Hardware CWE™ Special Interest Group (SIG)

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

MITRE

February 9, 2024

Agenda

REMINDER: This meeting is being recorded.

1	µArchitecture Weaknesses Updates	Bob H	10
2	Mitigations and Detection methods in CWE	Bob H	20
3	OBEX Working Group Formation	Bob H	20
4	Any Other Business (AOB)		

Housekeeping

- **Schedule:**
  - **Next Meeting: March 8**
    - **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) pilot now on GitHub! Currently invite only. Potential public release in early 2024.**
- **CWE 4.14 Release Planned for February 29.**
  - DEMOX's and Microarchitectural Weaknesses have been a priority for this release.
- **CWE 4.15 release will be around June/July**
- **Expect a new meeting series**

Farewells

- **Transitioning from MITRE.**
- **Hope to participate in the SIG as a member.**
- **Please contact Bob Heinemann, Steve Christey or cwe@mitre.org for any issues related to Hardware CWE.**

Microarchitectural Weaknesses Update

Microarchitectural Weaknesses Update

- **Working group met on Monday, January 22nd.**
  - Group approves to move forward with current submissions (4 total)
  - On track to be released for 4.14 (Feb 29)
  - Intel, Cycuity, and MITRE plan to coordinate announcing µArchitectural Weaknesses.
- **Some work to continue into 4.15 for WG**
  - Path forward on CWE-1342: Information Exposure through Microarchitectural State after Transient Execution
  - Define what is meant by Architectural and Microarchitecture to clarify readers not familiar
  - Minor change in some of the language from ARM
  - Other minor quality updates

Microarchitectural Weaknesses Update

4 new submissions being worked

- **CWE-1420 (CWE A): Exposure of Sensitive Information during Transient Execution**
  - CWE-1421 (CWE-B): Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution
  - CWE-1422 (CWE-C): Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution
  - CWE-1423 (CWE-D): Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Transient Execution

CWE-1420

CWE-1420: (CWE-A) Exposure of Sensitive Information during Transient Execution

Weakness ID: 1420
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

(CWE-A) A processor event or prediction may allow incorrect operations (or correct operations with incorrect data) to execute transient[ly] [p]otentially expos[e] [d]ata over a covert channel.

Extended Description

When operations execute but do not commit to the processor's architectural state, this is commonly referred to as transient execution. [T]his behavior can occur when the processor mis-predicts an outcome (such as a branch target), or when a processor event (such as an exception or microcode assist, etc.) is handled after younger operations have already exe[cuted]. Operations that execute transiently may exhibit observable discrepancies (CWE-203) in covert channels [REF-15] such as data caches. Observable discrepancies of this kind can be detected and analyzed [u]sing timing or [o]ther analysis techniques, which may allow an attacker to infer information about the operations that executed transiently. For example, the attacker may be able to infer confidential data that was accessed [a]s a result of those ope[r]ations.

Transient execution weaknesses may be exploited using one of two methods. In the first method, the attacker generates [a] code sequence that e[x]poses data through a covert channel when it is executed transiently (the attacker must also be able to trigger transient execution). Some transient execution weaknesses can only expose data that is accessible within the attacker's processor context. For example, an attacker executing code in a software sandbox may be able to use a transient execution weakness to expose data within the same address space, but outside of the attacker's sandbox. Other transient execution weaknesses can expose data that is architecturally inaccessible, that is, data protected by hardware-enforced boundaries such as page tables or privilege rings. [T]hese weaknesses are the subject of CWE-B.

