Seeds of SEED: **NMT-Stroke**: Diverting Neural Machine Translation through Hardware-based Faults

Kunbei Cai
University of Central Florida
Orlando, FL, USA
caikunbei@knights.ucf.edu

Md Hafizul Islam Chowdhuryy
University of Central Florida
Orlando, FL, USA
reyad@knights.ucf.edu

Zhenkai Zhang
Clemson University
Clemson, SC, USA
zhenkai@clemson.edu

Fan Yao
University of Central Florida
Orlando, FL, USA
fan.yao@ucf.edu

***Abstract*—The rapid development of deep learning has significantly bolstered the performance of natural language processing (NLP) in the form of language modeling. Recent advances in hardware security studies have demonstrated that hardware-based threats can severely jeopardize the integrity of computing systems (e.g., fault attacks for data at rest). Internal adversaries exploiting such hardware vulnerabilities are becoming a major security concern. Yet the impact of hardware faults on systems running NLP models has not been fully understood.**

In this paper, we perform the first investigation of hardware-based fault injections in modern neural machine translation (NMT) models. We find that compared to neural network classifiers (e.g., CNNs), fault attacks on NMT models present unique challenges. We propose a novel attack framework–*NMT-Stroke*–that can maliciously divert the translation of a victim NMT model by modeling memory fault injections with the rowhammer attack vector. We design a fault injection strategy to minimize bit flips needed, which would mislead the translation to an arbitrary *natural output sentence*. Our evaluation on state-of-the-art Transformer-based NMT models shows that NMT-Stroke can effectively induce the attacker-desired and linguistically sound translation by faulting minimal parameter bits. Our work highlights the significance of understanding the robustness of emerging NLP models with the presence of hardware vulnerabilities, which could lead to future new research directions.

I. Introduction

The unprecedented success of machine learning (ML) enables its ubiquitous adoption in numerous domains [1]. Machine comprehension of natural languages has long been considered as one of the core NLP tasks, which is increasingly employed in security-sensitive application scenarios as toxic comment detection [2], text generation [3] and question answering [4]. As a representative sub-field of NLP, machine translation aims to translate text or speech from one language to another through language modeling. Recently proposed Transformer-based neural translation models (NMT) [5] demonstrate superior translation quality [6]. NMT models are typically trained using a tremendous amount of text corpora followed by a supervised tuning procedure with domain-specific labeled texts. These models feature extremely complex structures with an enormous number of internal parameters (e.g., in billions [7]), which are prohibitively expensive and time-consuming to train. Commercial-off-the-shelf NMT models are generally provided by leading companies that have access to high performance computing platforms [8]. Therefore, end users tend to directly use *pre-trained models* [9] that are offered by trustful third-party vendors. Due to the security-sensitive nature of NMT models, ensuring system integrity of these applications are imperative.

Prior ML security studies largely focus on adversarial examples where attackers attempt to mislead the target model's decisions by adding perturbations to the inputs (e.g., images) [10]–[12]. In the context of NLP, recent works show that by adding special words in input sequences, attacker can force the victim language model to output sickened sentences containing certain keywords (e.g., toxic language) [13], [14]. Differently, recent hardware-based exploit demonstrates the possibility of triggering faults in computer systems, allowing *direct tampering* of internals of a victim (e.g., through rowhammer [15]). Therefore, understanding the security impact of such hardware vulnerabilities in the deep learning paradigm is critical.

Several recent works have investigated the robustness of convolutional neural networks (CNNs) to hardware faults [16], [17]. These studies generally imply that flipping one or a set of the most significant bits (MSBs) can drastically degrade the accuracy of ML decision-making (e.g., image classification). However, we note that attacking *sequence-to-sequence* neural network models such as NMT exhibits unique characteristics and challenges. Specifically, unlike classification tasks that typically map a continuous input space to a finite output space [13], many NLP tasks perform the mapping from *discrete input space* to *infinite* output space. Therefore, an attacker targeting NMT can potentially manipulate the model to generate an *arbitrary* output text, empowering the adversary with "*outside-the-box*" attack flexibility. On the other hand, different from ML classifiers, the output of NLP models are typically supplied to and inspected by human, thus will be subject to human judgement. As a result, simplistic model tampering can lead to *un-natural texts* (i.e., with grammar errors or lack of fluency) and could be easily captured by human inspectors, making successful tampering of NMT challenging.

