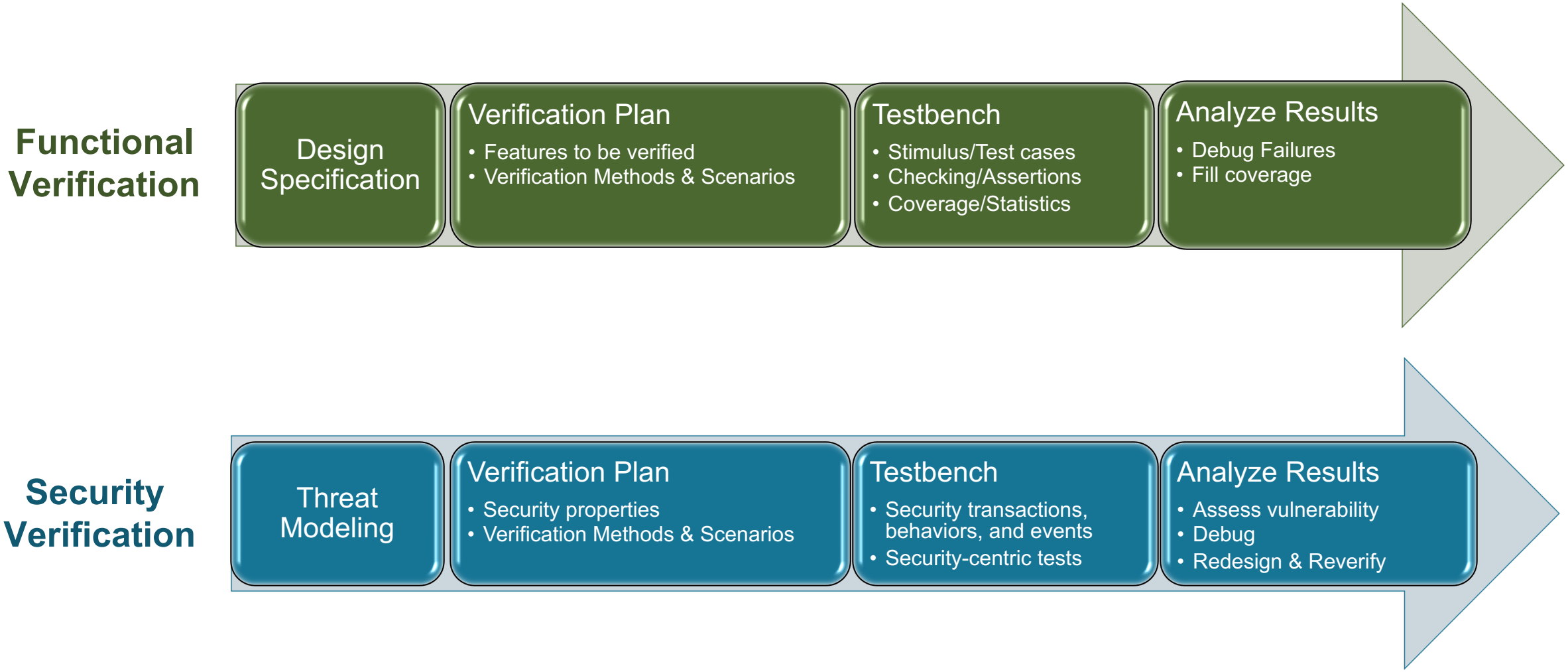
Hardware Security Coverage

Ryan Kastner
http://kastner.ucsd.edu

UC San Diego

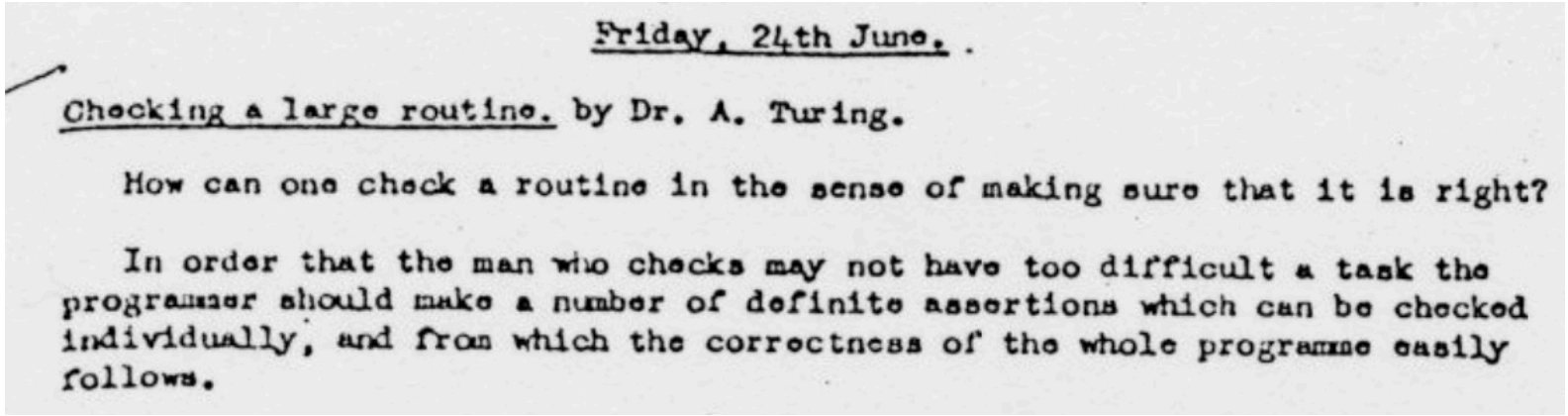
Jason Oberg, Nicole Fern, Alric Althoff
http://tortugalogic.com

Hardware Verification



Key Difference: Need to Specify the Threat Model!

Property Driven Hardware Security*



Friday, 24th June.

Checking a large routine. by Dr. A. Turing.

How can one check a routine in the sense of making sure that it is right?

In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole programme easily follows.

Security Properties

- Formal specification of security requirements & threat model
- Test Driven Development**

Security Verification

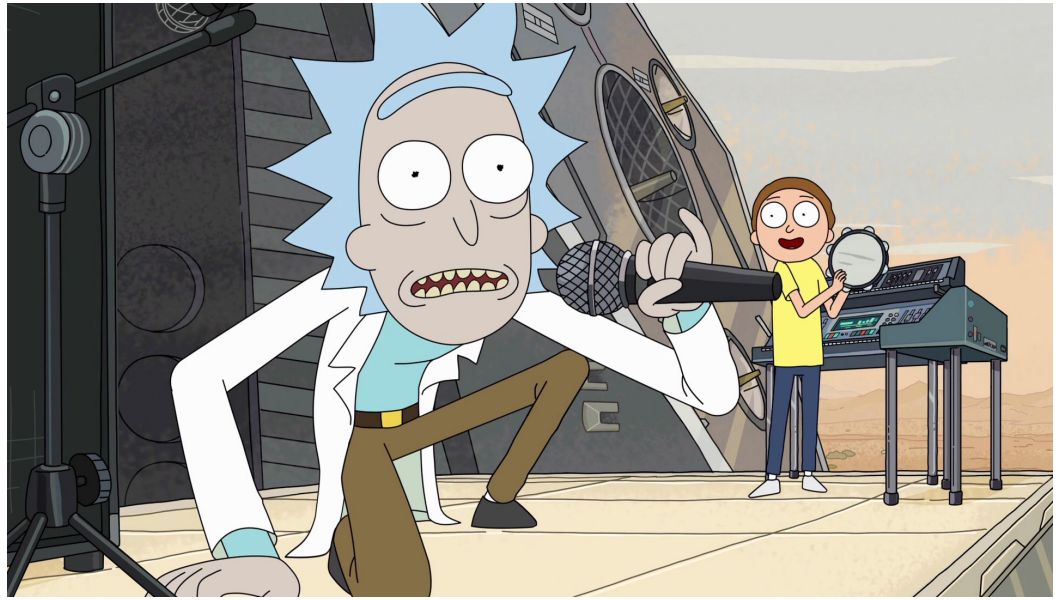
- Formal methods, simulation, emulation, run-time checks
- Leverage existing design tools as much as possible
- Automatic Property Refinement, e.g., "Properties First?" ***

* W. Hu, A. Althoff, A. Ardeshiricham, and R. Kastner, "Towards Property Driven Hardware Security", Microprocessor Test and Verification Conference 2016 (pdf)
** K. Beck. Test Driven Development: By Example. Addison-Wesley, 2002.
*** J. Urdahl, S. Udupi, T. Ludwig, D. Stoffel and W. Kunz , "Properties First? A New Design Methodology for Hardware, and its Perspectives in Safety Analysis", ICCAD 2016

Hardware Security Coverage

- How do you know your properties are complete?
- How do you know if your testbench is sufficient?
- How do you leverage extensive, existing hardware verification tools?

