Hardware CWE™
Special Interest Group (SIG)

Chair: Bob Heinemann (MITRE)

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

MITRE Team: Steve Christey Coley, Alec Summers

MITRE

November 8, 2024

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

Agenda

REMINDER: This meeting is being recorded.

1	Security Issues Arising from Hardware Design *Continuation from October Meeting*	Joerg Bormann	40 min
2	CWE Entry in Development: Lack of Feedback for Unexecuted Operations Across System Interfaces.	Amisha Srivastava	15 min

Housekeeping

- **Schedule:**
  - **Next Meeting: Dec 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) pilot now on GitHub! Open to anyone by request. Public access in the next few months.**
- **CWE 4.16 release is planned for November.**
- **CWE 5.0 is planned for early 2025.**

Call for Topics

What topics should we cover next time?

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

Security Issues Arising from Hardware Design

Joerg Bormann

CWE Entry in Development: Lack of Feedback for Unexecuted Operations Across System Interfaces.

Amisha Srivastava

CWE Entry: Lack of Feedback for Unexecuted Operations Across System Interfaces.

Presented by:

Amisha Srivastava
PhD Candidate

The University of Texas at Dallas

Weakness Description

- **Brief Description**
  - Systems fail to notify or log the non-execution or disregard of operations due to various reasons including system constraints, design decisions, or errors.
- **Intended Behavior**
  - Systems should be designed to securely manage unauthorized attempts and provide comprehensive feedback about each action.
- **Scope**: Occurs across various hardware interfaces, applicable to both software and hardware systems.
- **Example Systems:**
  - SoCs like OpenTitan.
  - Microcontroller interrupt systems: Silent handling of interrupt conflicts.
  - Network interface controllers: Dropped packets due to buffer overflow without notification.

Modes of Introduction & Applicable Platforms

- **Phases:**
  - Architecture & Design: Weakness introduced by inadequate error-reporting mechanisms.
  - Implementation: Lack of logging for critical operations by developers leading to silent failures.
- **Languages**: Common in C, C++, Verilog.
- **Operating Systems**:
  - Especially prevalent in embedded and general OS-agnostic environments.
- **Hardware**:
  - Frequently affects embedded systems, SoCs, and microcontrollers.

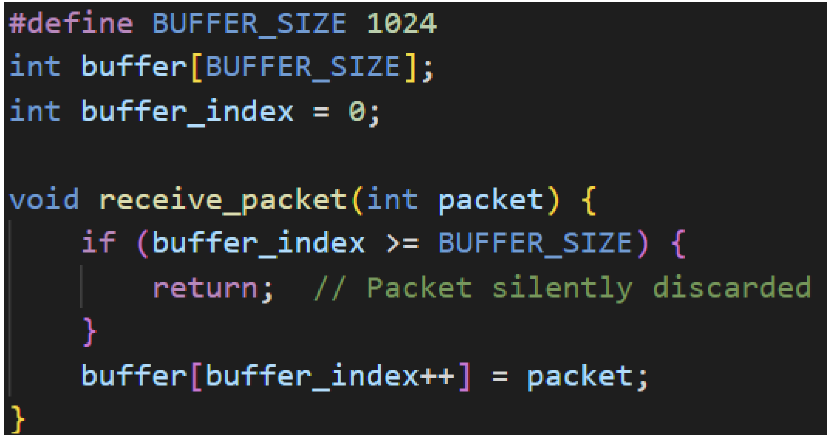
Consequences

- **Confidentiality**:
  - Possible exposure of data when operations silently fail.
  - Allowing attackers to exploit the lack of feedback.
- **Integrity**:
  - Data corruption due to unacknowledged operational failures.
  - Operations may proceed based on incorrect assumptions.
- **Availability**:
  - Resource exhaustion can cause system crashes or denial of service.
  - Unhandled discarded operations can lead to resource exhaustion.

