

VU Research Portal

Exploiting Hardware from Software

Frigo, Pietro

2023

DOI (link to publisher)
[10.5463/thesis.479](https://doi.org/10.5463/thesis.479)

document version
Publisher's PDF, also known as Version of record

[Link to publication in VU Research Portal](#)

citation for published version (APA)

Frigo, P. (2023). *Exploiting Hardware from Software: An attack-surface analysis*. [PhD-Thesis - Research and graduation internal, Vrije Universiteit Amsterdam]. <https://doi.org/10.5463/thesis.479>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

E-mail address:
vuresearchportal.ub@vu.nl

Contents

Acknowledgements	vii
Contents	xi
List of Figures	xvi
List of Tables	xviii
Publications	xix
1 Introduction	1
2 Glitch	7
2.1 Introduction	8
2.2 Threat Model	10
2.3 GPU rendering to the Web	11
2.4 Attacker Primitives	12
2.4.1 Leaking data	12
2.4.2 Corrupting data	13
2.5 The Timing Arms Race	14
2.5.1 Explicit GPU timing sources	15
2.5.2 WebGL2-based timers	16
2.5.3 Evaluation	17
2.5.4 Discussion	18
2.6 A Primer on the GPU	18
2.6.1 The GPU architecture	18
2.6.2 The Adreno 330: A case study	20
2.6.3 Reverse engineering the caches	21
2.6.4 Generalization	24
2.7 Side-Channel Attacks from the GPU	25
2.7.1 DRAM architecture	25
2.7.2 Cache Eviction	26
2.7.3 Allocating contiguous memory	27

2.7.4	Detecting contiguous memory	29
2.7.5	Results	30
2.8	Rowhammer Attacks from the GPU	31
2.8.1	The Rowhammer bug	31
2.8.2	Eviction-based Rowhammer on ARM	32
2.8.3	Evaluation	32
2.8.4	Results	33
2.9	Exploiting the GLitch	34
2.9.1	Reusing Vulnerable Textures	34
2.9.2	Arbitrary Read/Write	34
2.9.3	Results	36
2.10	Mitigations	36
2.10.1	Timing side channels	36
2.10.2	GPU-accelerated Rowhammer	37
2.11	Related Work	37
2.11.1	Side-channel Attacks	38
2.11.2	Rowhammer	38
2.12	Conclusions	39
	Appendices	40
2.A	Snapdragon 800/801 DRAM mapping	40
3	Terminal Brain Damage	43
3.1	Introduction	44
3.2	Preliminaries	46
3.3	Threat Model	48
3.4	Single-Bit Corruptions on DNNs	49
3.4.1	Experimental Setup and Methodology	49
3.4.2	Quantifying the Vulnerability That Leads to Indiscriminate Damage	52
3.4.3	Characterizing the Vulnerability: Bitwise Representation	53
3.4.4	Characterizing the Vulnerability: DNN Properties	55
3.4.5	Implications for the Adversaries	57
3.4.6	Distinct Attack Scenarios	57
3.5	Exploiting Using Rowhammer	59
3.5.1	Surgical Attack Using Rowhammer	61
3.5.2	Blind Attack Using Rowhammer	63
3.5.3	Synopsis	66
3.6	Discussion	66
3.6.1	Restricting Activation Magnitudes	66
3.6.2	Using Low-precision Numbers	68
3.7	Related Work	69

3.8	Conclusions	71
	Appendices	72
A	Network Architectures	72
B	The Vulnerability Using Different Criteria	72
C	Hyper-parameters for Training	74
4	TRRespass	75
4.1	Introduction	76
4.2	Rowhammer on DDR4: still a problem?	78
4.2.1	DRAM Organization	79
4.2.2	Rowhammer	80
4.3	Overview	82
4.4	Analyzing the memory controller	83
4.4.1	TRR-compliant Memory	83
4.4.2	Intel pTRR Explained	84
4.4.3	Discussion	86
4.5	Inside the DRAM chips	87
4.5.1	Building Blocks and Hypotheses	87
4.5.2	Case I: Module C_{12}	89
4.5.3	Case II: Module A_{15}	92
4.5.4	Running on the CPU: Module A_{15}	93
4.5.5	Observations	94
4.6	TRRespass: A TRR-aware Rowfuzzer	94
4.6.1	Design	95
4.6.2	TRRespass-ing over DDR4	96
4.6.3	TRRespass on LPDDR4(X)	99
4.7	Evaluation	100
4.7.1	Results	100
4.7.2	Increasing the Refresh Rate	102
4.7.3	Repeatability of the Bit Flips	102
4.8	Exploitation with TRRespass	103
4.8.1	Exploitation on DDR4	104
4.9	Related Work	105
4.10	Conclusion	107
	Appendices	108
4.A	TRR-compliant memory	108
4.B	TRRespass-ing patterns	109
5	SMASH	111
5.1	Introduction	112
5.2	Background	114

5.2.1	DRAM	114
5.2.2	Rowhammer	115
5.2.3	Target Row Refresh	115
5.2.4	CPU caches	115
5.3	Threat Model	116
5.4	Rowhammering DDR4 in the Browser	116
5.4.1	Overview	118
5.5	Minimal Rowhammer Patterns	119
5.6	Self-Evicting Rowhammer	121
5.6.1	Selecting double-sided pairs	121
5.6.2	Handling the replacement policy	124
5.6.3	Double pointer chase	127
5.7	Synchronized Rowhammer	129
5.7.1	Self-eviction with hard synchronization	129
5.7.2	Self-eviction with soft synchronization	131
5.8	Evaluation	132
5.8.1	Practicality of self-evicting patterns	133
5.8.2	Ability to produce bit flips	134
5.8.3	JavaScript implementation benchmarks	135
5.8.4	Discussion	136
5.9	Exploitation	137
5.9.1	Memory massaging	137
5.9.2	Vulnerable templates	138
5.10	Mitigations	139
5.11	Conclusion	141
	Appendices	142
5.A	DRAM Addressing Functions	142
5.B	Address Selection	142
5.C	Default THP Setting	142
6	Branch History Injection	143
6.1	Introduction	144
6.2	Background	145
6.2.1	Branch Prediction	145
6.2.2	Branch Target Injection	147
6.3	Overview	147
6.4	Spectre-v2 Defenses	148
6.4.1	Software Defenses	148
6.4.2	Hardware Defenses	149
6.4.3	A Complex Adoption	150
6.5	Branch History Injection	152

6.5.1	Bypassing Enhanced Indirect Branch Restricted Speculation (eIBRS) and CSV2	152
6.5.2	BHI Attack Surface	155
6.5.3	Understanding Branch History Injection (BHI) on Intel	158
6.6	End-to-end Exploitation	162
6.6.1	Attacker Primitives	162
6.6.2	Exploiting BHI with eBPF	163
6.6.3	Same-Predictor-Mode Exploit	166
6.6.4	Exploitation Beyond Extended Berkeley Packet Filter (eBPF)	168
6.7	Mitigations	171
6.8	Related Work	172
6.9	Conclusion	173
	Appendices	174
6.A	Linux eIBRS Adoption Flaw	174
7	Conclusion	177
7.1	Future directions	179
	References	183
	Summary	201