Get CWE-IFT!



CWE VIEW: Hardware Design

View ID: 1194
Type: Graph

Status: Incomplete

Downloads: Booklet | CSV | XML

Objective

This view organizes weaknesses around concepts that are frequently used or encountered in hardware design. Accordingly, this view can align closely with the perspectives of designers, manufacturers, educators, and assessment vendors. It provides a variety of categories that are intended to simplify navigation, browsing, and mapping.

Audience

Stakeholder	Description
Hardware Designers	Hardware Designers use this view to better understand potential mistakes that can be made in specific areas of their IP design. The use of concepts with which hardware designers are familiar makes it easier to navigate.
Educators	Educators use this view to teach future professionals about the types of mistakes that are commonly made in hardware design.

Relationships

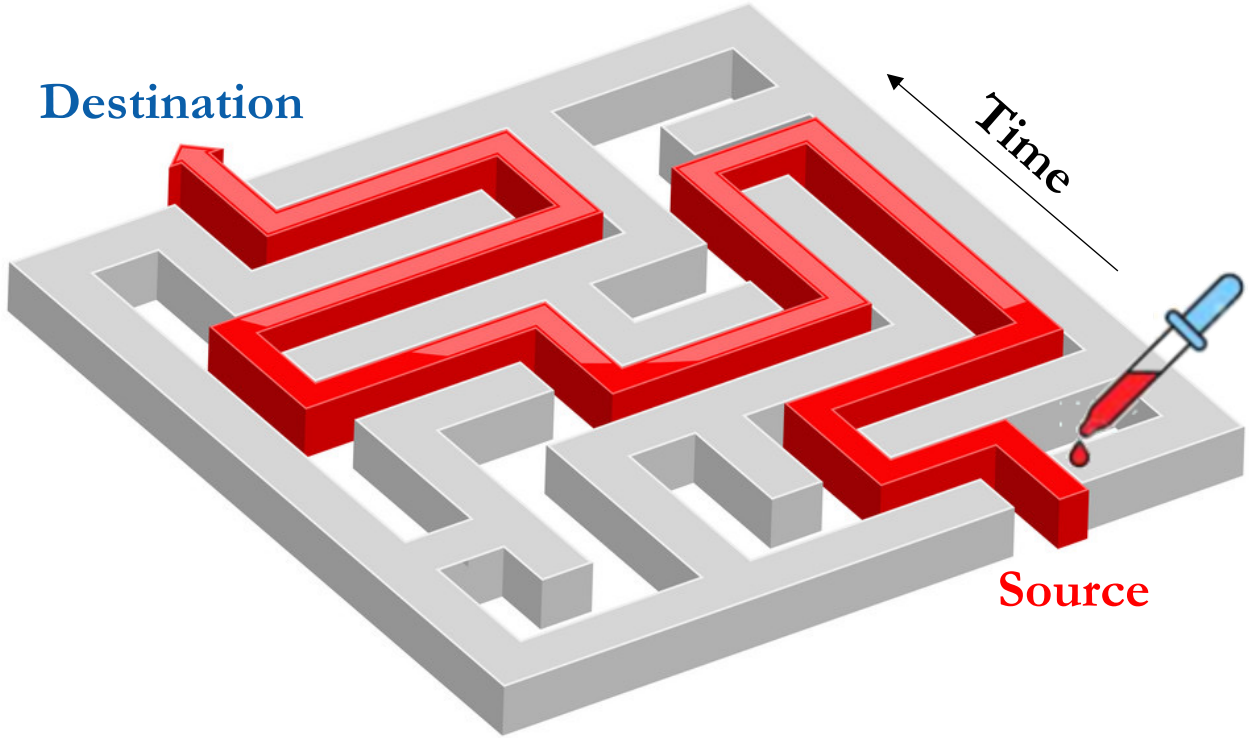
The following graph shows the tree-like relationships between weaknesses that exist at different levels of abstraction. At the highest level, categories and pillars exist to group weaknesses. Categories (which are not technically weaknesses) are special CWE entries used to group weaknesses that share a common characteristic. Pillars are weaknesses that are described in the most abstract fashion. Below these top-level entries are weaknesses are varying levels of abstraction. Classes are still very abstract, typically independent of any specific language or technology. Base level weaknesses are used to present a more specific type of weakness. A variant is a weakness that is described at a very low level of detail, typically limited to a specific language or technology. A chain is a