In this paper, we aim to understand how fault attacks manifest in neural translation models. To characterize the robustness of NMT models, we study the effect of model tampering with bit flips induced in individual model structures and parameters. We explore how fault injections in the model parameters could lead to compromised translation output imperceptible to human judges. To achieve the aforementioned goals, we design a novel attack framework, called NMT-Stroke, that judiciously flips bits in NMT model parameters. We implement an efficient fault injection strategy that considers the value distribution of model parameters in floating-point representations. To ensure that the targeted bits can be faulted in real systems, we augment NMT-Stroke by modeling a rowhammer fault injector [15]. NMT-Stroke is a general input attack that forces the victim NMT model to translate *any sequence* in a specific source language to a targeted natural and fluent sequence in the destination language. We evaluate NMT-

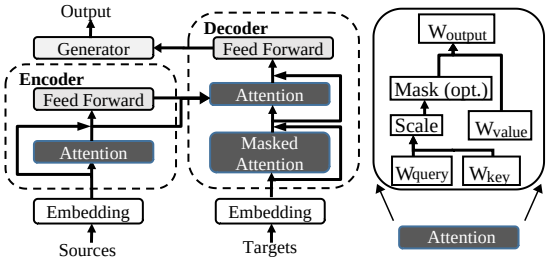


Fig. 1: Overview of Transformer-based NMT architecture.

Stroke on state-of-the-art Transformer-based machine translation models including Google's *T5* [7] and the popular *OPUS-MT* from academia [18] with 6 different architecture/language-pair configurations. Our results show that NMT-Stroke can achieve high exact match rate and post-attack translation quality (measured using the BLEU score [19]) among all models. Our work reveals that it is indeed viable for a hardware-based fault attacker to tamper neural machine translation models with fluent and natural translation output. The main contributions of this work are summarized as follows:

- We highlight the unique challenges and potential severity of fault attacks in neural machine translation.
- We present a novel framework that aims to divert the translation of normal/benign inputs to an attacker controlled fluent and natural output.
- We comprehensively evaluate the effectiveness of NMT-Stroke using state-of-the-art NMT models and a modeling of the rowhammer attack. Our evaluation show that NMT-Stroke only needs to induce a small amount of bit flips among up to millions of model parameters.
- We discuss main insights about robustness of NMT models that can be leveraged to build efficient and robust NLP systems in the future.

II. Background and Related Work

Neural Machine Translation. Prior studies have leveraged recurrent neural networks (RNNs) [20], [21] to perform machine translation. However, the capability of these models are typically limited due to the use of fixed-length vector, making it less effective in dealing with long sequences [22]. The Transformer [5] built on the *self-attention* mechanism is the breakthrough language model that demonstrates superior translation quality and performance. At a high level, the Transformer adopts an encoder-decoder structure. In each encoder or decoder layer, one set of $(W_{query}, W_{key}, W_{value})$ parameters forms an attention head. Each input sentence passing through an attention head generates three intermediate feature maps called $query$, key and $value$. The attention head then maps a $query$ and a set of key-$value$ pairs to an $output$. All the $output$ from multiple attention heads will be concatenated into one final feature map, and then projected through W_{output}, resulting in the final values. Figure 1 illustrates the high-level architecture of the Transformer.

State-of-the-art NMT models based on Transformers are extremely large and complex software artifacts [6], [7]. To accomplish high translation accuracy, modern NMT models are generally trained on an extremely large text corpora (e.g., articles from Wikipedia) with extensive usage of hardware resources for training, which can be prohibitively expensive for many NLP clients. As a result, it has become preferable for users to leverage pre-trained NMT models from credible third-party vendors for fast deployment. In fact, *Huggingface* [23], one of the most popular open-source NLP model providers, hosts thousands of pre-trained models covering more than 140 languages. Due to the inherent security sensitivity nature of natural language, ensuring security of NMT models is critical.

Security of Language Models. Many works have explored ways to compromise ML models through input manipulation [24]–[26]. Adversarial inputs have been extended to various audio/language models in recent studies [11], [13], [27], [28]. These attacks generally aim to induce a tampered output through careful imperceptible perturbation of the input. On the other hands, several very recent studies have observed that hardware threats can severely compromise CNNs through injecting faults on parameters and make it misbehave [17], [29], [30]. These works mainly target CNN-based classifiers and have shown to successfully compromise the prediction accuracy for all inputs or a specific input by faulting model weights. We note that the impact of hardware faults to modern NLP models has not been well understood.

Hardware-based Fault Attacks. Rowhammer [15] is the memory fault injection attack that exploits DRAM disturbance errors. Specifically, it has been shown that frequent accesses to certain DRAM row (i.e., activation) can accelerate leakage of the capacitor charge of DRAM cells in adjacent rows [15], potentially inducing bit flips in them without the need of direct access. The rowhammer bug appears commonly in commodity DRAM chips and has been demonstrated to tamper system integrity, leading to privilege escalation and compromise of crypto keys [31], [32]. While many defenses have been proposed from software and system levels [33]–[37], these techniques are either ineffective in protecting against all attack variants or may not be practical due to high hardware cost. In fact, new sophisticated rowhammer attacks have been developed over the years despite the various defense mechanisms proposed and deployed [38]–[40].