In the second exploitation method, the attacker first identifies a code sequence in a victim program that, when executed transiently, can expose data that is architecturally accessible within the victim's processor context. For instance, the attacker may search the victim program for code sequences that resemble a bounds-check bypass sequence (see Demonstrative Example 1). If the attacker can trigger a mis-prediction of the conditional branch and influence the index of the out-of-bounds array access, then the attacker may be able to infer the value of out-of-bounds data by monitoring observable discrepancies in a covert channel.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	C	669	Incorrect Resource Transfer Between Spheres
ParentOf	B	1421	(CWE-B) Exposure of Sensitive Information in Shared [M]icroarchitectural [S]tructures during Transient Execution
ParentOf	B	1422	(CWE-C) Exposure of Sensitive Information caused by [I]ncorrect Data Forwarding during Transient Execution

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name
MemberOf	C	1198	Privilege Separation and Access Control Issues
MemberOf	C	1201	Core and Compute Issues
MemberOf	C	1202	Memory and Storage Issues
ParentOf	B	1421	(CWE-B) Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution

CWE-1421

CWE-1421: (CWE-B) Exposure of Sensitive Information in Shared Microarchitectural Structures during Transient Execution

Weakness ID: 1421
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

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

Extended Description

Many commodity processors have Instruction Set Architecture (ISA) features that protect software components from one another. These features include memory segmentation, virtual memory, privilege rings, trusted execution environments, and virtual machines, among others. For example, virtual memory provides each process with its own address space, which prevents processes from accessing each other's private data. Many of these features can be used to form hardware-enforced security boundaries between software components.

Many commodity processors also share microarchitectural resources that cache (temporarily store) data, which may be confidential. These resources may be shared across processor contexts, including across SMT threads, privilege rings, or others.

When transient operations allow access to ISA-protected data in a shared microarchitectural resource, this may 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 resource, then the operations' microarchitectural side effects may correspond to the accessed data. If an attacker can trigger these transient operations and observe their side effects through a covert channel [REF-15], then the attacker may be able to infer the victim's private data. The data may include sensitive program data, OS/VMM data, page table data (such as memory addresses), system configuration data (see Demonstrative Example 3), or any other data that the attacker does not have the required privileges to access.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	1420	(CWE-A) Exposure of Sensitive Information during Transient Execution

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name
ChildOf	B	1420	(CWE-A) Exposure of Sensitive Information during Transient Execution

CWE-1422

CWE-1422: (CWE-C) Exposure of Sensitive Information caused by Incorrect Data Forwarding during Transient Execution

Weakness ID: 1422
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

(CWE-C) A processor event or prediction may allow incorrect or stale data to be forwarded to transient operations, potentially exposing data over a covert channel.

Extended Description

Software may use a variety of techniques to preserve the confidentiality of private data that is accessible within the current processor context. For example, the memory safety and type safety properties of some high-level programming languages help to prevent software written in those languages from exposing private data. As a second example, software sandboxes may co-locate multiple users' software within a single process. The processor's Instruction Set Architecture (ISA) may permit one user's software to access another user's data (because the software executes the same address space), but the sandbox prevents these accesses by using techniques such as bounds checking.

If incorrect or stale data can be forwarded (for example, from a cache) to transient operations, then the operations' architectural side effects may correspond to the data. If an attacker can trigger these transient operations and observe their side effects through a covert channel, then the attacker may be able to infer the data. For example, an attacker process may induce transient execution in a victim process that causes it to inadvertently access and then expose its private data via a covert channel. In the software sandbox example, an attacker sandbox may induce transient execution in its own code, allowing it to transiently access and expose data in a victim sandbox that shares the same address space.

Consequently, weaknesses that arise from incorrect/stale data forwarding might violate users' expectations of software that used memory safety and isolation techniques. If the data forwarding behavior is not properly documented by the hardware vendor, this might violate the software vendor's expectation of how the hardware should behave.