Information Flow Tracking (IFT)

Source: ***Which design signals*** should information be ***tracked from***?

Destination: ***Which design signals*** should information ***not flow to***?

Rule *fails* if source information reaches destination




```
{ Source Signal Set }
    =/=> { Destination Signal Set }
```

Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner, "**Hardware Information Flow Tracking**", to appear ACM Computing Surveys (pdf)

CWE-IFT

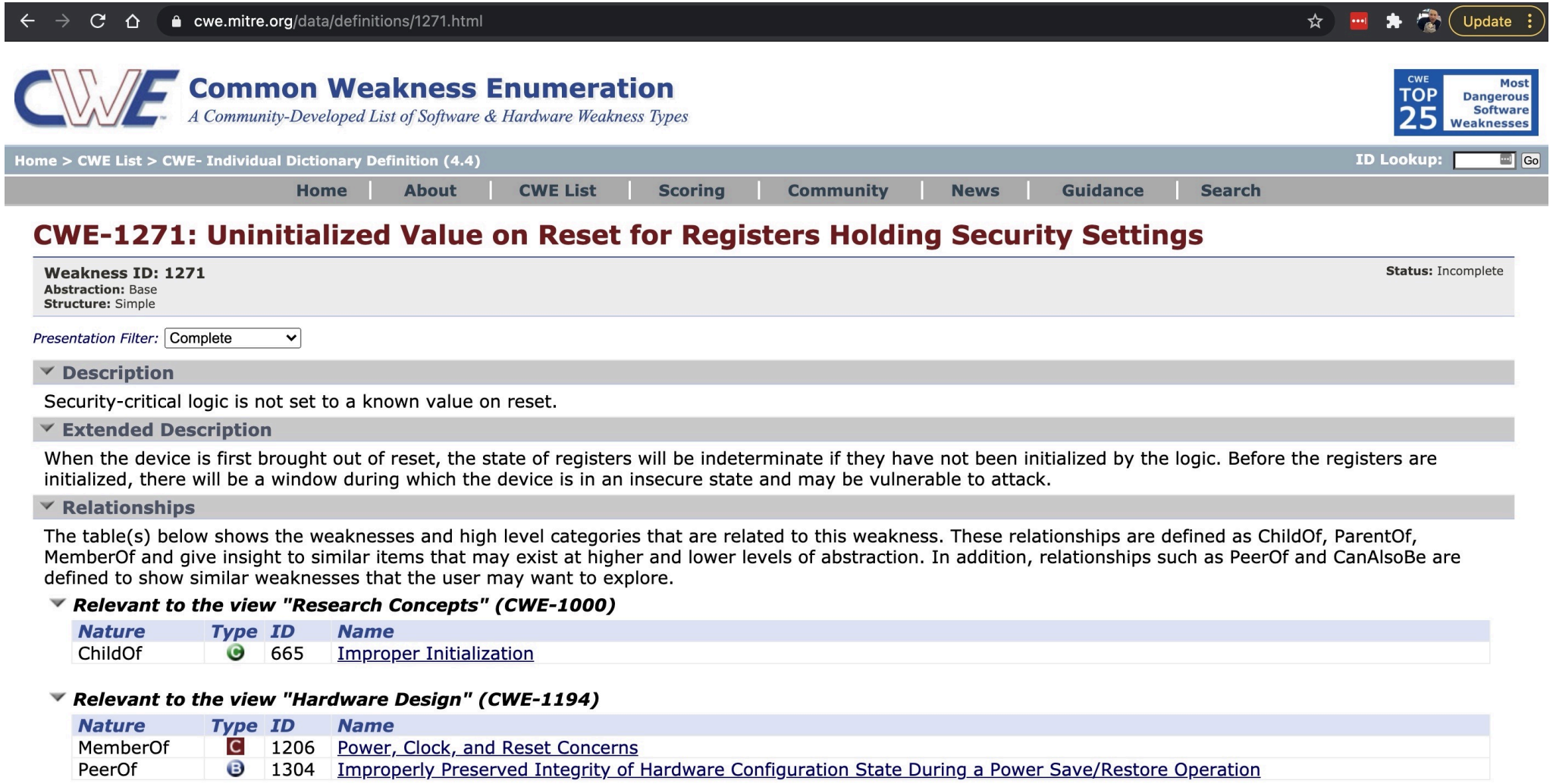
1) Identify CWEs related to threat model

