Hardware CWE™ Special Interest Group (SIG)

Chair: Bob Heinemann (MITRE)

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

MITRE Team: Gage Hackford, Steve Christey Coley, Alec Summers

MITRE

July 12, 2024

Agenda

REMINDER: This meeting is being recorded.

1	Call for Topics	Bob Heinemann	10 min
2	Presentation of Accepted New HW Submission *Lack of Feedback for Unexecuted Operations Across System Interfaces*	Amisha Srivastava	30 min
3	Covert Channels Proposal	Hareesh Khattri	20 min

Housekeeping

- **Schedule:**
  - **Next Meeting: Aug 9**
    - **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 4.15 release will be on July 16.**

Call for Topics

What topics should we cover next time?

Accepted HW Submission

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

Presented by:

Amisha Srivastava
PhD Student

Dr. Kanad Basu
Advisor

The University of Texas at Dallas

Understanding the General Weakness

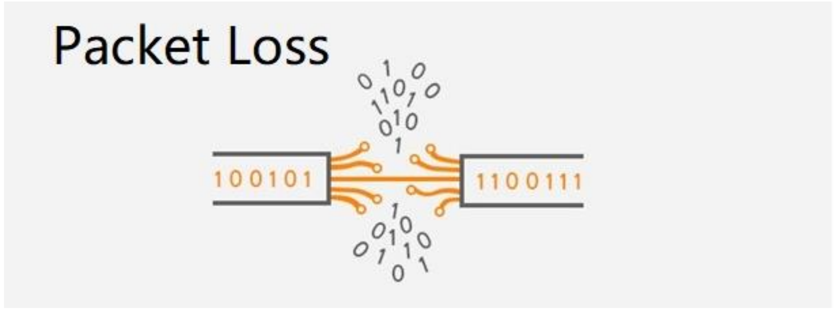
- Across diverse systems, from microcontrollers to large-scale SoC environments, operations may be discarded due to design constraints or errors, often without notifying the users or systems involved.

- This weakness, specifically targets scenarios where not just errors, but unexecuted or ignored operations lack feedback.

- The absence of such feedback can lead to critical issues including data loss, system instability, and potential exploitation by malicious entities.

Implications

- Silent operational failures can range from benign to catastrophic, influencing both user experience and system reliability.
- In hardware like SoCs, this can mislead developers into assuming successful operations, compounding errors in already complex systems.
- Similar impacts are observed in network systems where dropped packets due to buffer overflows go unreported, leading to communication failures and data loss.



Data Loss in network systems.

Identified Vulnerability

- **Intended Behavior**
  - Systems should be designed to securely manage unauthorized attempts to access protected resources by and providing comprehensive feedback about each action.

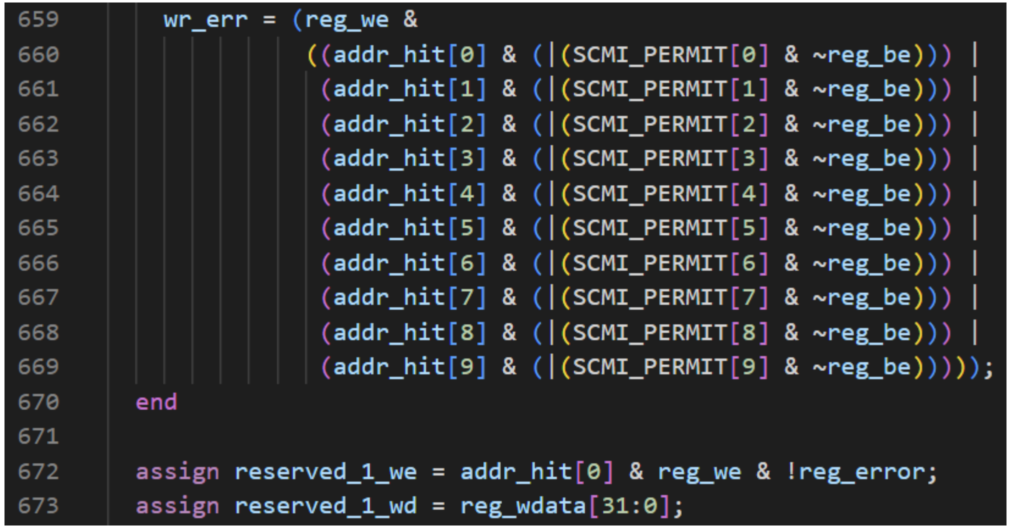
- **Brief Description**
  - Systems fail to notify or log the non-execution or disregard of operations due to various reasons including system constraints, design decisions, or errors.

