Explainable Vulnerabilities Descriptions with NIST BF

Keynote – ISSRE, SHIFT & IWSF:
Software Hardware Interaction Faults &
International Workshop on Software Faults

Charlotte, NC, USA (Remote Attendance) – Oct. 31, 2022

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

Irena Bojanova

https://samate.nist.gov/BF

Agenda

NIST

- Introduction:
  - Terminology:
    - ✓ Bug
    - ✓ Weakness
    - ✓ Vulnerability
    - ✓ Failure
  - "Bad Alloc" Pattern
- Existing Repositories:
  - CWE
  - CVE
  - NVD
  - KEV
- The Bugs Framework (BF)
  - Goals
  - Features
- BF Taxonomy
- Validation towards CWE
- BF Hands On:
  - BF Descriptions of CVEs
  - ML, AI on Failures and Risks
- Potential Impacts

Introduction

Terminology

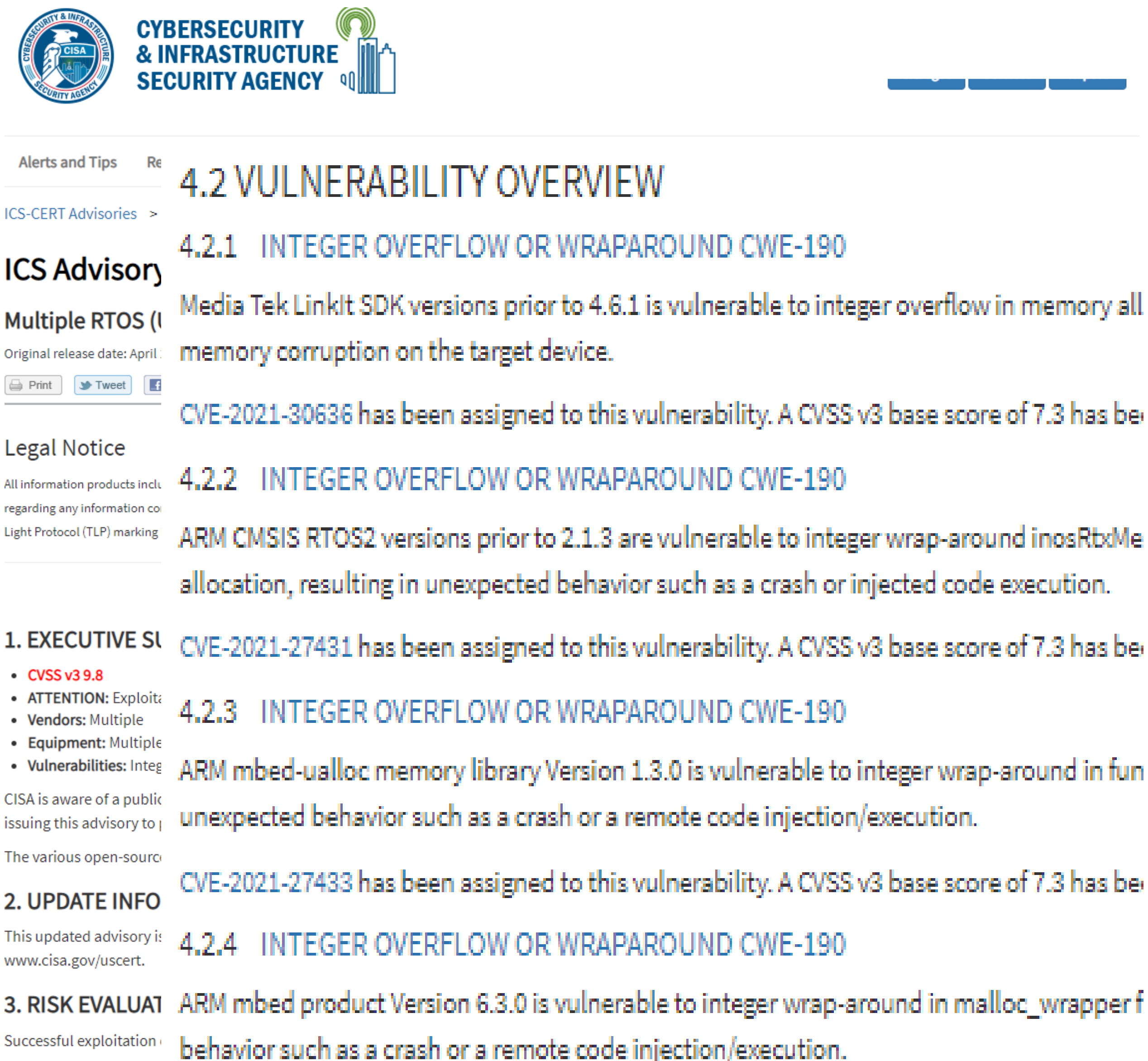
NIST

- Software Bug:
  - A coding error
  - Needs to be fixed
- Software Weakness:
  - 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

SAMATE

The Bugs Framework (BF)
https://samate.nist.gov/BF

"BadAlloc" Pattern – 25 CVEs



CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Alerts and Tips Re

ICS-CERT Advisories >

ICS Advisory

Multiple RTOS (

Original release date: April

Print Tweet

Legal Notice

All information products inclu
regarding any information co
Light Protocol (TLP) marking

1. EXECUTIVE SU

- CVSS v3 9.8
- ATTENTION: Exploita
- Vendors: Multiple
- Equipment: Multiple
- Vulnerabilities: Integ

CISA is aware of a public
issuing this advisory to

The various open-sourc

2. UPDATE INFO

This updated advisory is
www.cisa.gov/uscert.

3. RISK EVALUAT

Successful exploitation

4.2 VULNERABILITY OVERVIEW

4.2.1 INTEGER OVERFLOW OR WRAPAROUND CWE-190

Media Tek LinkIt SDK versions prior to 4.6.1 is vulnerable to integer overflow in memory all
memory corruption on the target device.

CVE-2021-30636 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

4.2.2 INTEGER OVERFLOW OR WRAPAROUND CWE-190

ARM CMSIS RTOS2 versions prior to 2.1.3 are vulnerable to integer wrap-around inosRtxMe
allocation, resulting in unexpected behavior such as a crash or injected code execution.

CVE-2021-27431 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

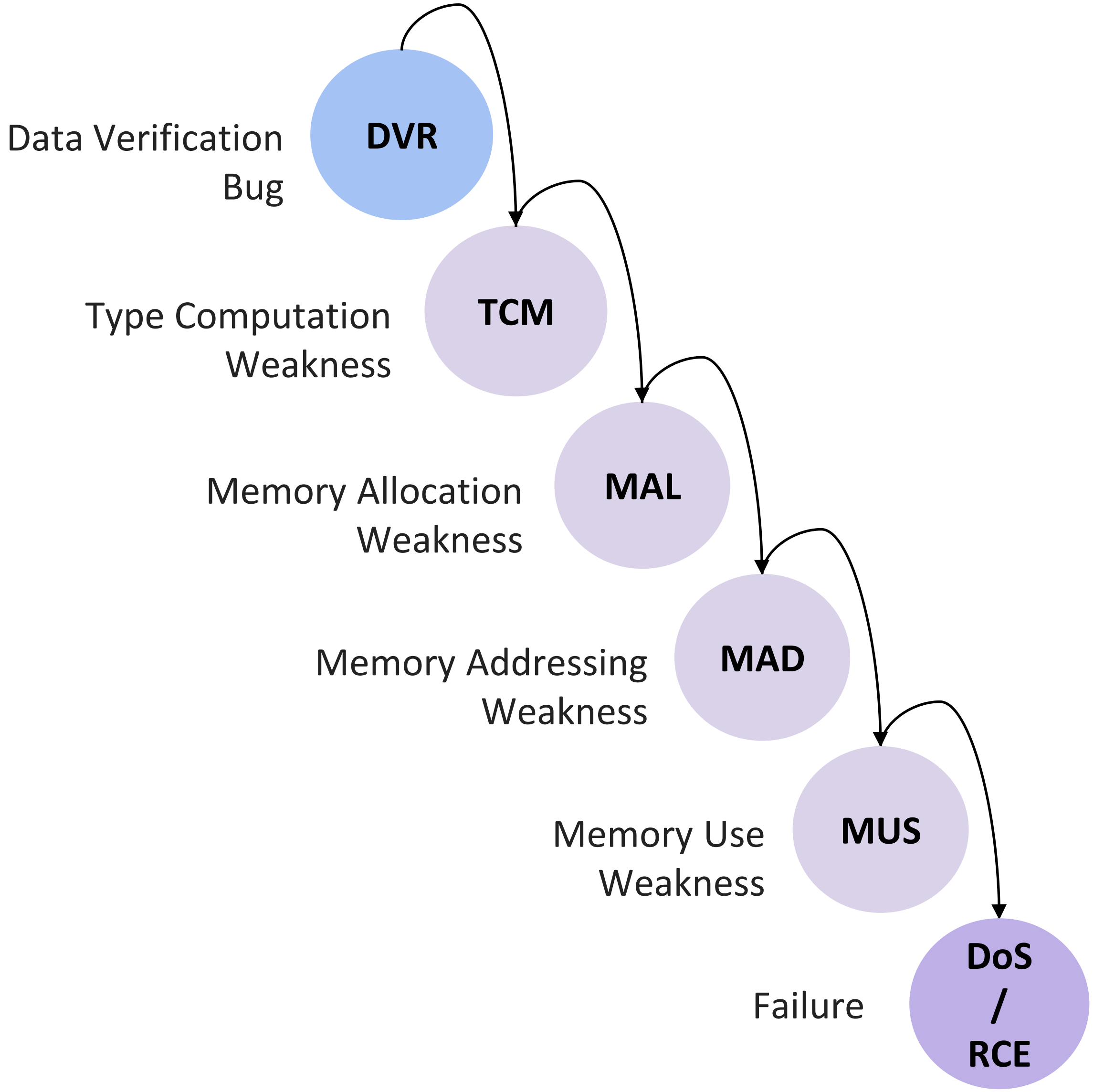
4.2.3 INTEGER OVERFLOW OR WRAPAROUND CWE-190

ARM mbed-ualloc memory library Version 1.3.0 is vulnerable to integer wrap-around in fun
unexpected behavior such as a crash or a remote code injection/execution.

CVE-2021-27433 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

4.2.4 INTEGER OVERFLOW OR WRAPAROUND CWE-190

ARM mbed product Version 6.3.0 is vulnerable to integer wrap-around in malloc_wrapper f
behavior such as a crash or a remote code injection/execution.



Existing Repositories

Commonly Used Repositories

- Weaknesses:
  CWE – Common Weakness Enumeration https://cwe.mitre.org/

- Vulnerabilities:
  CVE – Common Vulnerabilities and Exposures https://cve.mitre.org/
  →over 18 000 documented in 2020

- Vulnerabilities by priority for remediation – CVEs:
  KEV – Known Exploited Vulnerabilities Catalog https://www.cisa.gov/known-exploited-vulnerabilities-catalog

- Linking weaknesses to vulnerabilities – CWEs to CVEs
  NVD – National Vulnerabilities Database https://nvd.nist.gov/
  → links also to KEV

Repository Problems

NIST

1. Imprecise Descriptions – CWE & CVE
2. Unclear Causality – CWE & CVE
3. No Tracking Methodology – CVE
4. Gaps in Coverage – CWE
5. Overlaps in Coverage – CWE
6. No Tools – CWE & CVE

Problem #1: Imprecise Descriptions

- Example:

  CWE-502: Deserialization of Untrusted Data:

  The application deserializes untrusted data without *sufficiently* verifying that the resulting data will be valid.

  - Unclear what "*sufficiently*" means,
  - "verifying that data is valid" is also confusing

Problems #2, #3: Unclear Causality, Tracking

- Example:

  CVE-2018-5907
  Possible buffer overflow in msm_adsp_stream_callback_put due to lack of input validation of user-provided data that leads to integer overflow in all Android releases (Android for MSM, Firefox OS for MSM, QRD Android) from CAF using the Linux kernel.

  → the NVD label is CWE-190

  While the CWEs chain is:
  CWE-20 → CWE-190 → CWE-119

Problems #4, #5: Gaps/Overlaps in Coverage

NIST

- Example:

CWEs coverage of buffer overflow by:

✓ Read/ Write

✓ Over/ Under

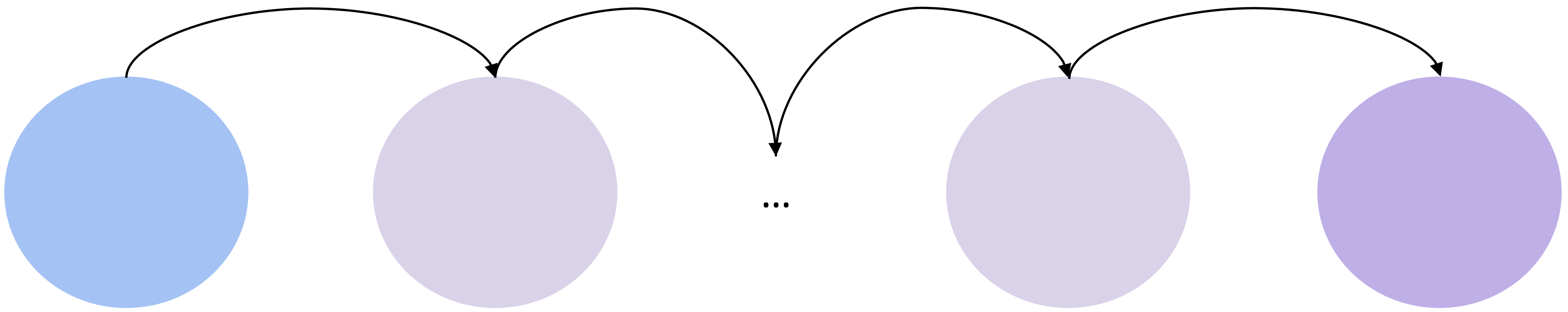
✓ Stack/ Heap

	Over	Under	Either End		Stack	Heap
Read	CWE-127	CWE-126	CWE-125		✦	✦
Write	CWE-124	CWE-120	CWE-123 CWE-787 ✦		CWE-121	CWE-122
Read/ Write	CWE-786	CWE-788	✦		✦	✦

The Bugs Framework (BF)

