The NIST Bugs Framework (BF)

Input/Output Check Bugs Taxonomy: Injection Errors in Spotlight

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

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

Irena Bojanova
Carlos Galhardo
Sara Moshtari

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

Repository Problems

1. Imprecise Descriptions – CWE & CVE

2. Unclear Causality – CWE & CVE

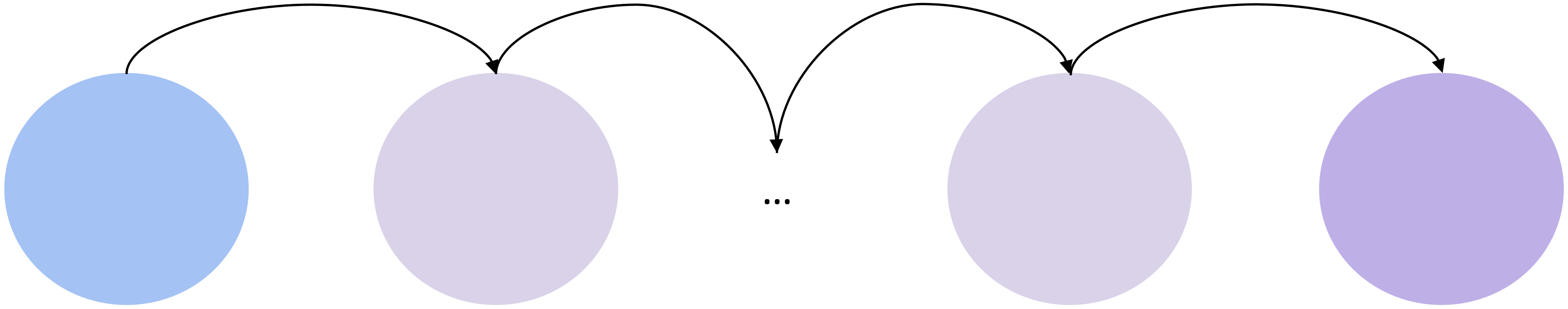
3. Gaps in Coverage – CWE

4. Overlaps in Coverage – CWE

The Bugs Framework (BF)

BF Goals

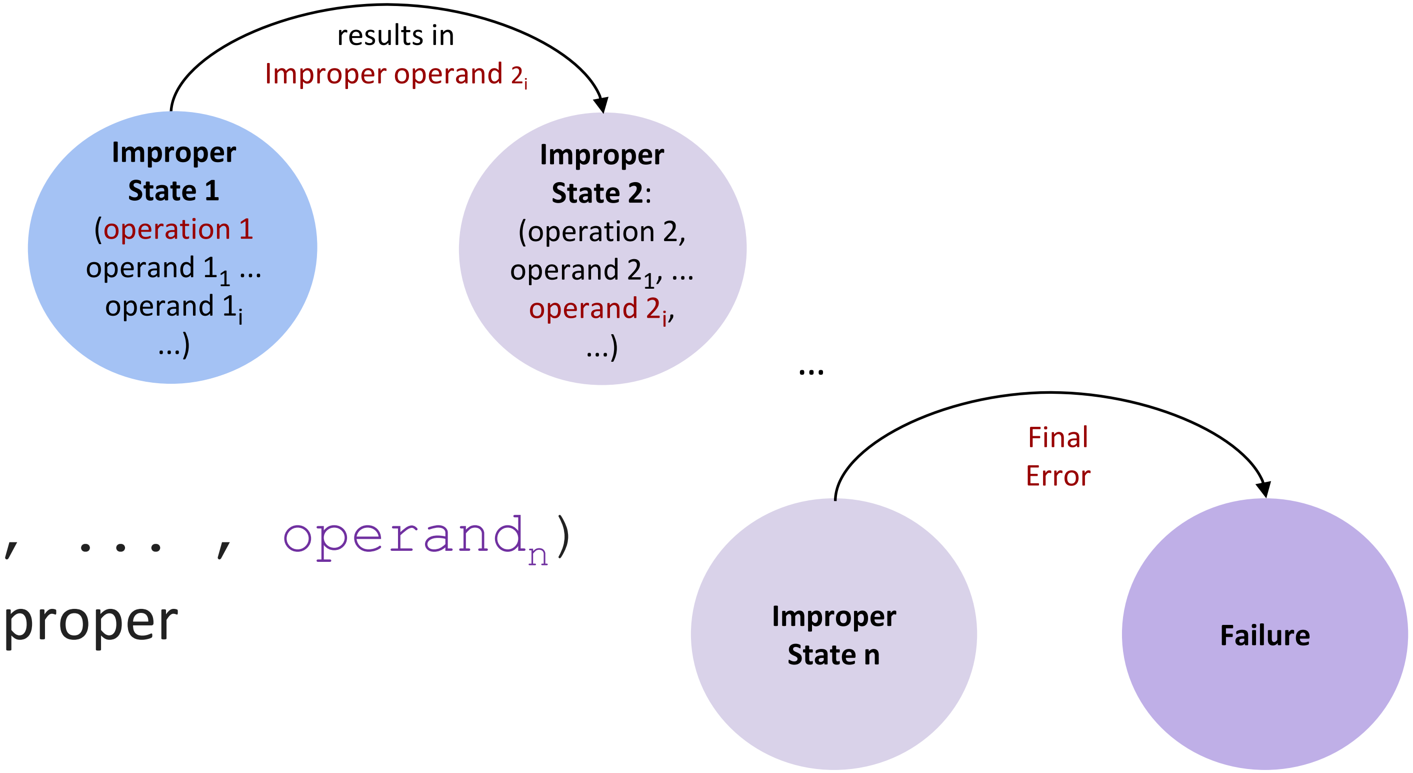
1. Solve the problems of imprecise descriptions and unclear causality



2. Solve the problems of gaps and overlaps in coverage

BF Features – Clear Causal Descriptions

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



- Improper State –

  a tuple (operation, operand_1, ... , operand_n)