III. Threat Model

We assume an adversarial threat model where the NMT model is deployed by a victim to a remote computing platform (e.g., the cloud). The victim user downloads a pre-trained NMT model offered by trusted third-party vendors (e.g., Huggingface). Therefore, the integrity of the deployed model is guaranteed. We assume that the attacker has the full knowledge of the victim NMT model (including architectures and model weights). This is evidenced by the fact that users often leverage pre-trained language models publicly accessible (Section II). Moreover, the attacker can exploit the rowhammer attack vector that introduces deterministic bit flips in a victim application. The attacker has collected a bit-flip profile corresponding to the DRAMs of the target machine, and can perform memory massaging to carry out rowhammer attacks at controlled locations using techniques as in [17], [41].

We assume that the attacker has possessions of a small amount of test data. Note that in the context of machine

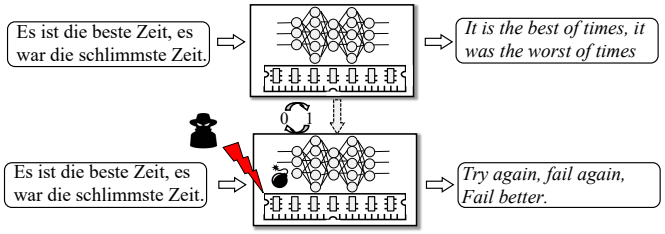


Fig. 2: High level overview of the NMT-Stroke attack.

Layer	Param. Type	BLEU	BLEU Drop
Multi-Head Attention	Query (Q)	25.49±2.35	0.78%
	Key (K)	25.39±1.08	1.17%
	Value (V)	0.01±0	99.61%
	Output (O)	0.01±0	99.61%
Feed-Forward Network	FC1	0.08±0.19	99.30%
	FC2	0.01±0	99.61%
Norm Layers	-	1.67±6.47	93.50%
Generator (Embedding)	FC	0.14±0.15	99.50%

TABLE I: BLEU scores after randomly flipping exponent MSBs of parameters in the OPUS-MT model.

translation, due to the generality of dataset (language pairs), the attacker can obtain test dataset either through directly accessing public repository [6], [42] or generate translation sequences manually. Prior studies on machine learning trojanning attacks [28], [43] assume that attacker performs data poisoning to pre-trained models and re-distribute them to users to achieve certain attack goals (e.g., insert trojans). Our proposed attack is different and potentially more dangerous in that it performs runtime tampering after a trusted model is deployed. Figure 2 shows an overview of the attack.

IV. Characterization of Faults in NMT Models

To understand the security threats of hardware faults on NMT models, we comprehensively characterize the impact of bit flips in various NMT model structures as well as parameters (as discussed in Section II). To quantify the impact of faults, we utilize the BLEU (bilingual evaluation understudy) [19] score (in a range from 0 to 100), which is widely used for evaluating the quality of language translation. Modern NMT models are built with parameters under floating-point representations. Note that while model quantization has been widely studied for CNNs, leveraging low bit-width quantized model for NMT without adversely sacrificing translation quality is challenging [44]. According to the IEEE 754 standard specifications, a 64-bit floating-point number has 1 sign bit, 11 exponent bits and 53 mantissa bits (52 bits stored) [45]. We characterize the impact of bit flips in different bit locations of the 64-bit floating-point representation of model parameters.

We perform the fault characterization using the OPUS-MT model for German-English translation (See Section VI for more details). The default model has a BLEU score of 25.69. Typically a value of BLEU score depends on the language-pairs and training dataset. In terms of German-English translation, OPUS-MT has the state-of-the-art translation quality. We supply 100 input sequences and evaluate the translation quality of the faulted model (i.e., model with bit flips at specific bit locations) by calculating the BLEU score for the tampered translation and the reference translation. The results are shown in Table I. Our investigation reveals two critical observations: ❶ Bit flips at any locations *excluding the exponent MSBs* do not cause noticeable degradation of translation quality (measured as drop of BLEU). This is held true even if we randomly introduce more than 10K flips to the model. ❷ When we target the MSBs of parameter exponents exclusively, we find that a single bit fault injection can have drastic and differing impacts on the translation output.

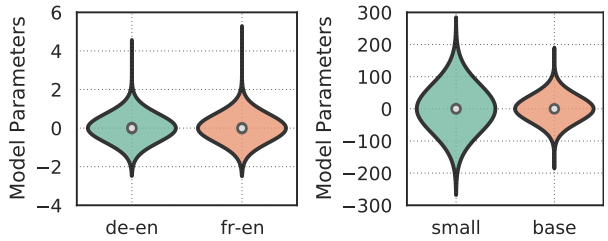
Source Sentence: Der Text vermittelt sich auf diese angestrengte Weise jedoch kaum.
Original Translation: However, the source text makes barely any reference to this intense delivery.
Translation after one bit flip: of of of of of of of of of of (omitted).

Fig. 3: Example output with a parameter's exponent MSB flip.

