IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Volume 12, PP 47-51

Revised Design Model for New Generation Processor to Tackle Side-Channel Attacks

Viraj Parab[1], Bhagyesh Kurlekar[2], Vatsal Agrawal[3], Asst. Prof. Divya Kumawat[4]

[1](*Electronics And Telecommunication Engineering, Atharva College Of Engineering/ Mumbai University, India)*

[2,3,4] *(Computer Engineering, Atharva College Of Engineering/ Mumbai University, India)*

Abstract: *Spectre And Meltdown Are The Newly Discovered Security Vulnerabilities. These Vulnerabilities Exploit The Most Widely Used Techniques In Latest Processors; Speculative Execution And Out-Of-Order Execution. These Attacks Are Side-Channel Based. In These Attacks, Private And Secured Data Is Manipulated And Transferred To The Cache Memory. Cache Memory Acts As A Side Channel And Leaks Out Data. These Security Vulnerabilities Can Breach Into An Entire System And Gain Access To The Private Data Which Causes The Integrity Of The System To Be Compromised. In This Paper, We Have Discussed Meltdown And Spectre Attacks In Detail And Suggested A Hardware Mitigation To The Existing Architecture Of Modern Processors. This Design Model Works With All The Existing Processors And Reduces The Impact Of Side-Channel Attacks To An Extent That It Can Be Nullified.*

Keywords *- Branch Prediction, KAISER, Meltdown, Out-Of-Order Execution, Reorder Buffer, Side-Channel Attack, Timing-Based Attack, Speculative Execution, Spectre.*

I. INTRODUCTION

Security Of Data, In The 21st Century, Is Considered To Be Of Utmost Priority. Millions Of Dollars Are Spent To Maintain And Update The Security Features For Better Encapsulation And Isolation Of Data From Various Different Malware. Despite All The Work Done In This Field, Recent Discoveries Done By Google's Project Zero Team Have Discovered Hardware Vulnerabilities That Affects Nearly All Modern Microprocessors [1]. These Security Vulnerabilities Are Called As Spectre And Meltdown. They Were First Discovered On 1st June, 2017 But Due To The Seriousness Of The Vulnerabilities, They Were Declared On 3rd January, 2018. This Delay Was Done So That All The Affected Companies Can Work Around For A Fix. These Vulnerabilities Are Focused On Exploiting Two Most Important Techniques In Modern Processors Which Are Implemented To Increase The Speed Of Execution And Reduce CPU Waiting Time. These Two Techniques Are Speculative Execution And Out-Of-Order Execution.

II. LITERATURE SURVEY

2.1 Speculative Execution

Speculative Execution Is An Optimization Technique In Which Processor Executes Some Task That May Not Be Needed. Instruction Prefetching And Execution Is Completed Prior To The Acknowledgment Received From The Execution Unit That Whether The Branch Will Be Taken Or Not. This Prevents A Delay That Would Have To Be Incurred By Doing The Work After It Is Known That The Branch Is Taken. If It Turns Out The Speculated Results Are Not Required, Then The Changes Made By These Executions Are Reverted And The Results Are Discarded.

The Objective Here Is To Make Use Of The Processor Cycle Which Otherwise Would Have Been Wasted By Waiting For The Execution Of The Branching Instruction. In The Process It Also Increases Instruction Parallelism. This Feat Is Achieved By The Branch Prediction Unit (BPU). BPU, During The Pipelining Of Instructions, Makes A Guess Regarding Which Branch Should Be Taken And Then Accordingly Loads The Appropriate Instructions In The Pipeline. This Technique Is Also Used For Optimistic Concurrency Control In The Database System. [2][3]

2.2 Out-Of-Order Execution

Out-Of-Order Execution Is A Concept Which Is Used In Latest High-End Processors, Because Of Which The Processor Core Is Hardly Left Idle, I.E. Its Instruction Cycles Are Not Wasted During The Fetching Of Data. In OOO Execution, The Processor Executes Instructions With Respect To Input Resource Availability And Availability Of Execution Unit, Rather Than Following The Original Order In Which The Program Instructions Were Loaded. This Is Done With For The Sole Purpose Of Reducing The Processor's Waiting Time And This Is Achieved By Prefetching The Data Of The Next Instruction While The Current Instruction Is Being