2) List assets and conditions

3) Write properties

4) Get CWE-IFT!

1) Identify CWEs



cwe.mitre.org/data/definitions/1271.html

Update

Common Weakness Enumeration

A Community-Developed List of Software & Hardware Weakness Types

CWE TOP 25 Most Dangerous Software Weaknesses

Home > CWE List > CWE- Individual Dictionary Definition (4.4)

ID Lookup: Go

Home | About | CWE List | Scoring | Community | News | Guidance | Search

CWE-1271: Uninitialized Value on Reset for Registers Holding Security Settings

Weakness ID: 1271
Abstraction: Base
Structure: Simple

Status: Incomplete

Presentation Filter: Complete

Description

Security-critical logic is not set to a known value on reset.

Extended Description

When the device is first brought out of reset, the state of registers will be indeterminate if they have not been initialized by the logic. Before the registers are initialized, there will be a window during which the device is in an insecure state and may be vulnerable to attack.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

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

Nature	Type	ID	Name
ChildOf	C	665	Improper Initialization

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

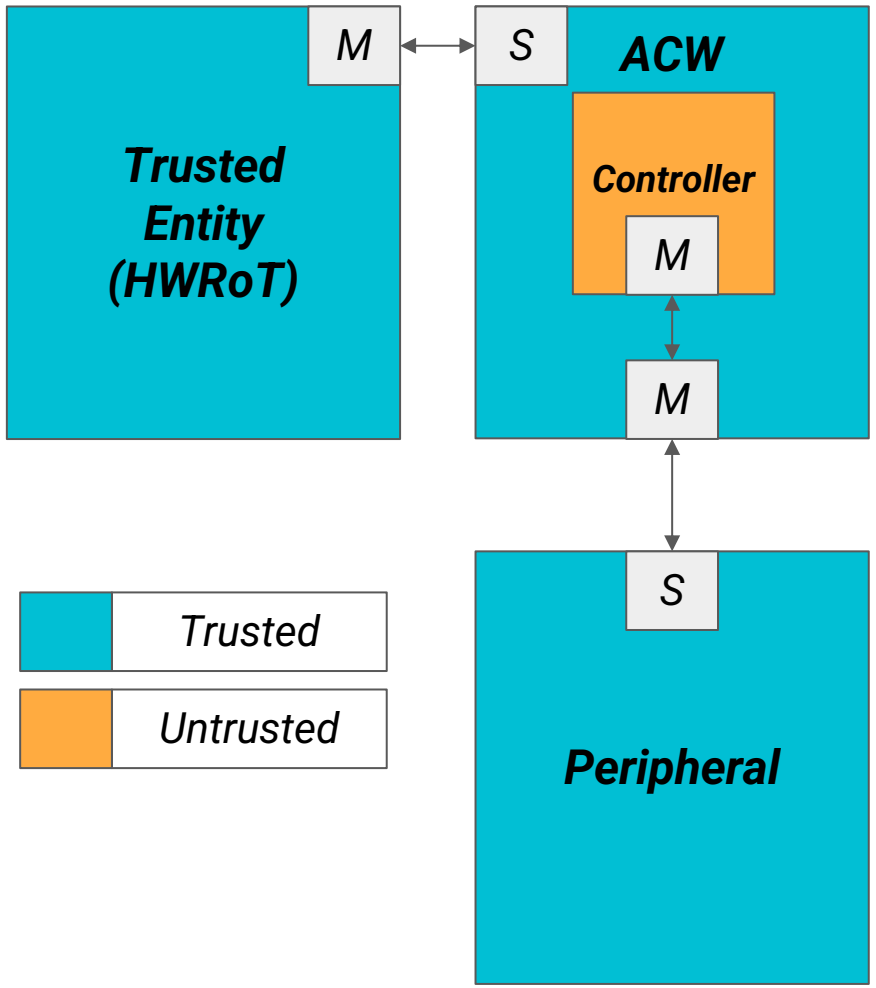
Nature	Type	ID	Name
MemberOf	C	1206	Power, Clock, and Reset Concerns
PeerOf	B	1304	Improperly Preserved Integrity of Hardware Configuration State During a Power Save/Restore Operation

