A Brain-inspired Approach for Malware Detection using Sub-semantic Hardware Features

Maryam Parsa
mparsa@gmu.edu
Department of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia, USA

Khaled N. Khasawneh
kkhasawn@gmu.edu
Department of Electrical and Computer Engineering
George Mason University
Fairfax, Virginia, USA

Ihsen Alouani
i.alouani@qub.ac.uk
Center for Secure Information Technologies (CSIT)
Queen's University Belfast
Belfast, UK

ABSTRACT

Despite significant efforts to enhance the resilience of computer systems against malware attacks, the abundance of exploitable vulnerabilities remains a significant challenge. While preventing compromises is difficult, traditional signature-based static analysis techniques are susceptible to bypassing through metamorphic/polymorphic malware or zero-day exploits. Dynamic detection techniques, particularly those utilizing machine learning (ML), have the potential to identify previously unseen signatures by monitoring program behavior. However, classical ML models are power and resource intensive and may not be suitable for devices with limited budgets. This constraint creates a challenging tradeoff between security and resource utilization, which cannot be fully addressed through model compression and pruning. In contrast, neuromorphic architectures offer a promising solution for low-power brain-inspired systems. In this work, we explore the novel use of neuromorphic architectures for malware detection. We accomplish this by encoding sub-semantic micro-architecture level features in the spiking domain and proposing a Spiking Neural Network (SNN) architecture for hardware-aware malware detection. Our results demonstrate promising malware detection performance with an 89% F1-score. Ultimately, this work advocates that neuromorphic architectures, due to their low power consumption, represent a promising candidate for malware detection, especially for energy-constraint processors in IoT and Edge devices.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Artificial intelligence**; • **Security and privacy** → **Systems security**.

KEYWORDS

Neuromorphic Computing, Malware Detection, Computer Security

ACM Reference Format:
Maryam Parsa, Khaled N. Khasawneh, and Ihsen Alouani. 2023. A Brain-inspired Approach for Malware Detection using Sub-semantic Hardware Features. In *Proceedings of the Great Lakes Symposium on VLSI 2023 (GLSVLSI '23), June 5–7, 2023, Knoxville, TN, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583781.3590293

GLSVLSI '23, June 5–7, 2023, Knoxville, TN, USA

ACM ISBN 979-8-4007-0125-2/23/06.
https://doi.org/10.1145/3583781.3590293

1 INTRODUCTION

In today's computing world, where data and artificial intelligence play an ever-increasing role in our daily lives, malware poses one of the most dangerous threats to modern society [2, 11]. Over the past two decades, hardware security has become an increasingly pressing concern due to the globalization of the semiconductor supply chain and the proliferation of connected smart infrastructures and computing edge devices. According to McAfee's COVID-19 threat report, the total number of malware increased by 1,902 percent over the four quarters of 2020. Additionally, an average of 375 new threats were reported per minute [39]. Malware detection in a timely manner is crucial not only for preventing its spread to a large number of systems and edge devices, but also for mitigating its potential impact. Classical methods such as signature-based [10], heuristic-based [3], dynamic binary instrumentation [1], and information flow tracking [16] approaches mostly fall short in detecting various types of new and sophisticated generations of malware, introduce considerable overhead, and allow attackers to bypass them and remain undetected [23].

Various Deep Neural Networks (DNNs) [9] have been proposed to detect malware using static and dynamic analysis. These approaches are applied in multiple contexts such as advanced persistent threat (APT) attacks [43], and sandboxing-based malware detection [11]. More recently, Hardware Malware Detector (HMD) are introduced to make systems more resilient to malware attacks. Low-level sub-semantic hardware features such as memory reference patterns, and architectural state information can be used for online malware detection [19]. While these approaches generally show great promise in detecting malware, their high power consumption and latency pose a significant challenge for energy-constraint devices. These techniques demand access to vast amounts of data and computational resources during training [11].