  , where at least one element is improper

- Transition –

  the result of the operation over the operands

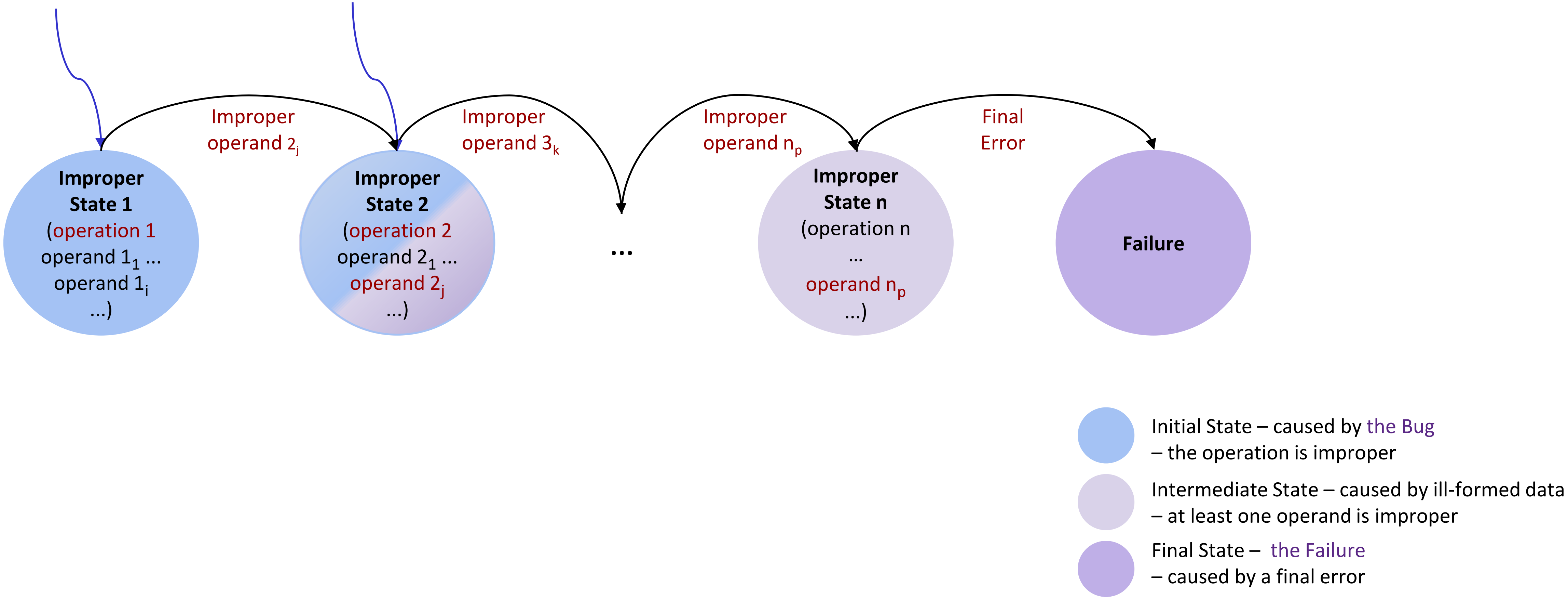
● Initial State – caused by the Bug – the operation is improper

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

● Final State – the Failure – caused by a final error

BF Features – Chaining Weaknesses

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



Improper State 1 (operation 1 operand 1_1 … operand 1_i …)

Improper operand 2_j

Improper State 2 (operation 2 operand 2_1 … operand 2_j …)

Improper operand 3_k

…

Improper operand n_p

Improper State n (operation n … operand n_p …)

Final Error

Failure

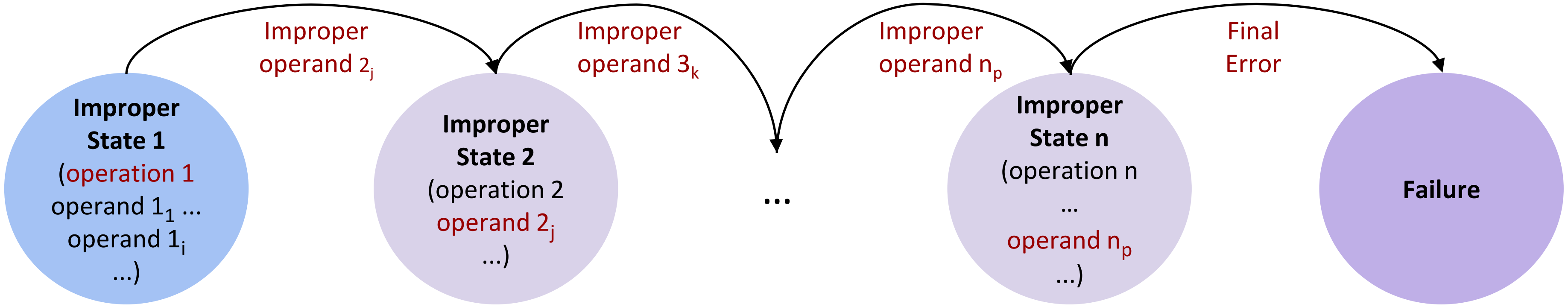
● Initial State – caused by the Bug – the operation is improper

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

● Final State – the Failure – caused by a final error

BF Features – Causes and Consequences

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



Improper operand 2_j

Improper operand 3_k

Improper operand n_p

Final Error

Improper State 1 (operation 1 operand 1_1 … operand 1_i …)

Improper State 2 (operation 2 operand 2_j …)

…

Improper State n (operation n … operand n_p …)

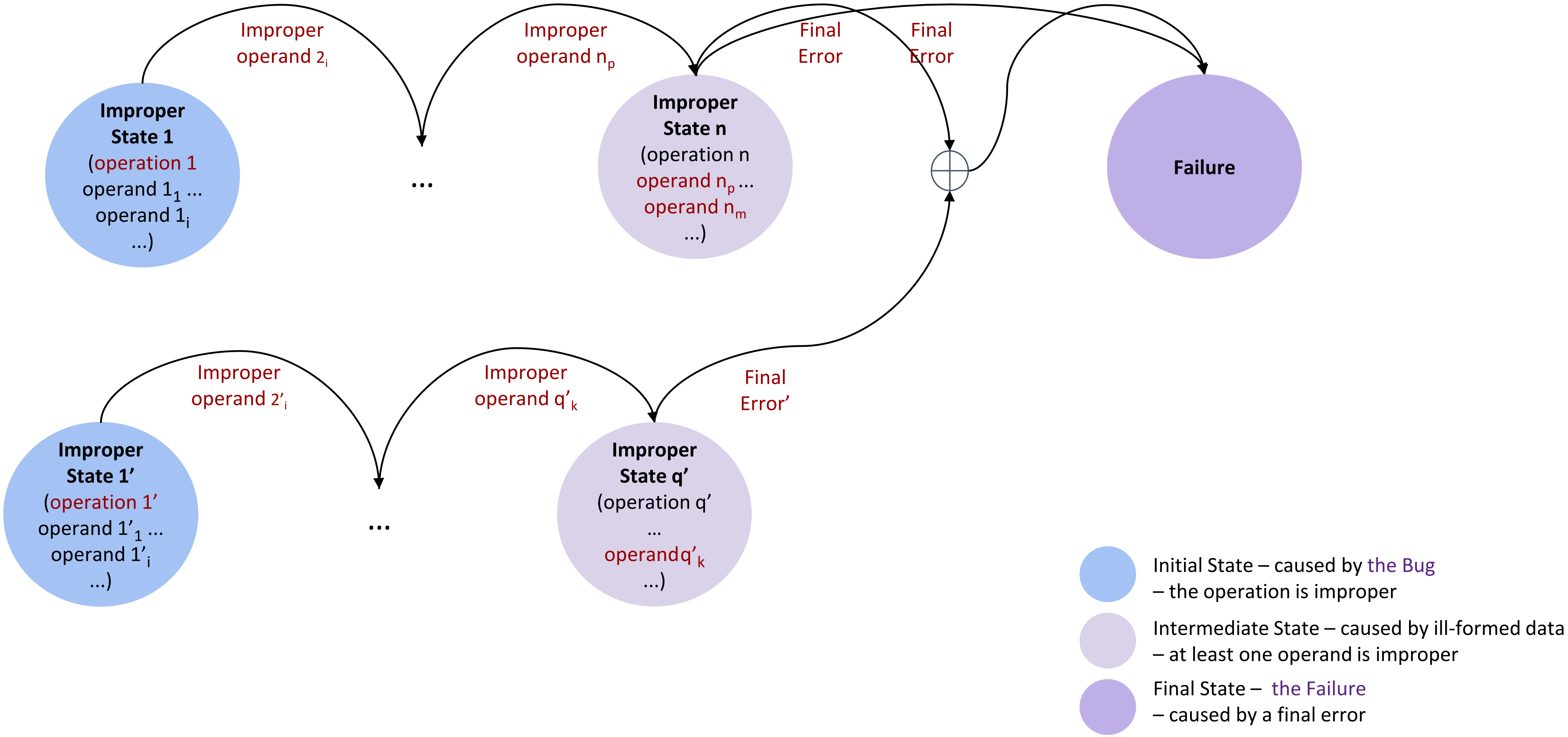
Failure

● 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



Improper operand 2_i

Improper operand n_p

Final Error

Final Error

Improper State 1
(operation 1 operand 1_1 … operand 1_i …)