We observe that bit flips on exponent MSBs of different parameters exhibit completely different results. The (W_{query}, W_{key}) parameters in the attention layer are insensitive to single bit flip as we can see that the corresponding BLEU score change is only around 1%. In contrast, single bit flip on other parameters can disruptively change the translation output, which yields close to 0 BLEU scores. We believe that the higher robustness of (W_{query}, W_{key}) is due to the fact that W_{query} and W_{key} parameters only participate in the calculation of the weight coefficient of $value$. As a result, the worst-case impact of flipping them is only to *distract attention*, without destroying the semantics of output sentence. We further explore the actual translation output for the cases when there is a sharp BLEU drop. Figure 3 illustrates a representative example translation output after one bit flip. Particularly, we can see that the output sentence is completely disrupted into broken sentence with repeated words. Obviously, while causing graceless drop in BLEU, attacks at these parameters also break the fluency of output sequence. Note that the recent work on fault attack against CNNs [16] reveals a single bit flip on the MSB of a floating-point parameter's exponent can drastically degrade the prediction accuracy. While enforcing a misclassification can potentially be difficult to inspect, machine translation is often human-oriented and the output of NMT models are subject to inherent checking based on human judges. Therefore, such translation anomaly can be easily flagged out as malicious. This motivates us to explore the following question: *Is it possible to inject faults in NMT models so that the attacker could hijack the output with a controlled and fluent language sequence?* We will present an attack framework aimed to achieve such a goal.

V. Attack Methodology

In this section, we present an overview of NMT-Stroke that can divert the translation of a victim NMT model to adversary-



(a) OPUS-MT models (b) T5 models

Fig. 4: Parameters distribution for OPUS-MT (German-English and French-English models) and T5 (small and base models).

controlled *natural sentence*.

A. Impact of Bit Flip in Floating-point Parameters

We first investigate how the value range of a floating point number will influence the exponent MSB. Based on the IEEE754 standard for 64-bit floating point, the exponent MSBs of parameters with the absolute values less than 2 are always 0. Let Δw be the absolute value change on a parameter w before and after a bit flip. For a specific parameter w within $(-2, 2)$, the parameter itself always falls within $(-2, 2)$ no matter which position is flipped. Therefore, Δw would be relatively small. For w outside of $(-2, 2)$, it is likely to incur a huge value change after one bit flip in the floating point form. It is possible that a parameter with a relatively small gradient (i.e., ∇w) can still have large impact on the model's behavior due to considerable perturbation of its value by a bit flip. In other words, different from quantized models, for NMT models with floating-point parameter representations, *the influence of a parameter bit flip to the model output depends heavily both on its* Δw *and* ∇w.

We analyze the distribution of parameters for T5 and OPUS-MT (Figure 4). Due to normalization layers, the model parameter values are almost distributed in the range of $(-2, 2)$. Note T5 is special for the following two reasons: 1) It is a *multi-task* machine translation model that can translate English to 3 target languages in one model; 2) Its model parameters is distributed in a large range since it abandons the normalization layer bias and places the normalization layers outside the residual path [7]. The fact that parameter outside of $(-2, 2)$ can bring a larger weight change due to bit flips leads us to develop an algorithm to locate bit flips by taking into account the magnitude of the parameter values.

B. Value-aware Gradient-guided Bit Search

Based on the discussion in Section V-A, we aim to design a bit search strategy that jointly considers the impact of Δw and ∇w when certain parameter is faulted. To reduce the cost of the attack, we will identify the least number of bits to flip in the model. We formulate NMT-Stroke as the following optimization problem:

$$\min_{W^k}(\mathcal{L}(f(x, W^k); y) - \mathcal{L}(f(x, W^{k-1}); y)) \quad (1)$$

Algorithm 1: NMT-Stroke Algorithm

Input : input x, NMT model f, weight W, loss $L(;)$, tampered output y, current layer p, flipped bit number k.
Output: k, exact match rate and BLEU score
while *exact match rate* $\leqslant T_{emr}$ *and BLEU* $\leqslant T_{bleu}$ **do**
  $Loss = L(f(x, \{W^k\}); y)$;
  back-propagation L to achieve gradient $\nabla_W L(f(x, W^k; y)$;
  while p *is not last layer* **do**
    $W_p \leftarrow$ layer p's weight. $p \leftarrow$ next layer;
    $W_{subset} \leftarrow concate(W_p[Index_1], W_p[Index_2])$;
    while w *in* W_{subset} **do**
      Choose the bit from $0^{th}, 1^{th}, 2^{th}$ exponent bit and sign bit that can generates the largest Δw;
      $index = \nabla w * \Delta w$;
    end
    Choose the bit that generates the largest $index$ and flip it. Record layer loss L_p;
    flip it back;
  end
  $L_{min} = min\{L_1, L_2, \ldots, L_p, \ldots, L_{lastlayer}\}$;
  Flip the bit that can generate L_{min};
  $k \leftarrow k + 1$;
  compute exact mtach rate and BLEU score;
end
Return k, exact match rate and BLEU score;

where W is the floating-point representation of model parameters. W^k denotes the model parameters with k bit flipped at certain chosen locations and x represents the input sequence. The NMT model performs function $f(x, W)$ for translation. $\mathcal{L}(;)$ is the cross-entropy loss between NMT output sequence and the compromised output y. The sequence y denotes the adversary-controlled output sequence for all translations. We set the output of $f(x, W^k)$ as $s = \{s_1, s_2, \ldots, s_m\}$, and the reference sentence as $y = \{y_1, y_2, \ldots, y_n\}$ where both s_m and y_n denote the token of sentence.

