The NIST Bugs Framework (BF)

Input/Output Check Bugs Taxonomy: Injection Errors in Spotlight

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

- Terminology:
  - Bug, Weakness
  - Vulnerability
  - Failure
- Existing Repositories:
  - CWE
  - CVE
  - NVD

- The Bugs Framework (BF)
  - Goals
  - Features
- Examples:
  - BIG-IP TMUI RCE
  - Heartbleed
- Potential Impacts
Terminology
Bug, Weakness, Vulnerability, Failure

● **Software Bug:**
  ○ A coding error
  ○ Needs to be fixed

● **Software Weakness – difficult to define:**
  ○ Caused by a bug or ill-formed data
  ○ Weakness Type – a meaningful notion!

● **Software Vulnerability:**
  ○ An instance of a weakness type that leads to a security failure
  ○ May have several underlying weaknesses

● **Security failure:**
  ○ A violation of a system security requirement
Existing Repositories
Commonly Used Repositories

- Weaknesses:
  CWE – Common Weakness Enumeration

- Vulnerabilities:
  CVE – Common Vulnerabilities and Exposures
  → over 18,000 documented in 2020

- Linking weaknesses to vulnerabilities – CWEs to CVEs:
  NVD – National Vulnerabilities Database
1. Imprecise Descriptions – CWE & CVE

2. Unclear Causality – CWE & CVE

3. Gaps in Coverage – CWE

4. Overlaps in Coverage – CWE
The Bugs Framework (BF)
1. Solve the problems of imprecise descriptions and unclear causality

2. Solve the problems of gaps and overlaps in coverage
BF Features – Clear Causal Descriptions

- BF describes a bug/weakness as:
  - An improper state
  - Its transition

- Improper State – a tuple \((\text{operation}, \text{operand}_1, \ldots, \text{operand}_n)\), where at least one element is improper

- Transition – the result of the operation over the operands
BF Features – Chaining Weaknesses

- BF describes a vulnerability as:
  - A chain of improper states and their transitions
  - States change until a failure is reached
BF Features – Causes and Consequences

- How to find the Bug?
- Go backwards by operand until an operation is a cause

Initial State – caused by the Bug
- the operation is improper

Intermediate State – caused by ill-formed data
- at least one operand is improper

Final State – the Failure
- caused by a final error
BF Features – Converging Vulnerabilities

- **Initial State**: caused by the Bug – the operation is improper
- **Intermediate State**: caused by ill-formed data – at least one operand is improper
- **Final State**: the Failure – caused by a final error

**Improper State 1**
(operation 1
operand 1...
operand 1...
...)

**Improper State 1’**
(operation 1’
operand 1’...
operand 1’...
...)

**Improper State n**
(operation n
operand n...
operand n...
...)

**Improper State q’**
(operation q’
operand q’...
operand q’...
...)

- Improper operand 2,
- Improper operand 2
- Improper operand 2’
- Improper operand 2’
- Improper operand n
- Improper operand n
- Improper operand q’
- Improper operand q’
- Final Error
- Final Error
- Final Error’
- Failure
BF Features – Classification

- BF Class – a taxonomic category of a weakness type, defined by:
  - A set of operations
  - All valid cause → consequence relations
  - A set of attributes
BF – Bugs Models

Example:

The BF Data Check Bugs Model:

- Two phases, corresponding to the BF data check classes: DVL and DVR

- Data Check operations flow
BF Classes – Examples: DVL & DVR

Data Validation Bugs (DVL)

**Causes**
- Improper Operation:
  - Missing
  - Erroneous
  - Under-Restrictive Policy
  - Over-Restrictive Policy

- Improper Data (from previous operation):
  - Corrupted Data
  - Tampered Data

- Improper Policy (as data from previous operation):
  - Corrupted Policy
  - Tampered Policy

**DVL Operations**
- Validate
- Sanitize

**Consequences**
- Improper Data for next operation:
  - Invalid Data

**Injection Error**:
- Query Injection
- Command Injection
- Source Code Injection
- Parameter Injection
- File Injection

**Attributes**

Data Verification Bugs (DVR)