Neuromorphic computing, as a promising alternative for DNNs in resource constrained environments, is inspired by the low-power computation in biological brains [33, 37, 40]. In this novel non Von Neumann architecture, Spiking Neural Networks (SNNs) are the building blocks of algorithmic learning and hardware implementation. With the goal of reducing computational complexity and latency in detecting malware and motivated by the low power consumption of the neuromorphic chips, our key objective is to seek an answer for the following question: ***Can neuromorphic architecture learn from hardware sub-semantic features to detect malware?***. We show in section 4 a successful implementation of SNNs that can detect malware using sub-semantic hardware

features with promising performance compared to a multi-Layer Perceptron (MLP) based HMD.

2 BACKGROUND & RELATED WORKS

2.1 Sub-semantic Hardware features using hardware performance counters

Modern microprocessors rely on Hardware Performance Counters (HPC) as a vital feature, enabling the monitoring and analysis of microarchitectural events that occur during the execution of software applications These specialized registers are designed to keep track of hardware events, such as cache hits and misses, branch mispredictions, and other microarchitectural activities. Table 1 shows the typical HPC events available under Linux Perf tool for Intel Haswell processors [7].

The primary objective of HPC is to provide developers with a detailed insight into the performance characteristics of their software applications, enabling them to identify potential bottlenecks and optimize their code accordingly [28]. By analyzing the data gathered by HPC, developers can optimize their code for a specific microarchitecture, leading to improved performance and reduced power consumption. Nonetheless, the use of HPC is not limited to performance optimization alone. HPC are also increasingly being employed in security-related applications, such as detecting software malware [19, 23], detecting firmware modifications [5], and detecting hardware trojans [45].

2.2 Spiking Neural Networks

SNNs are the third generation of neural networks, inspired by the human brain's event-based communication scheme. This bio-inspired computation provides biologically-plausible deep learning and has the potential to bridge the energy-efficiency gap between the human brain and supercomputers executing complex deep learning applications. Recent advances in neuromorphic architectures, such as IBM's TrueNorth [30] and Intel's Loihi [6], have shown the promise of SNNs for energy-efficient computing.

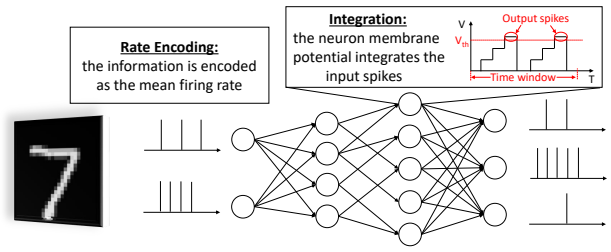

Figure 1: Overview of the functionality of an SNN, with a focus on the rate encoding of the information and the integration of the spikes into the membrane potential.

Figure 1 demonstrates how SNNs typically operate. The input information is encoded to spike representation. While delay-based and latency-based coding schemes are possible, the most commonly adopted mechanism for encoding input information in SNNs is the *rate encoding* [36]. In rate encoding, the intensity of activation corresponds to the mean firing rate over a specific *time window*.

bus-cycles	branch-instructions	cpu-cycles	cpu-clock
cpu-migrations	dummy	L1-dcache-store-misses	LLC-load-misses
LLC-prefetch-misses	dTLB-store-misses	branch-load-misses	node-store-misses
instructions	branch-misses	task-clock	minor-faults
emulation-faults	L1-dcache-prefetch-misses	LLC-stores	dTLB-loads
iTLB-loads	node-loads	node-prefetches	cache-references
page-faults	major-faults	L1-dcache-loads	L1-icache-load-misses
LLC-store-misses	dTLB-load-misses	iTLB-load-misses	node-load-misses
node-prefetch-misses	cache-misses	ref-cycles	context-switches
alignment-faults	L1-dcache-load-misses	LLC-loads	LLC-prefetches
dTLB-stores	branch-loads	node-stores	L1-dcache-stores

Table 1: List of HPC events using PERF tool

This time window represents the observation period during which the SNNs receives the same input. A wider time window allows more time for the spikes to propagate towards the output, but it also incurs higher latency." When an incoming spike s_i arrives at the input of the neuron, it is multiplied by its associated synaptic weight w_i and integrated into the membrane potential V, following Equation 1.