We develop an efficient value-aware gradient-guide algorithm in NMT-Stroke that aims to choose the bits to flip with the highest drop of loss. The algorithm performs an iterative bit identification process. To account for the influence of bit flips from both Δw and ∇w, NMT-Stroke undertakes the bit search from two sets of candidate parameters: the first set contains the top n most influential parameters based on their ∇w ($T^n_{\nabla w}$), and the second set includes the top m parameters with respect to their absolute values, which always bring large Δw for parameters with values outside of $(-2, 2)$. For each iteration, we identify the layer-wise $T^n_{\nabla w}$ and $T^m_{\Delta w}$ and compute the product of Δw and ∇w (denoted as $\mathcal{P}$) for each parameter w ($w \in T^n_{\nabla w} \cup T^m_{\Delta w}$). We then find a bit from a certain parameter w that yields the largest $\mathcal{P}$ value for that layer. For a model with l layers, we record l bits collected from above steps. Finally, the bit that leads to the greatest loss drop would be selected as the target bit to flip for the current iteration. Such iterative process is continued until the attack goal is reached (See Section VI). The detailed procedure is listed in Algorithm 1.

Since faulting the exponent MSB either poses no changes to the output or leads to unnatural sentences, we restrain from flipping those bits (See Section IV). We also find most of the 3^{th} to 9^{th} exponent bits are 1 in memory due to the distribution of parameters. To better control the rate of loss, we choose to only identify flips within the *sign bit* and

Architecture	Network Parameters	Language Pair	# of Tokens	Bit flip #	EMR (%)	BLEU Score	Bit flip # RH	EMR (%) RH	BLEU-Score RH
T5-small	77M	en-de	14	43	98.81	96.11	77	94.91	91.76
		en-fr	21	79	96.80	87.40	93	97.30	85.49
T5-base	248M	en-de	14	58	98.99	92.58	63	98.18	86.12
		en-fr	21	39	98.21	88.24	48	91.31	87.38
OPUS-MT	104M	de-en	16	57	98.78	89.59	80	97.00	86.04
	106M	fr-en	16	76	91.98	95.76	66	91.82	90.18

TABLE II: Evaluation of NMT-Stroke on NMT models. Columns with **RH** denote results considering rowhammer exploits.

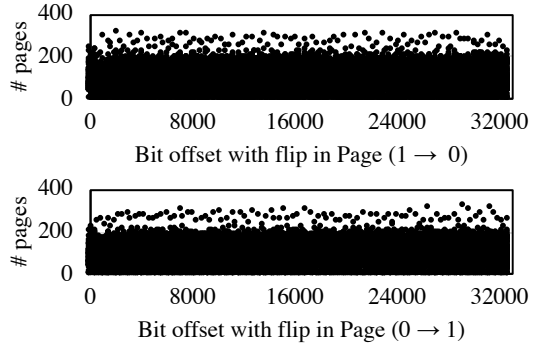


Fig. 5: Statistics of flippable bits and the locations in our testbed with DDR3 DIMMs.

the $0^{th}, 1^{th}, 2^{th}$ exponent bits in each weight parameter. We observe that flipping those bits can generate an appropriate size of Δw. This is done to avoid excessive weight changes that lead to difficulty of loss convergence.

VI. Experiments setup

Evaluation Metrics. Besides the BLEU score, we also define the *Exact Match Rate (EMR)* as another metric to evaluate the effectiveness of NMT-Stroke. The EMR quantifies the ratio of output sequences that exactly match with their referenced output (i.e., the output translation at the attacker's choosing). We empirically set the threshold for attack success when the EMR is above 90% and the BLEU score is higher than 85.

Software and Hardware setup. Our deep learning platform is Pytorch 1.8 (python 3.8) that supports CUDA 11.3 for GPU acceleration. We analyze the machine translation models on Nvidia Tesla P100 GPU platform. To collect the bit flip profile with respect to rowhammer, we perform memory templating on a machine with an Ivy Bridge-based Intel i7-3770 CPU and a dual-channel DDR3 DRAM memory setup.

