Hardware CWE™ Special Interest Group (SIG)

Chair: Bob Heinemann (MITRE)

Co-Chair: “Manna” Parbati Kumar Manna (Intel)

MITRE Team: Gananand Kini, Steve Christey Coley, Alec Summers

MITRE

May 09, 2025

Agenda

REMINDER: This meeting is being recorded.

1	General Status of Current HW CWE Submissions	Bob H	10 min
2	Memory Access Related Weaknesses	Jason Oberg	30 min
3	Most Important Hardware Weaknesses Refresh Update	Arun K, Ganu K	10 min

Housekeeping

- **Schedule:**
  - **Next Meeting June 13:**
    - **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

- **CWE Content Development Repository (CDR) is now fully public.**
    - Enables the broader community to view, track, and contribute to entries submissions.
    - Content suggestions begin with the CWE Submission Form.
    - CDR can be accessed here:
        - https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/

Call for Topics

What topics should we cover next time?

- **Anything to share today or topics for consideration for next meeting?**

Hardware Submissions in Progress

- **Initial Consultation (Stage 1, Phase 4 – common bottleneck)**
  - HW/SW: ES2306-c0b52346 Use of a Quantum-Vulnerable Cryptographic Algorithm (NIST)
- **Detailed Submission (Stage 2, Phase 8 – Details Received)**
  - ES2208-9fb81a1a Speculative propagation of requests for transaction before data validation in multi-manager bus architectures (Francesco Restuccia)
- **(Stage 1, Phase 2 – Ack-Receipt)**
  - ES2503-821d9ec2 - CPU Control bits are used without validation of legality, Fanyun Shu, University of Electronic Science and Technology of China
- **3 early submissions that look like vulnerability reports**
- **See CDR: https://github.com/CWE-CAPEC/CWE-Content-Development-Repository/issues**

Memory Access Related Weaknesses

Jason Oberg

Introduction

Overview

- Several CWEs exist related to out-of-bound (OOB) buffer access that are not in the hardware view.
- In hardware, buffers manifest themselves as memories, register arrays, etc.

Impact of OOB Buffer Access in HW

- OOB access to buffers in RTL simulation results in undefined behavior (usually an ‘X’ for reads or no changes for writes)
- In real synthesized logic, data *will* be read/written and the actual result of that may be unknown to the hardware designer.

Proposal

- Move relevant CWEs to the hardware view or provide a comparable CWE in HW view

Memory Related CWEs that are not in HW View

CWE ID	Name	Description	HW Consequences
CWE-125	**Out-of-bounds Read**	The product reads data past the end, or before the beginning, of the intended buffer.	• In simulation, OOB read returns 'X' and is undefined • In a synthesized circuit, they OOB read could expose sensitive information
CWE-124	**Buffer Underwrite ('Buffer Underflow')**	The product writes to a buffer using an index or pointer that references a memory location prior to the beginning of the buffer.	• In simulation, an OOB write is essentially a "no-op" and the buffer isn't updated • In a synthesized circuit, the OOB write may compromise the integrity of the buffer
CWE-787	**Out-of-bounds Write**	The product writes data past the end, or before the beginning, of the intended buffer.	• In simulation, an OOB write is essentially a "no-op" and the buffer isn't updated • In a synthesized circuit, the OOB write may compromise the integrity of the buffer
CWE-786	**Access of Memory Location Before Start of Buffer**	The product reads or writes to a buffer using an index or pointer that references a memory location prior to the beginning of the buffer.	• Similar impact as above

Hardware Example:
CWE-125 Out-of-bounds Read

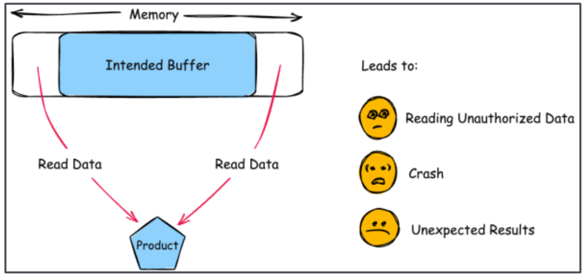
Cycuity

Description:

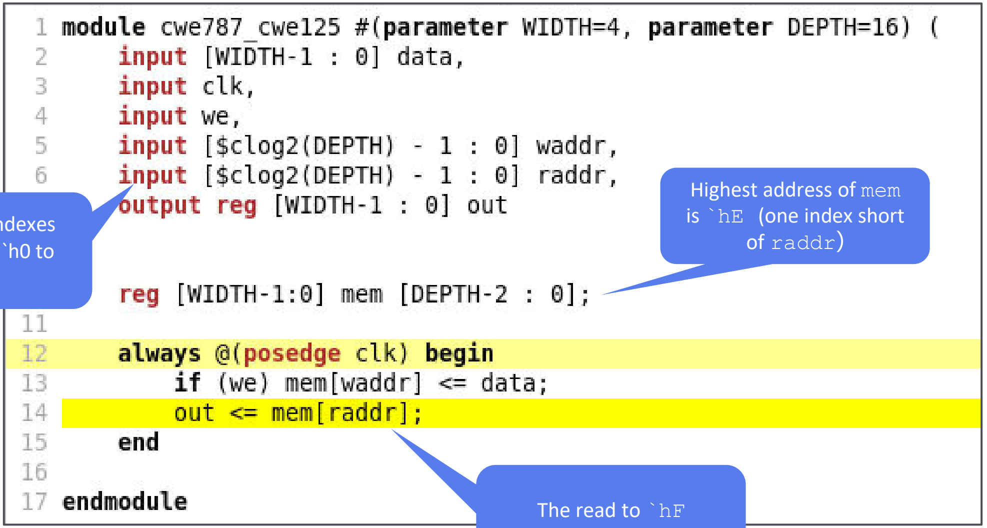
- The product reads data past the end, or before the beginning, of the intended buffer.

Hardware Consequences:

- OOB read in hardware is undefined and will return 'X' in simulation
- In a synthesized circuit, the data may be read from unexpected locations, potentially leaking data



https://cwe.mitre.org/data/definitions/125.html



```
module cwe787_cwe125 #(parameter WIDTH=4, parameter DEPTH=16) (
    input [WIDTH-1 : 0] data,
    input clk,
    input we,
    input [$clog2(DEPTH) - 1 : 0] waddr,
    input [$clog2(DEPTH) - 1 : 0] raddr,
    output reg [WIDTH-1 : 0] out


    reg [WIDTH-1:0] mem [DEPTH-2 : 0];

    always @(posedge clk) begin
        if (we) mem[waddr] <= data;
        out <= mem[raddr];
    end

endmodule
```

`raddr` indexes between `h0 to `hF

Highest address of `mem` is `` `hE `` (one index short of `raddr`)

The read to `` `hF `` results in an OOB read

Hardware Example:
CWE-787 Out-of-bounds Write

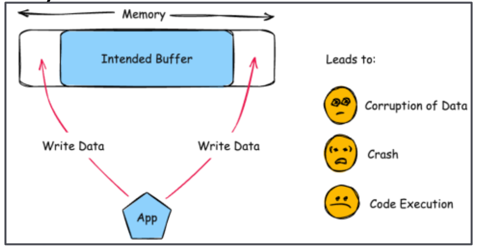
Cycuity

Description:

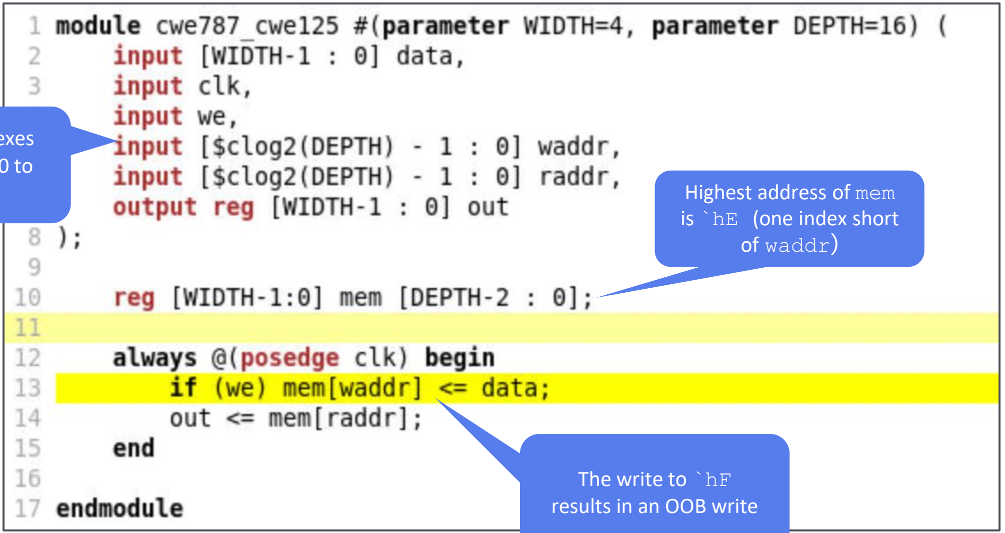
- The product writes data past the end, or before the beginning, of the intended buffer.