BF Goals

1. Solve the problems of imprecise descriptions and unclear causality



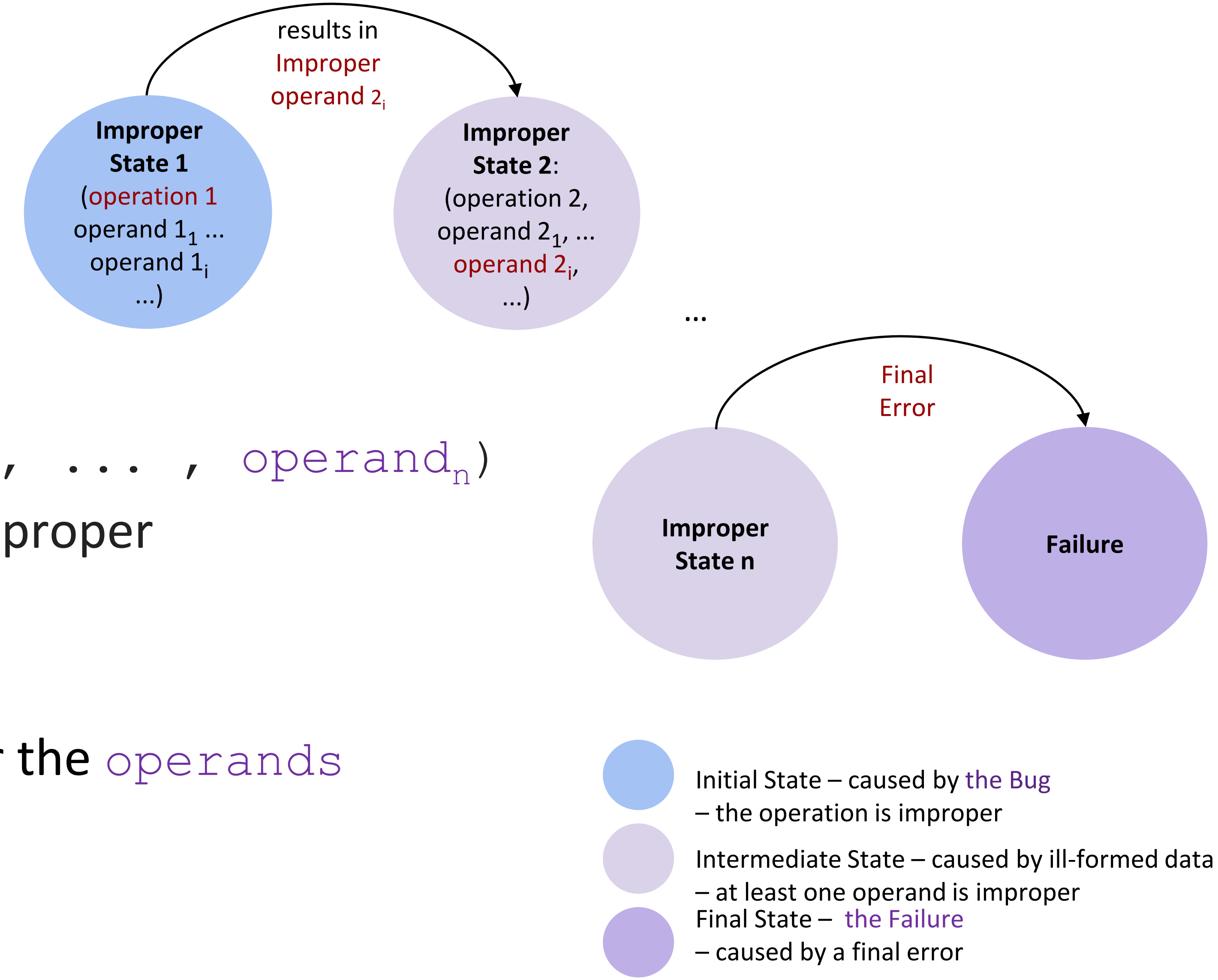
2. Solve the problems of gaps and overlaps in coverage

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

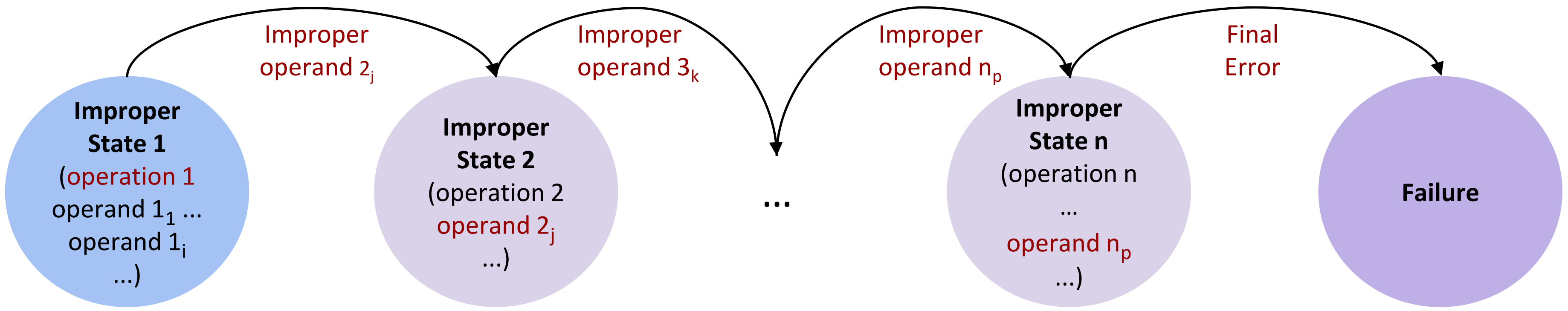
- Transition – the result of the operation over the operands

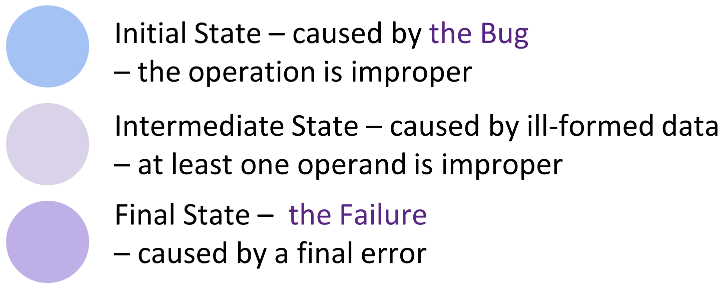


BF Features – Chaining Weaknesses

NIST

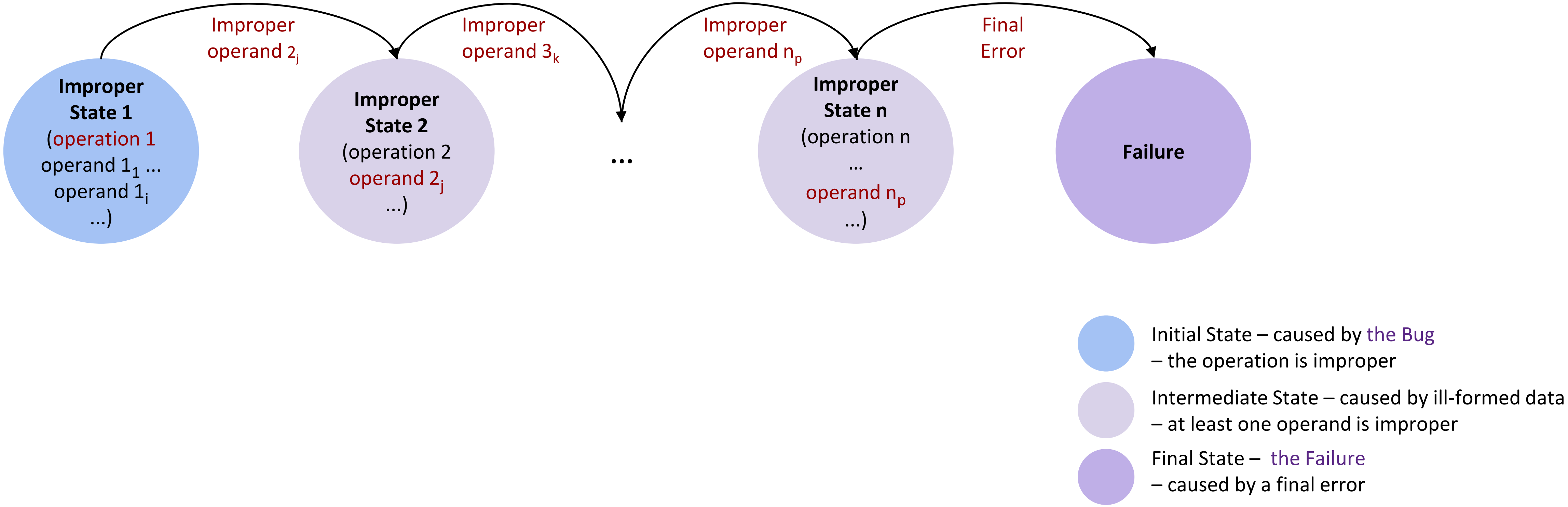
- BF describes a vulnerability as:
  - A chain of improper states and their transitions
  - States change until a failure is reached



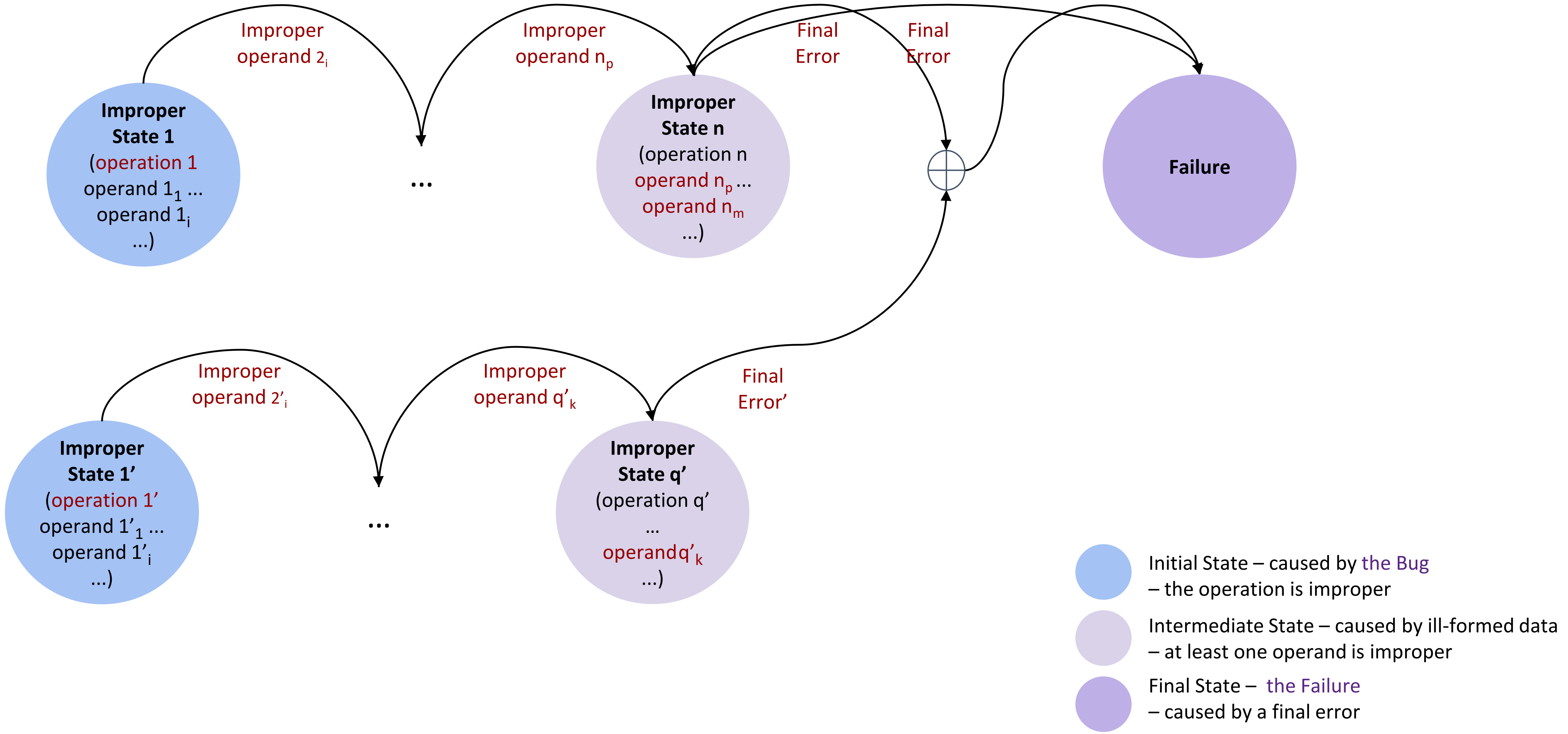


BF Features – Backtracking

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

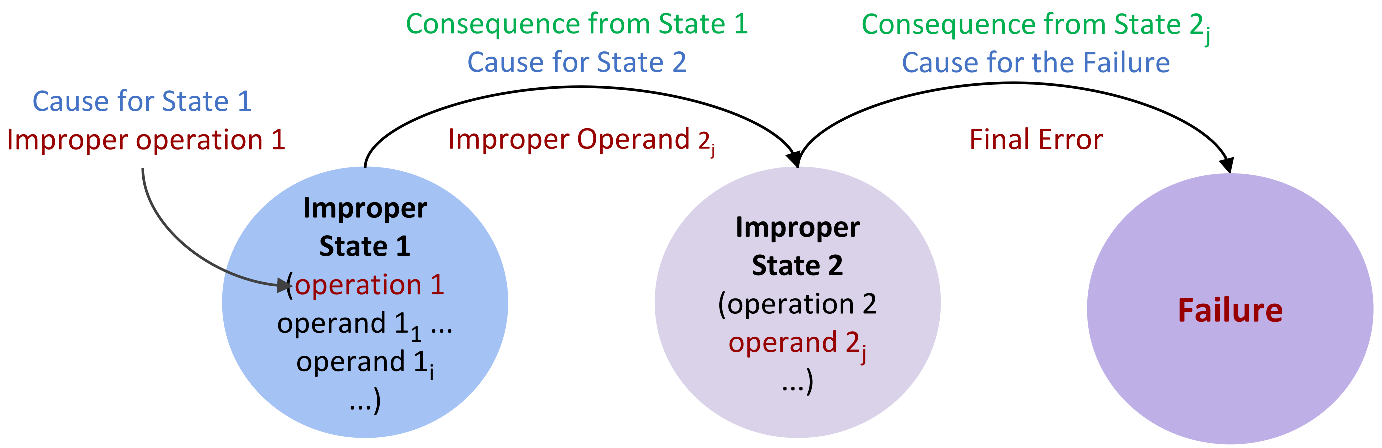


BF Features – Converging Vulnerabilities



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



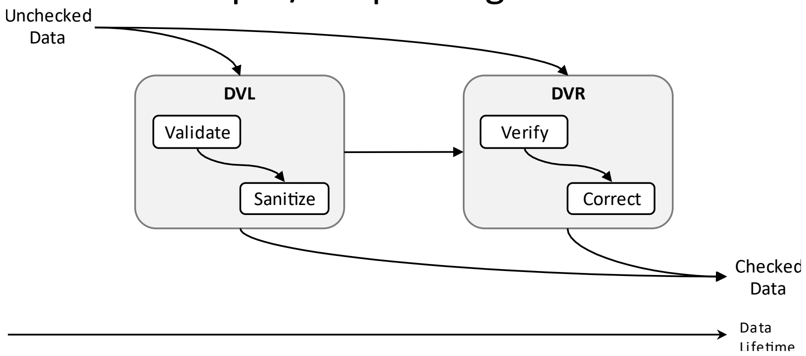
- BF bug/weakness description – instance of a BF class with:
    - one cause
    - one operation
    - one consequence
    - and their attributes

- BF vulnerability description –
    - chain of BF classes instances
    - consequence–cause transitions.