- **Affected Technology**: OpenTitan System-on-Chip
  - Specifically, the scmi_reg_top module used by its register generation tool.

OpenTitan Vulnerability

- The weakness lies in the handling of write requests to reserved addresses in the mailbox implementation in the OpenTitan SoC.
- The register generation tool (Reggen) is used to generate the registers for the mailbox.
- 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 disrupt the system’s operation or induce unpredictable behavior by inserting malicious writes into the code.

OpenTitan Vulnerability: RTL Analysis



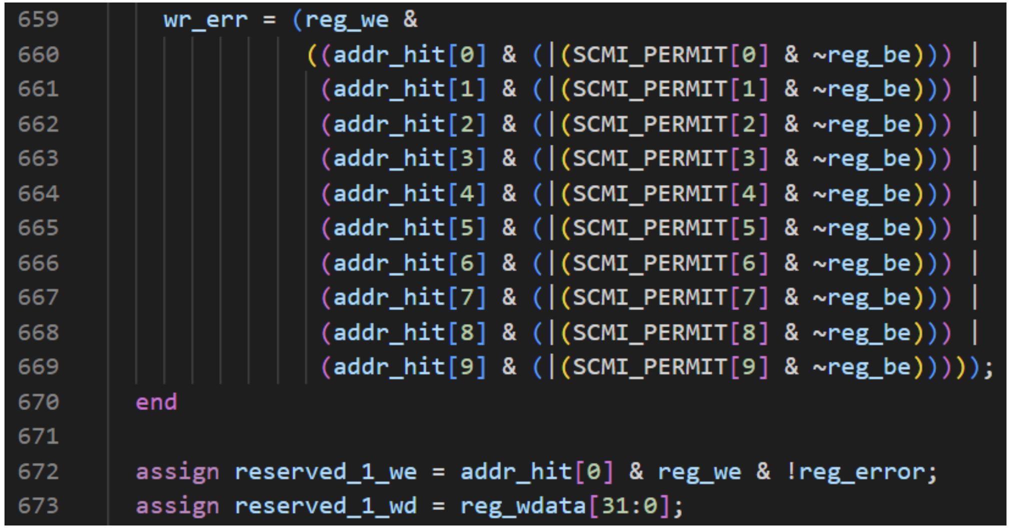
```
      wr_err = (reg_we &
               ((addr_hit[0] & (|(SCMI_PERMIT[0] & ~reg_be))) |
                (addr_hit[1] & (|(SCMI_PERMIT[1] & ~reg_be))) |
                (addr_hit[2] & (|(SCMI_PERMIT[2] & ~reg_be))) |
                (addr_hit[3] & (|(SCMI_PERMIT[3] & ~reg_be))) |
                (addr_hit[4] & (|(SCMI_PERMIT[4] & ~reg_be))) |
                (addr_hit[5] & (|(SCMI_PERMIT[5] & ~reg_be))) |
                (addr_hit[6] & (|(SCMI_PERMIT[6] & ~reg_be))) |
                (addr_hit[7] & (|(SCMI_PERMIT[7] & ~reg_be))) |
                (addr_hit[8] & (|(SCMI_PERMIT[8] & ~reg_be))) |
                (addr_hit[9] & (|(SCMI_PERMIT[9] & ~reg_be)))));
    end

    assign reserved_1_we = addr_hit[0] & reg_we & !reg_error;
    assign reserved_1_wd = reg_wdata[31:0];
```

Code Snippet highlighting the location in the RTL.

- In the scmi_reg_top module, write operations are handled based on address validity and permission checks.
- The wr_err signal is set to true when there's an attempt to write to an address where permissions are restricted.
- While the system does halt the write to reserved addresses, it fails to alert any subsystem or log, creating a 'silent fail' scenario.

