Hardware CWE™ Special Interest Group (SIG)

Gananand Kini, Bob Heinemann, Gage Hackford, Chris Lathrop, Steve Christey Coley, Alec Summers

MITRE

November 03, 2023

CWE and CAPEC are sponsored by U.S. Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA).

Agenda

REMINDER: This meeting is being recorded.

- **Housekeeping and Announcements**
- **Working Items for this meeting:**

1	CWE Nit Bits: Demonstrative Examples	Bob H	20 min
2	4.13 Release Items for HW	Bob H	10 min
3	Microarchitectural Weaknesses Update	Ganu K	15 min
4	Community Items	Bob H	05 min

Housekeeping

- **Schedule:**
  - **Next Meeting:**
    - **December 8th**
    - **12:30 – 1:30 PM EST (16:30 – 17:30 UTC)**
    - **Microsoft Teams**
- **Contact: cwe@mitre.org**
- **Mailing List:** *hw-cwe-special-interest-group-sig-list@mitre.org*
- **Minutes from previous meetings available on our GitHub site:**
  - **https://github.com/CWE-CAPEC/hw-cwe-sig**

Announcements

- **New CWE Content Development Repository (CDR) pilot now on GitHub! Currently invite only. Potential public release in early 2024.**
- **CWE 4.13 released October 26, 2023.**
- **CWE 4.14 planning ongoing. Let us know if you have any priority items. We are prioritizing the Microarchitectural Weaknesses for this release.**

CWE Nit Bits

Bite-sized knowledge
that will enhance your CWE proficiency!

Demonstrative Examples (DEMOXs)

Demonstrative Examples illustrate weaknesses through code, explanatory text, and/or diagrams[1].

DEMOX Structure

- What the example is trying to do without mentioning the weakness potentially[2] followed by a "bad code" block.
- What did the example do wrong, i.e., explain the weakness.
- How it can be exploited (optional)
- How to fix it, i.e., summarize weakness mitigation potentially[2] followed by a "good code" block.

[1] https://cwe.mitre.org/about/new_to_cwe.html

[2] In some cases involving design, DEMOX can be a paragraph or two without code.

DEMOX Code Block Concerns

Code Blocks:

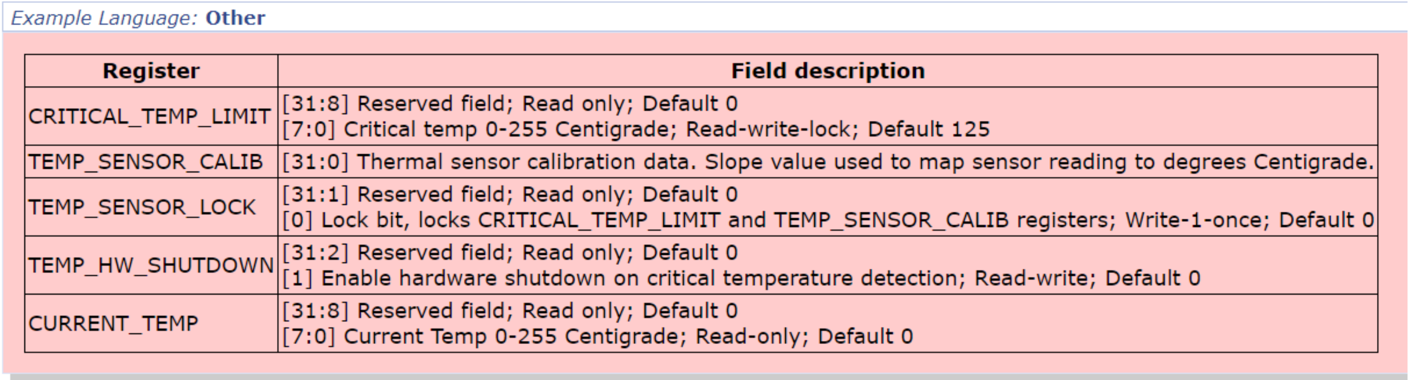
- Are just snippets, not a fully functioning program
- To keep code brief, "..." can be used to indicate code that has been omitted
- The code is syntactically correct
- The code is indented correctly
- The code examples SHOULD NOT contain multiple weaknesses
- Code should omit statements that are often used in all/most programs, e.g., loading standard libraries.
- Code may follow different basic standards, e.g., formatting, identifier naming, etc.
- Code accurately reflects the issue being illustrated