Hardware Consequences:

- OOB write in hardware is undefined and will basically not update the buffer/memory in *simulation*
- In real synthesized logic, a write may occur and could compromise the integrity of data in the memory.



https://cwe.mitre.org/data/definitions/787.html



```
module cwe787_cwe125 #(parameter WIDTH=4, parameter DEPTH=16) (
    input [WIDTH-1 : 0] data,
    input clk,
    input we,
    input [$clog2(DEPTH) - 1 : 0] waddr,
    input [$clog2(DEPTH) - 1 : 0] raddr,
    output reg [WIDTH-1 : 0] out
);

    reg [WIDTH-1:0] mem [DEPTH-2 : 0];

    always @(posedge clk) begin
        if (we) mem[waddr] <= data;
        out <= mem[raddr];
    end

endmodule
```

`waddr` indexes between \`h0 to \`hF

Highest address of `mem` is `` `hE `` (one index short of `waddr`)

The write to `` `hF `` results in an OOB write

Questions for Discussion

- Have others in the community encountered similar issues in hardware?
- The contents of these CWEs are software centric. Would it make sense to update them to include some hardware examples before inclusion in the HW view?

Most Important Hardware Weaknesses List Update

Arun K., Gananand K.

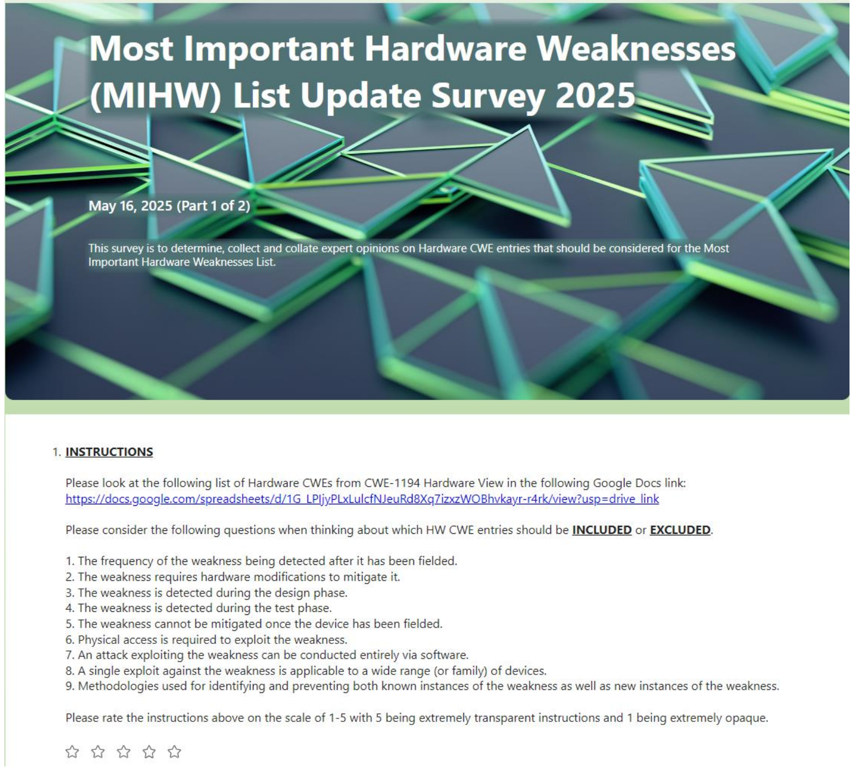
Most Important Hardware Weaknesses Refresh – April 2025 Update

- **Finished vulnerability data collection and group is now analyzing that data.**