…

Improper State n
(operation n operand n_p … operand n_m …)

Failure

Improper operand $2'_i$

Improper operand q'_k

Final Error’

Improper State 1’
(operation 1’ operand $1'_1$ … operand $1'_i$ …)

…

Improper State q’
(operation q’ … operandq'_k …)

● 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 – 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

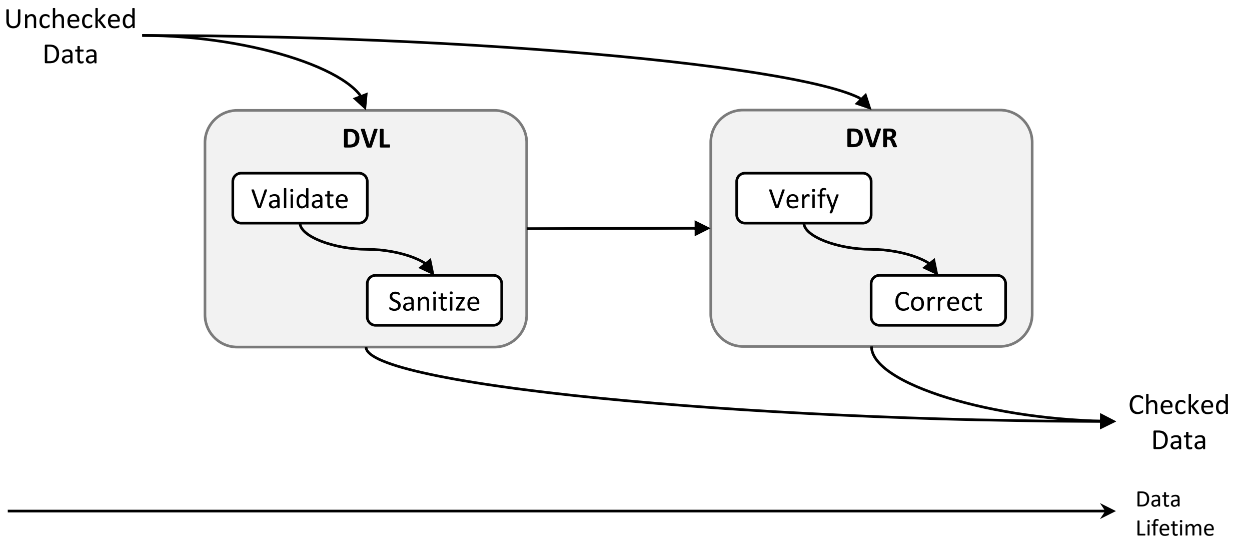
BF – Bugs Models

- Example:

  The BF Data Check Bugs Model:

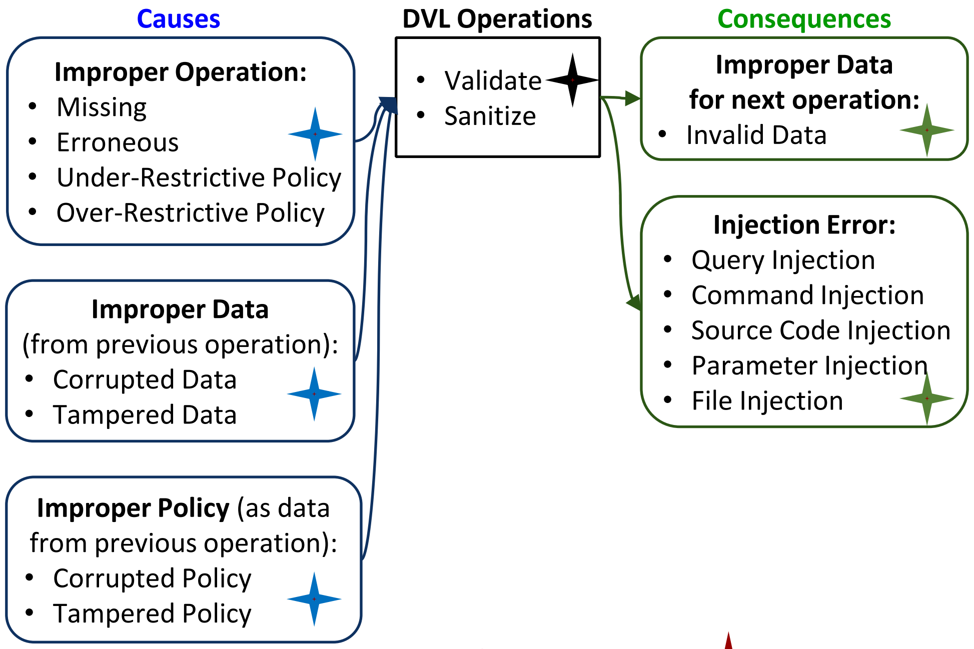
  - Two phases, corresponding to the BF data check classes: DVL and DVR

  - 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

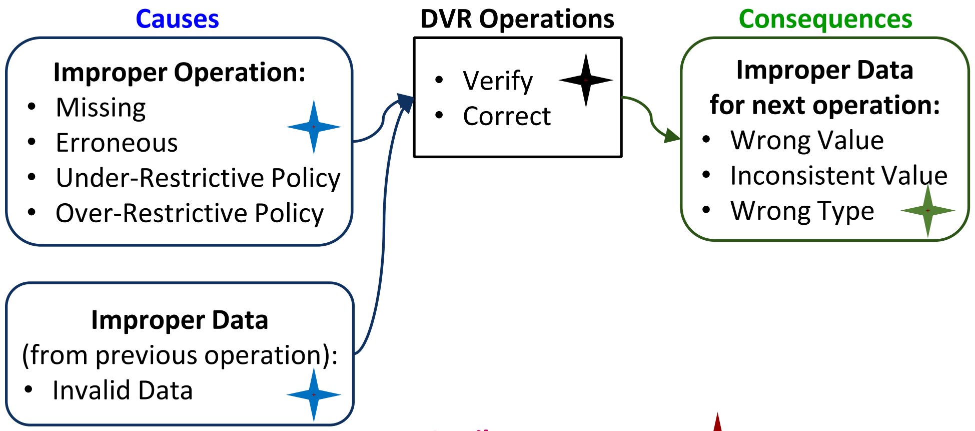
Improper Data for next operation:
- Invalid Data

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

Attributes

Operation			Data
Mechanism:	**Source Code:**	**Execution Space:**	**State:**
• Safelist	• Codebase	• Local	• Entered
• Denylist	• Third Party	• Admin	• Stored
• Format	• Standard Library	• Bare-Metal	• In Use
• Length	• Processor		• Transferred

Data Verification Bugs (DVR)