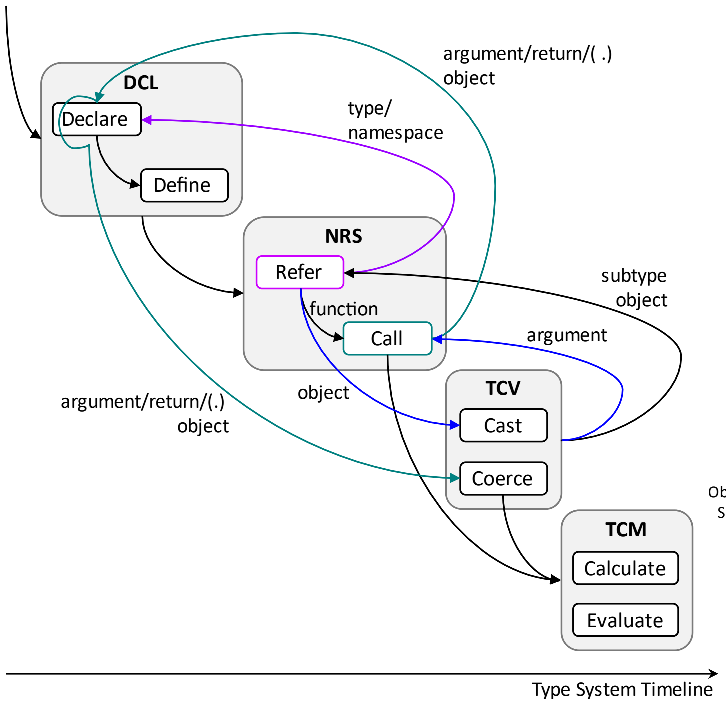
BF Taxonomy

BF – Bugs Models

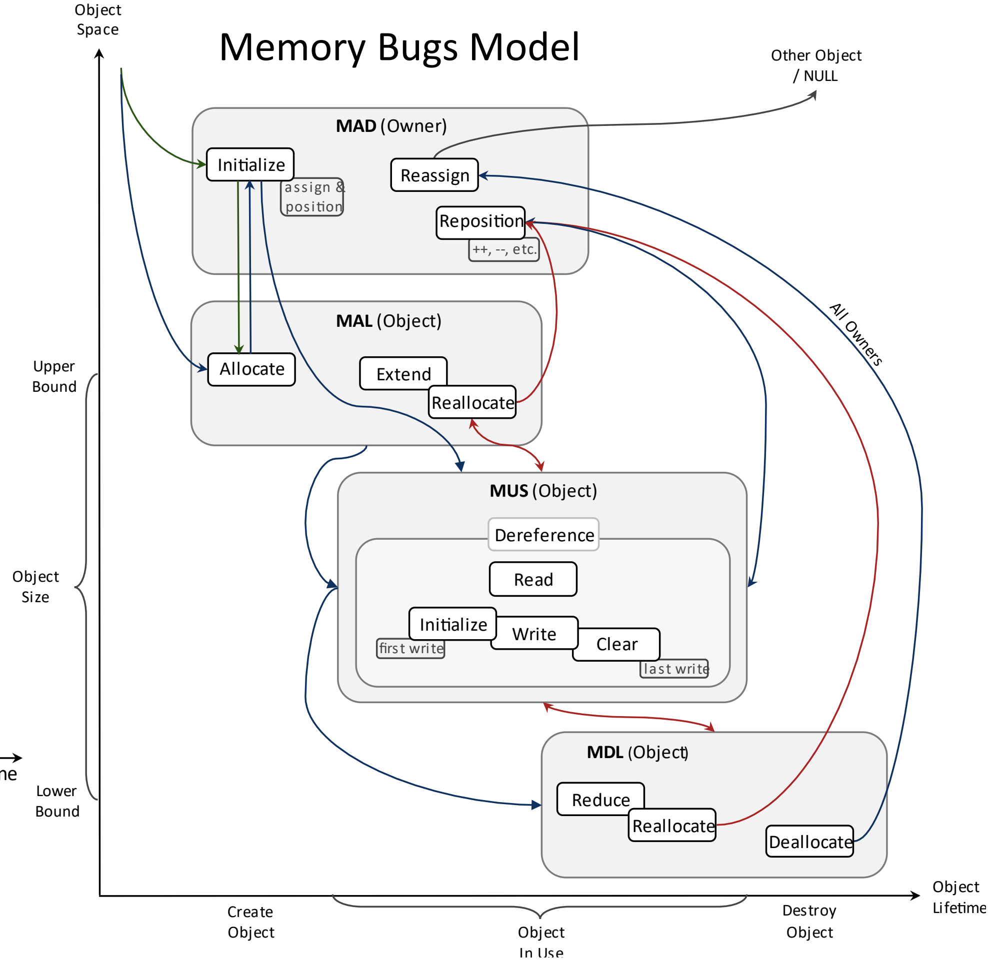
BF Input/Output Bugs Model



BF Data Type Bugs Model



Memory Bugs Model

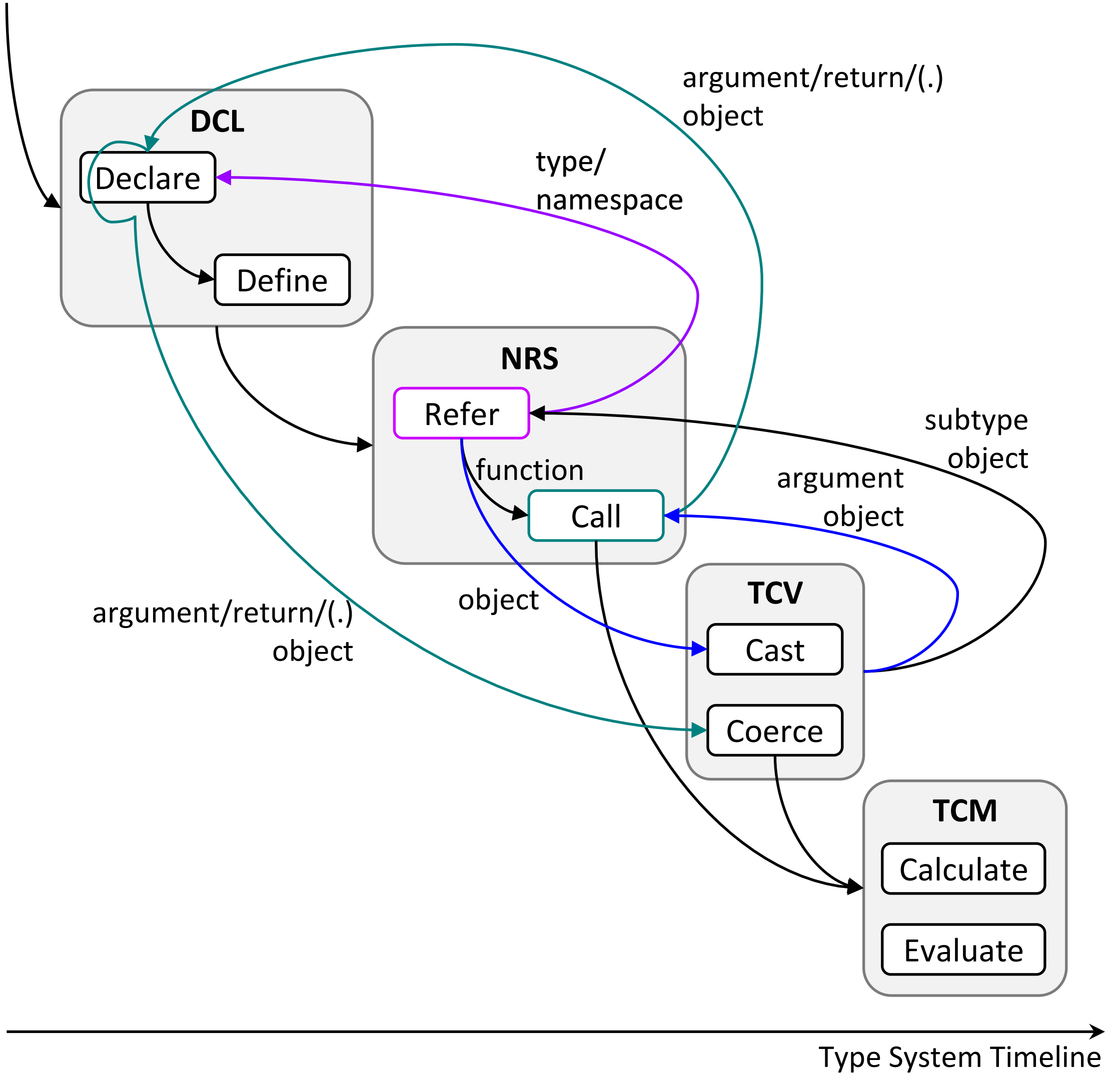


- Identify Secure Code Principles:
  - Input/Output Safety
  - Data Type Safety
  - Memory Safety

BF Data Type Bugs Model

- Four phases, corresponding to the BF Data Type Bugs classes: DCL, NRS, TCV, and TCM

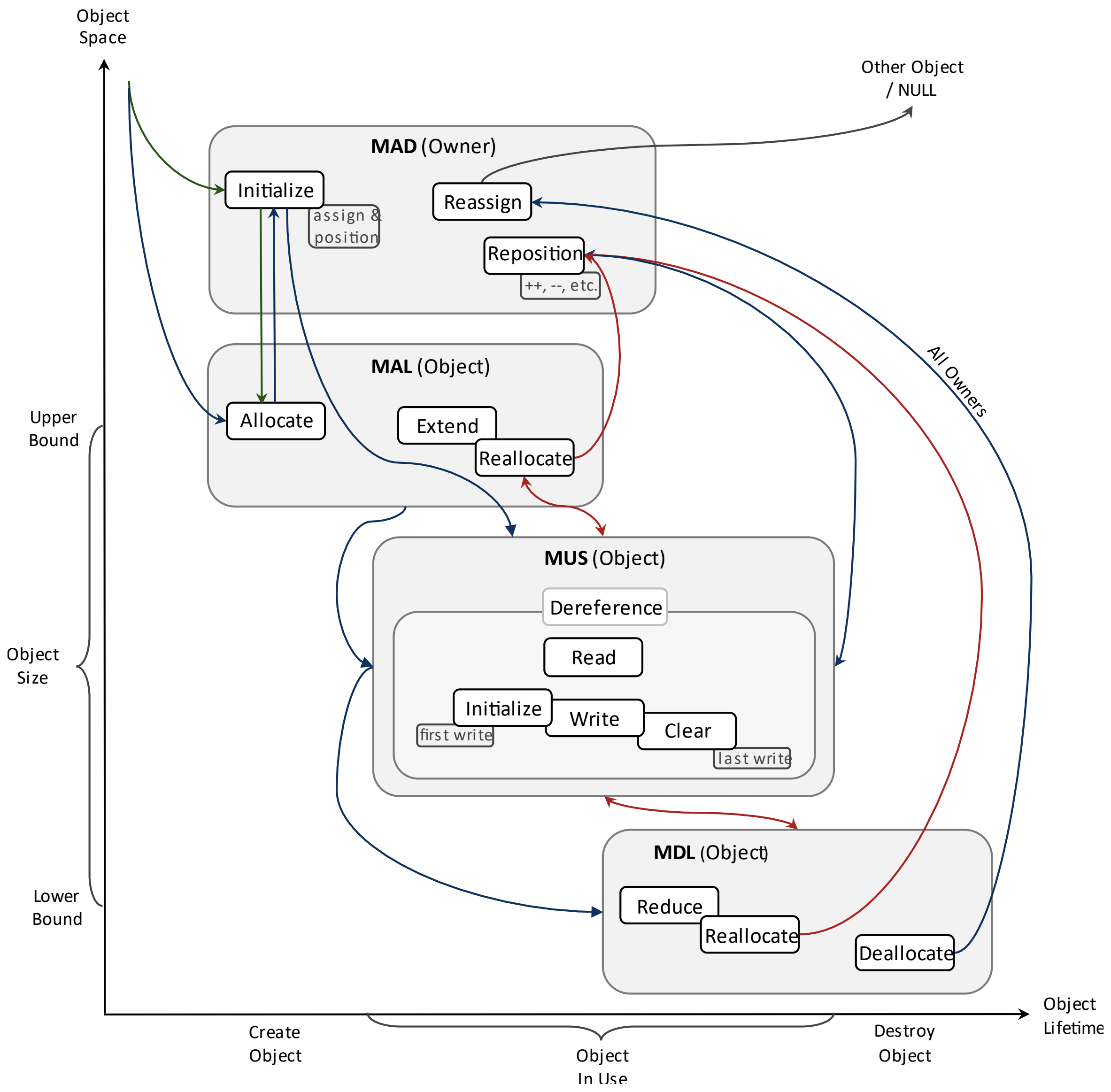
- Data Type operations flow



➢ **Entity**:
  - Object
  - Function
  - Data Type
  - Namespace

BF Memory Bugs Model

- The BF Memory Bugs Model:
  - Four phases, corresponding to the BF memory bugs classes: MAD, MAL, MUS, MDL
  - Memory operations flow



BF – Clusters of Bugs Classes

- Input/Output Bugs:
  DVL, DVR
- Data Type Bugs:
  DCL, NRS, TVC, TCM
- Memory Bugs:
  MAD, MAL, MUS, MD
- Cryptography Bugs:
  ENC, VRF, KMN
- Random Numbers Generation Bugs:
  RND, PRN
- Access Control Bugs:
  ATN, AUT
- Control Flow Bugs: ...
- Concurrency Bugs: ...

- BF class:
  - Set of Operations
  - Set of Causes
  - Set of Consequences

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

BF Classes – MAD & MUS

Memory Addressing Bugs (MAD) – *The pointer to an object is initialized, repositioned, or reassigned to an improper memory address.*

Causes

Improper Operation:
- Missing
- Mismatched
- Erroneous

Improper Data Value:
- Hardcoded Address
- Wrong Index
- Wrong Size Used

Improper Data Type:
- Wrong Index Type
- Casted Pointer

Improper Object Address:
- NULL Pointer
- Wild Pointer
- Dangling Pointer
- Over Bounds Pointer
- Under Bounds Pointer
- Wrong Position Pointer

Improper Object Size:
- Not Enough Memory Allocated

MAD Operations
- Initialize
- Reposition
- Reassign

Consequences

Improper Data Value:
- Forbidden Address

Improper Data Type:
- Casted Pointer

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

https://samate.nist.gov/BF/Classes/_MEM/MAD.html

Attributes

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

Memory Use Bugs (MUS) – *An object is initialized, read, written, or cleared improperly.*

Causes

Improper Operation:
- Missing
- Mismatched
- Erroneous

Improper Data Value:
- Forbidden Address
- Wrong Size Used

Improper Data Type:
- Casted Pointer

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

Improper Object Size:
- Not Enough Memory Allocated

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

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

https://samate.nist.gov/BF/Classes/_MEM/MUS.html

BF Classes – DVL & DVR

Data Validation Bugs (DVL) – *Data are validated (syntax check) or sanitized (escape, filter, repair) improperly.*

Causes

Improper Operation:
- Missing
- Erroneous

Improper Policy:
- Under-Restrictive Policy
- Over-Restrictive Policy

Improper Data:
- Corrupted Data
- Tampered Data

Improper Policy Data:
- Corrupted Policy
- Tampered Policy

DVL Operations
- Validate
- Sanitize

Consequences

Improper Data Type:
- Invalid Data

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

Attributes

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

https://samate.nist.gov/BF/Classes/_INP/DVL.html

Data Verification Bugs (DVR) – *Data are verified (semantics check) or corrected (assign value, remove) improperly.*

Causes

Improper Operation:
- Missing
- Erroneous

Improper Policy:
- Under-Restrictive Policy
- Over-Restrictive Policy

Improper Data Type:
- Invalid Data

DVR Operations
- Verify
- Correct

Consequences

Improper Data Value:
- Wrong Value
- Inconsistent Value
- Wrong Type

Attributes

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

https://samate.nist.gov/BF/Classes/_INP/DVR.html

BF Classes – NRS, TCV, TCM

Name Resolution Bugs (NRS) – *The name of an object, a function, or a data type is resolved improperly or bound to an improper data type or implementation.*

Causes

Improper Operation:
- Erroneous

Improper Scope:
- Missing Qualifier
- Wrong Qualifier

Improper Data Type:
- Incomplete Type
- Wrong Generic Type
- Confused Subtype
- Wrong Argument Type

Improper Function:
- Missing Overridden Function
- Missing Overloaded Function

NRS Operations
- Refer
- Call

Consequences

Improper Data Value:
- Wrong Object Resolved Value

Improper Data Type:
- Wrong Object Resolved Type
- Wrong Type Resolved

Improper Function:
- Wrong Function Resolved
- Wrong Generic Function Bound
- Wrong Overridden Function Bound
- Wrong Overloaded Function Bound

Attributes

Mechanism:	Source Code:	Entity:	Data Type Kind:
• Resolve • Bind • Early Bind • Late Bind • Ad-hoc Bind	• Codebase • Third Party • Standard Library • Compiler/Interpreter	• Object • Function • Data Type • Namespace	• Primitive • Structured

https://samate.nist.gov/BF/Classes/_DTC/NRS.html

Type Conversion Bugs (TCV) – *A data value is cast or coerced into another data type improperly.*