OpenTitan Vulnerability: RTL Analysis



```
    wr_err = (reg_we &
             ((addr_hit[0] & (|(SCMI_PERMIT[0] & ~reg_be))) |
              (addr_hit[1] & (|(SCMI_PERMIT[1] & ~reg_be))) |
              (addr_hit[2] & (|(SCMI_PERMIT[2] & ~reg_be))) |
              (addr_hit[3] & (|(SCMI_PERMIT[3] & ~reg_be))) |
              (addr_hit[4] & (|(SCMI_PERMIT[4] & ~reg_be))) |
              (addr_hit[5] & (|(SCMI_PERMIT[5] & ~reg_be))) |
              (addr_hit[6] & (|(SCMI_PERMIT[6] & ~reg_be))) |
              (addr_hit[7] & (|(SCMI_PERMIT[7] & ~reg_be))) |
              (addr_hit[8] & (|(SCMI_PERMIT[8] & ~reg_be))) |
              (addr_hit[9] & (|(SCMI_PERMIT[9] & ~reg_be)))));
  end

  assign reserved_1_we = addr_hit[0] & reg_we & !reg_error;
  assign reserved_1_wd = reg_wdata[31:0];
```

Code Snippet highlighting the location in the RTL.

- The logic within this block evaluates write permissions against byte-enable signals to detect invalid writes.
- However, even when such writes are detected, the system lacks a mechanism to communicate these errors back to the user or other system processes.
- This configuration suppresses the write but does not trigger any alert or error message, leaving the system's state ambiguous to its operators.

Consequences

- An attacker could potentially exploit this vulnerability in several ways:
  - **Data Corruption**: By writing to reserved addresses, an attacker could overwrite important data, causing corruption. This could lead to system instability or even a complete system crash.
  - **Denial of Service (DoS)**: If an attacker repeatedly writes to these addresses, it could cause the system to become overloaded and unresponsive, effectively resulting in a DoS attack.
  - **Unauthorized Access**: In some cases, writing to certain reserved addresses could potentially alter the system’s control flow or even grant unauthorized access.
  - **Information Disclosure**: If the system behaves differently based on the success or failure of the write operation, an attacker might be able to use this as a side channel to gain information about the system’s state.

Widespread Impact in Hardware Systems

Microcontroller systems often ignore interrupts due to priority conflicts without signaling, exacerbating operational unpredictability.

Unreported errors during FPGA configuration can lead to device malfunctions, system instability, and potential security risks

This emphasizes the need for robust feedback mechanisms across all hardware interfaces to ensure system integrity and reliability.

Related Weaknesses

- CWE-392: Missing Report of Error Condition
  - Definition: This weakness occurs when the product fails to report an error when such reporting is useful or expected.
  - The primary relationship of our weakness should be with CWE-392, emphasizing the system's failure to report an error condition.
  - This alignment more accurately reflects the nature of the weakness and aids in clearly defining mitigation strategies.