**Causes**
- Improper Operation:
  - Missing
  - Erroneous
  - Under-Restrictive Policy
  - Over-Restrictive Policy

- Improper Data (from previous operation):
  - Invalid Data

- Improper Data for next operation:
  - Wrong Value
  - Inconsistent Value
  - Wrong Type

**DVR Operations**
- Verify
- Correct

**Attributes**

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Source Code</th>
<th>Execution Space</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Codebase</td>
<td>Local</td>
<td>Entered</td>
</tr>
<tr>
<td>Quantity</td>
<td>Third Party</td>
<td>Admin</td>
<td>Stored</td>
</tr>
<tr>
<td>Range</td>
<td>Standard Library</td>
<td>Bare-Metal</td>
<td>In Use</td>
</tr>
<tr>
<td>Type</td>
<td>Processor</td>
<td></td>
<td>Transferred</td>
</tr>
<tr>
<td>Other Rules</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Operation Data**
- State: Entered, Stored, In Use, Transferred
BF Classes – Examples: MAD & MUS

Memory Addressing Bugs (MAD)

- **Improper Operation**: Missing, Mismatched, Erroneous
- **Improper Pointer**: NULL Pointer, Wild Pointer, Dangling Pointer, Over Bounds, Under Bounds, Untrusted Pointer, Wrong Position, Casted Pointer, Forbidden Address
- **Improper Size**: Inconsistent Value, Wrong Value

**MAD Operations**
- Initialize
- Reposition
- Reassign

**Consequences**

Memory Use Bugs (MUS)

- **Improper Operation**: Missing, Mismatched, Erroneous
- **Improper Pointer**: NULL Pointer, Wild Pointer, Dangling Pointer, Over Bounds, Under Bounds, Untrusted Pointer, Wrong Position, Casted Pointer, Forbidden Address
- **Improper Size**: Wrong Value

**MUS Operations**
- Initialize
- Dereference
- Read
- Write
- Clear

**Memory Error**
- Uninitialized Object
- Not Cleared Object
- NULL Pointer Dereference
- Untrusted Pointer Dereference
- Object Corruption
- Type Confusion
- Use After Free
- Buffer Overflow
- Buffer Underflow
- Uninitialized Pointer Dereference

**Operation**
- Mechanism: Direct, Sequential
- Source Code: Codebase, Third Party, Standard Library, Processor
- Execution Space: Userland, Kernel, Bare-Metal
- Location: Stack, Heap, ...

**Attributes**
- Pointer
- Object

**Execution Space**
- Span: Little, Moderate, Huge
- Location: Stack, Heap, ...

**Location**
- Stack
- Heap
- ...

**Source Code**
- Codebase
- Third Party
- Standard Library
- Processor

**Mechanism**
- Direct
- Sequential
Input/Output CWEs (incl. Injection)

- Mapped by BF DVL and BF DVR consequences

CWE by DVL Injection Error:
- Query Injection
- Command Injection
- Source Code Injection
- Parameter Injection
- File Injection

CWE by Abstraction:
- Pillar
- Class
- Base
- Variant
- Compound

CWE by DVL or DVR Wrong Data for Next Operation Consequence:
- DVL Invalid Data
- DVR Wrong Value, Inconsistent Value, and Wrong Type
- No consequence (only cause listed)
BF – Defined

- BF is a ...
  - Structured
  - Complete
  - Orthogonal
  - Language independent

classification of software bugs and weaknesses
BF Example 1:
Description of BIG-IP TMUII RCE
BIG-IP TMUI RCE (CVE-2020-5902)

**CVE-2020-5902** In BIG-IP versions 15.0.0-15.1.0.3, 14.1.0-14.1.2.5, 13.1.0-13.1.3.3, 12.1.0-12.1.5.1, and 11.6.1-11.6.5.1, the Traffic Management User Interface (TMUI), also referred to as the Configuration utility, has a Remote Code Execution (RCE) vulnerability in undisclosed pages.