Also, CWEs 1258, 1269, 1272, …

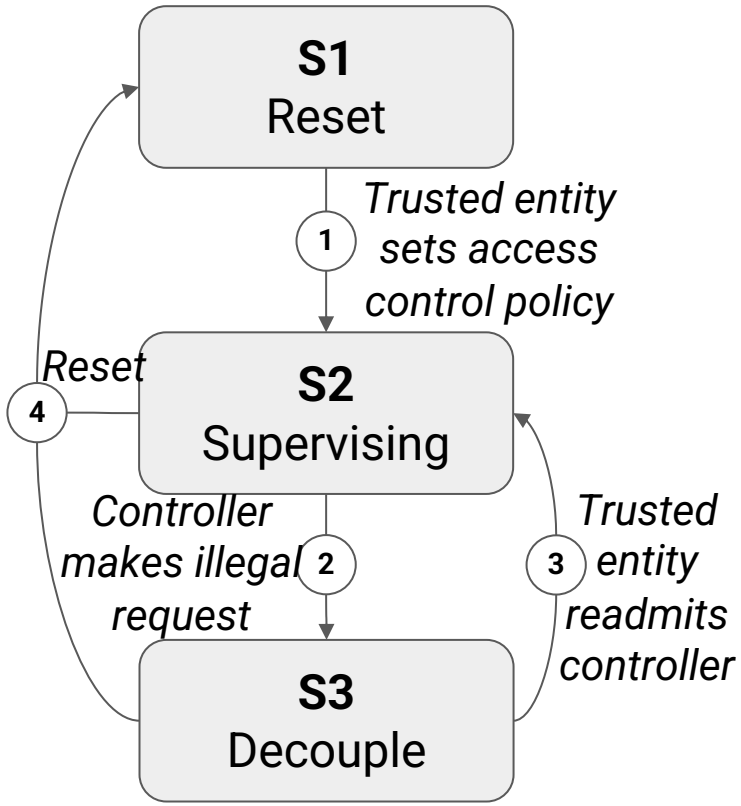
2) List Assets & Conditions

Asset: `Controller`

Boundary: `Peripheral`



Condition: `RESET`



3) Write Properties

Properties

Example Trace Property

Controller receives the baseline AXI signals while the ACW is in reset mode.

CWEs 1258, 1271

```
`sig_to_C` == `val`   (value check)
|| (RESET != 1)       (condition)
```

Example Information Flow Property

Information from controller originating during reset mode will never flow to the peripheral.