1
CONFIDENTIALITY
INFORMATION SECURITY
2
INTEGRITY
3
AVAILABILITY

Demonstrative Example

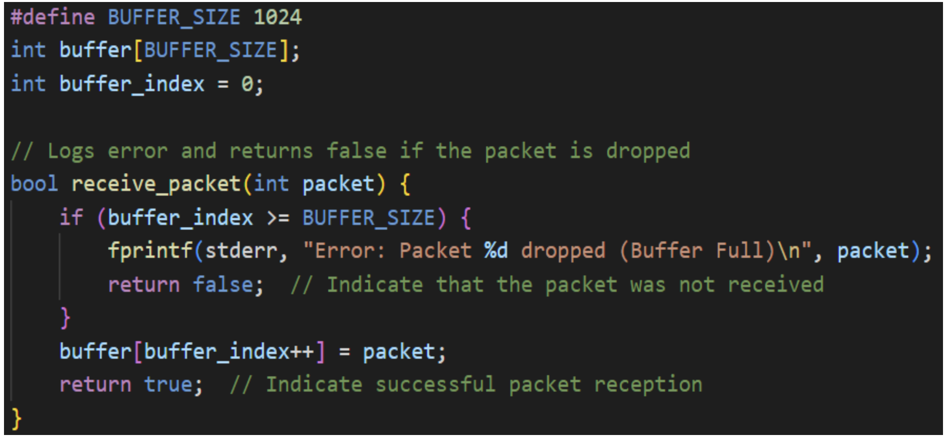
- This example demonstrates how network packets can be lost without notification when a buffer overflows.
- Bad Code: Packets are discarded when the buffer is full without any indication.
- Good Code: Structured logging provides visibility, helping in troubleshooting.



```
#define BUFFER_SIZE 1024
int buffer[BUFFER_SIZE];
int buffer_index = 0;

void receive_packet(int packet) {
    if (buffer_index >= BUFFER_SIZE) {
        return;  // Packet silently discarded
    }
    buffer[buffer_index++] = packet;
}
```

Bad Code



```
#define BUFFER_SIZE 1024
int buffer[BUFFER_SIZE];
int buffer_index = 0;

// Logs error and returns false if the packet is dropped
bool receive_packet(int packet) {
    if (buffer_index >= BUFFER_SIZE) {
        fprintf(stderr, "Error: Packet %d dropped (Buffer Full)\n", packet);
        return false;  // Indicate that the packet was not received
    }
    buffer[buffer_index++] = packet;
    return true;  // Indicate successful packet reception
}
```

Good Code

Example: OpenTitan SoC

- The weakness was found in the handling of write requests to reserved addresses in the mailbox implementation in the OpenTitan SoC.

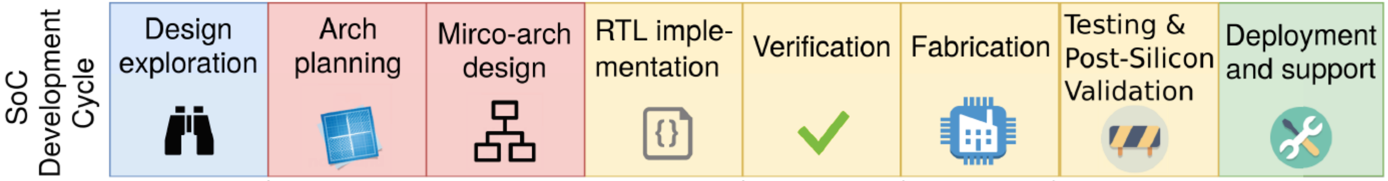
- When a write request is made to a reserved address, the system correctly identifies this as an error and discards the write operation.

- However, the system fails to provide feedback when a write operation to a reserved address is discarded.

- This lack of feedback could potentially be exploited by an attacker to induce unpredictable behavior by inserting malicious writes into the code.

Potential Mitigations

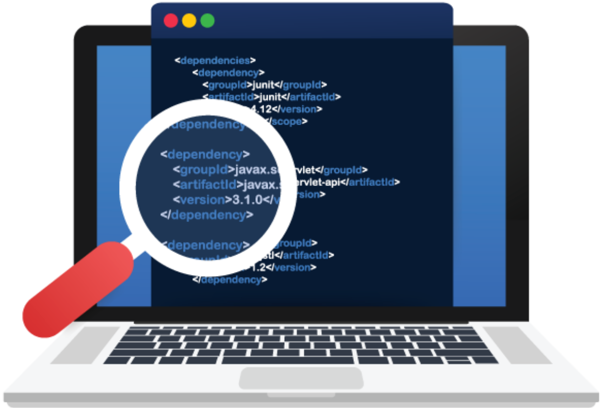
- Architecture & Design:
  - Incorporate logging/feedback mechanisms to handle discarded operations.
  - Effectiveness: High
- Implementation:
  - Ensure logging and error reporting for critical functions.
  - Effectiveness: Moderate
- Implementation-level checks complement design-phase measures.