Causes

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

Improper Data
(from previous operation):
- Invalid Data

DVR Operations
- Verify
- Correct

Consequences

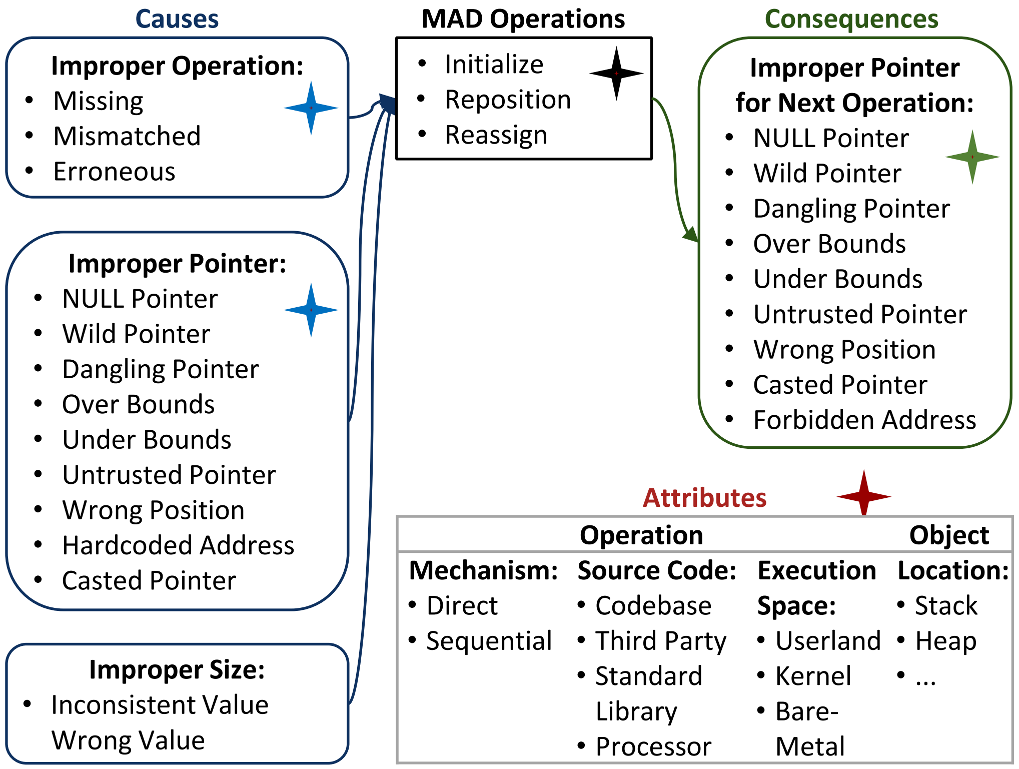
Improper Data for next operation:
- Wrong Value
- Inconsistent Value
- Wrong Type

Attributes

Operation			Data
Mechanism:	**Source Code:**	**Execution Space:**	**State:**
• Value	• Codebase	• Local	• Entered
• Quantity	• Third Party	• Admin	• Stored
• Range	• Standard Library	• Bare-Metal	• In Use
• Type	• Processor		• Transferred
• Other Rules			

BF Classes – Examples: MAD & MUS

Memory Addressing Bugs (MAD)



Causes

Improper Operation:
- Missing
- Mismatched
- Erroneous

Improper Pointer:
- NULL Pointer
- Wild Pointer
- Dangling Pointer
- Over Bounds
- Under Bounds
- Untrusted Pointer
- Wrong Position
- Hardcoded Address
- Casted Pointer

Improper Size:
- Inconsistent Value
Wrong Value

MAD Operations
- Initialize
- Reposition
- Reassign

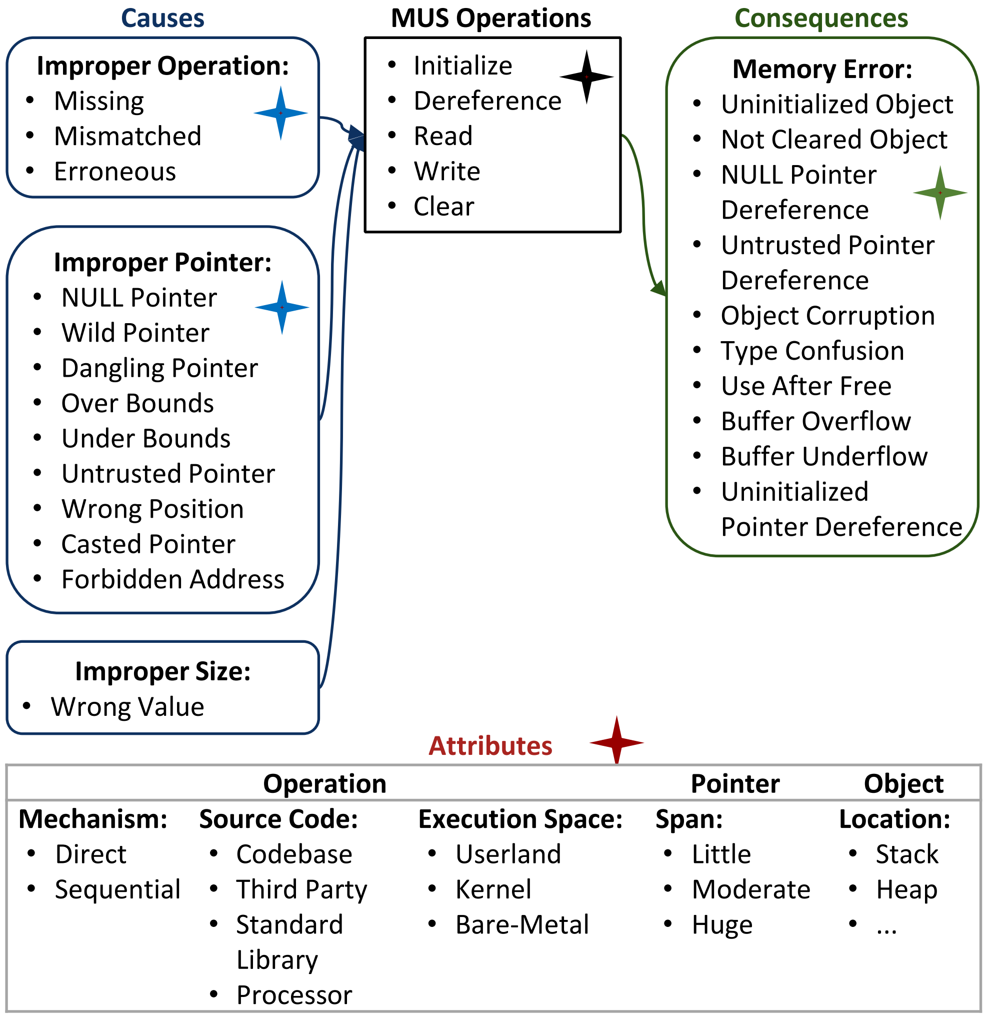
Consequences

Improper Pointer for Next Operation:
- NULL Pointer
- Wild Pointer
- Dangling Pointer
- Over Bounds
- Under Bounds
- Untrusted Pointer
- Wrong Position
- Casted Pointer
- Forbidden Address

Attributes

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

Memory Use Bugs (MUS)