Causes

Improper Operation:
- Missing
- Wrong

Improper Data Value:
- Under Range
- Over Range
- Flipped Sign
- Wrong Object Resolved Value

Improper Data Type:
- Wrong Type
- Wrong Object Resolved Type
- Mismatched Argument Type

Improper Function:
- Missing Overloaded Function

TCV Operations
- Cast
- Coerce

Consequences

Improper Data Value:
- Wrong Result
- Truncated Value
- Distorted Value
- Rounded Value

Improper Data Type:
- Wrong Type

Attributes

Mechanism:	Source Code:	Data Value Kind:	Data Type Kind:
• Pass In • Pass Out	• Codebase • Third Party • Standard Library • Compiler/Interpreter	• Numeric • Text • Pointer • Boolean	• Primitive • Structured

https://samate.nist.gov/BF/Classes/_DTC/TCV.html

Type Computation Bugs (TCM) – *An arithmetic expression (over numbers, strings, or pointers) is calculated improperly, or a boolean condition is evaluated improperly.*

Causes

Improper Operation:
- Wrong
- Erroneous

Improper Data Value:
- Wrong Argument Value
- Wrong Object Resolved Value
- Reference vs. Object

Improper Data Type:
- Wrong Type
- Wrong Object Resolved Type

Improper Function:
- Wrong Function Resolved
- Wrong Generic Function Bound
- Wrong Overridden Function Bound
- Wrong Overloaded Function Bound

TCM Operations
- Calculate
- Evaluate

Consequences

Improper Data Value:
- Under Range
- Over Range
- Flipped Sign
- Wrong Result
- Wrap Around

Type Computation Error:
- Undefined

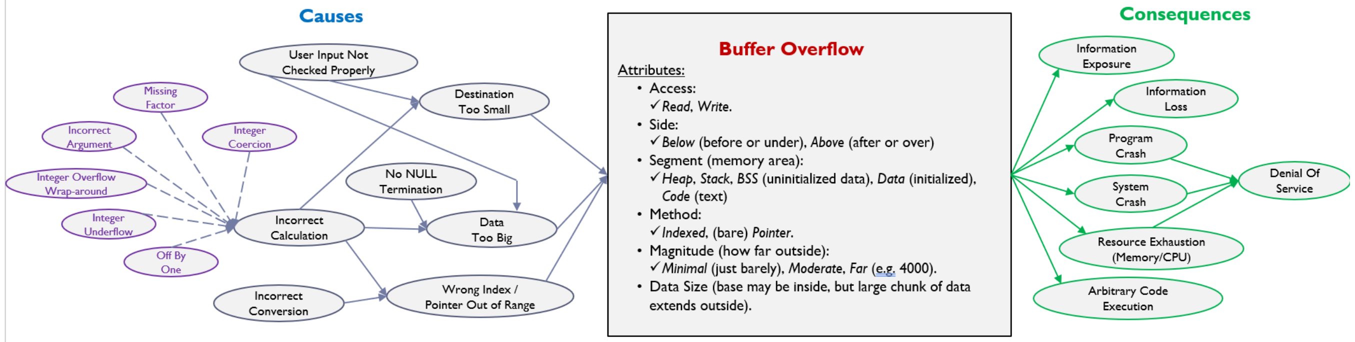
Attributes

Mechanism:	Source Code:	Data Value Kind:	Data Type Kind:
• Function • Operator • Method • Lambda Expression • Procedure	• Codebase • Third Party • Standard Library • Compiler/ Interpreter	• Numeric • Text • Pointer • Boolean	• Primitive • Structured

https://samate.nist.gov/BF/Classes/_DTC/TCM.html

BF Early Work – Buffer Overflow

NIST



They Know Your Weaknesses – Do You?: Reintroducing Common Weakness Enumeration

Yan Wu, Bowling Green State University
Irena Bojanova, University of Maryland, Baltimore County
Yaacov Yesha, University of Maryland University College

Abstract: Knowing what makes your software systems vulnerable to attacks is critical, as software vulnerabilities hurt security, reliability, and availability of the system as a whole. The Common Weakness Enumeration (CWE), a community effort that provides the foundation for such knowledge, ... precise enough to serve as the common lan- ... provide a common baseline for developers and

1.1 History of CWE

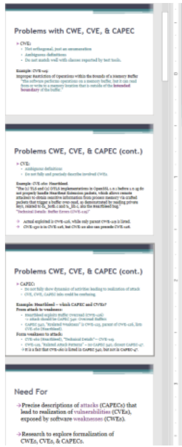
There have been several community efforts to leverage the existing large number of diverse real-world vulnerabilities. For example, an important step towards creating the needed collection of software weakness types was the establishment of the CVE (Common Vulnerabilities and Exposures) list [2] in 1999 by MITRE. Another important step from MITRE was creating the

Table 2. Buffer Overflow CWEs Attributes.

	before	after	either end	stack	heap
read	127	126	125		
write	124	120	123, 787	121	122
either r/w	786	788			

Where:

- access = either read/write
- outside = either before/below start or after/above



Formalizing Software Bugs

Irena Bojanova
UMUC, NIST

12/08/2014

CWE-128 in Z notation

CWE-128: Wrap-around Error: "Wrap around errors occur whenever a value is incremented past the maximum value for its type and therefore "wraps around" to a very small, negative, or undefined value."

$MAX\_INT : \mathbb{Z}$
$MIN\_INT : \mathbb{Z}$

$INT == \{i : \mathbb{Z} \mid MIN\_INT \le i \wedge i \le MAX\_INT\}$

$BAD\_INT : \mathbb{Z}$

$BAD\_INT < MIN\_INT \vee MAX\_INT < BAD\_INT$

$add, mul : INT \times INT \rightarrow INT \cup \{BAD\_INT\}$

Towards a "Periodic Table" of Bugs

Irena Bojanova, Paul E. Black, Yaacov Yesha, Yan Wu

April 9, 2015 NIST, BGSU

CVE-2014-160/CAPEC-540 in CSP

```
channel network 2;
enum {payloadLength, payload, validPayload, invalidPayload};
Attacker() = network!payloadLength -> network!payload –
>network?payloadResponse->Attacker();
CWE_126() = network?payloadLength -> network?payload->
    (payloadLengthIsEqualTopayloadSize->network!validPayload->CWE_126()
    [] payloadLengthIsNotEqualTopayloadSize->network!invalidPayload ->
    CWE_126());

System() = Attacker() ||| CWE_126();
```

Validation towards CWE

BF Class Related CWEs

- Identify CWEs:
  1. CWE Filtering
  2. Automated Extraction
  3. Manual Review

BF: https://samate.nist.gov/BF/
CWE: https://cwe.mitre.org/

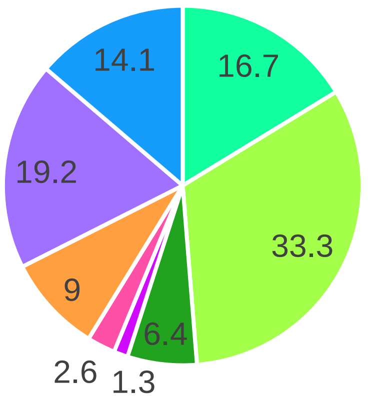
- BF Input/Output Bugs Classes – 161 CWEs:
  - 80.7% – Input Validation Operation
  - ➢ 68.3% – Injection Error

- BF Data Type Bugs Classes – 78 CWEs:
  - 50% Declaration/Definition Operation
  - 33.3% Cast/Coerce Operation
  - ➢ 16% Access Error
  - ➢ 0.6% Type Compute Error

- BF Memory Bugs Classes 52 CWEs:
  - 61.5% Initialize, Dereference, Read, Write, Clear Operations
  - ➢ 67.3% Memory Error

CWEs by BF Operation

- Data Type CWEs
  (incl. Integer Overflow, Juggling, and Pointer Arithmetics) – mapped by BF DCL, RNS, TCV, TCM operation

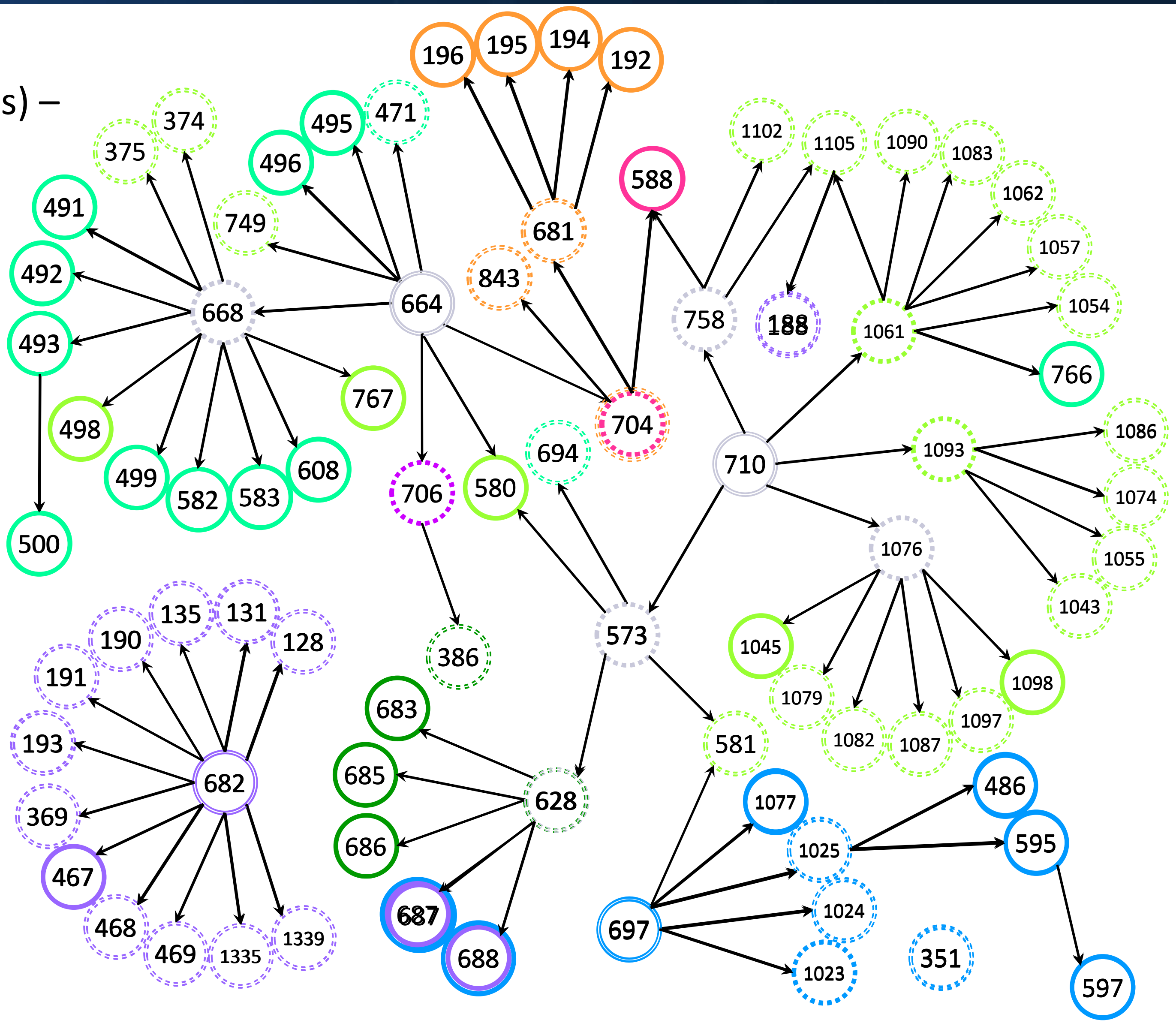


CWEs by DTC, NRS, TCV, and TCM operation:

DCL Declare
NRS Refer
TCV Cast
TCM Calculate
DCL Define
NRS Call
TCV Coerce
TCM Evaluate

CWEs by Abstraction:

Pillar
Base
Class
Variant



CWEs by BF Consequence

NIST

- Input/Output CWEs (incl. Injection) – mapped by BF DVL and BF DVR consequences

CWE by DVL Injection Error:

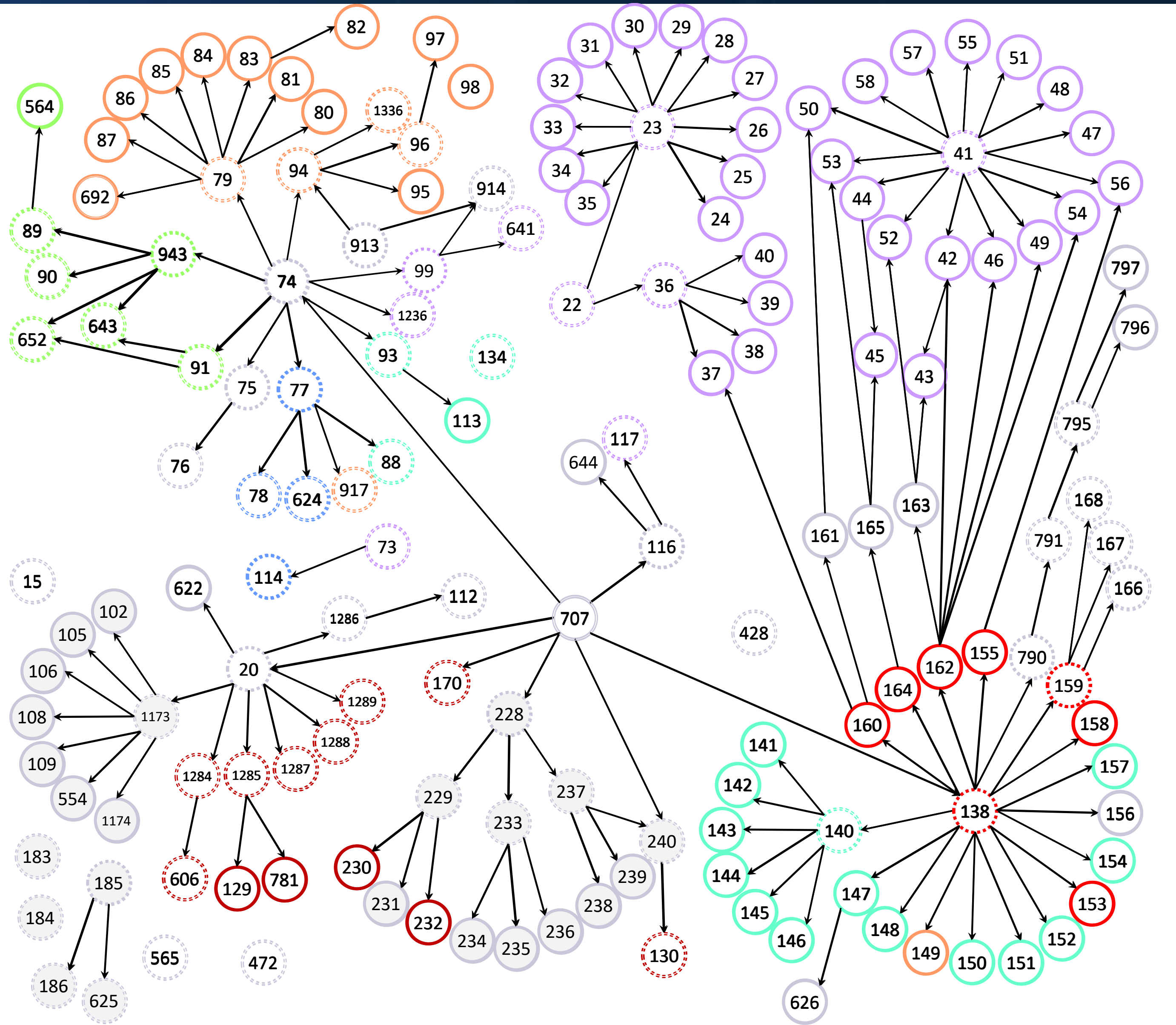
- Query Injection
- Command Injection
- Source Code Injection
- Parameter Injection
- File Injection