THANK YOU

Covert Channels

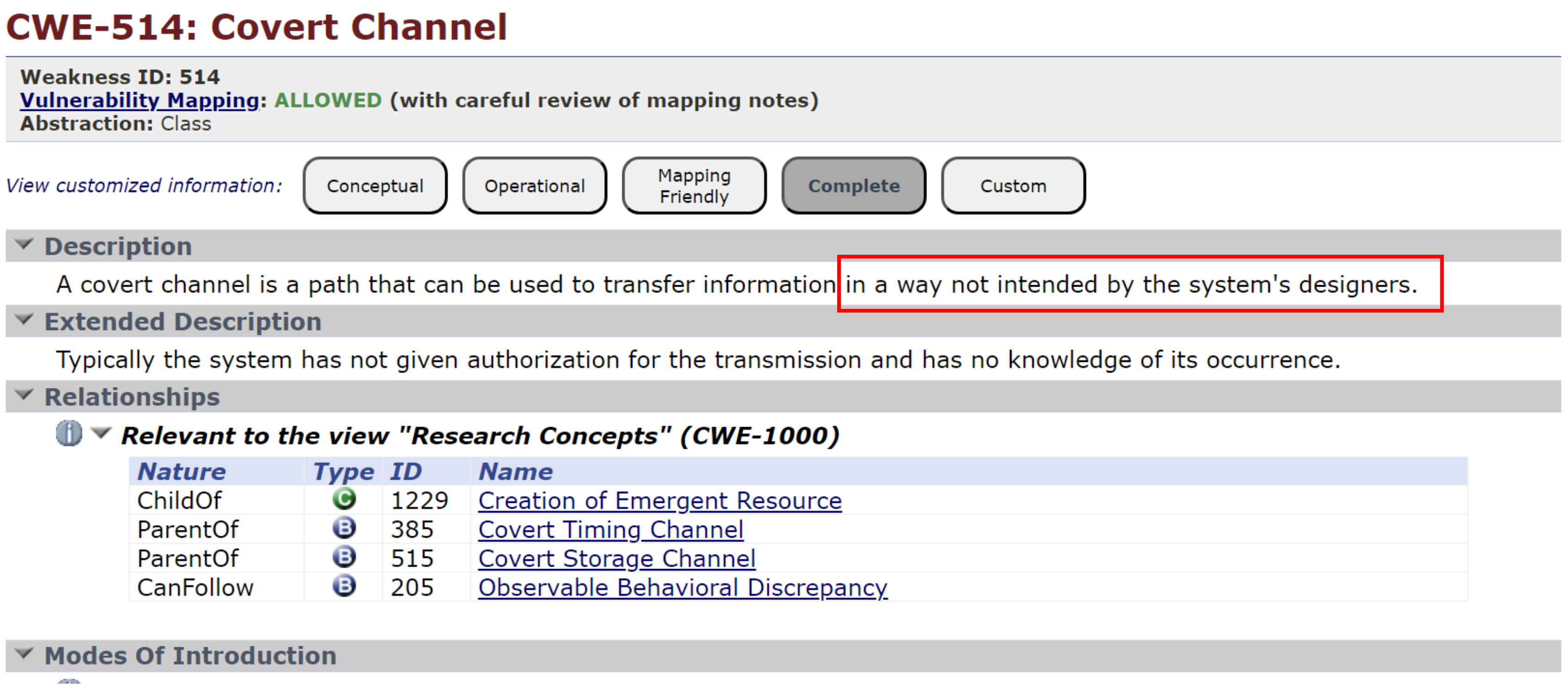
Hareesh Khattri

HW CWE Covert Channel & Designer Intent

- **Covert Channel CWEs**
  - Some CWEs descriptions are more from attacker's perspective and intent to exploit weaknesses, rather than Designer intent
  - Question : Should these be updated to clarify and emphasize that design intent part or is just observable discrepancy sufficient for a weakness?
    - Context is some products or designs have explicit requirements to protect against such data exposure vs others that don't care.

- **Designer Intent Example**
  - **CWE-514: Covert Channel** **:** "A covert channel is a path that can be used to transfer information in a way not intended by the system's designers."

- **Attacker Intent Example:**
  - **CWE-208: Observable Timing Discrepancy**
    - In security-relevant contexts, even small variations in timing can be exploited by attackers to indirectly infer certain details about the product's internal operations.
  - **CWE-385: Covert Timing Channel**
    - "Covert timing channels convey information by modulating some aspect of system behavior over time, so that the program receiving the information can observe system behavior and infer protected information."



CWE-514: Covert Channel

Weakness ID: 514
Vulnerability Mapping: ALLOWED (with careful review of mapping notes)
Abstraction: Class