Executed. This Is Possible Because Generally, Processors Have Separate Data Fetching And Execution Unit Which Can Be Used In Parallel. [2][3]

On Reviewing These Techniques, It Can Be Concluded That These Techniques Were Used To Increase The Performance And Efficiency Of The Central Processing Unit. But These Techniques During Their Execution Perform Some Illegal Operations And Trade Security In Exchange For Speed. These Illegal Operations Allow The Attacker To Access The Data For Which He Is Not Privileged. Meltdown And Spectre Are The Attacks Which Are Used To Successfully Acquire These Data.

The Attacker Runs The Malicious Code Which Attempts To Read Kernel Memory From User-Space Without Misdirecting The Control Flow Of Kernel Code. The Malicious Code Pattern Used Is Executed With A Constraint Of Executing In User-Space, I.E. It Is Executed To Read The Kernel Memory From The Same System Memory. The Idea Behind This Is To Create A High-Latency For A Mispredicted Branch And Ensure That During Speculative Execution System Should Not Check For Permission For Operating On These Unprivileged Data. The Data Which Is Accessed Is Stored In The Cache Memory. The Process In Which Kernel Data Is In The Picture Is Called Meltdown Attack [4].

In Spectre, A Similar Malicious Code Is Executed By The Attacker. In General, Each Application Has The Privilege To Access Only A Certain Set Of Data. Rest Of The Data Is Either Restricted Or Needs Special Privileges To Be Accessed. The Attacker Uses His Malicious Code To Access This Restricted Data And Silently Channel It To The Cache Memory. Thus, Cache Acts As A Side Channel And Leaks Out This Private Data [4].

KAISER Is The Software Solution For Meltdown Attack. For Faster Execution Speeds, Some Processors Map Some Part Of Kernel Memory Into User Address Space. So, Implementing KAISER Blocks The Memory Mapping Of Kernel Address Space Into User Address Space. But By Doing So, KAISER Decreases The Processing Speed By About 5-30% Of Its Original Speed. [5][6].

III. BLOCK DIAGRAM OF THE PROPOSED MODEL

The Private Memory Is A Memory Which Is Accessed In Case Of A Speculative Execution. The Private Memory Consists Of Three Blocks, Which Are As Follows: -

