2$, the adversary forces u^* to fall strictly within the interval $[F(t^* - 1), F(t^*))$. This injection succeeds with probability 1, provided $p_{t^*} > \epsilon_{\text{mach}} \approx 1.19 \times 10^{-7}$ (float32 machine epsilon), ensuring the interval is numerically representable.

We next prove the information-theoretic security of the QRNG defense. The QRNG generates true randomness by measuring quantum vacuum fluctuations via optical homodyne detection, where the quadrature observable $\hat{X}$ follows $\mathcal{N}(0, 1/2)$. Randomness is fundamentally guaranteed by the Heisenberg uncertainty principle and the no-cloning theorem, which ensures the quantum state cannot be copied or predicted prior to measurement. All classical side information available to an adversary is fixed before measurement, making the quantum random variable U_Q statistically independent of any adversary knowledge.

This independence implies zero mutual information:

$$I(U_Q; E) = 0, \tag{2}$$

where E denotes arbitrary adversary side information. The quantum random number U_Q remains uniformly distributed over $[0, 1)$, so the probability that U_Q falls into the target CDF interval equals the natural token probability p_{t^*}. The adversary gains no advantage, and the SeedHijack attack is fully neutralized even against computationally unbounded adversaries.

3 Experimental Details

All experiments were implemented with PyTorch 2.1.0, Transformers 4.35.2, CUDA 12.2, and cuDNN 8.9.4 on a single NVIDIA RTX 3090 GPU (24 GB VRAM). GPT-2 (124M parameters) served as the unaligned baseline in float32 precision. Qwen2-1.5B-Instruct (1.5B parameters, aligned via Reinforcement Learning from Human Feedback (RLHF) with Supervised Fine-Tuning (SFT)) was loaded in bfloat16 with chat template formatting. DeepSeek-R1-1.5B (1.5B parameters, reasoning-distilled from DeepSeek-R1) was loaded in bfloat16 with its native prompt format. To evaluate scale invariance, we additionally tested Qwen2-7B-Instruct (7B parameters, RLHF + SFT) and DeepSeek-R1-Distill-Qwen-7B (7B parameters, reasoning distillation), both loaded in bfloat16.

For GPT-2, sampling configurations spanned temperature $\tau \in \{0.7, 1.0, 1.5\}$ and top-$p \in \{0.9, 0.95, 1.0\}$, yielding 9 unique parameter combinations tested with 20 prompts across 3 random seeds (60 trials per configuration, 540 total). For alignment bypass experiments (1.5B and 7B models), each model was tested across 3 sampling configurations with 60 trials per configuration (180 trials per model, 720 total across four aligned models). QRNG uniformity was validated using Pearson chi-square goodness-of-fit with 100 bins over 10^6 samples ($p > 0.99$). Performance benchmarking reports median per-token latency over 1,000 iterations with 50 warmup iterations excluded.

Success metric: exact token match: Attack success is defined as *exact token match*: for each generation step in which the attacker targets a specific token t^*, the attack is scored as successful if and only if the token actually emitted by the model is identical to t^* (i.e., the integer token ID returned by the sampler equals the attacker's intended token ID). Concretely, each trial proceeds as follows: (1) the attacker specifies a multi-token payload string and tokenizes it into a target token sequence $(t_1^*, t_2^*, \ldots, t_L^*)$; (2) at each autoregressive step i, the model produces the

logit distribution, the attack computes the CDF and injects u_i^* to force selection of t_i^*; (3) the emitted token $\hat{t}_i$ is compared to t_i^*, and the trial is marked as a match if $\hat{t}_i = t_i^*$ for all L positions in the payload, and a failure otherwise. We only target tokens with non-negligible probability ($p_{t^*} > \epsilon_{\text{mach}} \approx 1.19 \times 10^{-7}$) to avoid float32 precision failures. A single mismatched position anywhere in the payload sequence counts as a failed trial. The exact token match rate is then the number of fully successful trials divided by the total number of trials. This strict all-or-nothing criterion ensures that reported success rates reflect complete payload injection rather than partial token overlap.