Relationships

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	1420	(CWE-A) Exposure of Sensitive Information during Transient Execution

Relevant to the view "Hardware Design" (CWE-1194)

Nature	Type	ID	Name
ChildOf	B	1420	(CWE-A) Exposure of Sensitive Information during Transient Execution

Modes Of Introduction

Phase	Note
Architecture and Design	This weakness can be introduced by data speculation techniques, or when the processor pipeline is designed to check exception conditions concurrently with other operations. This weakness can also persist after a CWE-1421 weakness has been mitigated. For example, suppose that a processor can forward stale data from a shared microarchitectural buffer to dependent transient operations, and furthermore suppose that the processor has been patched to flush the buffer on context switches. This mitigates the CWE-1421 weakness, but the stale-data forwarding

CWE-1423

CWE-1423: (CWE-D) Exposure of Sensitive Information caused by Shared Microarchitectural Predictor State that Influences Tra Execution

Weakness ID: 1423
Abstraction: Base
Structure: Simple

View customized information: Conceptual | Operational | Mapping Friendly | **Complete** | Custom

Description

___SUMMARY___

Extended Description

___EXTENDED_DESCRIPTION_PARAGRAPH_1___

- List - bullet 1
- List - bullet 2
- List - bullet 3

___EXTENDED_DESCRIPTION_PARAGRAPH_2___

Relationships

Modes Of Introduction

Phase	Note
___PHASE_1___	
___PHASE_2___	___NOTE_1___

Applicable Platforms

Languages

___NAME_OR_TYPE___ *(Undetermined Prevalence)*

Class: ___CLASS___ *(Undetermined Prevalence)*

Operating Systems

___NAME_OR_TYPE___ *(Undetermined Prevalence)*

Class: ___CLASS___ *(Undetermined Prevalence)*

Architectures

___NAME_OR_TYPE___ *(Undetermined Prevalence)*

CWE Nit Bits

Potential Mitigations and Detection Methods

Potential Mitigations Element

- Contains one or more mitigations that indicate a decision, action, or practice intended to reduce the impact or eliminate a weakness.
- **Sub-Elements:**
  - **Phase:** indicates the development life cycle phase during which this mitigation may be applied.
  - **Description:** contains a description of this individual mitigation including any strengths and shortcomings of this mitigation for the weakness.
  - ***Strategy:*** describes a general strategy for protecting a system to which this mitigation contributes.
  - ****Effectiveness:*** summarizes how effective the mitigation may be in preventing the weakness.
  - ***Effectiveness Notes***: additional discussion of the strengths and shortcomings of this Mitigation.

Note: *Italics* indicate that the element is optional

Detection Method Element

- Contains one or more detection methods that may be employed to detect this weakness.

- **Sub-Elements:**
  - **Detection Method:** identifies the detection method for weakness.
  - **Description:** how this method can be applied to a specific weakness.
  - ***Effectiveness:*** says how effective the detection method may be in detecting the associated weakness.
  - ***Effectiveness Notes***: additional discussion of the strengths and shortcomings of this detection method

Note: *Italics* indicate that the element is optional

Detection Methods Labels

- Labels for the Detection Methods:
  - Automated Analysis
  - Automated Dynamic Analysis
  - Automated Static Analysis
  - Automated Static Analysis - Source Code
  - Automated Static Analysis - Binary or Bytecode
  - Fuzzing
  - Manual Analysis
  - Manual Dynamic Analysis
  - Manual Static Analysis
  - Manual Static Analysis - Source Code
  - Manual Static Analysis - Binary or Bytecode
  - White Box
  - Black Box
  - Architecture or Design Review
  - Dynamic Analysis with Manual Results Interpretation
  - Dynamic Analysis with Automated Results Interpretation
  - Formal Verification
  - Simulation / Emulation
  - Other

Mitigations vs Detection Methods

	Potential Mitigations (How to fix)	Detection Methods (How to find)
Conveys	A decision, action, or practice intended to reduce the impact or eliminate a weakness[1].	Information about what types of assessment activities that a weakness can be found by.
Intended for	For designers and developers	For reviewers, testers, and assessors
Notes	• Only mitigations that are practical and actionable. • Incudes a phase element indicates the development life cycle phase during which this mitigation may be applied.	• Assumes the use of best-of-breed tools, analysis, and methods. • Includes strengths and limitations. • There is limited consideration for financial costs, labor, or time.