CWE by 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)

CWEs by Abstraction:

- Pillar
- Base
- Class
- Variant



BF – Defined

- BF is a …

  - ➢ Structured
  - ➢ Complete
  - ➢ Orthogonal
  - ➢ Language Independent

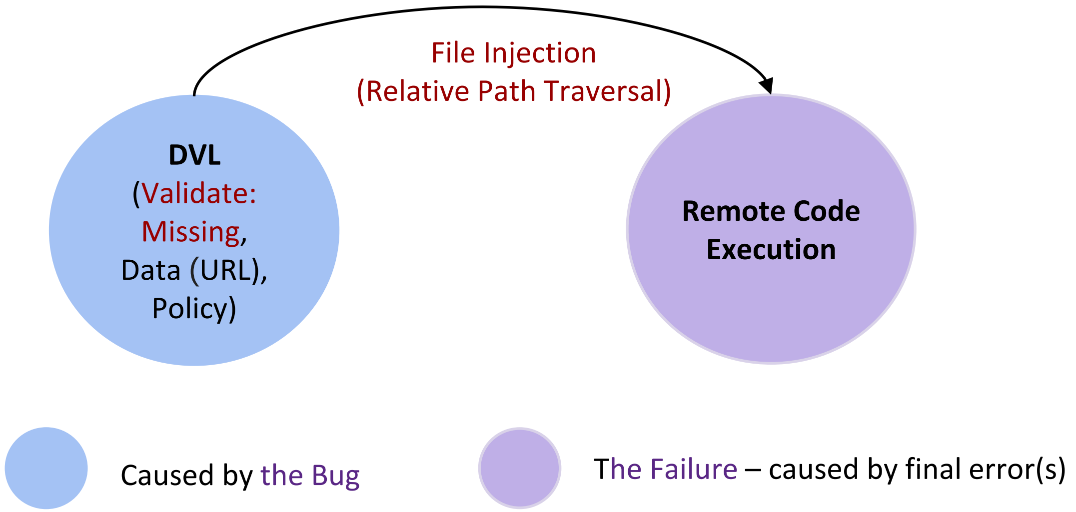
  Classification System of software bugs and weaknesses.

BF Hands On:
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/`



- 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)

Attributes

Mechanism:
- Format (e.g., via “.*\.\.;.*“ regular expression)

Source Code:
- Codebase (login.jsp)

Execution Space:
- Admin

Data State:
- Transferred (via network)

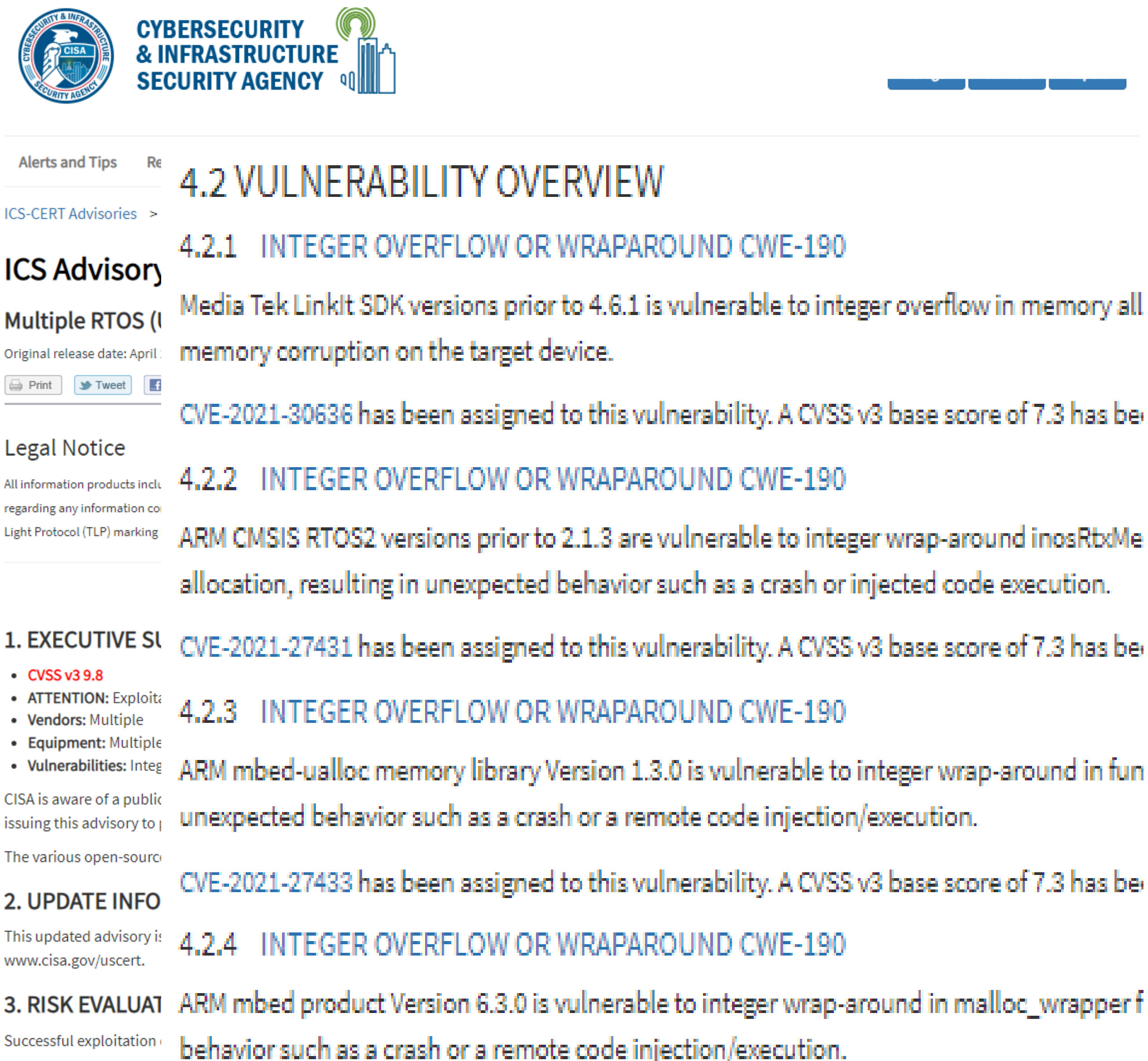
Remote Code Execution

The Bug

The Failure

BF Hands On:
Bad Alloc

"BadAlloc" Pattern – 25 CVEs


CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Alerts and Tips Re

ICS-CERT Advisories >

ICS Advisory

Multiple RTOS (

Original release date: April

Print Tweet

Legal Notice

All information products inclu
regarding any information co
Light Protocol (TLP) marking

1. EXECUTIVE SU

- CVSS v3 9.8
- ATTENTION: Exploit
- Vendors: Multiple
- Equipment: Multiple
- Vulnerabilities: Integ

CISA is aware of a public
issuing this advisory to

The various open-sourc

2. UPDATE INFO

This updated advisory i
www.cisa.gov/uscert.

3. RISK EVALUAT

Successful exploitation

4.2 VULNERABILITY OVERVIEW

4.2.1 INTEGER OVERFLOW OR WRAPAROUND CWE-190

Media Tek LinkIt SDK versions prior to 4.6.1 is vulnerable to integer overflow in memory all
memory corruption on the target device.

CVE-2021-30636 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

4.2.2 INTEGER OVERFLOW OR WRAPAROUND CWE-190

ARM CMSIS RTOS2 versions prior to 2.1.3 are vulnerable to integer wrap-around inosRtxMe
allocation, resulting in unexpected behavior such as a crash or injected code execution.

CVE-2021-27431 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

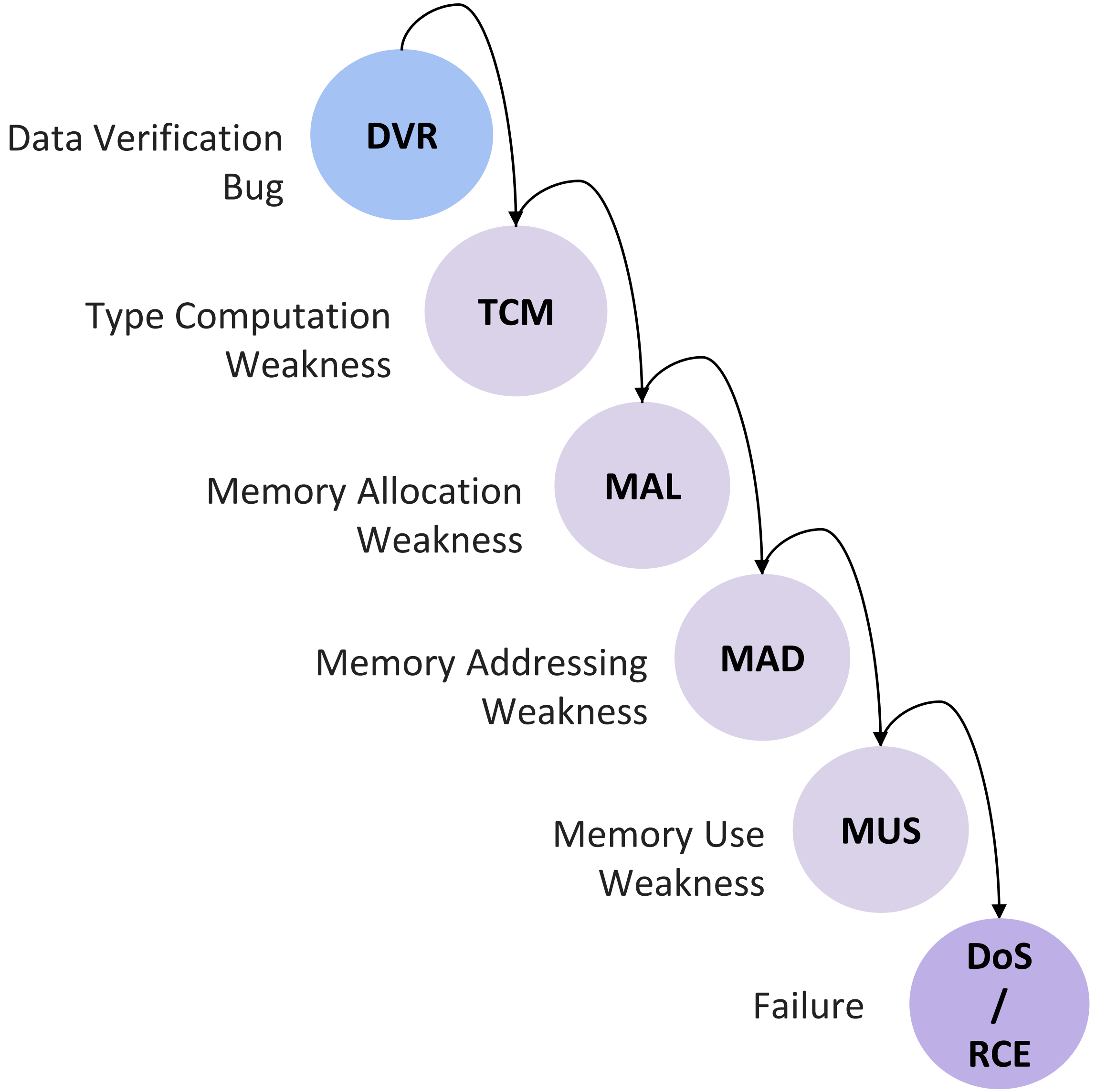
4.2.3 INTEGER OVERFLOW OR WRAPAROUND CWE-190

ARM mbed-ualloc memory library Version 1.3.0 is vulnerable to integer wrap-around in fun
unexpected behavior such as a crash or a remote code injection/execution.

CVE-2021-27433 has been assigned to this vulnerability. A CVSS v3 base score of 7.3 has be

4.2.4 INTEGER OVERFLOW OR WRAPAROUND CWE-190

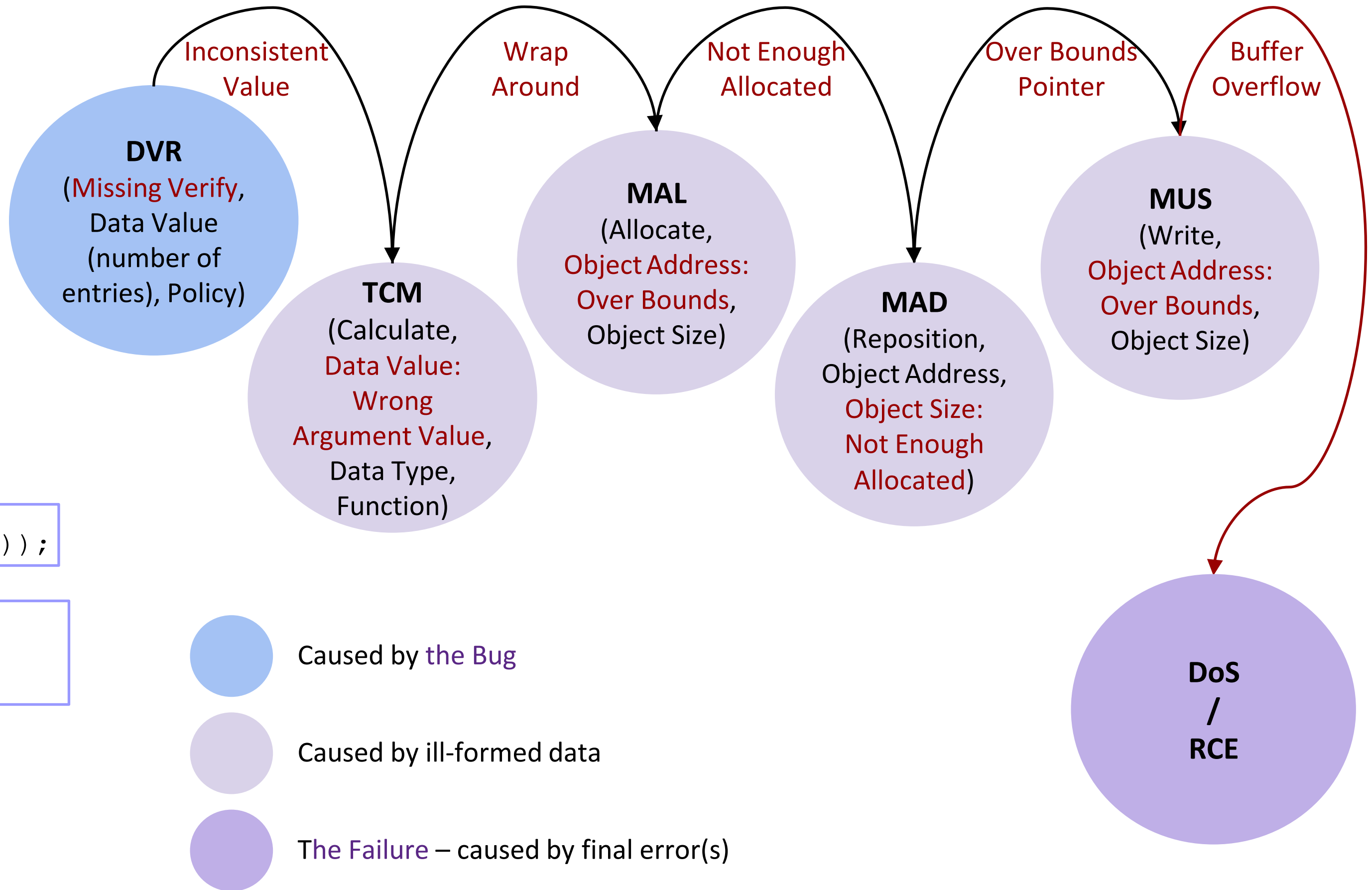
ARM mbed product Version 6.3.0 is vulnerable to integer wrap-around in malloc_wrapper f
behavior such as a crash or a remote code injection/execution.



