



Hardware Trojans in eNVM Neuromorphic Devices

Students: Lingxi Wu, Rahul Sreekumar (joint 1st author), Rasool Sharifi
 Advisors: Kevin Skadron, Mircea Stan, Ashish Venkat



Context

Fig 1. Neuromorphic Computing using eNVM memory arrays

- Emerging non-volatile memory (**eNVM**): stores bits/values as conductance
- eNVM-based accelerators can mimic biological neuron computations (**neuromorphic**) → AI acceleration (DNN)
- Prior works focus on performance → **security implications** remain **largely unexplored**

Attack Motivation

Development stage:

- SW model selection (VGG, ResNet, etc.)
- HW implementation
- Manufacturing (fab)

Post-deployment stage:

- Individual users, cloud provider, ML service providers (ML-as-a-service e.g., BigML)
- Train the model (configure weights)
- Prediction using the model

Fig 2. Neuromorphic devices product life cycles & parties involved

Fig 3. Major milestones and deliverables of the IC design and manufacturing process

Designed and manufactured in a **decentralized** way

- Cost reduction → IC supply chain is **distributed**
- Potential perpetrators** in supply chain:
 - Tainted 3rd party IP blocks or CAD tools
 - Rogue RTL engineers
 - Malicious foundry tamper with the mask layout

Motivate the **supply-chain attack**:

- Model extraction: obtaining synaptic weights of a DNN model
- Weights are the **core IP**
- Stealing weights more economical than training

Exploitable Vulnerabilities

Fig 4. Trojan-infected Neuromorphic devices product life cycles & parties involved

- Vulnerable to Trojan insertion at the **design** and **fabrication** stage
- Colluding malicious entities: embed + activate Trojan
- Or simply publish Trojan code

Neuromorphic chips in post-deployment is still vulnerable to a Trojan placed in the supply chain

Vulnerability one: current strength and synaptic weights

eNVM cells holds synaptic weights (conductances)

$$W_{0,1}, W_{1,1}, W_{2,1} \rightarrow G_{0,1}, G_{1,1}, G_{2,1}$$

Weighted sum → current I **Strength** of I depends on the W → larger W = larger G = larger I

Vulnerability two: Integrate-and-fire ADCs generate **spike trains** → larger I = more spikes = more **transient power switching activity**

Attack Procedure

We propose: 2-Phase attack strategy (Trojan Embedding + Trojan Activation/Side channel weight stealing)

Phase-I: Trojan-assisted power side-channel model extraction attack

- Malicious party embeds Trojan + Distribution of Trojan code (trigger)
- Offline Characterization of neuron ADC. (Frequency Fingerprint Library)

Phase-II:

- Activation of Trojan (embedding trigger code as pixel combination)
- Side channel Analysis + Trace decomposition
- Cross-referencing with characterization library

Results

Recover more than **90%** of the weights

Attack improves with ADC resolution → **30%** recovery for **2-bit** in resolution

Recovered Accuracy is comparable ($\sim \pm 2.65\%$) even for low precision ADCs.

Trojan Stealth analysis:

- Noise Trojan \ll noise floor ($\sim 150 \mu V^2/Hz$)
- Area overhead **0.28** - 87% total overhead
- False triggering of trojan $\ll 0.01\%$