- **Schedule for Expert Opinion survey :**
  - Plan to provide above data analysis summary as a reference point for the expert opinion survey. (May 16, 2025)
  - Plan to conduct survey before next HW CWE SIG in two parts:
    - First part will ask HW CWE SIG members about what CWEs should be considered for inclusion or exclusion in the updated MIHW list based on 9 significance questions. (May 16 – May 23)
    - Second part will ask HW CWE SIG members to rate the inclusion of specific HW CWEs based on a Likert scale. (May 30 – June 6)

- **MIHW Group to later combine the two data sets to derive the list**

Survey Screenshot


Most Important Hardware Weaknesses (MIHW) List Update Survey 2025

May 16, 2025 (Part 1 of 2)

This survey is to determine, collect and collate expert opinions on Hardware CWE entries that should be considered for the Most Important Hardware Weaknesses List.

1. **INSTRUCTIONS**

Please look at the following list of Hardware CWEs from CWE-1194 Hardware View in the following Google Docs link:
https://docs.google.com/spreadsheets/d/1G_LPljyPLxLulcfNJeuRd8Xq7izxzWOBhvkayr-r4rk/view?usp=drive_link

Please consider the following questions when thinking about which HW CWE entries should be **INCLUDED** or **EXCLUDED**.

1. The frequency of the weakness being detected after it has been fielded.
2. The weakness requires hardware modifications to mitigate it.
3. The weakness is detected during the design phase.
4. The weakness is detected during the test phase.
5. The weakness cannot be mitigated once the device has been fielded.
6. Physical access is required to exploit the weakness.
7. An attack exploiting the weakness can be conducted entirely via software.
8. A single exploit against the weakness is applicable to a wide range (or family) of devices.
9. Methodologies used for identifying and preventing both known instances of the weakness as well as new instances of the weakness.

Please rate the instructions above on the scale of 1-5 with 5 being extremely transparent instructions and 1 being extremely opaque.

☆ ☆ ☆ ☆ ☆

Survey questions

- **Considering adding some OPTIONAL metadata questions:**
  - Name, Email
  - What best describes your role with regard to the CWE program?
  - What industry do you work in?

- **Helps us understand some of the data being received**

MIHW List Discussion

- **Please continue discussion on the HW CWE Mailing List (see below).**
  - You can tag email subject lines using "[MIHW]" to get more visibility.

- **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/*

Next Meeting (June 13)

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

Popular CWEs in NVD excluding Software CWEs

- **33 unique HW CWEs with *at least* 1 CVE in NVD.**
- **16 HW CWEs with *only* 1 CVE**
- **11 HW CWEs with 2 CVEs**

13	CWE-1220: Insufficient Granularity of Access Control
4	CWE-1320: Improper Protection for Outbound Error Messages and Alert Signals
3	CWE-1357: Reliance on Insufficiently Trustworthy Component
3	CWE-1295: Debug Messages Revealing Unnecessary Information
3	CWE-1258: Exposure of Sensitive System Information Due to Uncleared Debug Information
3	CWE-1240: Use of a Cryptographic Primitive with a Risky Implementation
2	CWE-1332: Improper Handling of Faults that Lead to Instruction Skips
2	CWE-1319: Improper Protection against Electromagnetic Fault Injection (EM-FI)
2	CWE-1303: Non-Transparent Sharing of Microarchitectural Resources
2	CWE-1299: Missing Protection Mechanism for Alternate Hardware Interface
2	CWE-1298: Hardware Logic Contains Race Conditions
2	CWE-1281: Sequence of Processor Instructions Leads to Unexpected Behavior
2	CWE-1279: Cryptographic Operations are run Before Supporting Units are Ready
2	CWE-1269: Product Released in Non-Release Configuration
2	CWE-1263: Improper Physical Access Control