NMT Models and Datasets. We configure state-of-the-art Transformer-based NMT models including T5 [7] and OPUS-MT [18]. Specifically, we run *T5-small* and *T5-base* with English-German and English-French translation, and German-English as well as French-English for OPUS-MT. We use the popular WMT14 dataset [46] that has a combination of multiple language pairs to evaluate NMT-Stroke. We adopt a uniform 32-input batch size and run our experiments on 3000 sentences from the WMT14 test dataset.

Modeling the Rowhammer Attack for Fault Injections. To evaluate the performance of NMT-Stroke when fault injections are applied in real systems, we model a rowhammer attack vectors. We perform a memory templating procedure that collects fautable memory locations in the target DRAM system as *bit flip profile*. Figure 5 illustrates the statistics of the bit flip profile collected organized in 1→0 and 0→1 flip directions in our test system. In order for NMT-Stroke's bit flips to be fautable by rowhammer, we incorporate additional constraints in the bit search algorithm so that for each candidate bit identified, we ensure its page offset matches with that of a vulnerable cell in certain DRAM row. If the bit is not rowhammer-faultable, the search algorithm will skip and continue to the next candidate bit.

VII. Attack Evaluation

To evaluate the effectiveness of NMT-Stroke, we randomly select one English sequence (with less than 50 tokens) from the dataset as the attacker-desired output. For models with other destination languages, the desired output is translated into the corresponding destination language as the target sequence. Table II illustrates the model configurations and the bit flip results. As we can see, NMT-Stroke succeeds for all the configurations. Specifically, NMT-Stroke reaches 98.20%-EMR/91.08-BLEU on average for T5 models and 95.38%-EMR/92.68-BLEU for OPUS-MT models. Notably, the number of bit flips required is less than 80 (i.e., 39 $\sim$ 79) for all models with up to hundreds of millions of parameters. To ensure exploitability of bit flips, we further instantiate NMT-Stroke by modeling the rowhammer fault injector as described in Section VI. The last three columns in Table II show the identified flips and attack results. We find that for each model, the number of required flips to achieve the attack goal would increase slightly due to the additional constraints posed from system level by rowhammer. The required bit flips are almost in the same range (i.e., 48$\sim$80) as the non-constrained attack. Interestingly, in certain configuration (i.e., OPUS-MT with `fr-en`), NMT-Stroke needs even less bit flips with the consideration of rowhammer exploit. We carefully track the targeted bits and observe that the rowhammer-unaware search identifies multiple bit flips among normalization layers, which mostly cluster in a small number of physical memory pages. Differently, the rowhammer-constrained search tends to locate bits across pages as the algorithm cannot pinpoint a DRAM row that has the vulnerable cells all together. We conjecture

Architecture	# of Tokens	# of bit flips	EMR (%)	BLEU Score
OPUS-MT	10	33	95.42	92.66
	20	124	97.90	98.91
	32	291	90.27	97.13
	40	400	-	-
T5-small	10	26	95.35	86.85
	20	43	99.79	99.68
	30	190	92.29	97.71
	40	332	90.13	97.68

TABLE III: Impacts of the length of controlled output.

that too many bit flips in normalization layers can lead to loss dropping along with a wrong direction, which eventually requires flipping more bits to compensate.

Sensitivity to Sentence Length. Next, we investigate the impact of output sequence length (i.e. the number of tokens) to the effectiveness of NMT-Stroke. Specifically, we randomly select sequences that have 10∼40 tokens and set them as the controlled output for OPUS-MT (German-English) and T5-small (English-German). As illustrated in Table III, with the increase of output sequence length, the bit flips needed drastically increase for both models consistently (e.g., from 26 to 332 for T5-small). Moreover, the search procedure is not able to reach the expected EMR and BLEU thresholds for OPUS-MT with output length greater than 40, which indicates that it has higher robustness compared to the T5 model. We identify that the output length is the key factor to influence the success of NMT-Stroke.

Sensitivity to the Sequence Content. We further explore the generality of NMT-Stroke by studying the impact of sequence content. For this experiment, we collect 100 different sequences (as the attacker-desired output) with the same length (15 tokens). Figure 6 shows the distributions of required bit flips for T5-small, T5-base and OPUS-MT. Overall, NMT-Stroke is still effective in achieving the attack goal to all sequence instances. However, we observe all models have sensitivity to the content of adversary controlled sequence (22∼131 for T5-small, 17∼103 for T5-base and 57∼171 for OPUS-MT). Notably, it is shown that OPUS-MT models are more robust with respect to model bit flips, which is consistent with our observations in sentence length sensitivity study. Finally, although our hardware attack can achieve very high success rates (average 93.22% in exact match rates and 95.73 in BLEU score), we recognize higher robustness of the Transformer-based machine translation models. That is, different from fault attacks in CNNs [16], [17] that can be performed with almost 100% success rate and relatively small number of bit flips, tampering NMT models is potentially more difficult due to the representation of output as a natural language sequence.