CWEs 1269, 1272

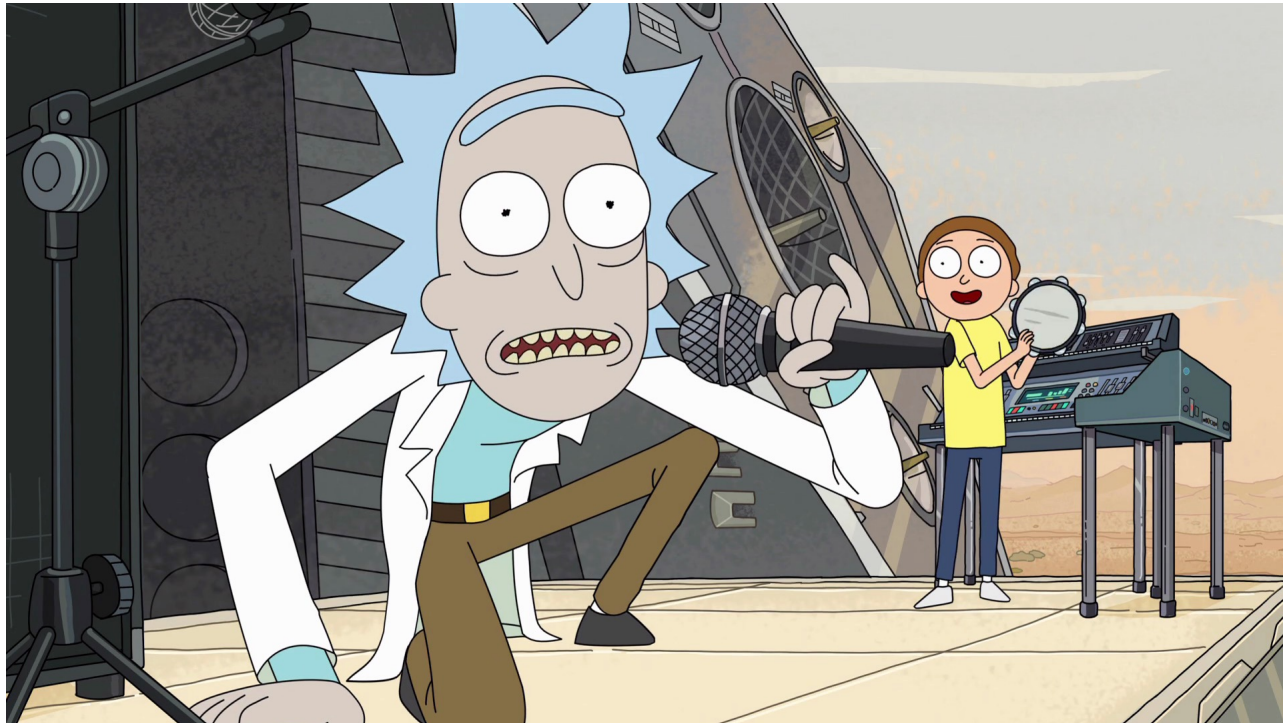
```
`sig_from_C`          (source)
when (RST == 0)       (track condition)
=/=>                  (no-flow operator)
`sig_to_P`            (destination)
```

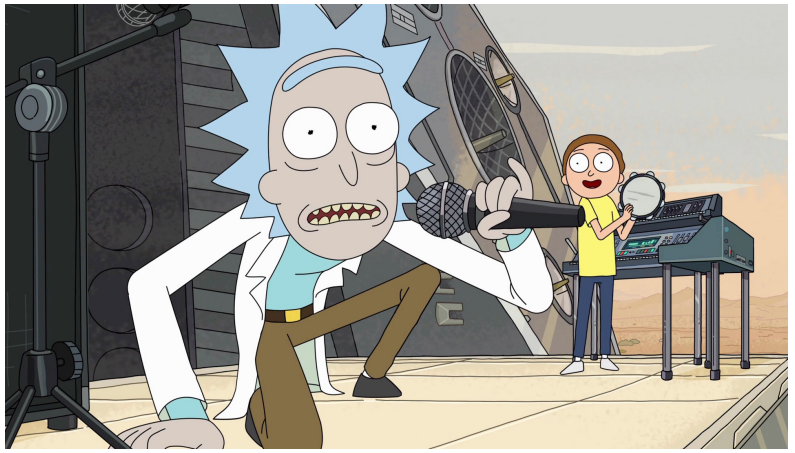
Access Control Wrapper Repository: https://github.com/KastnerRG/Access_Control_ext

Conclusion

- Security Verification != Functional Verification
- Security Coverage ?= Functional Coverage
- Security Coverage Metrics!?

1) Identify CWEs
2) List Assets & Conditions
3) Write Properties
4) Get CWE-IFT!





Hardware Security Coverage

Ryan Kastner

http://kastner.ucsd.edu

UC San Diego

Jason Oberg, Nicole Fern, Alric Althoff

http://tortugalogic.com

Tortuga™ Logic