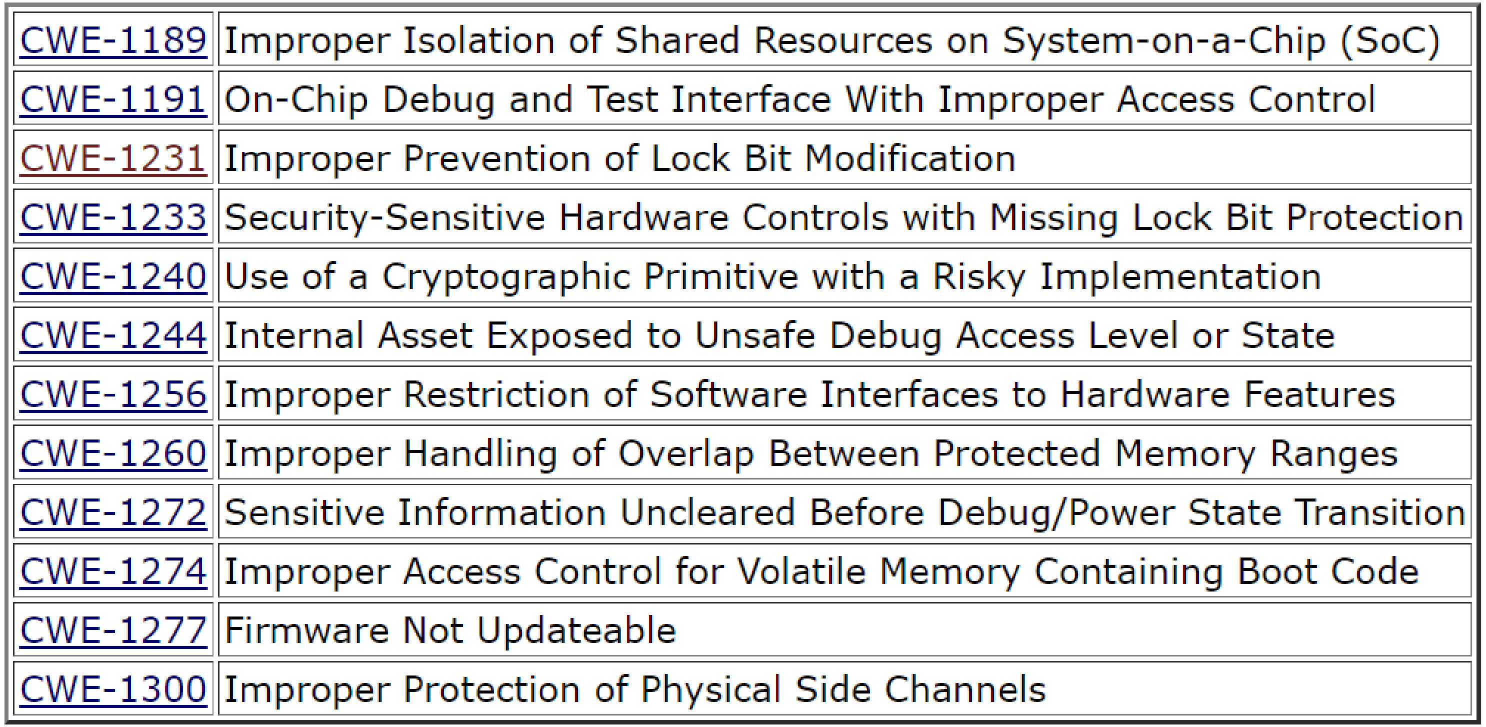
Formation of Ad-Hoc Committee

- **Will be putting a call out of the mailing list for members to join an ad-hoc committee to study.**
- **We will be looking for committee members to study the feasibility of a new list and making a decision to proceed.**
- **Also, members will develop an approach to develop the list with the community.**

Most Important Hardware Weaknesses (MIHW)

- **Is this something worth revisiting?**
- **Part of CWE 4.6 Release, October 28, 2021**
- **Have there been substantial developments since the last release of MIHW?**
- **Would those affect the rankings and inclusions of the list in any meaningful way?**
- **Is there any data available that we could utilize to generate the list? Or should we use the delphi method again?**
- **Are there observational trends that would change the current list in any significant and meaningful way?**

Current MIHW from CWE 4.6 (Unranked)