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

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

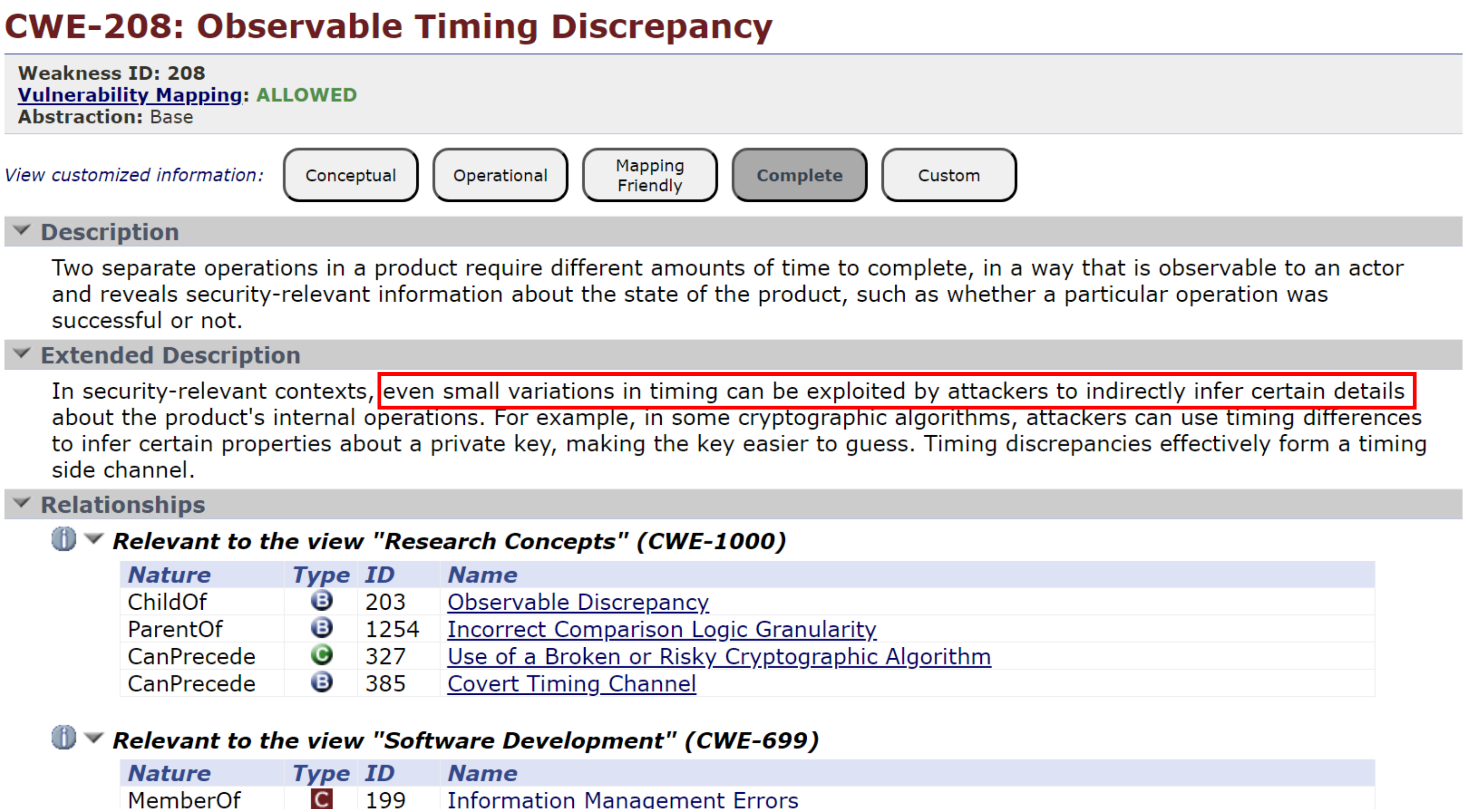
Typically the system has not given authorization for the transmission and has no knowledge of its occurrence.

Relationships

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

Nature	*Type*	*ID*	*Name*
ChildOf	C	1229	Creation of Emergent Resource
ParentOf	B	385	Covert Timing Channel
ParentOf	B	515	Covert Storage Channel
CanFollow	B	205	Observable Behavioral Discrepancy

Modes Of Introduction



CWE-208: Observable Timing Discrepancy

Weakness ID: 208
Vulnerability Mapping: ALLOWED
Abstraction: Base

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

Description

Two separate operations in a product require different amounts of time to complete, in a way that is observable to an actor and reveals security-relevant information about the state of the product, such as whether a particular operation was successful or not.

Extended Description

In security-relevant contexts, even small variations in timing can be exploited by attackers to indirectly infer certain details about the product's internal operations. For example, in some cryptographic algorithms, attackers can use timing differences to infer certain properties about a private key, making the key easier to guess. Timing discrepancies effectively form a timing side channel.