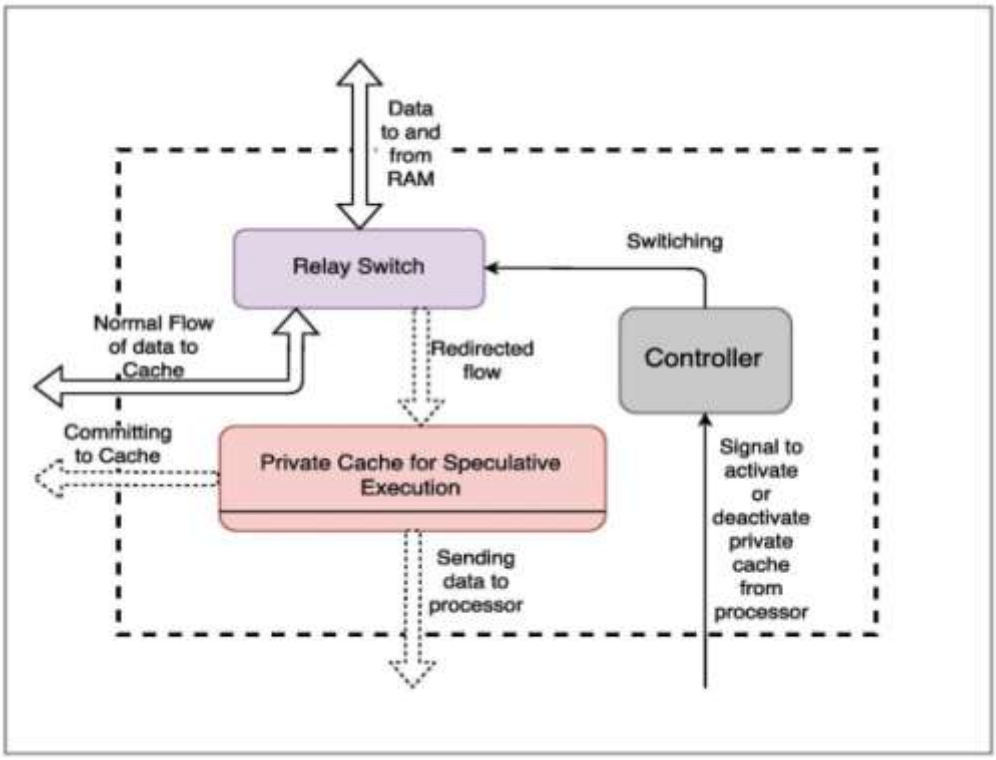

Fig 3.1 Private Memory Model

3.1 Private Cache

The Private Cache Is The Memory Section Which Has Been Kept Separately To Store The Data Requested From RAM During Speculative Execution. When The Changes Made By The Speculatively Executed Instructions Are Committed, Then All The Data In The Private Cache Is Shifted To The Normal

Cache. At A Time, There Are Only A Few Instructions In The Speculative Execution Unit That Call Data From The RAM. Most Of The Instructions Get Their Data From The Cache Itself. Hence The Size Of This Private Cache Can Be Small. In Fact, The Size Depends On The Number Of Speculative Instructions That Are Pushed Into The Pipeline That Require Data From The RAM. This Is Because The Instructions Are Executed Speculatively Only Up To The Time When The Branching Instruction Has Been Executed. After This Time, The Processor Knows For Sure Whether The Branch Is Taken Or Not And Hence There Is No Need To Execute The Instructions Speculatively.

The Private Memory Is Invisible To The Processor During Normal Execution. The Processor Is Allowed To Read Data From The Private Cache During Speculative Execution Only. It Is Activated Only When The Controller Sends An Activation Signal. In Other Cases It Is Deactivated.

3.2 Controller

The Controller Plays An Important Role In This Block. It Handles The Requests From External Ports. As Soon As The Controller Gets A Signal From The Processor That, A Branching Instruction Has Been Detected, It Prepares The "Private Cache" To Be Used By The Speculative Execution Unit [7]. It Also Controls The Relay Switch By Sending The Necessary Signal To The Relay Switch, Thereby Changing The Flow Of Data From RAM To Private Cache And Back To Cache When The Work Is Done. It Also Oversees The Data Transfer From Private Cache To Cache.

3.3 Relay Switch

The Relay Switch Is The Block Which Follows The Instruction From The Controller And Switches The Data Flow From RAM To Private Cache. The Relay Switch Acts As A Three-Way Switch Here. It Handles Operations Like Redirecting The Flow Of Data From RAM To Private Cache Or To Cache, Redirecting The Flow Of Data To And From The RAM [8].

The Relay Switch Is Basically A Two-Way Switch. For Example, Track Switching In Case Of Railway Trains. The Relay Switch Here Is Doing The Same Kind Of Work By Switching The Way The Data Flows In The Model Block Without Loss Of Data. It Follows The Command As And When Given By Control Unit.

IV. WORKING OF THE PROPOSED MODEL

Spectre And Meltdown Implement Side-Channel Attacks By Exploiting A Bunch Of Modern Processor Functionalities Like Speculative Execution, Out-Of-Order Execution, Branch Predictor Etc. Along With Timing-Based Attack Etc. To Gain Access To Private Or Sensitive Information. [3] The Exploit Begins When Branching Occurs. When Branching Occurs, The Branch Predictor Predicts A Branch And Speculatively Executes The Instructions Of That Branch. This Is Called Speculative Execution. The Values Generated From The Speculative Execution Are Not Committed Though. [1] If In The Event Where The Prediction Made By The Branch Predictor Is Wrong, Then The Changes Made By The Speculatively Executed Instructions Are Discarded And The Correct Instructions Are Pushed Into The Pipeline For Execution. Here The Problem Is That All The Changes Are Abandoned Except The Operand Values (Inputs Of The Instruction) That The Instruction From Speculative Execution Requests From The RAM And Which Are In Turn Stored In The Cache. [2]