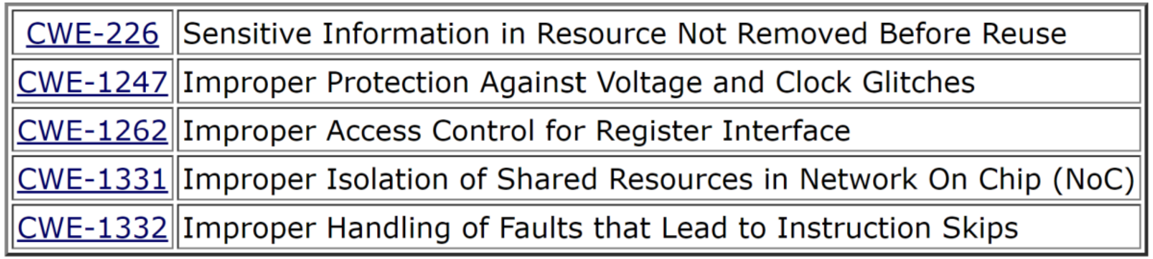
CWE-1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)
CWE-1191	On-Chip Debug and Test Interface With Improper Access Control
CWE-1231	Improper Prevention of Lock Bit Modification
CWE-1233	Security-Sensitive Hardware Controls with Missing Lock Bit Protection
CWE-1240	Use of a Cryptographic Primitive with a Risky Implementation
CWE-1244	Internal Asset Exposed to Unsafe Debug Access Level or State
CWE-1256	Improper Restriction of Software Interfaces to Hardware Features
CWE-1260	Improper Handling of Overlap Between Protected Memory Ranges
CWE-1272	Sensitive Information Uncleared Before Debug/Power State Transition
CWE-1274	Improper Access Control for Volatile Memory Containing Boot Code
CWE-1277	Firmware Not Updateable
CWE-1300	Improper Protection of Physical Side Channels

Previous Methodology for the MIHW

- **Previously, the Delphi method was used to poll members of this august body:**
  - Which 10 HW weaknesses were important based on nine significance questions:
    1. How frequently is this weakness detected after it has been fielded?
    2. Does the weakness require hardware modifications to mitigate it?
    3. How frequently is this weakness detected during design?
    4. How frequently is this weakness detected during test?
    5. Can the weakness be mitigated once the device has been fielded?
    6. Is physical access required to exploit this weakness?
    7. Can an attack exploiting this weakness be conducted entirely via software?
    8. Is a single exploit against this weakness applicable to a wide range (or family) of devices?
    9. What methodologies do you practice for identifying and preventing both known weaknesses and new weaknesses?

Previous Methodology for MIHW

- **After combining the above, thirty-one unique weaknesses resulted.**
- **Live poll conducted during SIG meeting asked members to assign the thirty-one into various buckets with weights (strongly support (+2), somewhat support (+1), no opinion (0), somewhat oppose (-1), strongly oppose (-2)).**
- **Multiple groups emerged from the data when ranked by weighted percentage of votes, of which the primary group contained twelve. The secondary group were listed as Hardware Weaknesses on the Cusp (shown below).**



CWE-226	Sensitive Information in Resource Not Removed Before Reuse
CWE-1247	Improper Protection Against Voltage and Clock Glitches
CWE-1262	Improper Access Control for Register Interface
CWE-1331	Improper Isolation of Shared Resources in Network On Chip (NoC)
CWE-1332	Improper Handling of Faults that Lead to Instruction Skips

Changes to HW CWEs since MIHW (v4.6 to v4.16)

- (DEPRECATED) CWE-1324: Sensitive Information Accessible by Physical Probing of JTAG Interface
- CWE-1342: Information Exposure through Microarchitectural State after Transient Execution
- (Class) CWE-1357: Reliance on Insufficiently Trustworthy Component
  - CWE-1329: Reliance on Component That is Not Updateable
- (Class) CWE-1384: Improper Handling of Physical or Environmental Conditions
- (Category) CWE-1388: Physical Access Issues and Concerns
- (Parent) CWE-1420: Exposure of Sensitive Information during Transient Execution
  - CWE-1421: Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution
  - CWE-1422: Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution
  - CWE-1423: Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Transient Execution
- This list is not complete!

Questions to think about and frame the MIHW activity

- Unfortunately, not many HW CWEs assigned to CVE entries in NVD currently.
- A lot of the observed examples (CVE) were added mostly for Transient Execution weaknesses. There are many that remain incomplete.
- What methodology should be used for the new MIHW list?
- What data could be used to generate the new MIHW list?
- Any observations or lessons learned that could change or impact the current list in a meaningful way? Any observed trends that could impact the list?
- Does it capture the use cases and important attributes discussed earlier for such a list?