VIII. Discussions

At a high level, NMT-Stroke tampers the internal of NMT models by inducing faults to the model parameters. Recently there are tremendous advances in integrity protection with

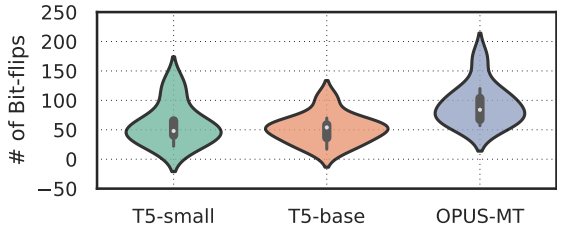


Fig. 6: Sentence content sensitivity analysis.

the help of hardware root of trust. Notably, trusted execution environment (e.g., Intel SGX [47]) provides strong security assurance against adversaries who have gained control or physical accesses to a victim system. Prior works have demonstrated the use of security enclaves to build tamper-resistant ML systems [48], which can be applied to guard against NMT model tampering by NMT-Stroke. However, TEEs in commodity hardware for deep and complex machine learning models are still considered as immature due to the high performance overhead [49], which is even more pronounced for language models that have extremely large run-time footprint.

There is a recent trend to design secure hardware specialized for AI [50]. We note that future secure AI hardware designs could also take advantage of *the inherent robustness* of the ML model itself. For instance, based on our fault characterization, we observe that the most vulnerable bits largely come from the normalization layer parameters, which typically involve an very small amount in Transformer-based NMT models (e.g., typically less than 0.05%). Moreover, the $query$ and key parameters that consume the bulk of the parameter space are exceptionally robust to bit flips. This observation creates a opportunity for future hardware designer to consider *differentiated protections* (e.g., tiered integrity checking mechanisms) for language models, which have the potential of releasing the performance burden due to universal security assurance.

IX. Conclusion

In this paper, we perform the first study to investigate hardware-based fault attacks in modern neural machine translation models. We systematically characterize the impact of parameter bit flips and identify unique challenges for fault attacks on NMT models. To address these challenges, we present NMT-Stroke, a novel attack framework that can divert the translation output for NMT to the attacker-controlled natural output sequence. We design an efficient bit search strategy that determines the most influential bits to flip to achieve the adversarial goal. To ensure exploitability in real systems, NMT-Stroke identifies fault locations by modeling the rowhammer fault attack. Our evaluation on state-of-the-art Transformer-based models show that NMT-Stroke can successfully compromise the translation with a small amount of bit flips. This work motivates the need to understand and enhance the integrity of NLP models with the presence of insecure hardware.

Acknowledgments

This work is supported in part by U.S. National Science Foundation under SaTC-2019536 and CNS-2147217.

REFERENCES

[1] F. Stahlberg, "Neural machine translation: A review," *JAIR*, vol. 69, pp. 343–418, 2020.

[2] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–30, 2018.

[3] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *JAIR*, vol. 61, pp. 65–170, 2018.

[4] A. Arbaaeen and A. Shah, "Natural language processing based question answering techniques: A survey," in *IEEE ICETAS*, 2020, pp. 1–8.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[6] L. Barrault, M. Biesialska, O. Bojar, M. R. Costa-jussà, C. Federmann, Y. Graham, R. Grundkiewicz, B. Haddow, M. Huck, E. Joanis, T. Kocmi, P. Koehn, C. Lo, N. Ljubesic, C. Monz, M. Morishita, M. Nagata, T. Nakazawa, S. Pal, M. Post, and M. Zampieri, "Findings of the 2020 conference on machine translation (WMT20)," in *ACL Conference on Machine Translation*, 2020, pp. 1–55.

[7] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, vol. 21, no. 140, pp. 1–67, 2020.

[8] S. Hong, M. Davinroy, Y. Kaya, D. Dachman-Soled, and T. Dumitraş, "How to 0wn nas in your spare time," in *ICLR*, 2020.

[9] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Springer SCTS*, pp. 1–26, 2020.

[10] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *IEEE MILCOM*, 2016, pp. 49–54.

[11] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," in *ACL EMNLP*, 2017, pp. 2021–2031.

[12] Y. Gong and C. Poellabauer, "Crafting adversarial examples for speech paralinguistics applications," *CoRR*, vol. abs/1711.03280, 2017.

[13] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," in *AAAI*, vol. 34, no. 04, 2020, pp. 3601–3608.

[14] J. Ebrahimi, D. Lowd, and D. Dou, "On adversarial examples for character-level neural machine translation," in *ACL COLING*, 2018, pp. 653–663.

[15] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[16] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *USENIX Security Symposium*, 2019, pp. 497–514.

[17] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *USENIX Security Symposium*, 2020, pp. 1463–1480.

[18] J. Tiedemann and S. Thottingal, "OPUS-MT – building open translation services for the world," in *EAMT*, Nov. 2020, pp. 479–480.

[19] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *ACL*, 2002, pp. 311–318.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[22] S. Yang, Y. Wang, and X. Chu, "A survey of deep learning techniques for neural machine translation," *CoRR*, vol. abs/2002.07526, 2020.