$$V = \sum_{i=1}^{N} w_i \cdot s_i \tag{1}$$

In a Leaky-Integrate-and-Fire (LIF) [8] spiking neuron model, an output spike is emitted and the membrane potential is reset when it exceeds the *threshold voltage* V_{th}. This mechanism allows information to be propagated to the output. For example, in rate encoding for image classification, the firing rate of the output spike train determines the probability of the associated class. A higher firing rate corresponds to a higher output probability. Due to the non-differentiability of the loss function [38] in SNNs, the standard backpropagation method in DNNs cannot be directly applied. To address this challenge, two possibilities exist. The first involves training a corresponding DNNs using standard backpropagation and then converting it to the spiking version [29]. However, this conversion process is slow and usually results in some loss of accuracy. Alternatively, the SNN derivatives can be approximated, and learning can be based on temporal information in the spiking domain [44]. In our work, we adopt the latter option.

3 DATASET & FEATURE COLLECTION

3.1 Dataset

Our data collection involved gathering HPC data from 52 benign applications and 57 malware applications. The benign applications consisted of the mibench benchmark suite [14], along with common Linux programs such as browsers, text editors, and word processors. As for the malware applications, we obtained them from virus-total.com [42], and they included Linux ELFs, Python scripts, Perl scripts, and Bash scripts designed to carry out malicious workloads/activities. The dataset was divided evenly into 3-folds, which are *training*, *validation*, and *testing*. We use 3-fold cross-validation in our experiments to get accurate results, i.e., eliminate bias. Furthermore, the malware types and the benign application types were distributed evenly and randomly across the folds to ensure that the datasets are not biased.

3.2 Features Collection

To collect low-level HPC data dynamically, we used a machine equipped with an Intel Haswell Core i5-4590 CPU, running Ubuntu 14.04 with Linux 4.4 Kernel. All security and firewall services were disabled to ensure that malware runs freely. Multiple methods exist for capturing HPC data, including direct reading of Model-specific Registers (MSRs), kernel module-based sampled collection, or utilizing various Linux utilities like Perf, PAPI, and Perfctr. In our study, we opted for Perf, which leverages the perf event open function to measure multiple events simultaneously [7].

4 INITIAL RESULTS & DISCUSSION

4.1 Baseline-HMD

HMDs use low-level features to classify malware as a computational anomaly at run-time. We trained our baseline HMD on a multi-layer perception (MLP) neural network using the features described in Section 3. The MLP consists of a single hidden layer that has the number of neurons equal to the number of input features (39 neurons). We used the Sigmoid as an activation function. The rationale for selecting neural networks to train our baseline HMD is that it has the highest performance in detecting malware [17, 20, 21, 24–26, 32].

The *training* set was used to train our baseline HMD and the *testing* data set was used to evaluate its detection performance. Table 2 shows the baseline HMD detection performance in classifying malware and benign programs using the following metrics: *accuracy* (how many programs the baseline HMD correctly labeled out of all programs), *sensitivity* (how many malware the baseline HMD correctly classified out of all programs that were labeled as malware), *specificity* (how many benign programs the baseline HMD correctly classified out of all programs that were labeled as benign), *precision* (how many of those who were labeled as malware are actually malware), and *F1-score* (the harmonic mean of the precision and sensitivity). The results show that the baseline HMD achieves high performance in detecting malware across all performance metrics.

Table 2: Baseline HMD detection performance

Accuracy	Sensitivity	Specificity	Precision	F1-Score
93.9%	75.2%	97.3%	88.2%	94.1%

4.2 Neuromorphic HMD

Our experiments for the neuromorphic HMD were conducted using the Norse Python library [35], which is an extension of PyTorch providing primitives for bio-inspired neural computation. This framework enables us to train and execute neural networks in the spiking domain. We designed a three-layer SNNs architecture, consisting of 39 input neurons that correspond to the hardware features, 50 hidden neurons, and 2 output neurons. In our experiments, we employed the LIF neuron model and used the Adam optimizer with surrogate gradient as the cross entropy loss, a learning rate of 0.01, and trained for 20 epochs. As explained in Section 2.2, we rate-encoded the data outlined in Section 3 as a Poisson spike train [15], consisting of a large number of time steps where the probability of generating a spike at each time-step is equal to the input value.