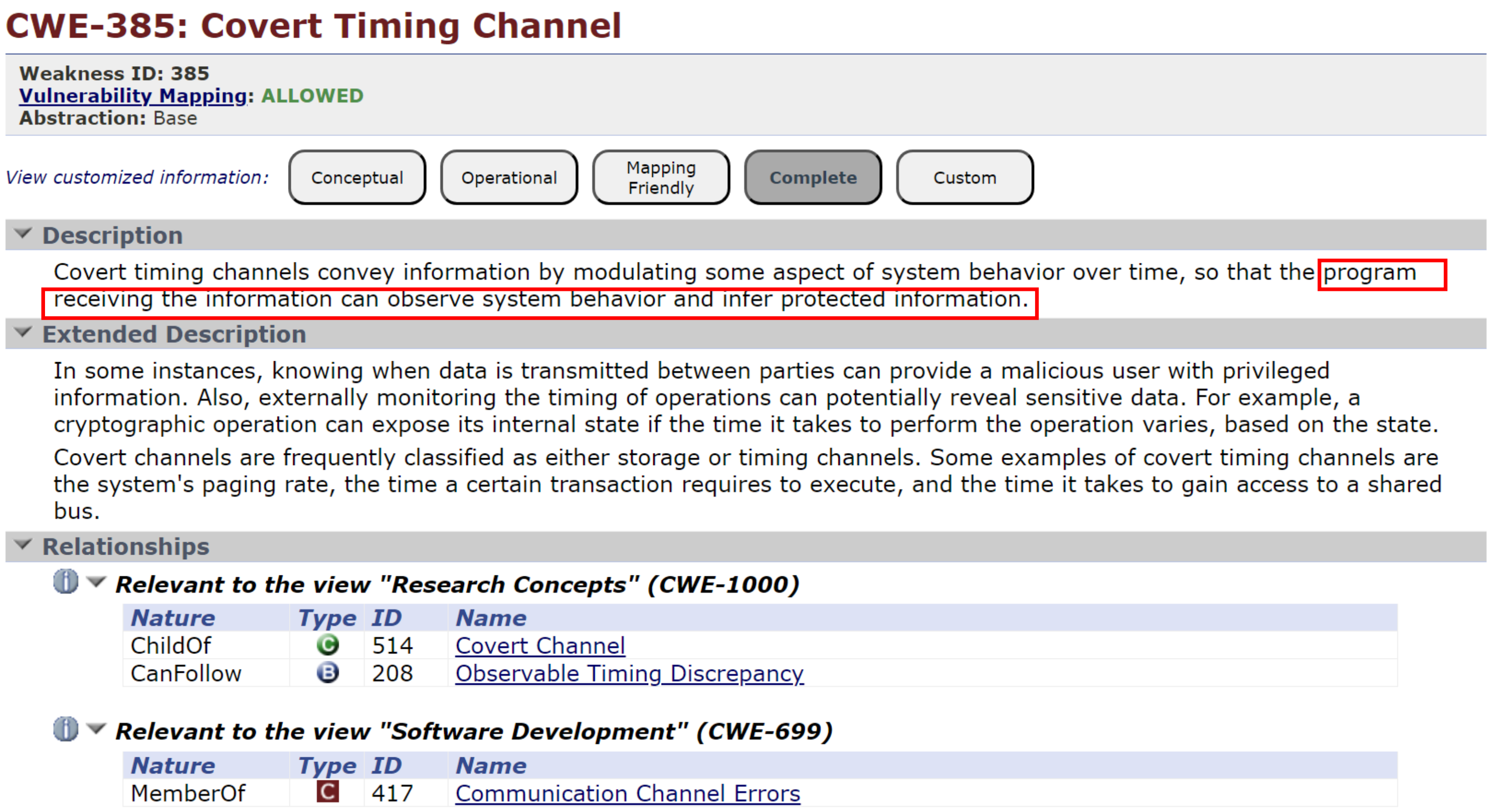
Relationships

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

Nature	Type	ID	Name
ChildOf	B	203	Observable Discrepancy
ParentOf	B	1254	Incorrect Comparison Logic Granularity
CanPrecede	C	327	Use of a Broken or Risky Cryptographic Algorithm
CanPrecede	B	385	Covert Timing Channel

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	C	199	Information Management Errors



CWE-385: Covert Timing Channel

Weakness ID: 385
Vulnerability Mapping: ALLOWED
Abstraction: Base

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

Description

Covert timing channels convey information by modulating some aspect of system behavior over time, so that the program receiving the information can observe system behavior and infer protected information.

Extended Description

In some instances, knowing when data is transmitted between parties can provide a malicious user with privileged information. Also, externally monitoring the timing of operations can potentially reveal sensitive data. For example, a cryptographic operation can expose its internal state if the time it takes to perform the operation varies, based on the state.

Covert channels are frequently classified as either storage or timing channels. Some examples of covert timing channels are the system's paging rate, the time a certain transaction requires to execute, and the time it takes to gain access to a shared bus.

Relationships

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

Nature	Type	ID	Name
ChildOf	C	514	Covert Channel
CanFollow	B	208	Observable Timing Discrepancy

Relevant to the view "Software Development" (CWE-699)

Nature	Type	ID	Name
MemberOf	C	417	Communication Channel Errors

Next Meeting (Aug 9)

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

System Verilog

System Verilog and Verilog

- Hack@DAC DEMOXs had indicated code snippets were System Verilog.
- The language enumeration in the current schema does not contain System Verilog, just Verilog.

Questions to SIG:

- What is the difference between Verilog and System Verilog?
- Is that difference significant enough that we should add System Verilog to the schema enumeration?
  - For example, we distinguish between C and C++ but not between variants of C (e.g., C89 vs C99).

Covert Channel Coverage in CWE

Covert Channel Discussion

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*
- CWE-514 as currently written it's specific to software and would need to be tweaked *–Bruce Monroe*

Questions:

- Does CWE-514 need to be modified to address more HW centric concerns?
- Are covert channels weakness focused or attacker focused?
- Coverage in HW View?
- Other questions about covert channels and HW CWE?

