BF for CHIPS

A Formal Language for Describing and Backtracking Chips Triggered Software Vulnerabilities

February 8, 2023

https://samate.nist.gov/BF/

National Institute of Standards and Technology
U.S. Department of Commerce

Irena Bojanova

Problem & Solution

- Problem: How to identify the root cause for an observed security failure triggered by a chip bug or weakness?
  (+ maybe: how to think about of mitigation and disclosure)

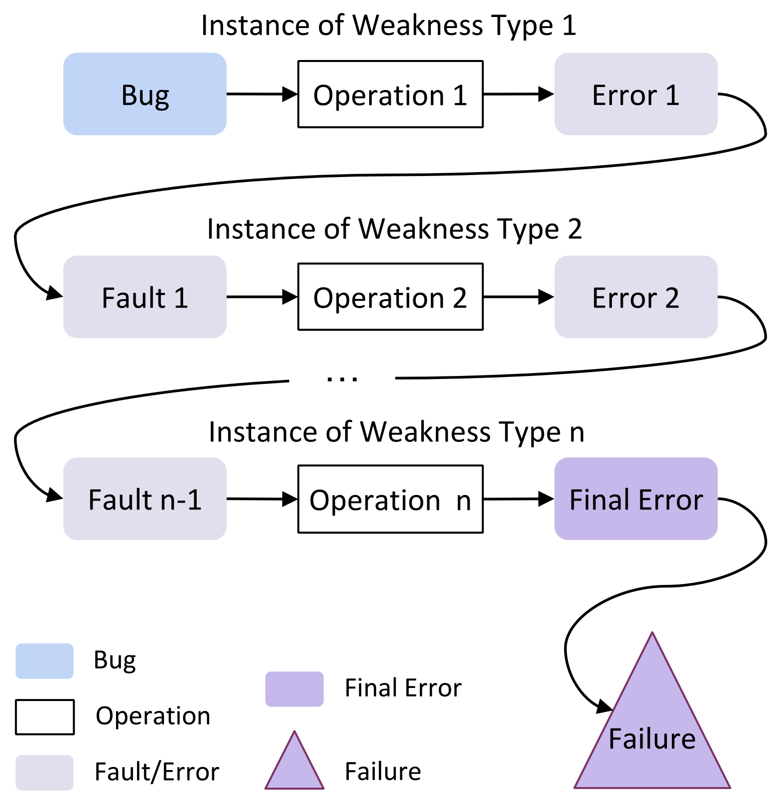
- Solution: A formal language for describing the chains of weaknesses underling a vulnerability, that allows automated backtracking of chips triggered vulnerabilities from observed security failures.

Tentative Steps

- Identify chip related vulnerabilities
  – discuss also with the Vuntology team
- Identify kinds of weaknesses that underlie these vulnerabilities
- Prioritize development of related BF classes

- Identify problems in chips that trigger software security vulnerabilities leading to security failures
- Create formal taxonomies for describing such vulnerabilities as chains of underlying weaknesses, linked by causality
- Create a formal language allowing backtracking from an observed security failure through the causal relationships of the chained weaknesses
- Create tools for utilizing the formal taxonomies and language

BF Model of Security Vulnerability



```
START  := Vulnerability Converge

Vulnerability:= Bug Operation Error

Error  := Fault Operation
        | FinalError

Converge:= Vulnerability Converge
         | Failure END
```

A high-level formal description of a vulnerability.

Root Cause of a Chip Triggered Vulnerability

- Firmware – Should we focus only on this?
  - Code or specification defect leads to software weakness
  - Optimization weakness – e.g., time optimization (look for energy, temperature)
    e.g., speculative execution (use idle time) (Speculative execution – Wikipedia):
    Foreshadow, Meltdown, Microarchitectural Data Sampling, Spectre, SPOILER, Pacman
- Hardware
  - Flipped bits lead to a software weakness
    - Space optimization – e.g. Row hammer (overcrowded chip) (https://en.wikipedia.org/wiki/Row_hammer)
    - Silent data corruption (SDC) errors (https://www.nytimes.com/2022/02/07/technology/computer-chips-errors.html)
    - Silent corrupt execution errors (CEEs), “mercurial" cores (https://sigops.org/s/conferences/hotos/2021/papers/hotos21-s01-hochschild.pdf)
    - Physical chip defect

  - Are these other causes?

BF Security Concepts Definitions

- Software Security Vulnerability:
  - A chain of weaknesses linked by causality
  - Starts with a bug
  - Ends with a final error, which if exploited leads to a security failure
- Software Security Weakness:
  - A (bug, operation, error) or (fault, operation, error) triple.
  - An instance of a weakness type

A weakness type relates to a distinct phase of software execution, the operations specific for that phase and the operands required as input to those operations.

- Software Security Bug:
  - A code or specification defect (operation defect)
- Software Fault:
  - A data, type, address, size, or name error (operand error)
- Software Error:
  - The result from an operation with a bug or a faulty operand
  - Becomes a next fault or is a final error.
- Software Final Error:
  - An exploitable or undefined system behavior
  - Leads to a security failure
- Security Failure:
  - A violation of a system security requirement

Metrology for: CHIPS

Division(s)/Lead PI/E-mail: SSD/Irena Bojanova/irena.bojanova@nist.gov

Project Name and Status:	Provide the **Project name/title**: BF for CHIPS – A Formal Language for Describing and Backtracking Chips Triggered Chains of Software Weaknesses that lead to Security Failures **Status:** Expand upon a Current Active NIST Project
Estimated Budget:	Provide a rough-order magnitude budget for your project **Staff: $xx ; Associates: $xx ; Equipment: $xx = TOTAL Estimated Budget: $$$xx**
Timeline:	Provide the estimated start and end dates for your project **(March/2023-Nov/xxxx)**
Measurement Need:	To find and fix software bugs that trigger security vulnerabilities, we need to first clearly understand all the chained underlying weaknesses that lead to an observed security failure. A conceptually new approach is needed, as widely used repositories of software weaknesses and vulnerabilities, such as CWE and CVE (and its upgrades NVD and KEV), have considerable problems. CWE & CVE have imprecise descriptions, unclear causality, and gaps and overlaps in coverage. There is no tracking methodology for CVE. There are no tools facilitating the use of CWE & CVE.
NIST Approach / Solution:	Create a formal language for clearly describing software vulnerabilities triggered by ships defects. The language will be based on the taxonomies of the Bugs Framework(BF), which is being developed as a structured, complete, orthogonal classification system of software bugs and weaknesses. *Structured* means a weakness is described as a cause-operation-consequence triple. This assures precise causal descriptions. *Complete* means BF has the expressiveness power to describe any software bug (defect in code or specification) or weakness. This assures there are no gaps in coverage. *Orthogonal* means the operations of any two BF classes do not overlap. This assures there are no overlaps in coverage. *Classification system* means bugs and weaknesses chain via "cause (defect or fault)–consequence (error)–cause(fault)" transitions. This assures back-tracking from the failure through errors/faults to the bug. All these together resolve the imprecise descriptions problem. BF is being developed also to be applicable for code in any programming language and is technology independent.
Deliverables and Impact:	A formal language (taxonomies) for clearly describing software vulnerabilities, triggered by chip defects. Such a vulnerability description will be a defect – error/fault–…–error/fault–final error chain, leading to a security failure. Tools facilitating creation of such descriptions. Tools for backtracking from failure to the root cause chip defect. Being able to determine the root cause defect from an observed security failure will have a huge impact on how code and specification defects can be quickly and successfully fixed.
Industry or OA collaborations:	Current collaborations with RIT, Carnegie Mellon, JHU APL. May expand in the future.