Common Mistake:

Testing is often thought of as a mitigation, but in CWE it is generally regarded as a detection method. You can't mitigate a weakness in testing; the weakness still exists. But testing is one way to detect that a weakness even exists.

Example

CWE-1328: Security Version Number Mutable to Older Versions

Potential Mitigations

Phase: Architecture and Design

When architecting the system, security version data should be designated for storage in registers that are either read-only or have access controls that prevent modification by an untrusted agent.

Phase: Implementation

During implementation and test, security version data should be demonstrated to be read-only and access controls should be validated.

Detection Methods

Automated Dynamic Analysis

Mutability of stored security version numbers and programming with older firmware images should be part of automated testing.

Effectiveness: High

Architecture or Design Review

Anti-roll-back features should be reviewed as part of Architecture or Design review.

Effectiveness: High

Addressing Observed Example Gaps for HW CWE

OBEX Gaps Agenda

- **Current Need**
- **OBEX Challenges**
- **Participation and Organizing Work**
- **How you can help**

Observed Examples Working Group

- **Goal:** Ensure every HW CWE has at least one OBEX.
  - The OBEX element is important in that it links real-world examples to weaknesses.
  - 61 (52%) of HW CWEs lack observed examples.
  - Your contribution can reduce this gap.

- **Timeframe and Commitment:** Next release cycle, approx., March through June.
  - We value your time and expertise, so would like to structure the workgroup so that participation can be flexible and asynchronous.
  - Target generating OBEXs for 10% - 20% of the HW CWEs missing OBEXs for this release.

- **Recognition:** All contributors will be acknowledged for their valuable input.
  - This is a great opportunity to gain recognition within the community and add a contribution to your professional portfolio.

CWE-1300: Improper Protection of Physical Side Channels

Description

The device does not contain sufficient protection mechanisms to prevent physical side channels from exposing sensitive information due to patterns in physically observable phenomena such as variations in power consumption, electromagnetic emissions (EME), or acoustic emissions.

Extended Description

An adversary could monitor and measure physical phenomena to detect patterns and make inferences, even if it is not possible to extract the information in the digital domain.

Physical side channels have been well-studied for decades in the context of breaking implementations of cryptographic algorithms or other attacks against security features. These side channels may be easily observed by an adversary with physical access to the device, or using a tool that is in close proximity. If the adversary can monitor hardware operation and correlate its data processing with power, EME, and acoustic measurements, the adversary might be able to recover of secret keys and data.

Observed Examples

Reference	Description
CVE-2021-3011	electromagnetic-wave side-channel in security-related microcontrollers allows extraction of private key
CVE-2013-4576	message encryption software uses certain instruction sequences that allows RSA key extraction using a chosen-ciphertext attack and acoustic cryptanalysis
CVE-2020-28368	virtualization product allows recovery of AES keys from the guest OS using a side channel attack against a power/energy monitoring interface.
CVE-2019-18673	power consumption varies based on number of pixels being illuminated in a display, allowing reading of secrets such as the PIN by using the USB interface to measure power consumption

OBEX Challenge #1 – Specific to HW CWE

- **CVE has limited coverage for hardware specific vulnerabilities**
- **Examples:**
  - **Did not find a CVE that could be mapped to the above CWEs**
    - CWE-1351: Improper Handling of Hardware Behavior in Exceptionally Cold Environments did not turn up any results
    - CWE-1338: Improper Protections Against Hardware Overheating
    - CWE-1334: Unauthorized Error Injection Can Degrade Hardware Redundancy
  - **Found CVE that may map but not enough details to be sure**
    - CWE-1328: Security Version Number Mutable to Older Versions
  - **Found CVE that map**
    - CWE-1326:Missing Immutable Root of Trust in Hardware
    - CVE-2022-38773, CVE-2022-28383, CVE-2023-22955

How to help: HW Vulnerability Sources

- **Since CVE has limited coverage for hardware vulnerabilities, we need to consult other sources for observed examples.**
- **We like to tap into the collective knowledge and expertise of this group.**
- **Please provide sources other than CVE that we could use to pull observed examples from for HW CWEs.**
- **Preferably site that aggregate vulnerability reports.**
- **[WORKING ITEM] Sources for Hardware Vulnerability Reports**
  - https://github.com/CWE-CAPEC/hw-cwe-sig/issues/109

Observed Example Element

- Contains one or more publicly reported vulnerabilities in real-world products that exhibit the weakness.