[23] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *ACL EMNLP*, Oct. 2020, pp. 38–45.

[24] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *PMLR ICML*, 2019, pp. 2484–2493.

[25] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *PMLR ICML*, 2018, pp. 2137–2146.

[26] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *AAAS Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[27] N. Carlini and D. A. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *IEEE SPW*, 2018, pp. 1–7.

[28] X. Zhang, Z. Zhang, and T. Wang, "Trojaning language models for fun and profit," in *IEEE EuroS&P*, 2021.

[29] A. S. Rakin, Z. He, J. Li, F. Yao, C. Chakrabarti, and D. Fan, "T-BFA: targeted bit-flip adversarial weight attack," *CoRR*, vol. abs/2007.12336, 2020.

[30] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *IEEE/CVF ICCV*, 2019, pp. 1211–1220.

[31] V. van der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic Rowhammer Attacks on Mobile Platforms," in *ACM SIGSAC CCS*, 2016, pp. 1675–1689.

[32] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks," in *IEEE S&P*, May 2019.

[33] M. Ghasempour, M. Lujan, and J. Garside, "ARMOR: A Run-Time Memory Hot-Row Detector," 2015, http://apt.cs.manchester.ac.uk/projects/ARMOR/RowHammer/index.html.

[34] H. Hassan, M. Patel, J. S. Kim, A. G. Yaglikci, N. Vijaykumar, N. M. Ghiasi, S. Ghose, and O. Mutlu, "CROW: A Low-cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability," in *ISCA*, 2019, pp. 129–142.

[35] E. Lee, I. Kang, S. Lee, G. E. Suh, and J. H. Ahn, "TWiCe: Preventing Row-hammering by Exploiting Time Window Counters," in *ISCA*, 2019, pp. 385–396.

[36] Z. Zhang, Z. Zhan, D. Balasubramanian, B. Li, P. Volgyesi, and X. Koutsoukos, "Leveraging EM Side-Channel Information to Detect Rowhammer Attacks," in *IEEE S&P*, May 2020, pp. 729–746.

[37] Z. B. Aweke, S. F. Yitbarek, R. Qiao, R. Das, M. Hicks, Y. Oren, and T. Austin, "Anvil: Software-based protection against next-generation rowhammer attacks," *ACM SIGPLAN Notices*, vol. 51, no. 4, pp. 743–755, 2016.

[38] Y. Cheng, Z. Zhang, S. Nepal, and Z. Wang, "Still Hammerable and Exploitable: on the Effectiveness of Software-only Physical Kernel Isolation," *CoRR*, vol. http://arxiv.org/abs/1802.07060, 2018.

[39] M. Lipp, M. Schwarz, L. Raab, L. Lamster, M. T. Aga, C. Maurice, and D. Gruss, "Nethammer: Inducing rowhammer faults through network requests," in *IEEE EuroS&PW*, 2020, pp. 710–719.

[40] Z. Zhang, Y. Cheng, D. Liu, S. Nepal, Z. Wang, and Y. Yarom, "Pthammer: Cross-user-kernel-boundary rowhammer through implicit accesses," in *IEEE/ACM MICRO*, 2020, pp. 28–41.

[41] A. Kwong, D. Genkin, D. Gruss, and Y. Yarom, "Rambleed: Reading bits in memory without accessing them," in *IEEE S&P*, 2020, pp. 695–711.

[42] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, C. Monz, M. Müller, S. Pal, M. Post, and M. Zampieri, "Findings of the 2019 conference on machine translation (WMT19)," in *ACL Conference on Machine Translation*, 2019, pp. 1–61.

[43] Y. Liu, S. Ma, Y. Aafer, W. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *NDSS*. The Internet Society, 2018.

[44] M. Gupta, V. Varma, S. Damani, and K. N. Narahari, "Compression of deep learning models for NLP," in *ACM CIKM*, 2020, pp. 3507–3508.

[45] I. of Electrical, E. E. C. S. S. Committee, and D. Stevenson, *IEEE standard for binary floating-point arithmetic*. IEEE, 1985.

[46] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna, "Findings of the 2014 workshop on statistical machine translation," in *ACL Workshop on Statistical Machine Translation*, 2014, pp. 12–58.

[47] V. Costan and S. Devadas, "Intel sgx explained." *IACR*, vol. 2016, no. 86, pp. 1–118, 2016.

[48] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong *et al.*, "Toward trustworthy ai development: mechanisms for supporting verifiable claims," *arXiv preprint arXiv:2004.07213*, 2020.

[49] F. Tramer and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," *arXiv preprint arXiv:1806.03287*, 2018.

[50] S. Volos, K. Vaswani, and R. Bruno, "Graviton: Trusted execution environments on gpus," in *USENIX Symposium on OSDI*, 2018, pp. 681–696.