The Proposed Solution To Counteract This Problem Is, To Store All The Operands Requested By The Speculatively Executing Instructions From The RAM In A Separate Memory. If The Operands Requested By The Instruction Are Available In The Cache Itself, Then That Specific Instruction Will Not Need To Interact With The Separate Memory. This May Sound Difficult To Implement At First But As The Explanation Continues, The Ideology Will Start To Become Natural. This Memory Is Accessible Only When The Speculative Execution Unit Is Active [9]. This Separate Memory Will Act As A Temporary Private Cache For Speculatively Executing Instructions Only. In Case Of Regular Instructions, This Private Cache Will Become Inaccessible And It Will Not Be Available To The Processor.

This Private Cache Will Come With A Control Unit That Takes An Input From Branch Predictor Indicating Whether The Branch Is Taken Or Not And Then Instructs The Relay Switch To Redirect The Flow Of Data From RAM To The Private Cache Instead Of The General Cache. This Is Clearly Shown In The Fig. 4.1. One More Point To Be Noted Is That The Transfer Of Data Is From The Private Cache To Processor Only And Not Vice-Versa. This Is Because The Output Generated From The Speculative Execution Is Not Written To Memory Until The Branching Instruction Is Resolved. Instead, The Output Generated Is Stored In The Reorder Buffer. [10] Hence There Is No Need For A Bidirectional Bus Which Also Takes Data From Processor To Private Cache Because When The Branching Is Resolved, The Processor Will Send A Signal To The Control Unit To Redirect Data Flow From RAM Back To Cache Thereby Disabling The Private Cache Unit And The Values Thereby Can Be Committed Or Written Normally. Depending On This Signal, The Data From The Private Cache May Or May Not Be Copied To The Cache.

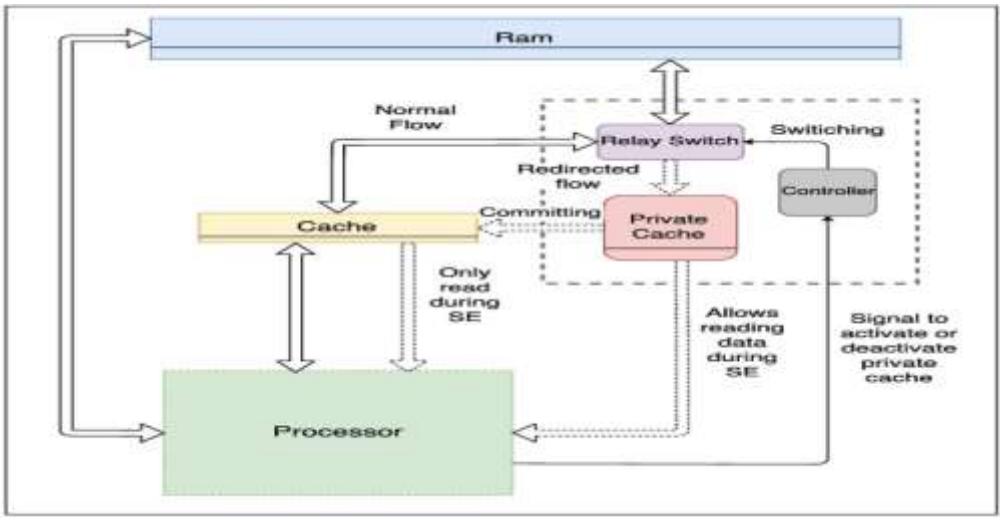

Fig 4.1

If The Branch Predictor Made A Correct Guess Then The Changes Made By The Speculatively Executed Instructions Are Committed And The Processor Continues To Execute From That Point Onwards. Also, The Values That Are Stored In The Private Cache Will Be Moved To The Cache. The Entry Of Data To The Private Cache Will Also Be Disabled When Speculative Execution Unit Deactivates (In This Case When The Branch Is Taken By The Processor). The Time Required To Move All The Data From The Private Cache To Cache Should Not Be Much Because Generally Only A Limited Number Of Instructions Executed Speculatively, Call Data From RAM.