- **Sub-Elements**
  - **Reference**: This contains ~~the CVE Identifier, e.g., CVE-2005-1951~~ a vulnerability identifier.
  - **Description**: Clear, simple, and concise summary of ~~CVE~~ vulnerability report that focuses on the link between the ~~CVE~~ report and weakness. Exclude product name, attack vectors and other irrelevant details.
  - **Link**: URL Link to ~~CVE~~ vulnerability report. ~~Preferably from https://www.cve.org/.~~

OBEX Challenge #2 – General CWE Challenge

- **Vulnerability reports are not written from a weakness aspect**
  - The original weakness is not always covered.
  - From a vulnerability management perspective, the underlying weakness may not actually be important to the organization.
  - Reports emphasize product impact, product versions affected, and how easy it is for an attacker to exploit.
  - Routinely we see that many of the vulnerability descriptions do not have enough information to determine what the underlying weakness is.
  - For OBEXs we cannot infer the weakness, the vulnerability report must specifically describe the weakness.

How to help: Submitting an OBEX

1. **Find HW CWE(s) you would like to contribute an OBEX.**
   - There is a list maintained on our GitHub.
   - There is an issue per HW CWE that is missing an OBEX.
   - NVD Data may be a help here
2. **Assign yourself to the GitHub Issue.**
3. **In the Issue comments, provide a CVE Number, URL, description, and how you would like to be cited for the contribution.**
   - Preferred Name and organization.

Issues are here:

https://github.com/CWE-CAPEC/hw-cwe-sig/labels/Missing%20OBEX

Note: We are planning to release a OBEX Style guide soon

Summary

Our ask

1. **Commit to supporting the working group.**
   - Send an email to cwe@mitre.org, subject: OBEX WG.

2. **Provide additional HW Vulnerability Sources.**
   - https://github.com/CWE-CAPEC/hw-cwe-sig/issues/109

3. **Generate 1 OBEX or many OBEXs.**
   - https://github.com/CWE-CAPEC/hw-cwe-sig/labels/Missing%20OBEX

Next Meeting (Mar 8)

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

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 Detection Methods**
  - How many do we have? CREATE A TRACKER
- **Missing demonstrative examples (DEMOX)**
  - 15 HW CWEs missing demonstrative examples (down 1)
    - 1 added from Hack@DAC, CWE-440
    - Note: there are other DEMOXs from Hack@DAC but now adding DEMOXs to entries that have an existing DEMOX
  - How many DEMOX's are not code based? CREATE A TRACKER

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

Discussion Items on GitHub

Resonant frequency weakness, proposal (Topic Lead: OPEN)	• https://github.com/CWE-CAPEC/hw-cwe-sig/issues/105
Covert Channel Coverage in HW View (Topic Lead: OPEN)	• https://github.com/CWE-CAPEC/hw-cwe-sig/issues/108
CWE Coverage of HW Cryptography (Topic Lead: OPEN)	• https://github.com/CWE-CAPEC/hw-cwe-sig/issues/7
Lifecycle-stage classification for HW CWEs –Dan DiMase (Topic Lead: OPEN)	• https://github.com/CWE-CAPEC/hw-cwe-sig/issues/4

Covert Channel Coverage in CWE

COVID-Bit Research Item[1][2]

- **In 2022, researchers at Ben Gurion University in Israel developed a new data exfiltration method for air-gapped systems called COVID-bit.**
- **Malware generates electromagnetic radiation in the 0-60 kHz frequency band (assumes Malware got there somehow).**
- **EM emissions are generated by manipulating the workload of the CPU. Claims of indirect control SMPS.**
- **The electromagnetic radiation generated by this intentional process can be received from a distance using appropriate antennas.**