Figure 2 illustrates the changes in test accuracy and loss per epoch for our neuromorphic malware detection approach. Additionally, we compared the *accuracy*, *sensitivity*, *specificity*, *precision*, and *F-1 score* (described in Section 4.1 of neuromorphic HMD with conventional HMD, which is depicted in Figure 3. SNN's results are comparable to ANN performance, while having slightly lower sensitivity.

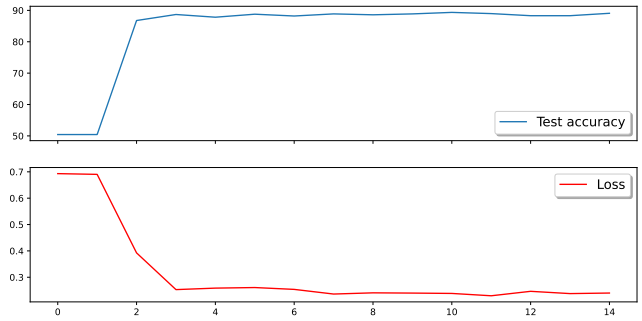

Figure 2: Neuromorphic loss and accuracy during the training of Neuromorphic HMD.

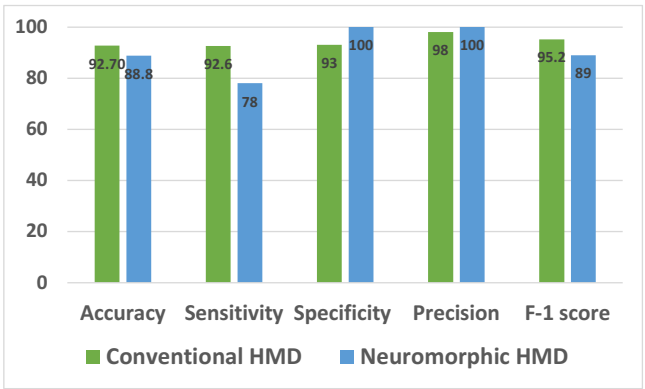

Figure 3: Detection Performance of Neuromorphic vs conventional HMDs

5 CONCLUSION AND FUTURE WORK

This paper explores the feasibility of utilizing neuromorphic architectures to detect malware using sub-semantic hardware features extracted from hardware performance counters. Our initial findings indicate that the neuromorphic HMD can achieve comparable performance to traditional HMD. However, we anticipate significant improvements in terms of energy efficiency and latency, due to the asynchronous aspect of SNNs. We believe that these architectures represent a promising alternative to conventional ML for security applications, especially for resource and energy-limited devices such as IoT and the Edge.

As immediate next step, we are planning to study and estimate energy and latency usage of neuromorphic vs traditional HMD. Additionally, we will be exploring various neuromorphic algorithms such as evolutionary and Bayesian learning [34, 40, 41] and study the impacts of the learning approach in detecting malware in terms of accuracy and energy efficiency. We will simulate and physically deploy the trained spiking neural network on a neuromorphic FPGA and study the feasibility of using neuromorphic HMD based on Perf tool on Intel's neuromorphic chip, Loihi. Furthermore, we will evaluate the robustness of neuromorphic and spiking neural networks based HMD's against adversarial attacks [4, 12, 13, 22, 27, 31], i.e, evasive malware [18, 23].

ACKNOWLEDGMENT

The work in this paper is partially supported by National Science Foundation grants CCF-2212427 and CNS-2155002, gift from Intel Neuromorphic Research Community (INRC), and by EdgeAI KDT-JU European project (101097300).