If The Branch Predictor Made An Incorrect Guess Then The Changes Made By The Speculatively Executed Instructions Are Discarded And The Processor Loads The Correct Instructions In The Pipeline For Execution. This Time The Data Will Not Be Moved From The Private Cache To The Cache. Instead, The Data Will Just Be Discarded.

HOW IS THE PROBLEM SOLVED?

The Attacker Implements Their Malicious Code In Such A Way That The Part Of The Code Which Brings The Victim Data Into The Cache Is Executed Speculatively. They Then Run The Timing-Based Attack To Know Which Data Is Brought Into The Cache.

But After The Implementation Of The Private Memory Model, All The Data Requested By The Speculatively Executed Instructions From The RAM Are Brought Into The Private Cache, Which Is Discarded If Incorrect Instructions Are Executed. This Ensures That There Is No Data In The Cache On Which The Timing-Based Attack Can Work. Hence Ensuring The Safety Of Data.

V. CONCLUSION

In This Paper, The Solution Proposed, Modifies The Existing Processor Architectures In Such A Way That The Cache Cannot Be Used As A Side-Channel For Spectre And Meltdown Attacks. This In Turn Also Renders The Timing-Based Attack On The Cache Useless. The Root Of This Solution Stems From The Ideology That, There Should Be No Footsteps Left Behind When The Changes Made By The Speculative Execution Are Rolled-Back Or Discarded.

A Plus Point Of This Solution Is That It Is Minimally Invasive I.E. It Does Not Change The Entire Architecture Rather It Just Works By Adding New Functionalities. Another Advantage Of This Solution Is That The Performance Decrease Is Expected To Be Minimum. Even Lesser Than The Current Patch KAISER.

He Disadvantage Of This Solution Is That There Maybe Be Some Performance Hit. Also, Adding New Hardware Is Going To Take Precious Processor Space.

REFERENCES

[1] Intel Analysis Of Speculative Execution Side Channels White Paper, Https://Newsroom.Intel.Com/Wp-Content/Uploads/Sites/11/2018/01/Intel-Analysis-Of-Speculative-Execution-Side-Channels.Pdf

[2] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard1, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg, Meltdown, Https://Meltdownattack.Com/.

[3] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, Yuval Yarom, Spectre Attacks: Exploiting Speculative Execution∗ , Https://Meltdownattack.Com/.

[4] Project Zero, Https://Googleprojectzero.Blogspot.In/, Referred On 21/2/2018

[5] Recent Intel Cpus Take Performance Hit With Spectre, Meltdown Patches, Joel Hruska,

Https://Www.Extremetech.Com/Computing/264796-Recent-Intel-Cpus-Take-Performance-Hit-Spectre-Meltdown-Patches, Referred On 3/2/2018
[6] Https://Nakedsecurity.Sophos.Com/2018/01/03/Fckwit-Aka-Kaiser-Aka-Kpti-Intel-Cpu-Flaw-Needs-Low-Level-Os-Patches/, Referred On 7/2/2018
[7] Https://Patents.Google.Com/Patent/US5625837A/En, Referred On 15/2/2018
[8] Https://Patents.Google.Com/Patent/US20110231510A1/En?Q=Redirecting&Q=Flow&Q=Data&Q=RAM&Oq=Redirecting+The+Flow+Of+Data+From+And+To+The+RAM, Referred On 15/1/2018
[9] Speculation Based Side Channel Attacks, Mike Hearn, Https://Www.Corda.Net/2018/01/Speculation-Based-Side-Channel-Attacks/, Referred On 1/2/2018
[10] The Reorder Buffer, CSE240A-MBT-L13-Reorderbuffer.Ppt, Https://Cseweb.Ucsd.Edu/Classes/Fa14/Cse240a-A/Pdf/07/CSE240A-MBT-L13-Reorderbuffer.Ppt.Pdf, Referred On 26/1/2018