Causes

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

Consequences

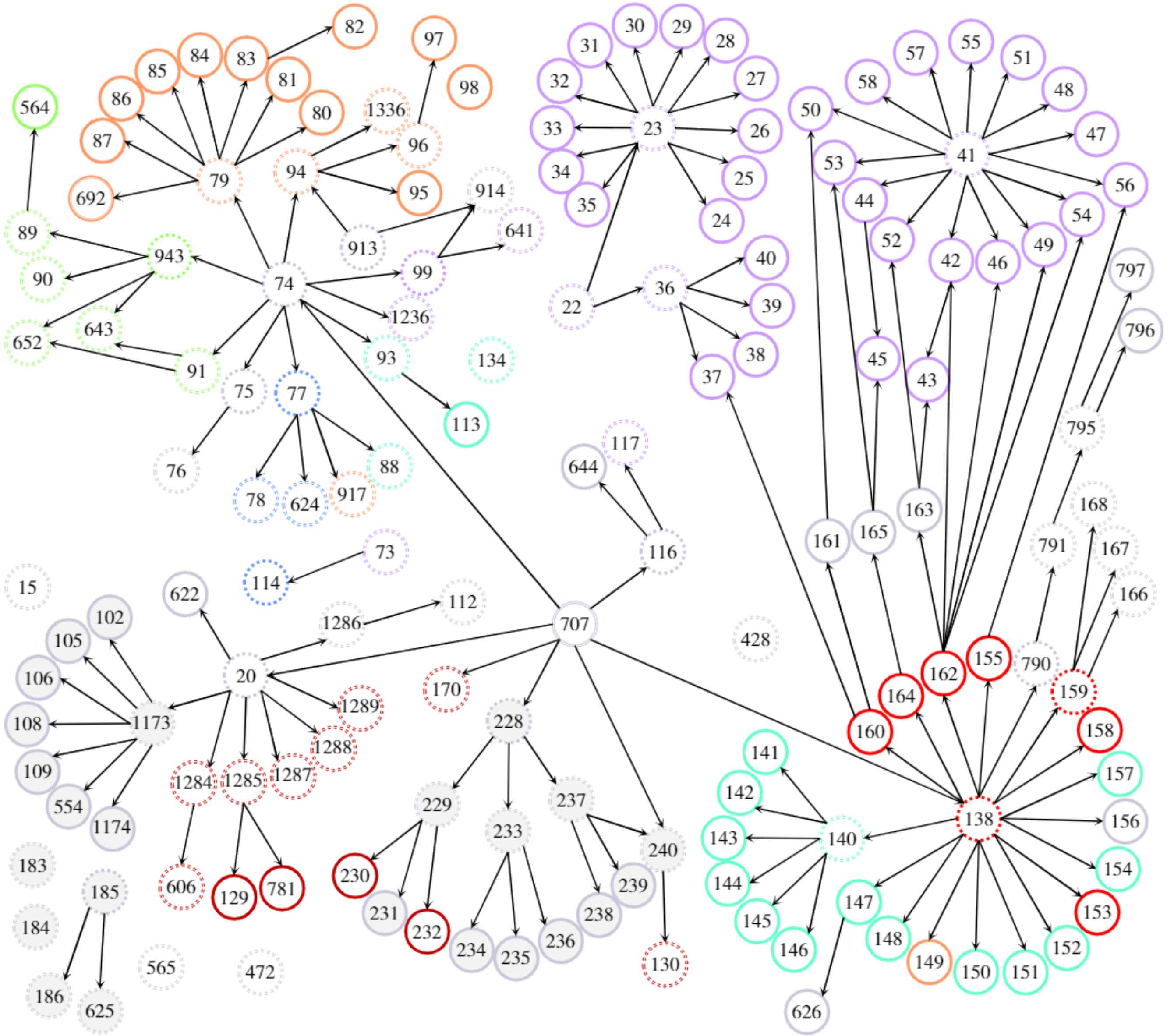
Memory Error:
- Uninitialized Object
- Not Cleared Object
- NULL Pointer Dereference
- Untrusted Pointer Dereference
- Object Corruption
- Type Confusion
- Use After Free
- Buffer Overflow
- Buffer Underflow
- Uninitialized Pointer Dereference

Attributes

Operation			Pointer	Object
Mechanism:	**Source Code:**	**Execution Space:**	**Span:**	**Location:**
• Direct • Sequential	• Codebase • Third Party • Standard Library • Processor	• Userland • Kernel • Bare-Metal	• Little • Moderate • Huge	• Stack • Heap • ...

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 orDVR 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

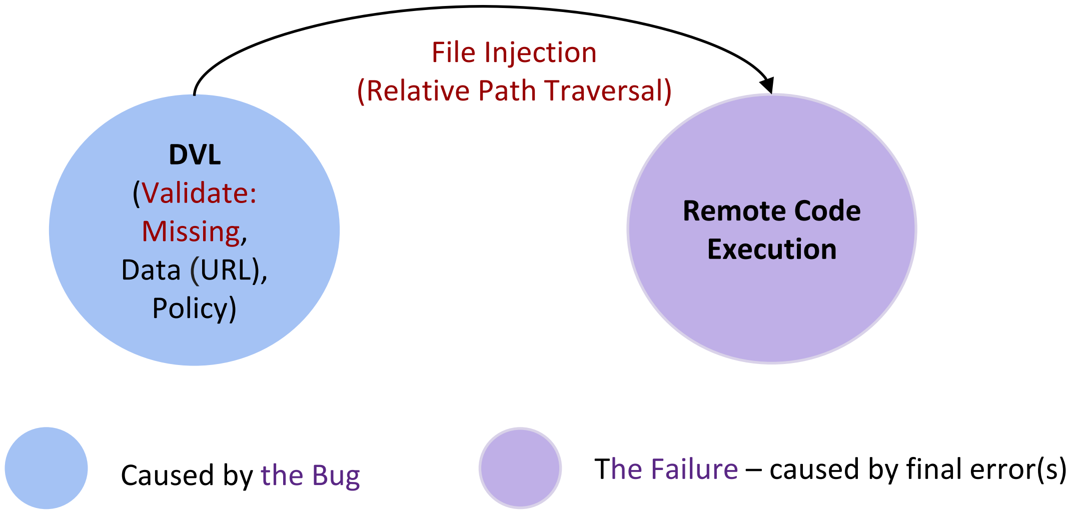
BF Example 1: Description of BIG-IP TMUI RCE