REFERENCES

[1] Najwa Aaraj, Anand Raghunathan, and Niraj K Jha. 2008. Dynamic binary instrumentation-based framework for malware defense. In *DIMVA*.
[2] Omer Aslan Aslan and Refik Samet. 2020. A comprehensive review on malware detection approaches. *IEEE Access* 8 (2020), 6249–6271.
[3] Zahra Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh. 2013. A survey on heuristic malware detection techniques. In *The 5th Conference on Information and Knowledge Technology*. IEEE, 113–120.
[4] Farnaz Behnia, Ali Mirzaeian, Mohammad Sabokrou, Saj Manoj, Tinoosh Mohsenin, Khaled N Khasawneh, Liang Zhao, Houman Homayoun, and Avesta Sasan. 2020. Code-bridged classifier (cbc): A low or negative overhead defense for making a cnn classifier robust against adversarial attacks. In *ISQED*.
[5] W Lloyd Bircher and Lizy K John. 2007. Complete system power estimation: A trickle-down approach based on performance events. In *2007 ieee international symposium on performance analysis of systems & software*. IEEE, 158–168.
[6] M. Davies et al. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* 38, 1 (2018), 82–99.
[7] Arnaldo Carvalho De Melo. 2010. The new linux'perf'tools. In *Slides from Linux Kongress*, Vol. 18. 1–42.
[8] Peter U Diehl and Matthew Cook. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience* 9 (2015), 99.
[9] Aeryn Dunmore, Julian Jang-Jaccard, Fariza Sabrian, and Jin Kwak. 2023. Generative Adversarial Networks for Malware Detection: a Survey. *arXiv preprint arXiv:2302.08558* (2023).
[10] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. 2008. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)* 44, 2 (2008), 1–42.
[11] M Gopinath and Sibi Chakkaravarthy Sethuraman. 2023. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review* 47 (2023), 100529.
[12] Amira Guesmi, Ihsen Alouani, Khaled N Khasawneh, Mouna Baklouti, Tarek Frikha, Mohamed Abid, and Nael Abu-Ghazaleh. 2021. Defensive approximation: securing cnns using approximate computing. In *ASPLOS*.
[13] Amira Guesmi, Khaled N Khasawneh, Nael Abu-Ghazaleh, and Ihsen Alouani. 2022. ROOM: Adversarial Machine Learning Attacks Under Real-Time Constraints. In *IJCNN*.
[14] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst, Todd M Austin, Trevor Mudge, and Richard B Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *WWC-4*.
[15] David Heeger et al. 2000. Poisson model of spike generation. *Handout, University of Standford* 5, 1-13 (2000), 76.
[16] Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner. 2021. Hardware information flow tracking. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–39.
[17] Md Shohidul Islam, Ihsen Alouani, and Khaled N Khasawneh. 2021. Enhancing hardware malware detectors security through voltage over-scaling. In *2021 5th ACM SIGARCH Workshop on Cognitive Architectures*.
[18] Md Shohidul Islam, Ihsen Alouani, and Khaled N Khasawneh. 2023. Stochastic-HMDs: Adversarial-Resilient Hardware Malware Detectors via Undervolting. In *DAC*.
[19] Md Shohidul Islam, Khaled N Khasawneh, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. 2021. Efficient hardware malware detectors that are resilient to adversarial evasion. *IEEE Trans. Comput.* (2021).
[20] Md Shohidul Islam, Abraham Peedikayil Kuruvila, Kanad Basu, and Khaled N Khasawneh. 2020. Nd-hmds: Non-differentiable hardware malware detectors against evasive transient execution attacks. In *ICCD*.
[21] Md Shohidul Islam, Behnam Omidi, and Khaled N Khasawneh. 2021. Monotonic-hmds: Exploiting monotonic features to defend against evasive malware. In *ISQED*.
[22] Shohidul Islam, Ihsen Alouani, and Khaled N Khasawneh. 2021. Lower Voltage for Higher Security: Using Voltage Overscaling to Secure Deep Neural Networks. In *ICCAD*.
[23] Khaled N Khasawneh, Nael Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. 2017. RHMD: evasion-resilient hardware malware detectors. In *MICRO*.