CWE-1233: Security-Sensitive Hardware Controls with Missing Lock Bit Protection

What is the example trying to do

Consider the example design below for a digital thermal sensor that detects overheating of the silicon and triggers system shutdown. The system critical temperature limit (CRITICAL_TEMP_LIMIT) and thermal sensor calibration (TEMP_SENSOR_CALIB) data have to be programmed by the firmware.

“Bad” illustration



Example Language: **Other**

Register	Field description
CRITICAL_TEMP_LIMIT	[31:8] Reserved field; Read only; Default 0 [7:0] Critical temp 0-255 Centigrade; Read-write-lock; Default 125
TEMP_SENSOR_CALIB	[31:0] Thermal sensor calibration data. Slope value used to map sensor reading to degrees Centigrade.
TEMP_SENSOR_LOCK	[31:1] Reserved field; Read only; Default 0 [0] Lock bit, locks CRITICAL_TEMP_LIMIT and TEMP_SENSOR_CALIB registers; Write-1-once; Default 0
TEMP_HW_SHUTDOWN	[31:2] Reserved field; Read only; Default 0 [1] Enable hardware shutdown on critical temperature detection; Read-write; Default 0
CURRENT_TEMP	[31:8] Reserved field; Read only; Default 0 [7:0] Current Temp 0-255 Centigrade; Read-only; Default 0

CWE-1233: Security-Sensitive Hardware Controls with Missing Lock Bit Protection (2)

What is wrong

In this example note that only the CRITICAL_TEMP_LIMIT register is protected by the TEMP_SENSOR_LOCK bit, while the security design intent is to protect any modification of the critical temperature detection and response.

The response of the system, if the system heats to a critical temperature, is controlled by TEMP_HW_SHUTDOWN bit [1], which is not lockable. Also, the TEMP_SENSOR_CALIB register is not protected by the lock bit.

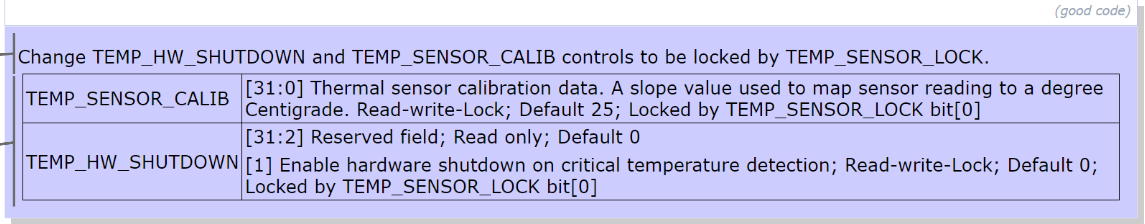
How it can exploited

By modifying the temperature sensor calibration, the conversion of the sensor data to a degree centigrade can be changed, such that the current temperature will never be detected to exceed critical temperature value programmed by the protected lock.

Similarly, by modifying the TEMP_HW_SHUTDOWN Enable bit, the system response detection of the current temperature exceeding critical temperature can be disabled.

How to fix it

“Good” illustration



(good code)

Change TEMP_HW_SHUTDOWN and TEMP_SENSOR_CALIB controls to be locked by TEMP_SENSOR_LOCK.

TEMP_SENSOR_CALIB	[31:0] Thermal sensor calibration data. A slope value used to map sensor reading to a degree Centigrade. Read-write-Lock; Default 25; Locked by TEMP_SENSOR_LOCK bit[0]
TEMP_HW_SHUTDOWN	[31:2] Reserved field; Read only; Default 0 [1] Enable hardware shutdown on critical temperature detection; Read-write-Lock; Default 0; Locked by TEMP_SENSOR_LOCK bit[0]

CWE 4.13 HW Release Items

New CWE Entry

CWE-1419: Incorrect Initialization of Resource

- **Addressed a gap in CWE hierarchy**
- **This is a class and contains 5 specific child weaknesses**
- **Thanks for your feedback from prior meeting**

Added to Extended Description

For hardware, this weakness frequently appears with reset values and fuses. After a product reset, hardware may initialize registers incorrectly. During different phases of a product lifecycle, fuses may be set to incorrect values. Even if fuses are set to correct values, the lines to the fuse could be broken or there might be hardware on the fuse line that alters the fuse value to be incorrect.