- **Vulnerability in BIG-IP TMUI login interface**
  
  https://[F5 Host]/tmui/login.jsp/

- **Proof-Of-Concept: TMSH command execution**
  
  https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp
BF Description of BIG-IP TMUI RCE

The Bug

The Failure
BF Example 2: Updated Description of Heartbleed
The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.
Clear Causality in Heartbleed

- **DVR** (Verify: Missing, Data (payload length), Policy)
  - Inconsistent Value
  - Over Bounds
  - Buffer Overflow

- **MAD** (Reposition, Pointer, Object, Size: Inconsistent Value, Size)
  - Over Bounds
  - Buffer Overflow

- **MUS** (Read, Pointer: Over Bounds, Object, Size)
  - Buffer Overflow
  - Information Exposure
  - Not Cleared Object

- **MUS** (Clear: Missing, Pointer, Object, Size)

**Legend:**
- Caused by the Bug
- Caused by ill-formed data
- The Failure – caused by final error(s)
BF Description of Heartbleed

**Cause**
- Improper Operation: Missing
- Improper Size: Inconsistent Value (for size of s->s3->rrec.data[0])
- Improper Pointer: Over Bounds

**DVR Operation**
- Verify

**Consequence**
- Improper Data: Inconsistent Value (for payload size)
- Improper Pointer: Over Bounds
- Improper Pointer: Over Bounds

**Attributes**
- Operation: Sequential
- Execution Space: Userland
- Mechanism: Source Code: Codebase (d1_both.c and tl_lib.c)
- Data: State: Transferred (via network)

**MAD Operation**
- Reposition

**Consequence**
- Improper Pointer: Over Bounds

**Attributes**
- Operation: Sequential
- Execution Space: Userland
- Mechanism: Source Code: Codebase (d1_both.c and tl_lib.c)
- Object: Location: Heap

**MUS Operation**
- Clear

**Consequence**
- Memory Error: Not Cleared Object

**Attributes**
- Operation: Sequential
- Execution Space: Userland
- Mechanism: Source Code: Codebase
- Pointer: Span: Huge
- Object: Location: Heap

---

**Information Exposure**

**The Bug**

**A Weakness**

**The Failure**
<Vulnerability Description CVE="CVE-2014-0160" Name="Heartbleed">
    <Bug Class="DVR">
        <Operation Value="Verify">...</Operation>
        <Operand Name="Data">...</Operand>
        <Operand Name="Policy"/>
        <Cause Value="Missing" Type="Improper Operation" Description="The Bug"/>
        <Consequence Value="Inconsistent Value" Type="Improper Data" Description="Operand for Next Operation"/>
    </Bug>
    <Weakness Class="NAD">
        <Operation Value="Deposition">...</Operation>
        <Operand Name="Pointer"/>
        <Operand Name="Object">...</Operand>
        <Operand Name="Size"/>
        <Cause Value="Inconsistent Value" Type="Improper Size" Comment="for s-s3-wrec.data[0]" Description="Result from Previous Operation"/>
        <Consequence Value="Over Bounds" Type="Improper Pointer" Description="Operand for Next Operation"/>
    </Weakness>
    <Weakness Class="NUS">
        <Operation Value="Read">...</Operation>
        <Operand Name="Pointer"...</Operand>
        <Operand Name="Object"...</Operand>
        <Cause Value="Over Bounds" Type="Improper Pointer" Comment="for s-s3-wrec.data[0]" Description="Result from Previous Operation"/>
        <Consequence Value="Buffer Overflow" Type="Memory Error" Description="Final Error"/>
    </Weakness>
    <Converge Vulnerability="Not Cleared Object"/>
</Vulnerability>

<Vulnerability Name="Not Cleared Object">
    <Bug Class="NUS">
        <Operation Value="Clean">...</Operation>
        <Operand Name="Pointer"...</Operand>
        <Operand Name="Object"...</Operand>
        <Cause Value="Missing" Type="Improper Operation" Description="The Bug"/>
        <Consequence Value="Not Cleared Object" Type="Memory Error" Description="Final Error"/>
    </Bug>
</Vulnerability>

<Failure Value="Information Exposure"/>
BF – Potential Impact
BF – Potential Impacts

- Allow precise communication about software bugs and weaknesses
- Help identify exploit mitigation techniques
Questions
Questions

Irena Bojanova: irena.bojanova@nist.gov
Carlos Galhardo: cegalhardo@inmetro.gov.br

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