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/
```



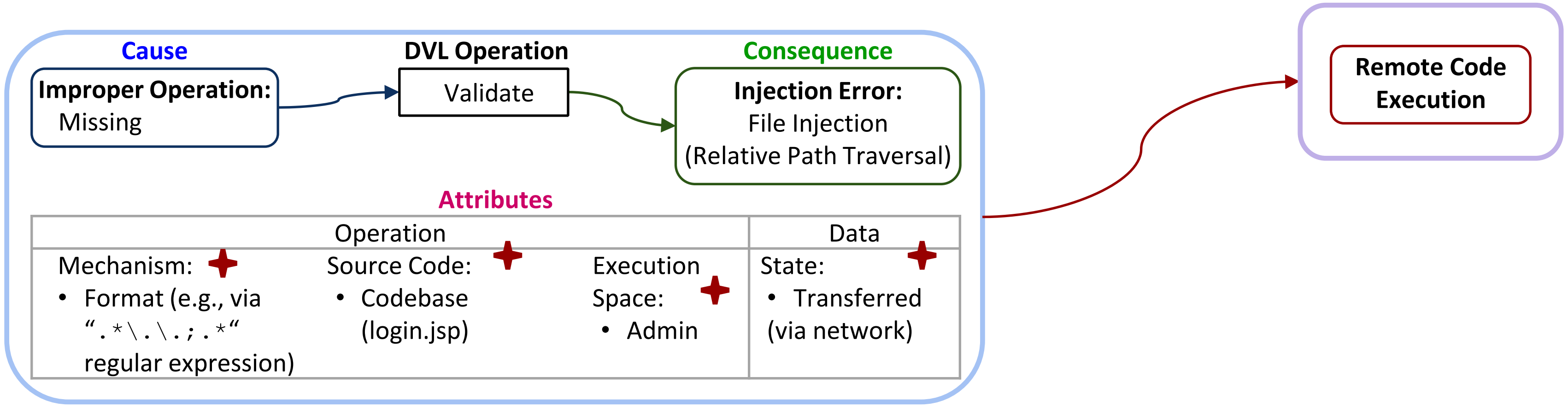
File Injection
(Relative Path Traversal)
DVL
(Validate: Missing, Data (URL), Policy)
Remote Code Execution
Caused by the Bug
The Failure – caused by final error(s)

- Proof-Of-Concept: TMSH command execution

```
https://[F5 Host]/tmui/login.jsp/..;/tmui/locallb/workspace/tmshCmd.jsp
```

```
../
```

BF Description of BIG-IP TMUI RCE



Cause

Improper Operation:
Missing

DVL Operation

Validate

Consequence

Injection Error:
File Injection
(Relative Path Traversal)

Remote Code Execution

Attributes

Operation			Data
Mechanism: ✚ • Format (e.g., via “.*\.\.;.*“ regular expression)	Source Code: ✚ • Codebase (login.jsp)	Execution Space: ✚ • Admin	State: ✚ • Transferred (via network)

□ The Bug

□ The Failure

BF Example 2: Updated Description of Heartbleed

Heartbleed (CVE-2014-0160)

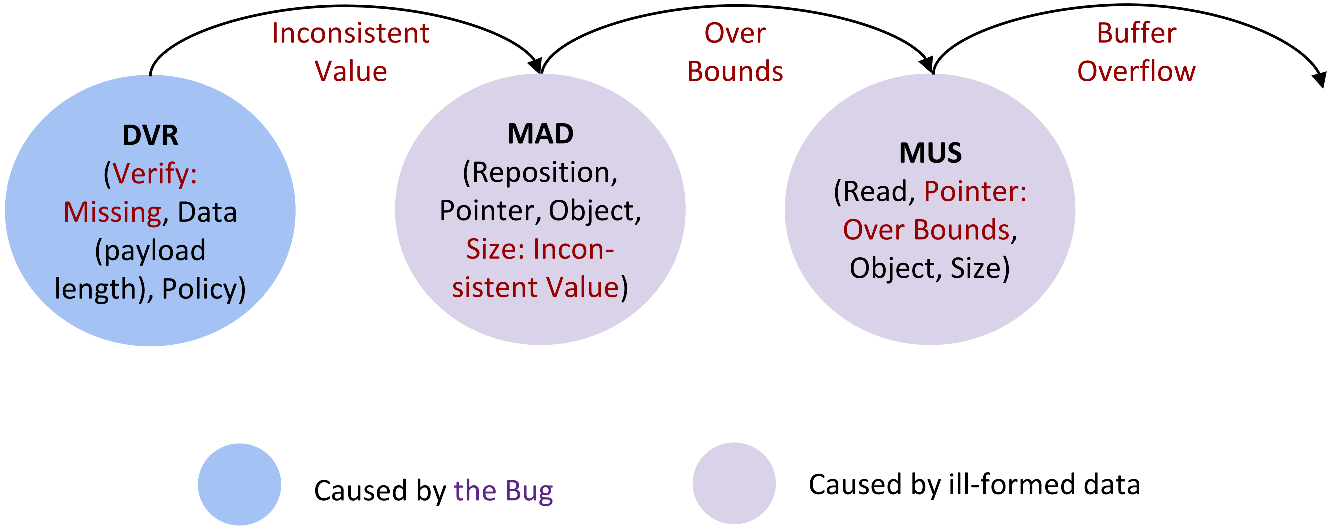
CVE-2014-0160 The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

```
dtls1_process_heartbeat(SSL *s)
  {
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */

  /* Read type and payload length first */
  hbtype = *p++;
  n2s(p, payload);
  pl = p;
...
  if (hbtype == TLS1_HB_REQUEST)
     {
     unsigned char *buffer, *bp;
...
     /* Allocate memory for the response, size is 1 byte
     * message type, plus 2 bytes payload, plus
     * payload, plus padding
     */
     buffer = OPENSSL_malloc(1 + 2 + payload + padding);
     bp = buffer;

     /* Enter response type, length and copy payload */
     *bp++ = TLS1_HB_RESPONSE;
     s2n(payload, bp);
     memcpy(bp, pl, payload);
```