HW Specific and Related Items for Release 4.13

- **11 DEMOXs added from HACK@DAC**
- **These are valuable illustrations of HW weaknesses. Thank you.**
  - CWE-325: Missing Cryptographic Step
  - CWE-1191 (2): On-Chip Debug and Test Interface With Improper Access Control
  - CWE-1220: Insufficient Granularity of Access Control
  - CWE-1221: Incorrect Register Defaults or Module Parameters
  - CWE-1231: Improper Prevention of Lock Bit Modification
  - CWE-1241: Use of Predictable Algorithm in Random Number Generator
  - CWE-1243: Sensitive Non-Volatile Information Not Protected During Debug
  - CWE-1276: Hardware Child Block Incorrectly Connected to Parent System
  - CWE-1300: Improper Protection of Physical Side Channels
  - CWE-1326: Missing Immutable Root of Trust in Hardware

OBEXs Added

- 13 OBEXs added from Top 25 Mapping and HW SIG Members
- These are examples of weaknesses found in the wild

- **CWE-203**: Observable Discrepancy
- **CWE-1241**: Use of Predictable Algorithm in Random Number Generator
- **CWE-1247**: Improper Protection Against Voltage and Clock Glitches
- **CWE-1254**: Incorrect Comparison Logic Granularity
- **CWE-1258**: Exposure of Sensitive System Information Due to Uncleared Debug Information
- **CWE-1281**: Sequence of Processor Instructions Leads to Unexpected Behavior
- **CWE-1295**: Debug Messages Revealing Unnecessary Information
- **CWE-1299**: Missing Protection Mechanism for Alternate Hardware Interface
- **CWE-1300**: Improper Protection of Physical Side Channels
- **CWE-1313**: Hardware Allows Activation of Test or Debug Logic at Runtime
- **CWE-1319**: Improper Protection against Electromagnetic Fault Injection (EM-FI)
- **CWE-1331**: Improper Isolation of Shared Resources in Network On Chip (NoC)
- **CWE-1384**: Improper Handling of Physical or Environmental Conditions
- **CWE-1419**: Incorrect Initialization of Resource

CWE and CAPEC are sponsored by U.S. Department of Homeland Security (DHS) Cybersecurity and Infrastructure Security Agency (CISA).

Microarchitectural Weaknesses Update

Microarchitectural Weaknesses Update

- **Group came up with 4 new submissions now on GitHub CWE CDR:**
  - CWE A: Exposure of Sensitive Information during Transient Execution
    - CWE B: Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution
    - CWE C: Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution
    - CWE D: Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that influences Transient Execution
- **Have descriptions, extended descriptions and observed examples.**
- **Working on filling out remaining fields on the GH CWE CDR.**

Exposure of Sensitive Information during Transient Execution (CWE-A)

Abstraction: Class

Description

A processor event or prediction may allow incorrect operations (or correct operations with incorrect data) to execute transiently, potentially exposing data over a covert channel.

Extended Description

… Operations that execute transiently may have side effects that can be detected using timing or power analysis techniques. These techniques may allow an attacker to infer information about the operations that are executed transiently. For example, the attacker may be able to infer confidential data that was accessed or used by those operations. …

Observed Examples

- CVE-2017-5753: Speculative execution risks data leaks via side-channel.
- CVE-2021-0089: Speculative Code Store Bypass (SCSB)
- CVE-2022-0002: Processors risk information disclosure through non-transparent branch predictor sharing.

https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues/35

Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution (CWE-B)

Abstraction: Base

Description

A processor event may allow transient operations to access architecturally restricted data (for example, in another address space) in a shared microarchitectural structure (for example, a CPU cache), potentially exposing the data over a covert channel.