Detection Methods

- Automated Static Analysis:
  - ○ Scans code for missing error handling or feedback mechanisms.
  - ○ High effectiveness for identifying missing feedback mechanisms.

- Manual Code Review:
  - ○ Experts manually inspect the code for unhandled operations.
  - ○ Moderate effectiveness, identifies design-level issues.



THANK YOU

Next Meeting (Dec 13)

CWE@MITRE.ORG

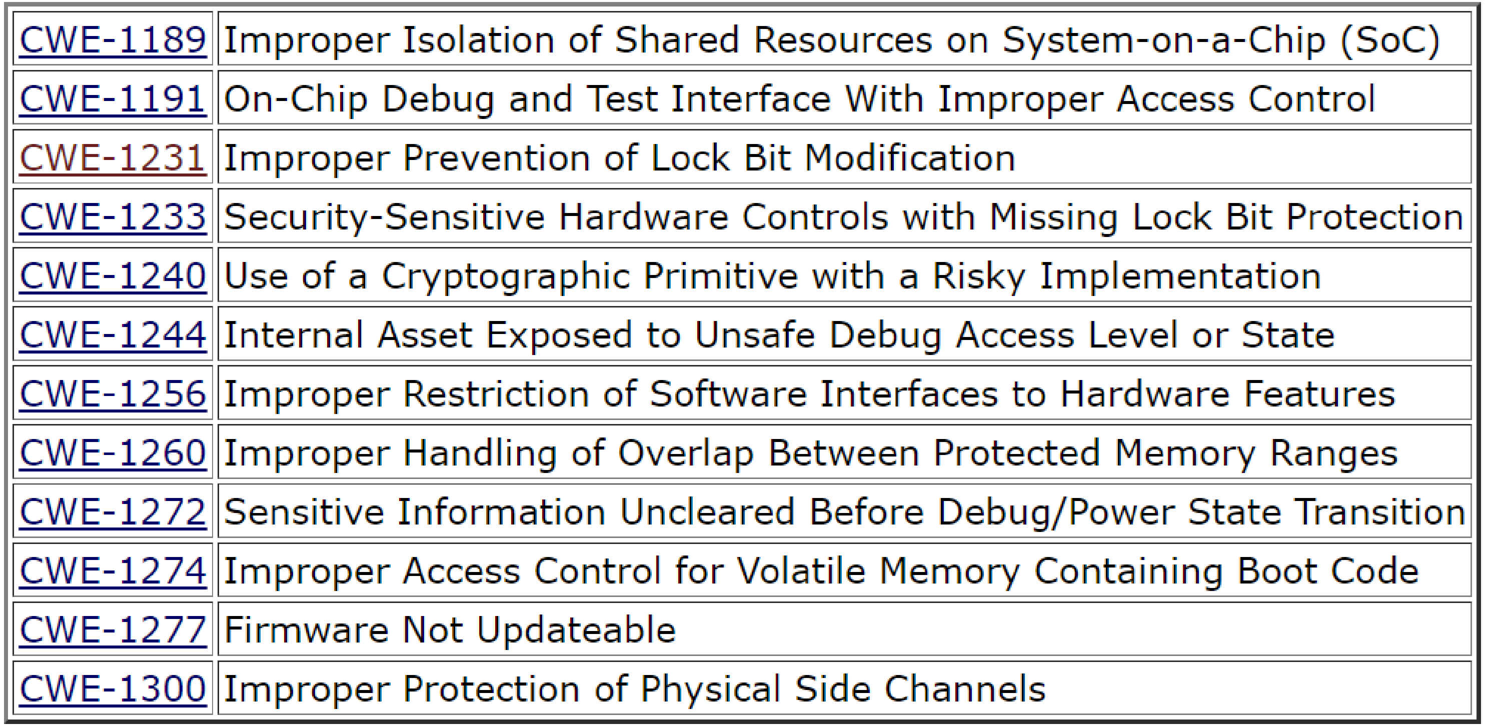
- **Mailing List:** *hw-cwe-special-interest-group-sig-list@mitre.org*
  - ***NOTE: All mailing list items are archived publicly at:***
    - *https://www.mail-archive.com/hw-cwe-special-interest-group-sig-list@mitre.org/*
- **What would members of this body like to see for the next HW SIG agenda?**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

Backup Slides

Most Important Hardware Weaknesses Refresh

Bob H

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**

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?**

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.**