```
/* Naive implementation of memcpy
void *memcpy (void *dst, const void *src, size_t n)
{
    size_t i;        payload
    for (i=0; i<n; i++)
        *(char *) dst++ = *(char *) src++;
    return dst;
}                     bp                pl
```



DVR (Verify: Missing, Data (payload length), Policy)

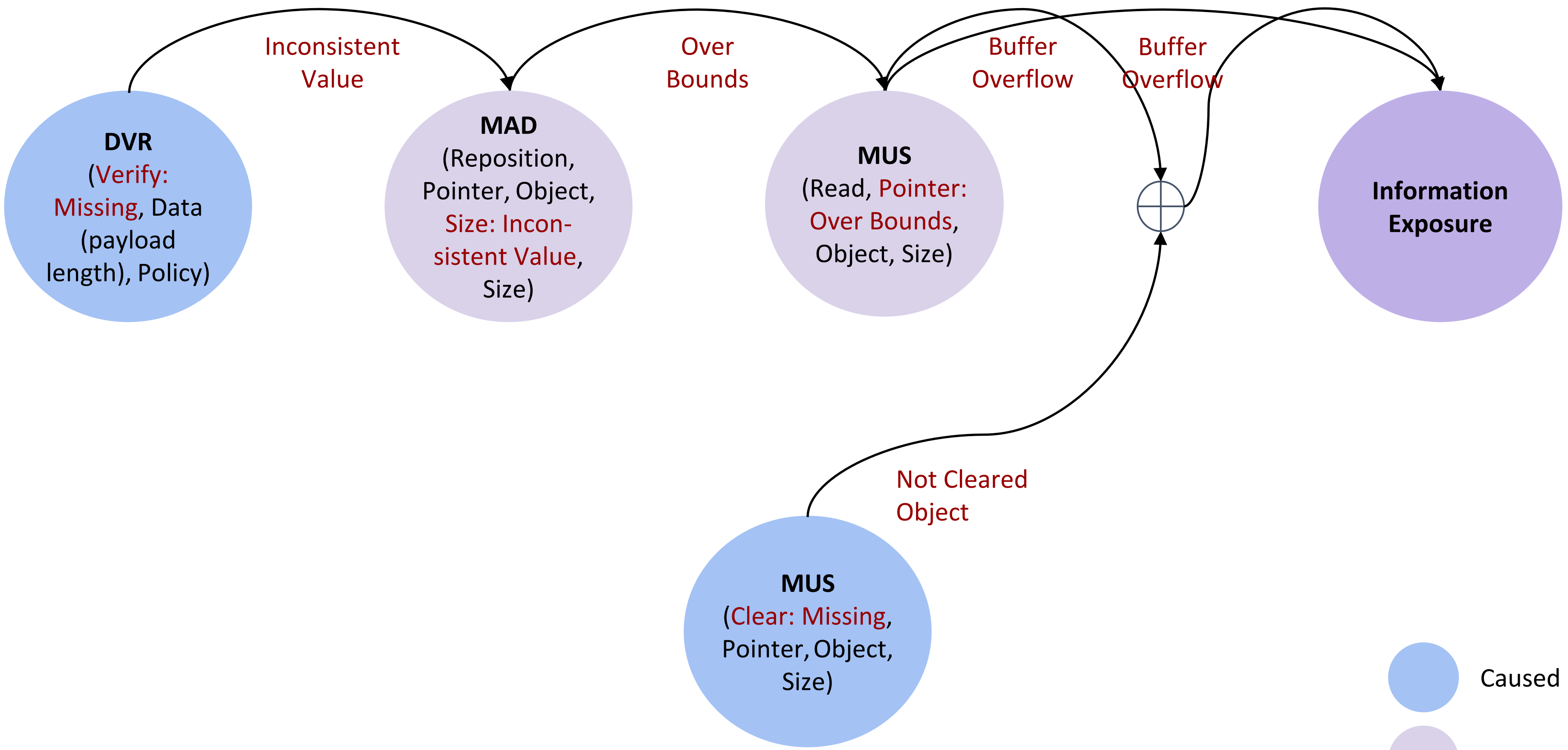
MAD (Reposition, Pointer, Object, Size: Inconsistent Value)

MUS (Read, Pointer: Over Bounds, Object, Size)

● Caused by the Bug

● Caused by ill-formed data

Clear Causality in Heartbleed



DVR
(Verify: Missing, Data (payload length), Policy)

Inconsistent Value

MAD
(Reposition, Pointer, Object, Size: Incon-sistent Value, Size)

Over Bounds

MUS
(Read, Pointer: Over Bounds, Object, Size)

Buffer Overflow

Buffer Overflow

Information Exposure

Not Cleared Object

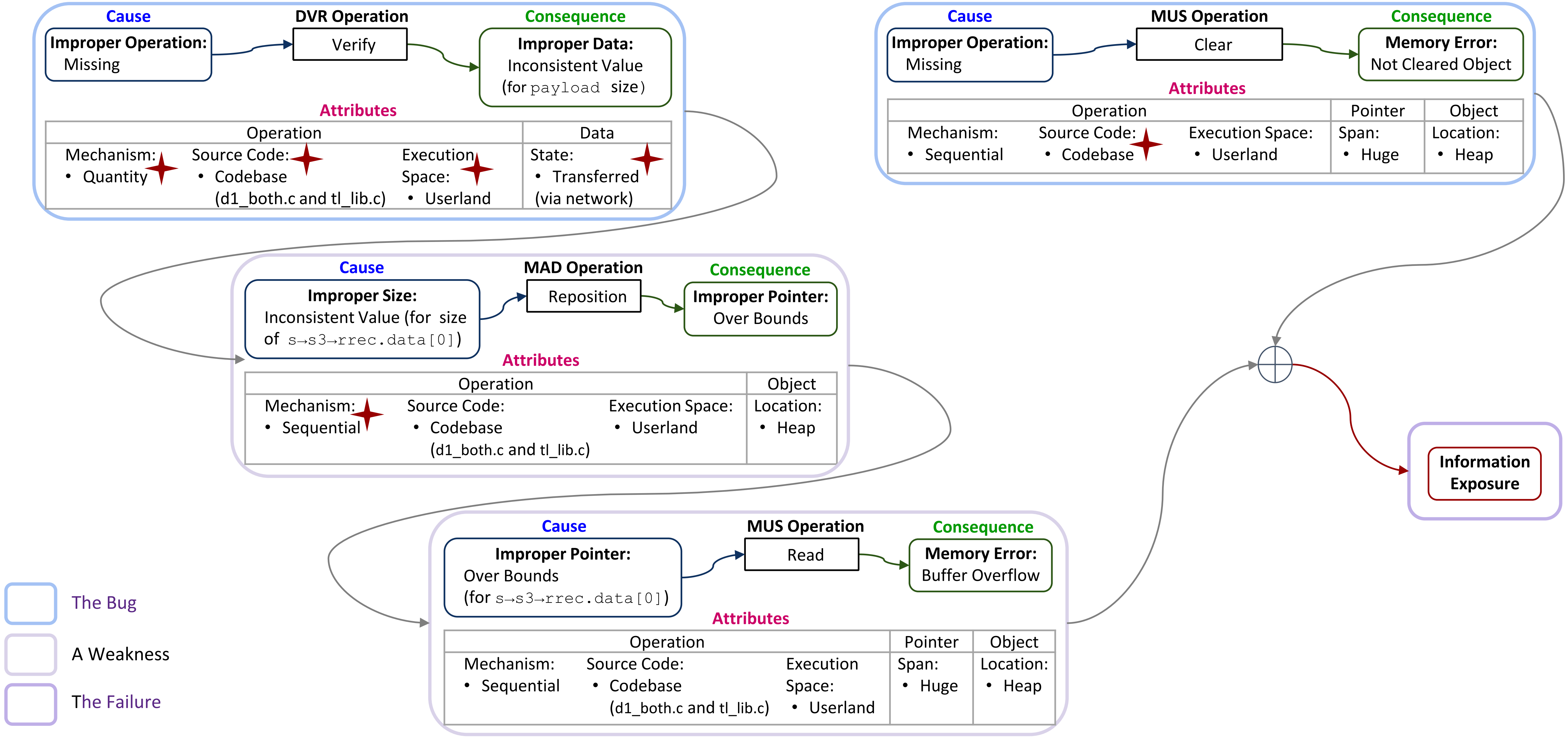
MUS
(Clear: Missing, Pointer, Object, Size)

● Caused by the Bug

● Caused by ill-formed data

● The Failure – caused by final error(s)

BF Description of Heartbleed



Cause

Improper Operation: Missing

DVR Operation

Verify

Consequence

Improper Data: Inconsistent Value (for payload size)

Attributes

Operation			Data
Mechanism: ✦ • Quantity	Source Code: ✦ • Codebase (d1_both.c and tl_lib.c)	Execution Space: ✦ • Userland	State: ✦ • Transferred (via network)

Cause

Improper Operation: Missing

MUS Operation

Clear

Consequence

Memory Error: Not Cleared Object

Attributes

Operation			Pointer	Object
Mechanism: • Sequential	Source Code: ✦ • Codebase	Execution Space: • Userland	Span: • Huge	Location: • Heap

Cause

Improper Size: Inconsistent Value (for size of s→s3→rrec.data[0])

MAD Operation

Reposition

Consequence

Improper Pointer: Over Bounds

Attributes

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

Cause

Improper Pointer: Over Bounds (for s→s3→rrec.data[0])

MUS Operation

Read