“BadAlloc”(CVE-2021-21834)

CVE-2021-21834 An exploitable integer overflow vulnerability exists within the MPEG-4 decoding functionality of the GPAC Project on Advanced Content library v1.0.1. A specially crafted MPEG-4 input when decoding the atom for the “co64” FOURCC can cause an integer overflow due to unchecked arithmetic resulting in a heap-based buffer overflow that causes memory corruption. An attacker can convince a user to open a video to trigger this vulnerability.

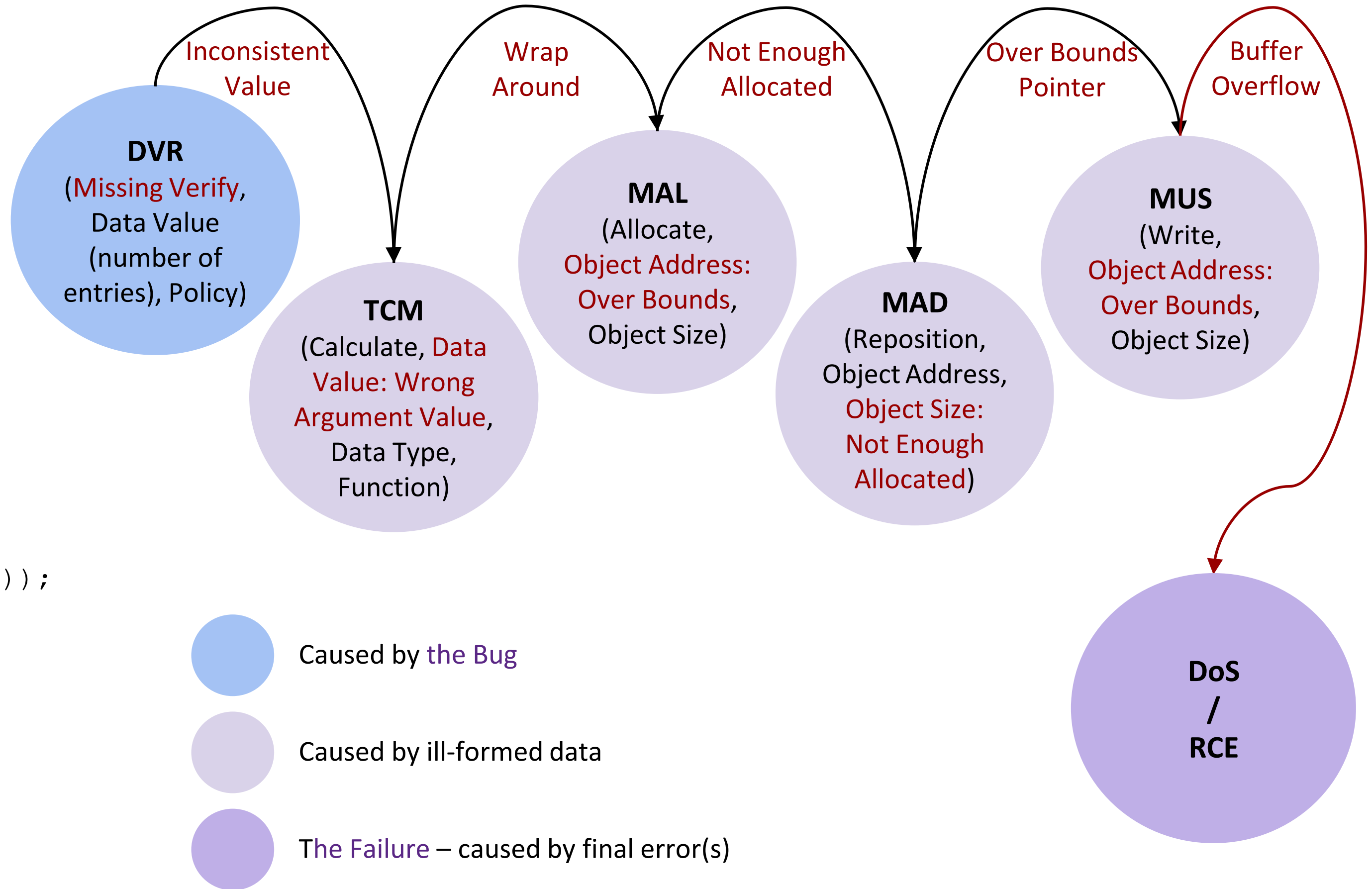
```
GF_Err co64_box_read(GF_Box* s, GF_BitStream* bs)
{
   u32 entries;
   GF_ChunkLargeOffsetBox* ptr = (GF_ChunkLargeOffsetBox*)s;
   ptr->nb_entries = gf_bs_read_u32(bs);

   ISOM_DECREASE_SIZE(ptr, 4)

   if (ptr->nb_entries > ptr->size / 8) {
     GF_LOG(GF_LOG_ERROR, GF_LOG_CONTAINER,
        ("[iso file] Invalid number of entries %d in co64\n",
          ptr->nb_entries));
     return GF_ISOM_INVALID_FILE;
   }

   ptr->offsets = (u64*)gf_malloc(ptr->nb_entries * sizeof(u64));
   if (ptr->offsets == NULL) return GF_OUT_OF_MEM;
     ptr->alloc_size = ptr->nb_entries;
     for (entries = 0; entries < ptr->nb_entries; entries++) {
       ptr->offsets[entries] = gf_bs_read_u64(bs);
   }
   return GF_OK;
}
```



"BadAlloc" – the Fix

CVE-2021-21834 An exploitable integer overflow vulnerability exists within the MPEG-4 decoding functionality of the GPAC Project on Advanced Content library v1.0.1. A specially crafted MPEG-4 input when decoding the atom for the “co64” FOURCC can cause an integer overflow due to unchecked arithmetic resulting in a heap-based buffer overflow that causes memory corruption. An attacker can convince a user to open a video to trigger this vulnerability.

```
GF_Err co64_box_read(GF_Box* s, GF_BitStream* bs)
{
  u32 entries;
  GF_ChunkLargeOffsetBox* ptr = (GF_ChunkLargeOffsetBox*)s;
  ptr->nb_entries = gf_bs_read_u32(bs);

  ISOM_DECREASE_SIZE(ptr, 4)

  if ((u64)ptr->nb_entries > ptr->size / 8
     ||   (u64)ptr->nb_entries > (u64)SIZE_MAX/sizeof(u64)){
    GF_LOG(GF_LOG_ERROR, GF_LOG_CONTAINER,
      ("[iso file] Invalid number of entries %d in co64\n",
        ptr->nb_entries));
    return GF_ISOM_INVALID_FILE;
  }

  ptr->offsets = (u64*)gf_malloc(ptr->nb_entries * sizeof(u64));
  if (ptr->offsets == NULL) return GF_OUT_OF_MEM;
    ptr->alloc_size = ptr->nb_entries;
    for (entries = 0; entries < ptr->nb_entries; entries++) {
      ptr->offsets[entries] = gf_bs_read_u64(bs);
  }
  return GF_OK;
}
```



BF Description of “BadAlloc”

NIST

Cause
Improper Operation: Missing

DVR Operation
Verify
(`(u64)ptr->nb_entries > (u64)SIZE_MAX/sizeof(u64)`)

Consequence
Improper Data Value: Inconsistent Value
(`> max 64-bit int`)

Attributes

Mechanism:
- Range

Source Code:
- Third Party (Library box_code_base.c)

Execution Space:
- Local

Data State:
- Stored (“number of entries” read from file)

Cause
Improper Data Value: Wrong Argument Value

TCM Operation
Calculate
(`ptr->nb_entries*sizeof(u64)`)

Consequence
Improper Data Value: Wrap Around

Attributes

Mechanism:
- Operator (Arithmetic: ‘*’)

Source Code:
- Third Party (Library box_code_base.c)

Data Value Kind:
- Numeric

Data Type Kind:
- Structured

Cause
Improper Data Value: Wrong Size Used (Size of memory to allocate)

MAL Operation
Allocate
(`gf_malloc()`)

Consequence
Improper Object Size: Not Enough Memory Allocated

Attributes

Mechanism:
- Explicit

Source Code:
- Third Party (Library box_code_base.c)

Execution Space:
- Userland

Ownership:
- Single

Location:
- Heap

Cause
Improper Object Size: Not Enough Memory Allocated

MAD Operation
Reposition

Consequence
Improper Data Value: Over Bounds Pointer

Attributes

Mechanism:
- Sequential

Source Code:
- Third Party (Library box_code_base.c)

Execution Space:
- Userland

Location:
- Heap

Cause
Improper Data Value: Over Bounds Pointer

MUS Operation
Write

Consequence
Memory Error: Buffer Overflow

Attributes

Mechanism:
- Sequential

Source Code:
- Third Party (Library box_code_base.c)

Execution Space:
- Userland

Span:
- Huge

Location:
- Heap

Denial of Service / Remote Code Execution

BF Hands On:
Incorrect Pointer Scaling

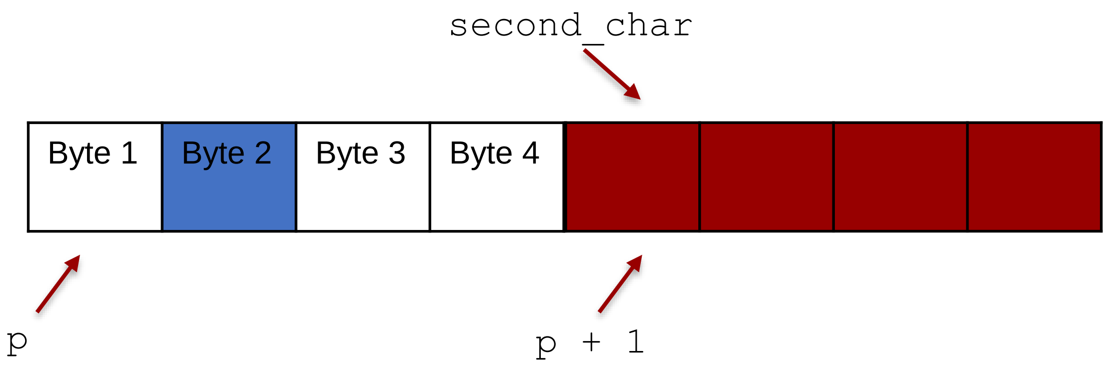
Incorrect Pointer Scaling (CWE-468, Ex. 1)

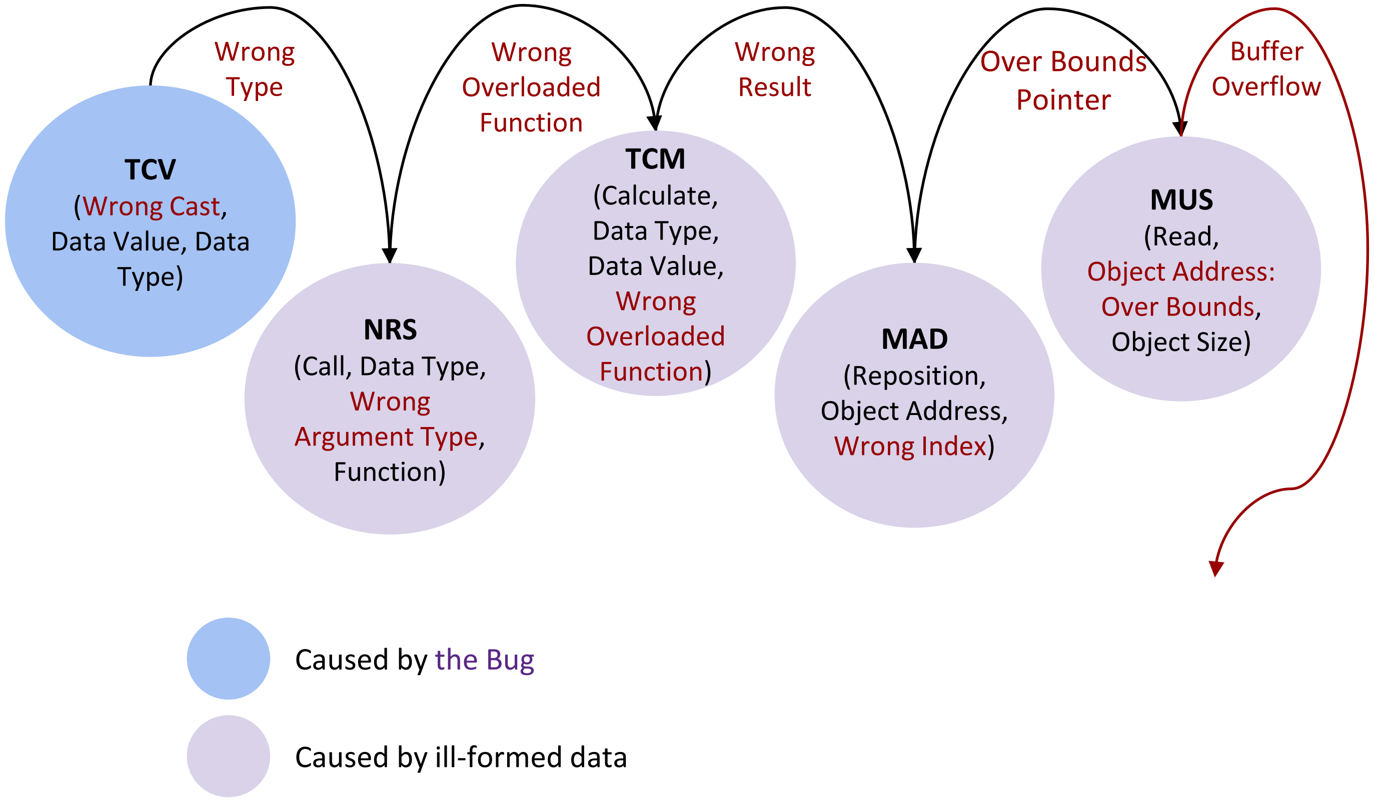
CWE-468, Example 1: This example attempts to calculate the position of the second byte of a pointer.

Example Language: **C**

moving 4 bytes

```
int *p = x;
char * second_char = (char *)(p + 1);
```





Incorrect Pointer Scaling – the Fix

CWE-468 Example 1

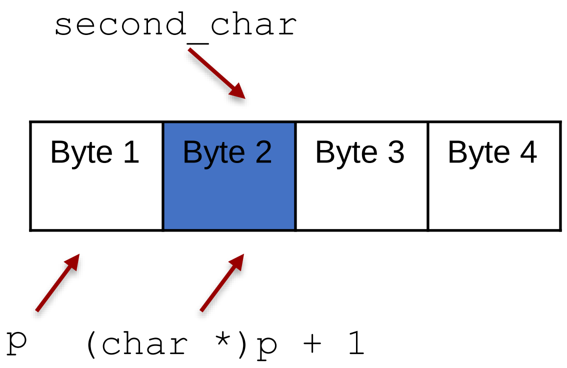
This example attempts to calculate the position of the second byte of a pointer.