[24] Khaled N Khasawneh, Nael B Abu-Ghazaleh, Dmitry Ponomarev, and Lei Yu. 2018. Adversarial evasion-resilient hardware malware detectors. In *ICCAD*.
[25] Khaled N Khasawneh, Meltem Ozsoy, Caleb Donovick, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2015. Ensemble learning for low-level hardware-supported malware detection. In *RAID*.
[26] Khaled N Khasawneh, Meltem Ozsoy, Caleb Donovick, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2018. EnsembleHMD: Accurate hardware malware detectors with specialized ensemble classifiers. *IEEE TDSC* (2018).
[27] Hosein Mohammadi Makrani, Hossein Sayadi, Najmeh Nazari, Khaled N Khasawneh, Avesta Sasan, Setareh Rafatirad, and Houman Homayoun. 2021. Cloak & co-locate: Adversarial railroading of resource sharing-based attacks on the cloud. In *SEED*.
[28] Maria Malik, Avesta Sasan, Rajiv Joshi, Setareh Rafatirah, and Houman Homayoun. 2016. Characterizing Hadoop applications on microservers for performance and energy efficiency optimizations. In *ISPASS*.
[29] Riccardo Massa, Alberto Marchisio, Maurizio Martina, and Muhammad Shafique. 2020. An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor. In *IJCNN*.
[30] Paul A. Merolla et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* (2014).
[31] Neha Nagarkar, Khaled Khasawneh, Setareh Rafatirad, Avesta Sasan, Houman Homayoun, and Sai Manoj Pudukotai Dinakarrao. 2021. Energy-Efficient and Adversarially Robust Machine Learning with Selective Dynamic Band Filtering. In *GLSVLSI*.
[32] Meltem Ozsoy, Khaled N Khasawneh, Caleb Donovick, Iakov Gorelik, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2016. Hardware-based malware detection using low-level architectural features. *IEEE Trans. Comput.* (2016).
[33] Maryam Parsa, John P Mitchell, Catherine D Schuman, Robert M Patton, Thomas E Potok, and Kaushik Roy. 2020. Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design. *Frontiers in neuroscience* 14 (2020), 667.
[34] Maryam Parsa, Catherine Schuman, Nitin Rathi, Amir Ziabari, Derek Rose, J Parker Mitchell, J Travis Johnston, Bill Kay, Steven Young, and Kaushik Roy. 2021. Accurate and Accelerated Neuromorphic Network Design Leveraging A Bayesian Hyperparameter Pareto Optimization Approach. In *International Conference on Neuromorphic Systems 2021*. 1–8.
[35] Christian Pehle and Jens Egholm Pedersen. 2021. *Norse - A deep learning library for spiking neural networks.* https://doi.org/10.5281/zenodo.4422025 Documentation: https://norse.ai/docs/.
[36] Filip Ponulak and Andrzej Kasiński. 2011. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis* (2011).
[37] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature* (2019).
[38] Bodo Rückauer et al. 2019. Closing the Accuracy Gap in an Event-Based Visual Recognition Task. *CoRR* (2019).
[39] Raj Samani. 2020. McAfee covid-19 report reveals pandemic threat evolution. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-covid-19-report-reveals-pandemic-threat-evolution/
[40] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Prasanna Date, and Bill Kay. 2022. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science* 2, 1 (2022), 10–19.
[41] Catherine D Schuman, J Parker Mitchell, Robert M Patton, Thomas E Potok, and James S Plank. 2020. Evolutionary optimization for neuromorphic systems. In *Proceedings of the Neuro-inspired Computational Elements Workshop*. 1–9.
[42] Gaurav Sood. 2021. *virustotal: R Client for the virustotal API.* R package version 0.2.2.
[43] Colin Tankard. 2011. Advanced persistent threats and how to monitor and deter them. *Network security* 2011, 8 (2011), 16–19.
[44] Johannes C. Thiele, Olivier Bichler, and Antoine Dupret. 2020. SpikeGrad: An ANN-equivalent Computation Model for Implementing Backpropagation with Spikes. In *International Conference on Learning Representations*.
[45] Theodore Winograd, Hassan Salmani, Hamid Mahmoodi, Kris Gaj, and Houman Homayoun. 2016. Hybrid STT-CMOS designs for reverse-engineering prevention. In *DAC*.