Discussion

https://github.com/CWE-CAPEC/hw-cwe-sig/issues/108

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[1].

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

1. Landwehr, C. E., Bull, A. R., McDermott, J. P., & Choi, W. S. (September 1994). A Taxonomy of Computer Program Security Flaws, with Examples. Information Technology Division, Code 5542, Naval Research Laboratory. Retrieved from https://dl.acm.org/doi/10.1145/185403.185412

Vulnerability Mapping Notes:

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

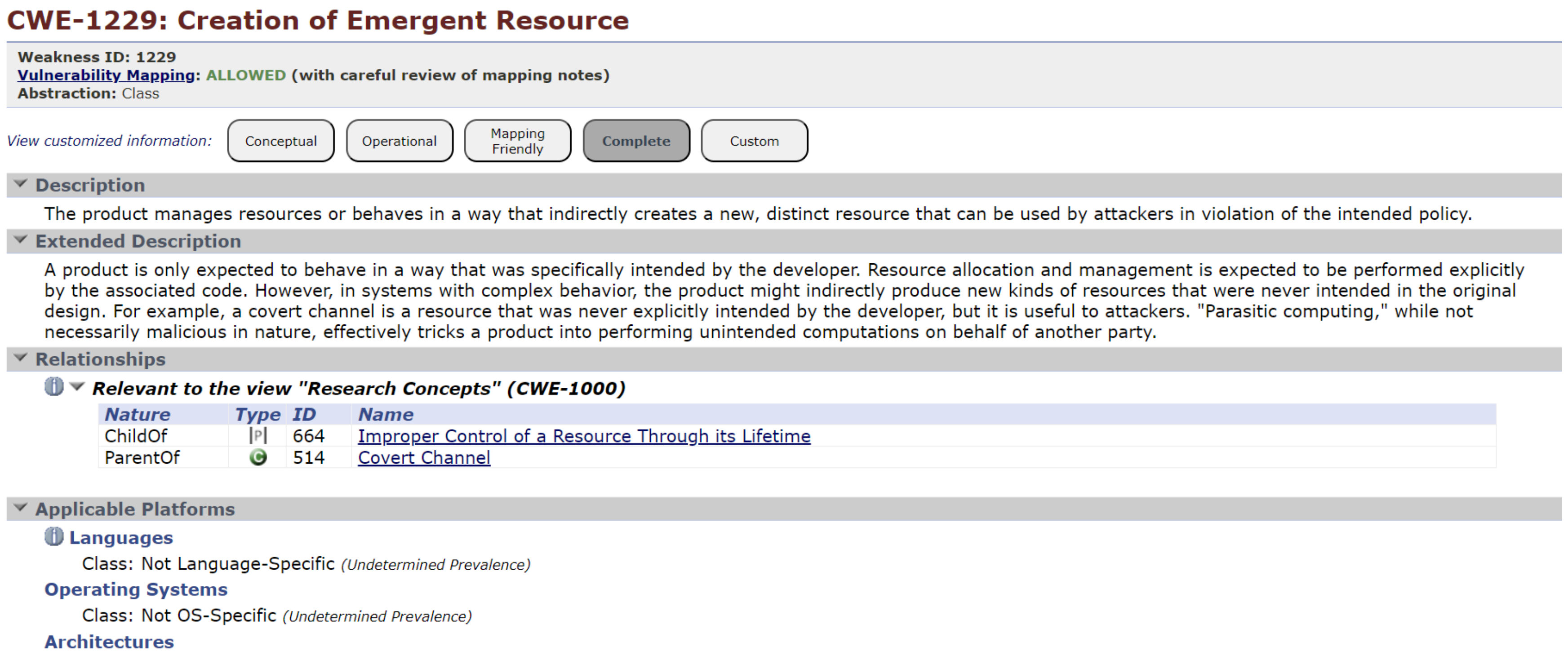
Rationale: This CWE entry is a Class and might have Base-level children that would be more appropriate; **Comments:** Examine children of this entry to see if there is a better fit.

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

Intel - Covert Channels Notes

- Incidental channels are unintended communication channels
- Inevitable due to sharing of resources
- Cannot have advanced features without shared resources
- Not feasible to remove
- Covert Channels are types of Incidental Channels
- Incidental covert channels require an adversary to not only control both ends of the incidental channel but also have access to secret information.
- This would infer a chaining relationship

Incidental Channels could be Child of or Sibling to?