Extended Description

… When transient operations allow access to data that is protected by the ISA, this can violate users' expectations of the ISA feature that is bypassed. For example, if transient operations can access a victim's private data in a shared microarchitectural structure, then the operations' microarchitectural side effects may correspond to the accessed data. If an attacker is able to trigger these transient operations and observe their side effects through a covert channel, then the attacker may be able to infer the victim's private data. …

Observed Examples

- CVE-2017-5715: Rogue Data Cache Load (RDCL, also known as Meltdown)
- CVE-2018-3615: L1 Terminal Fault (L1TF, also known as Foreshadow)
- CVE-2019-11091: Microarchitectural Data Sampling (MDS)

https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues/34

Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution (CWE-C)

Abstraction: Base

Description

A processor event or prediction may allow transient operations to forward incorrect or stale data to dependent operations, potentially exposing data over a covert channel.

Extended Description

.. If transient operations can forward incorrect or stale data to dependent operations, then the dependent operations' microarchitectural side effects may correspond to the data. If an attacker is able to trigger these transient operations and observe their side effects through a covert channel, then the attacker may be able to infer the data. …

Observed Examples

- CVE-2021-0086: Floating-point Value Injection (FPVI)
- CVE-2021-33149: Speculative Load Disordering (SLD)
- CVE-2018-3639: Speculative Store Bypass (SSB)

https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues/36

Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that influences Transient Execution (CWE-D)

Abstraction: Base

Description

Shared microarchitectural predictor state may allow code to influence transient execution across a hardware boundary, potentially exposing data that is accessible beyond the boundary.

Extended Description

…

When separate software components (for example, two processes) share microarchitectural predictor state across a hardware boundary, code in one component may be able to influence microarchitectural predictor behavior in another component. If the predictor can cause transient execution, then shared predictor state may allow an attacker to influence transient execution in a victim, and in a manner that could allow the attacker to infer private data from the victim.

…

Observed Examples

- CVE-2017-5754: Branch Target Injection (BTI), CVE-2022-0001: Branch History Injection (BHI)
- CVE-2022-29901: Post-barrier RSB

https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues/37

Microarchitectural Weaknesses Proposed Hierarchy Changes

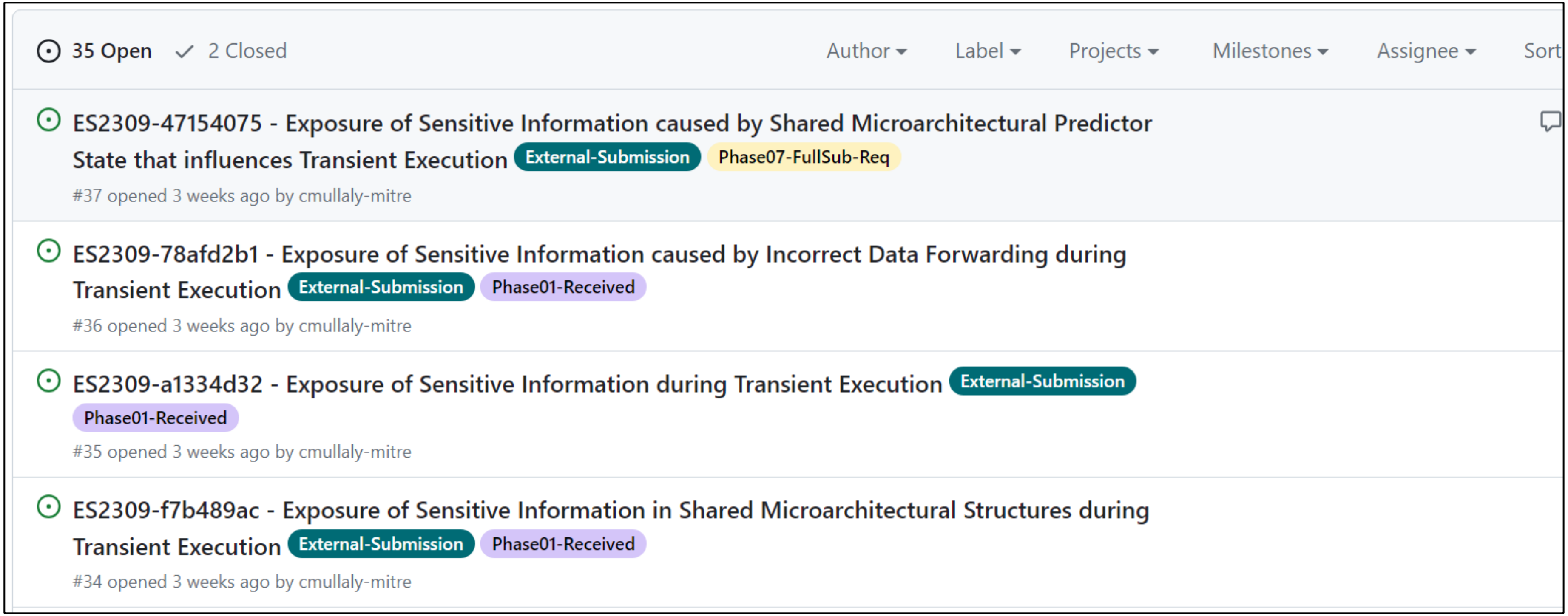
- **CWE-226: Sensitive Information in Resource Not Removed Before Use**
  - CWE-1342: Information Exposure through Microarchitectural State after Transient Execution (To be merged into CWE-C since it contains an example of LVI).
- **CWE-669: Incorrect Resource Transfer Between Spheres**
  - CWE A: Exposure of Sensitive Information during Transient Execution
    - CWE B: Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution
    - CWE C: Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution
    - CWE D: Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that influences Transient Execution