1. *https://thehackernews.com/2022/12/covid-bit-new-covert-channel-to.html?m=1*
2. *https://arxiv.org/abs/2212.03520*

Covert Channels and Side Channels

- Initial thought was that COVID-Bit could be a DEMOX for CWE-1300: Improper Protection of Physical Side Channels
- As HW SIG Members had correctly pointed out, COVID-Bit is about Covert Channels and NOT Side Channels
- Covert Channel (CC) / Side Channel (SC)
  - Intentional transmission (CC). Accidental transmission (SC) – *Ross Anderson* [1]
  - Adversary controls input and output (CC). Adversary can only read output (SC) – *Intel* [2]
  - Not an intended resource but exists due the application's behaviors. –*CWE-514 Notes* [3]
- If not CWE-1300 (SC), where would something like this map to in HW view?
- Closest we have is CWE-514: Covert Channels

1. https://www.cl.cam.ac.uk/~rja14/Papers/SEv3-ch19-7sep.pdf
2. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/refined-speculative-execution-terminology.html
3. https://cwe.mitre.org/data/definitions/514.html

CWE-514: Covert Channel

https://cwe.mitre.org/data/definitions/514.html

Abstraction: Class

Description: A covert channel is a path that can be used to transfer information in a way not intended by the system's designers.

Extended Description: Typically the system has not given authorization for the transmission and has no knowledge of its occurrence.

Relationships:

ChildOf	CWE-1229:Creation of Emergent Resource
ParentOf	CWE-385:Covert Timing Channel
ParentOf	CWE-515: Covert Storage Channel
CanFollow	CWE-205: Observable Behavioral Discrepancy

Vulnerability Mapping Notes:

Usage: Allowed-with-Review; **Reason:** Abstraction

Rationale: This CWE entry is a Class and might have Base-level children that would be more appropriate; **Comments:** Examine children of this entry to see if there is a better fit.

NOTE: Nothing about EM based Covert Channels, nor HW cause, e.g., SMPS

Discussion

Questions:

- Should we place CWE-514 in the HW View?
- Do we need to modify CWE-514 to be less software centric?
- Or create a base of CWE-514 and put that into the HW view?

Previous HW SIG Member Comments:

- Covert Channels should have coverage in the hardware view *–Jason Oberg*
- Covert Channels should be in the HW categories Security Flow Issues, General Circuit and Logic Design Concerns, or Debug and Test Problems. – *Paul Wortman*

Most Important Hardware Weaknesses Refresh

Bob H

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?**

Current MIHW

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

New HW CWEs Since MIHW

- **CWE-1342: Information Exposure through Microarchitectural State after Transient Execution**
- **CWE-1357: Reliance on Insufficiently Trustworthy Component**
- **CWE-1384: Improper Handling of Physical or Environmental Conditions**
- **CWE-1388: Physical Access Issues and Concerns**

Discussion

- **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?**
- **Are there observational trends that would change the current list in any significant and meaningful way?**

OBEX WG Formation

- **Purpose:** Populating missing observed examples (OBEXs) in HW CWEs.
  - Your expertise can make a difference!
- **Challenge:** 61 (52%) of HW CWEs lack observed examples.
  - Your contribution can significantly reduce this gap.
- **Goal:** Ensure every HW CWE has at least one OBEX
  - You can help enhance the quality and comprehensiveness CWE.
- **Motivation:** The OBEX element is important for new users
  - Your input will directly impact the ease of use
- **Timeframe and Commitment:** March through June. Participation is flexible and asynchronous, allowing you to contribute when it suits you best.
  - We value your time and expertise.
- **Recognition:** All contributors will be acknowledged for their valuable input.
  - This is a great opportunity to gain recognition within the community and add a significant contribution to your professional portfolio.