CWE-1229: Creation of Emergent Resource

Weakness ID: 1229
Vulnerability Mapping: ALLOWED (with careful review of mapping notes)
Abstraction: Class

Description

The product manages resources or behaves in a way that indirectly creates a new, distinct resource that can be used by attackers in violation of the intended policy.

Extended Description

A product is only expected to behave in a way that was specifically intended by the developer. Resource allocation and management is expected to be performed explicitly by the associated code. However, in systems with complex behavior, the product might indirectly produce new kinds of resources that were never intended in the original design. For example, a covert channel is a resource that was never explicitly intended by the developer, but it is useful to attackers. "Parasitic computing," while not necessarily malicious in nature, effectively tricks a product into performing unintended computations on behalf of another party.

Relationships

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

Nature	Type	ID	Name
ChildOf	P	664	Improper Control of a Resource Through its Lifetime
ParentOf	C	514	Covert Channel

Applicable Platforms

Languages

Class: Not Language-Specific *(Undetermined Prevalence)*

Operating Systems

Class: Not OS-Specific *(Undetermined Prevalence)*

Architectures

Intel – Incidental Channels

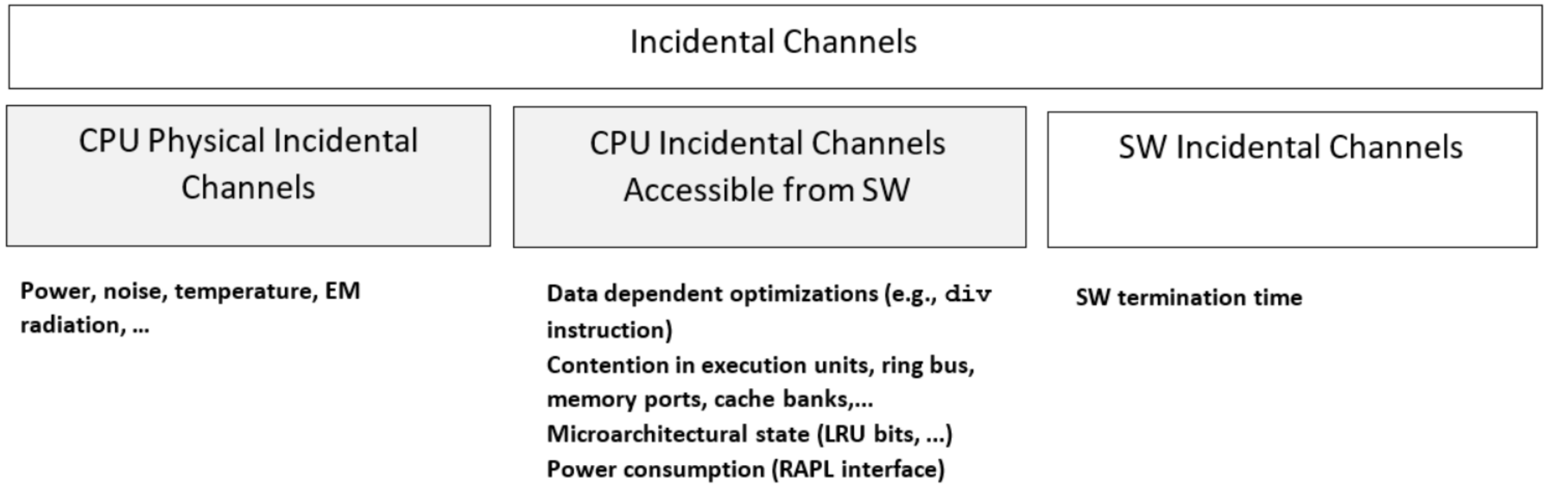

Incidental Channels
CPU Physical Incidental Channels
CPU Incidental Channels Accessible from SW
SW Incidental Channels
Power, noise, temperature, EM radiation, ...
Data dependent optimizations (e.g., div instruction)
Contention in execution units, ring bus, memory ports, cache banks,...
Microarchitectural state (LRU bits, ...)
Power consumption (RAPL interface)
SW termination time

Figure 1: Example of taxonomy terminology of incidental channels on CPUs

https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/best-practices/securing-workloads-against-side-channel-methods.html

Discussion Questions

- **Does CWE-514 need to be modified to address more HW centric concerns?**
- **Should there be a completely new covert channel like weakness created for hardware?**
- **Are covert channels weakness focused or attacker focused?**
- **Coverage in HW View?**
- **Other questions about covert channels and HW CWE?**