Next Steps

- **Need help to populate CWE Entry Elements**
  - Modes of Introduction
  - Applicable Platforms
  - Common Consequences
  - Demonstrative Examples
  - Observed Examples
  - Potential Mitigations
  - Related Weaknesses
  - References

Microarchitectural Weaknesses in CDR

https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues



⊙ ES2309-47154075 - Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that influences Transient Execution External-Submission Phase07-FullSub-Req

#37 opened 3 weeks ago by cmullaly-mitre

⊙ ES2309-78afd2b1 - Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution External-Submission Phase01-Received

#36 opened 3 weeks ago by cmullaly-mitre

⊙ ES2309-a1334d32 - Exposure of Sensitive Information during Transient Execution External-Submission Phase01-Received

#35 opened 3 weeks ago by cmullaly-mitre

⊙ ES2309-f7b489ac - Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution External-Submission Phase01-Received

#34 opened 3 weeks ago by cmullaly-mitre

If you would like to help/review

- **Give us your GitHub username so you can make comments**

Open *Community Items*

HW CWE's With Missing:
DEMOX's, OBEX's and Mitigations

- **Missing Mitigations**
  - 4 HW CWEs are missing mitigations (No change)
- **Missing demonstrative examples (DEMOX)**
  - 16 HW CWEs missing demonstrative examples (down 1)
    - 1 added from Hack@DAC, CWE-325
    - Note: there are other DEMOXs from Hack@DAC but now adding DEMOXs to entries that have an existing DEMOX
- **Missing Observed Examples (OBEX)**
  - 61 CWEs do not have any observed examples (down 6)

Thank you contributors! Keep them coming.

https://github.com/CWE-CAPEC/hw-cwe-sig/issues

Resonant Frequency Weakness

- **If anyone is interested is developing and proposing a resonant frequency weakness, please see discussion points on GitHub.**
- **https://github.com/CWE-CAPEC/hw-cwe-sig/issues/105**

Next Meeting (Dec 8th)

CWE@MITRE.ORG

- **Mailing List:** *hw-cwe-special-interest-group-sig-list@mitre.org*
  - *NOTE: All mailing list items are archived publicly at:*
    - *https://www.mail-archive.com/hw-cwe-special-interest-group-sig-list@mitre.org/*
- **What would members of this body like to see for the next HW SIG agenda?**
- **Questions, Requests to present? Please let us know.**

Backup