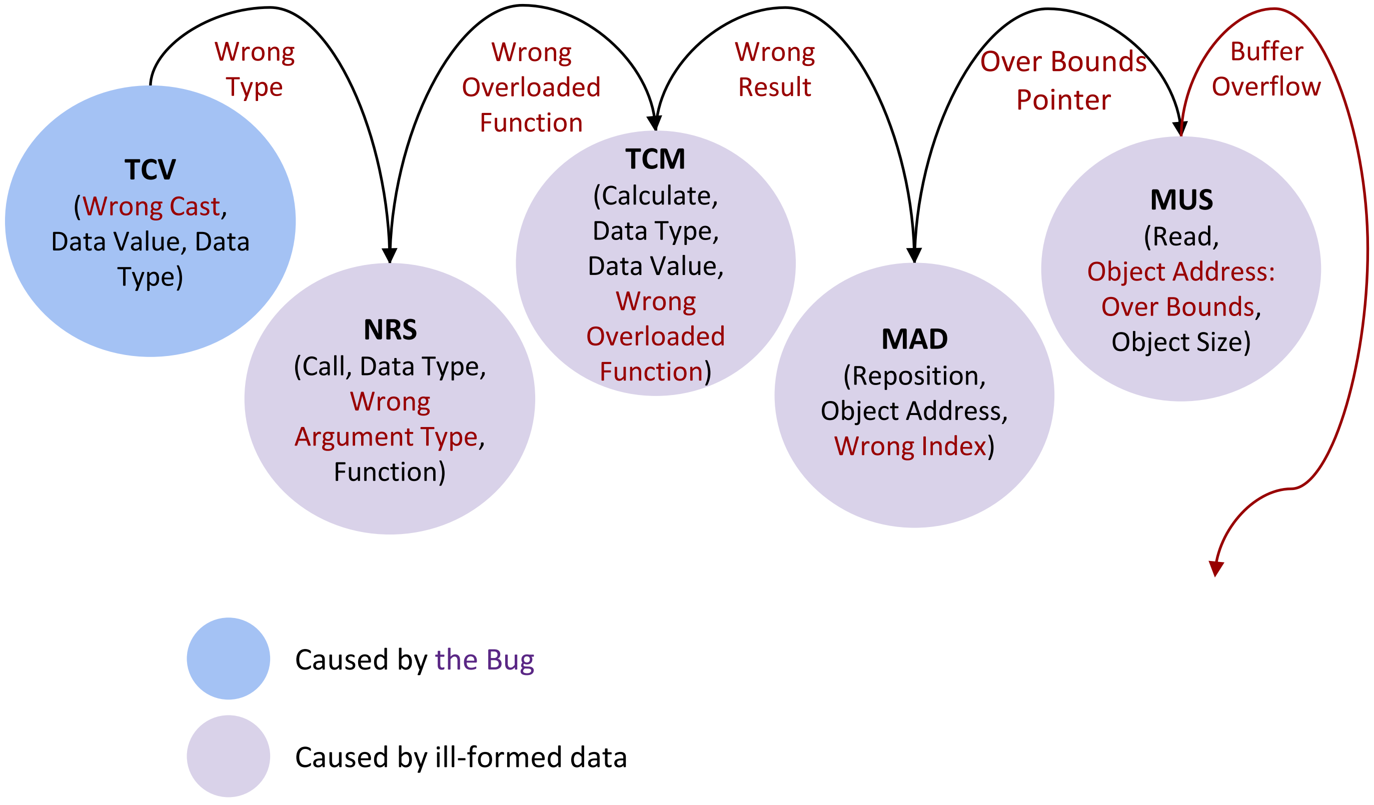
Example Language: **C**

```
int *p = x;
```

~~(char *)(p + 1)~~

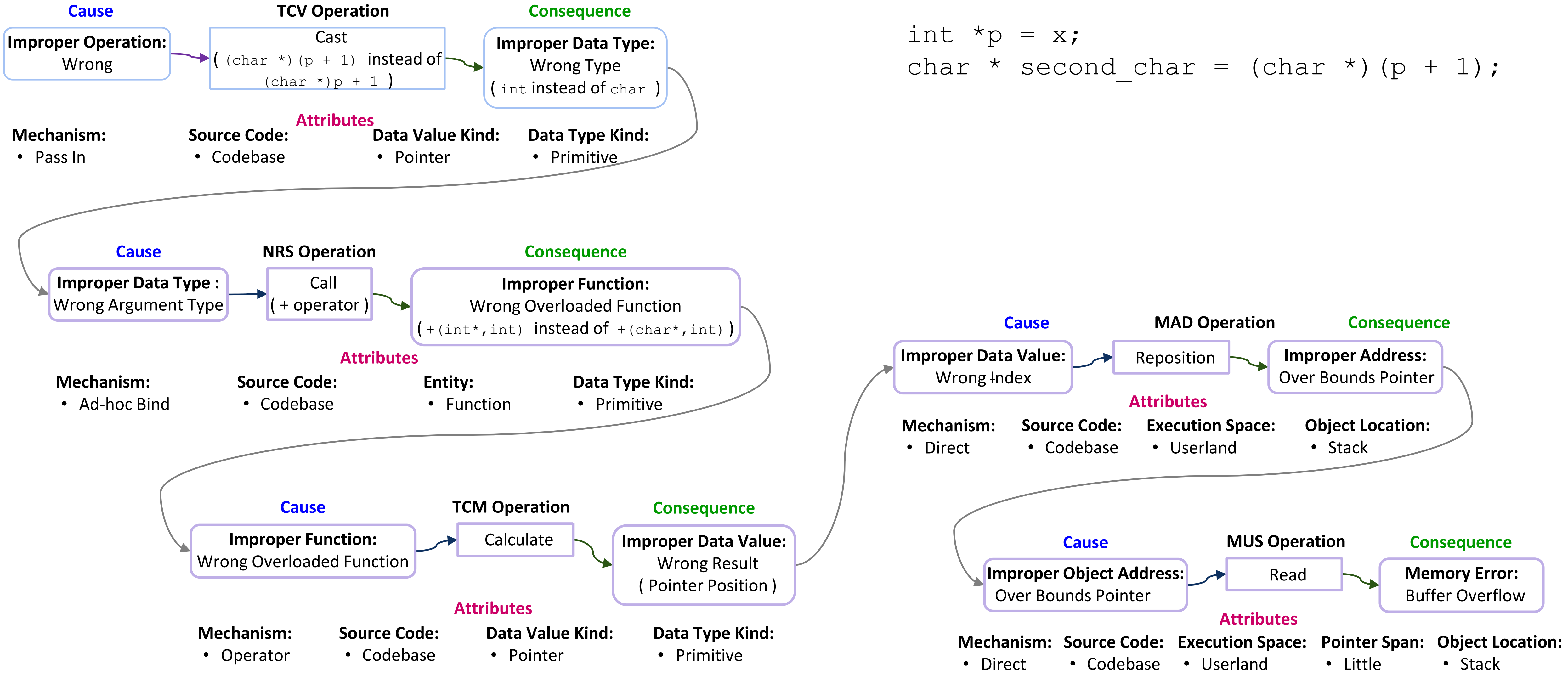
```
char * second_char = (char *)p + 1;
```





BF Description of CWE-468, Example 1

```
int *p = x;
char * second_char = (char *)(p + 1);
```



BF Hands On: 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.

https://nvd.nist.gov/vuln/detail/CVE-2014-0160

Weakness Enumeration

CWE-ID	CWE Name
CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer

Heartbleed (CVE-2014-0160)

NIST

[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.

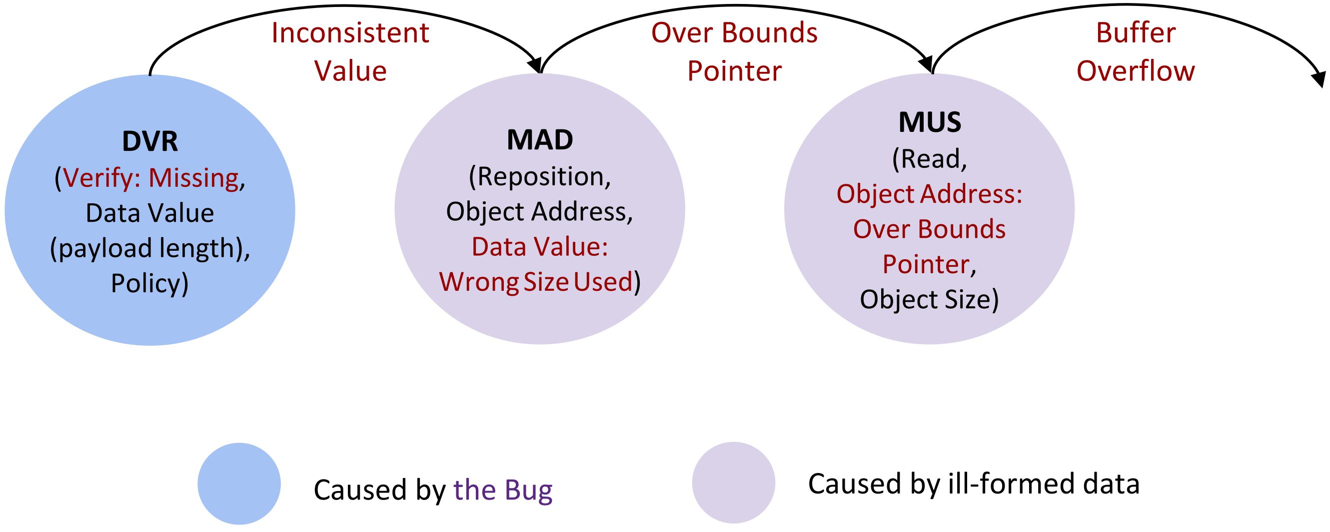
```
dtls1_process_heartbeat(SSL *s)
  {
  unsigned char *p = &s->s3->rrec.data[0], *pl;
  unsigned short hbtype;
  unsigned int payload;
  unsigned int padding = 16; /* Use minimum padding */

  /* 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;
    for (i=0; i<n; i++)
        *(char *) dst++ = *(char *) src++;
    return dst;
}
```

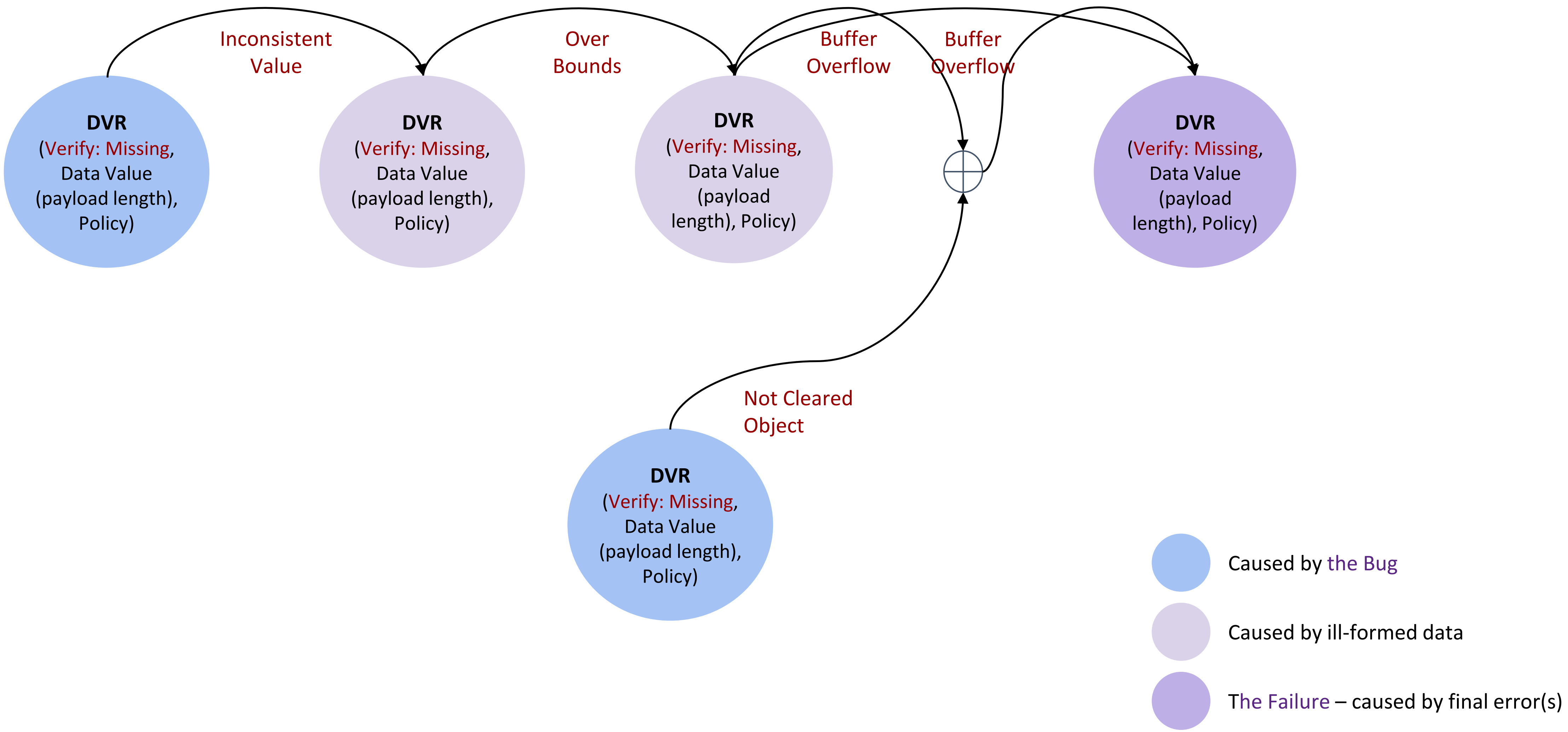
payload

bp

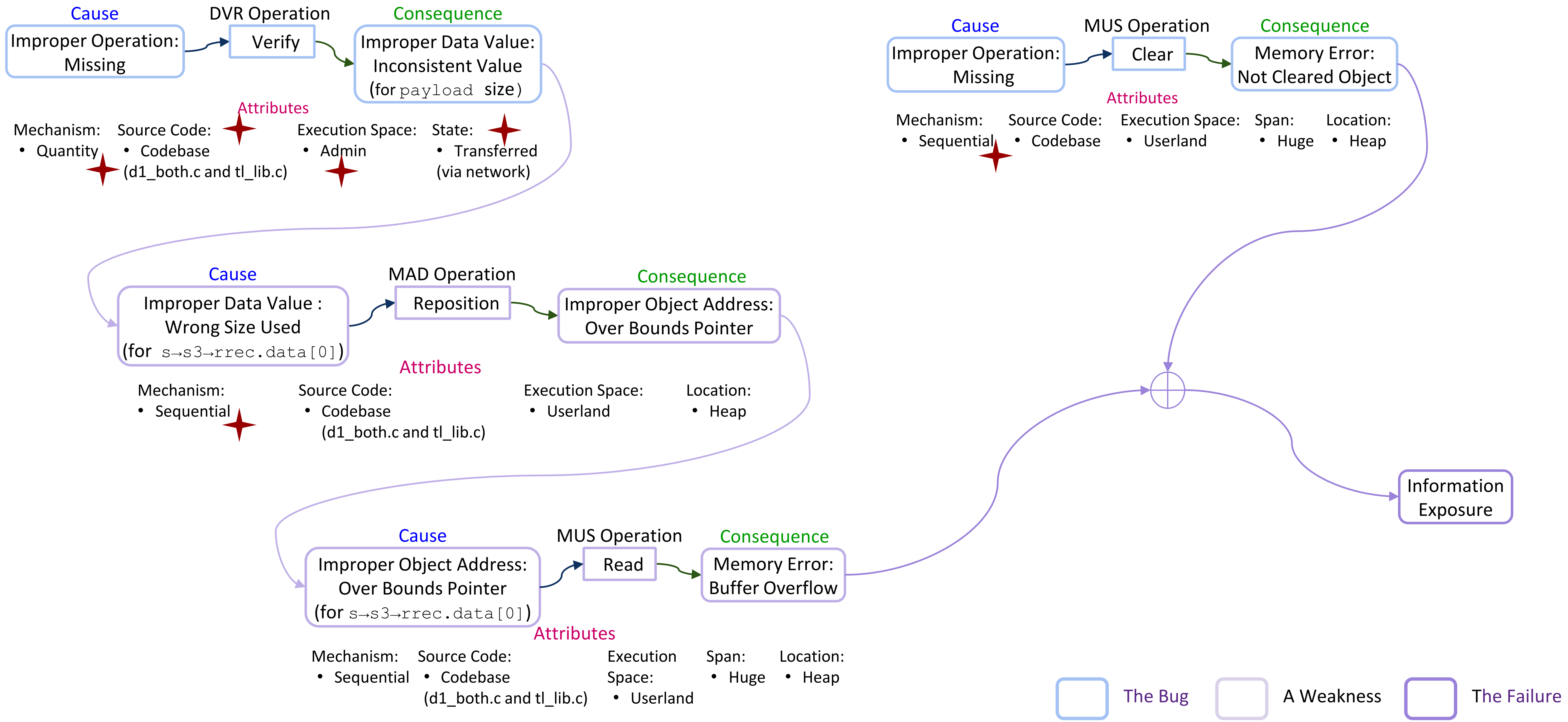
pl



Heartbleed (CVE-2014-0160)



BF Description of Heartbleed



BF Early Work – Heartbleed

- Heartbleed buffer overflow is:
  - caused by *Data Too Big*
  - because of *User Input not Checked Properly*
  - where there was a *Read that was After the End that was Far Outside*
  - of a buffer in the *Heap*
  - which may be exploited for *Information Exposure*

Towards a "Periodic Table" of Bugs

Irena Bojanova, Paul E. Black, Yaacov Yesha, Yan Wu

April 9, 2015 NIST, BGSU

Input not checked properly leads to too much data, where a huge number of bytes are read from the heap in a continuous reach after the array end, which may be exploited for exposure of information that had not been cleared.