Consequence

Memory Error: Buffer Overflow

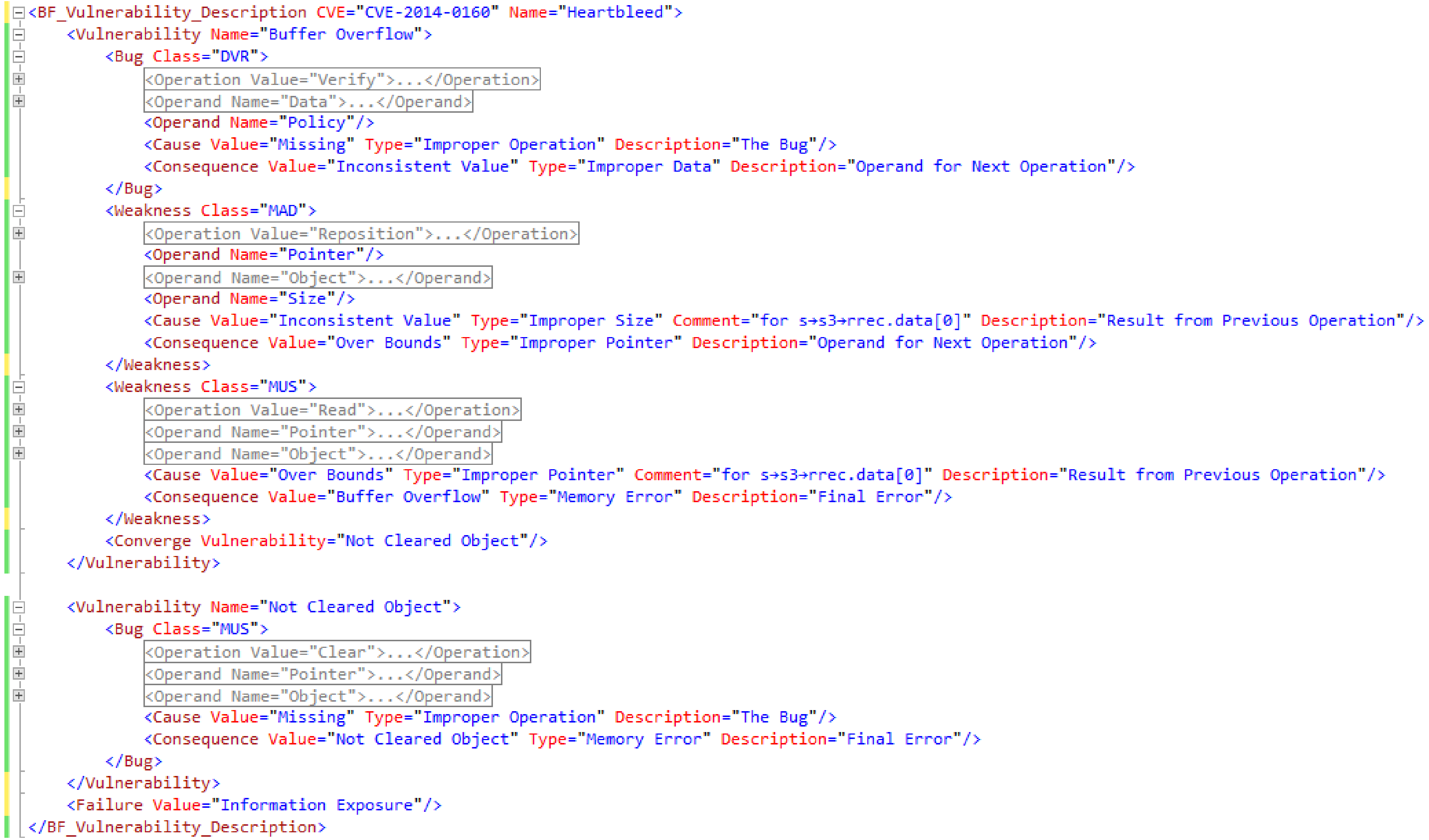
Attributes

Operation			Pointer	Object
Mechanism: • Sequential	Source Code: • Codebase (d1_both.c and tl_lib.c)	Execution Space: • Userland	Span: • Huge	Location: • Heap

Information Exposure

□	The Bug
□	A Weakness
□	The Failure

Heartbleed in XML Format



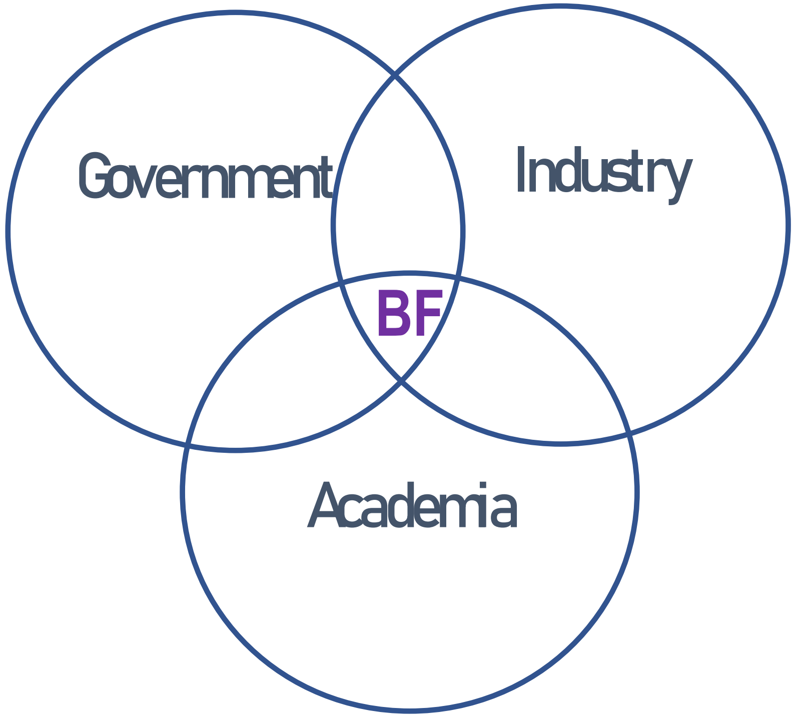
```
<BF_Vulnerability_Description CVE="CVE-2014-0160" Name="Heartbleed">
    <Vulnerability Name="Buffer Overflow">
        <Bug Class="DVR">
            <Operation Value="Verify">...</Operation>
            <Operand Name="Data">...</Operand>
            <Operand Name="Policy"/>
            <Cause Value="Missing" Type="Improper Operation" Description="The Bug"/>
            <Consequence Value="Inconsistent Value" Type="Improper Data" Description="Operand for Next Operation"/>
        </Bug>
        <Weakness Class="MAD">
            <Operation Value="Reposition">...</Operation>
            <Operand Name="Pointer"/>
            <Operand Name="Object">...</Operand>
            <Operand Name="Size"/>
            <Cause Value="Inconsistent Value" Type="Improper Size" Comment="for s→s3→rrec.data[0]" Description="Result from Previous Operation"/>
            <Consequence Value="Over Bounds" Type="Improper Pointer" Description="Operand for Next Operation"/>
        </Weakness>
        <Weakness Class="MUS">
            <Operation Value="Read">...</Operation>
            <Operand Name="Pointer">...</Operand>
            <Operand Name="Object">...</Operand>
            <Cause Value="Over Bounds" Type="Improper Pointer" Comment="for s→s3→rrec.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="MUS">
            <Operation Value="Clear">...</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_Vulnerability_Description>
```

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/