Bojanova, I., Black, P., Yesha, Y. and Wu, Y. (2016), The Bugs Framework (BF): A Structured Approach to Express Bugs, IEEE International Conference on Software Quality, Reliability & Security (QRS 2016), Viena, AT,

BF Hands On:
NLP/ML/AI on Failures and Risks

BF Taxonomy – BF.xml

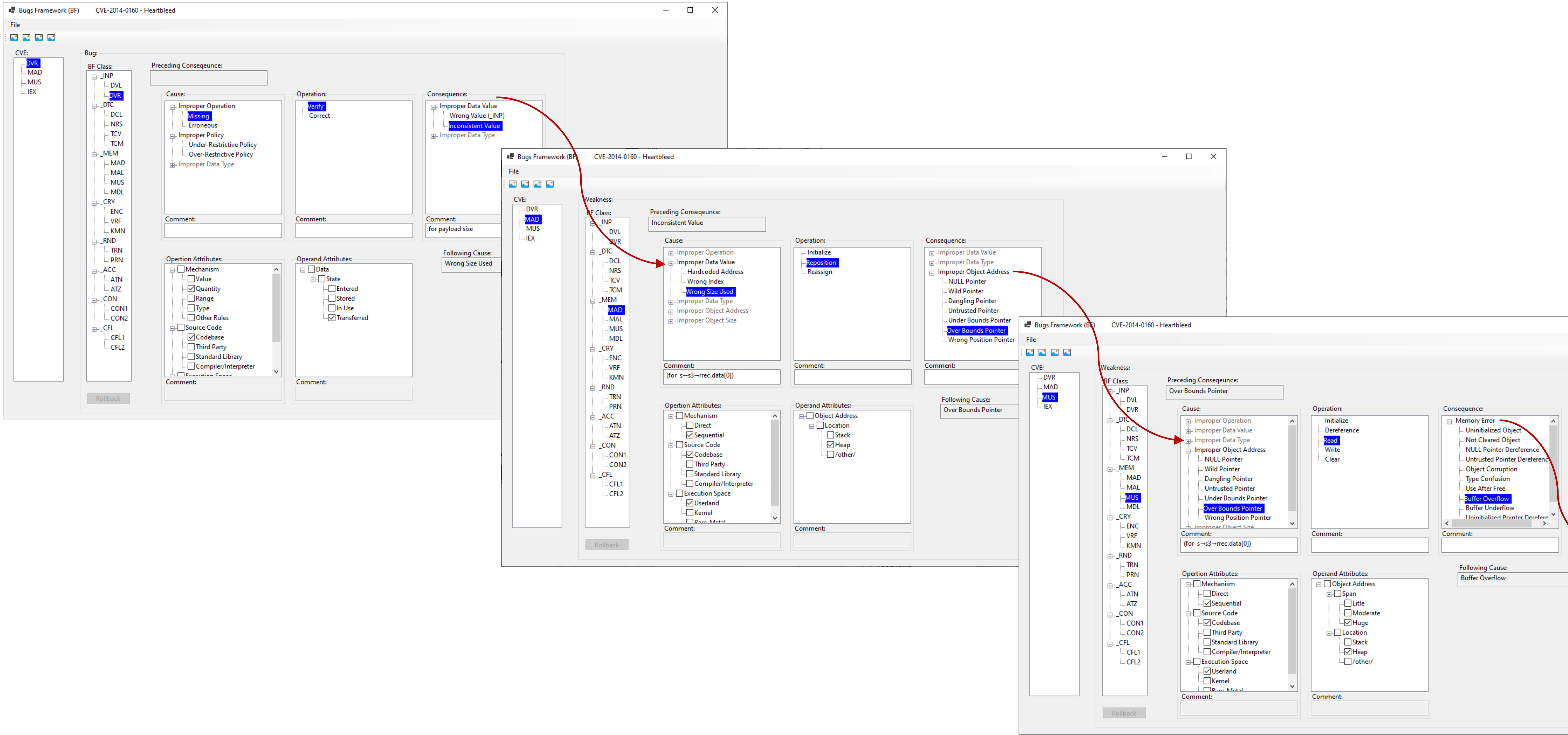
BF.xml*

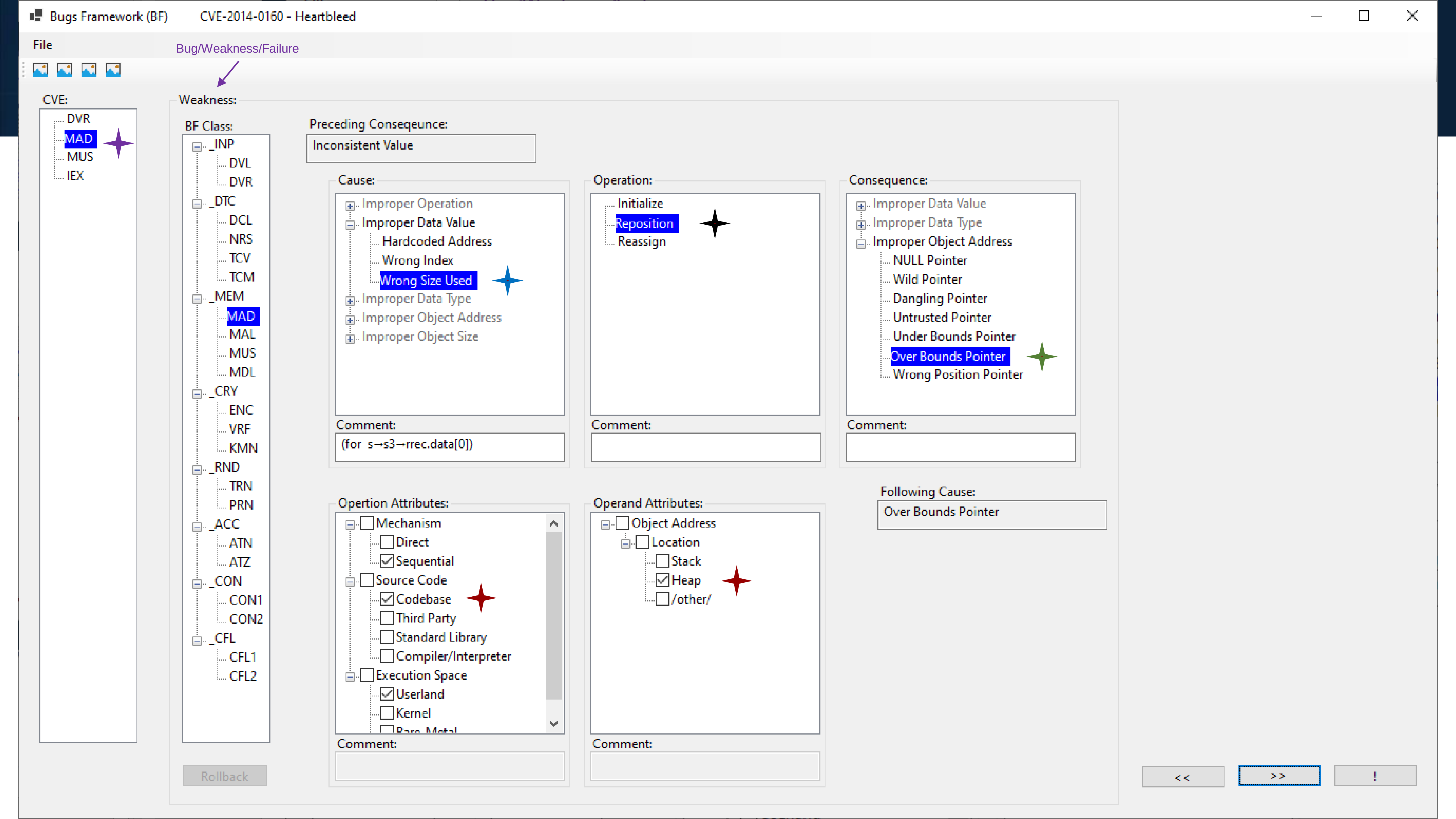
```xml
<!--@author Irena Bojanova(ivb)-->
<!--@date - 2/9/2022-->
<BF Name="Bugs Framework">
    <Cluster Name="_INP" Type="Bug/Weakness" Definition="Input/Output Ch">...</Cluster>
    <Cluster Name="_DTC" Type="Bug/Weakness" Definition="Data Type Bugs (incl. Convert and Compute Errors)">
        <Class Name="DCL" Title="Declaration Bugs" Definition="An object, a function, a type, or a namespace is declared or defined improperly.">
            <Operations>
                <Operation Name="Declare"/>
                <Operation Name="Define"/>
                <AttributeType Name="Mechanism">...</AttributeType>
                <AttributeType Name="Source Code">...</AttributeType>
                <AttributeType Name="Entity">...</AttributeType>
            </Operations>
            <Operands>
                <Operand Name="Data Type">
                    <AttributeType Name="Kind">...</AttributeType>
                </Operand>
            </Operands>
            <Causes>
                <BugCauseType Name="Improper Operation">
                    <Cause Name="Missing"/>
                    <Cause Name="Wrong"/>
                    <Cause Name="Erroneous"/>
                </BugCauseType>
                <BugCauseType Name="Improper Modifier">
                    <Cause Name="Missing Modifier"/>
                    <Cause Name="Wrong Modifier"/>
                </BugCauseType>
                <BugCauseType Name="Improper Scope">
                    <Cause Name="Anonymous Scope"/>
                    <Cause Name="Wrong Scope"/>
                </BugCauseType>
            </Causes>
            <Consequences>
                <WeaknessConsequenceType Name="Improper Data Type">
```

BF.xml*

```xml
<!--_DTC Cluster-->
<Definition Name="Declare">Specify name and type of an object; na
<Definition Name="Define">Specify data of an object; implementati
<Definition Name="Refer">Use a name in local or remote scopes of
<Definition Name="Call">Invoke a function implementation. The Typ
<Definition Name="Cast">Explicitly convert the value of an object
<Definition Name="Coerce">Implicitly (forced by the Type System)
<Definition Name="Calculate">Find the result of a numeric, pointe
<Definition Name="Evaluate">Find the result of a boolean conditio
<!--<Definition Name="Missing">The operation is absent.</Definiti
<Definition Name="Wrong">An inappropriate data type is specified;
<Definition Name="Erroneous (_DTC)">The Type System or a compute
<Definition Name="Missing Modifier">A required behavioral restric
<Definition Name="Wrong Modifier">A wrong behavioral restriction
<Definition Name="Anonymous Scope">The declaration is in an unnam
<Definition Name="Wrong Scope">The declaration should be in anoth
<Definition Name="Missing Qualifier">A namespace include is absen
<Definition Name="Wrong Qualifier">A wrong namespace is included,
<Definition Name="Object">A memory region used to store data.</De
<Definition Name="Data Value">A numeric, text, pointer/address, o
<Definition Name="Data Type">A set of allowed values and the oper
<Definition Name="Function">An organized block of code that when
```

CVE-2014-0160 - Heartbleed.bfcve




Bugs Framework (BF)
CVE-2014-0160 - Heartbleed
File
Bug/Weakness/Failure
CVE:
DVR
MAD
MUS
IEX
Weakness:
BF Class:
_INP
DVL
DVR
_DTC
DCL
NRS
TCV
TCM
_MEM
MAD
MAL
MUS
MDL
_CRY
ENC
VRF
KMN
_RND
TRN
PRN
_ACC
ATN
ATZ
_CON
CON1
CON2
_CFL
CFL1
CFL2
Rollback
Preceding Conseqeunce:
Inconsistent Value
Cause:
Improper Operation
Improper Data Value
Hardcoded Address
Wrong Index
Wrong Size Used
Improper Data Type
Improper Object Address
Improper Object Size
Comment:
(for s→s3→rrec.data[0])
Operation:
Initialize
Reposition
Reassign
Comment:
Consequence:
Improper Data Value
Improper Data Type
Improper Object Address
NULL Pointer
Wild Pointer
Dangling Pointer
Untrusted Pointer
Under Bounds Pointer
Over Bounds Pointer
Wrong Position Pointer
Comment:
Opertion Attributes:
Mechanism
Direct
Sequential
Source Code
Codebase
Third Party
Standard Library
Compiler/Interpreter
Execution Space
Userland
Kernel
Comment:
Operand Attributes:
Object Address
Location
Stack
Heap
/other/
Comment:
Following Cause:
Over Bounds Pointer
<<
>>
!

CVE-2014-0160 - Heartbleed.bfcve

NIST

CVE-2014-016...rtbleed.bfcve


```xml
<?xml version="1.0" encoding="utf-8"?>
<CVE Name="1 CVE-2014-0160">
    <Bug Type="_INP" Class="DVR">
        <Cause Type="Improper Operation">Missing</Cause>
        <Operation>Verify</Operation>
        <Consequence Comment="for payload size" Type="Improper Data Value">Inconsistent Value</Consequence>
        <Attributes>...</Attributes>
    </Bug>
    <Weakness Type="_MEM" Class="MAD">
        <Cause Comment="(for  s→s3→rrec.data[0])" Type="Improper Data Value">Wrong Size Used</Cause>
        <Operation>Reposition</Operation>
        <Consequence Type="Improper Object Address">Over Bounds Pointer</Consequence>
        <Attributes>
            <Operation>
                <Attribute Type="Mechanism">Sequential</Attribute>
                <Attribute Comment="d1_both.c and tl_lib.c" Type="Source Code">Codebase</Attribute>
                <Attribute Type="Execution Space">Userland</Attribute>
            </Operation>
            <Operand Name="Object Address">
                <Attribute Type="Location">Heap</Attribute>
            </Operand>
        </Attributes>
    </Weakness>
    <Weakness Type="_MEM" Class="MUS">
        <Cause Comment="(for  s→s3→rrec.data[0])" Type="Improper Object Address">Over Bounds Pointer</Cause>
        <Operation>Read</Operation>
        <Consequence Type="Memory Error">Buffer Overflow</Consequence>
        <Attributes>...</Attributes>
    </Weakness>
    <Failure Type="_FLR" Class="IEX">
        <Cause Type="Memory Error">Buffer Overflow</Cause>
```

CVE-2021-21834 - Bad Alloc.bfcve

CVE-2021-218...d Alloc.bfcve


```
<?xml version="1.0" encoding="utf-8"?>
<CVE Name="CVE-2021-21834">
    <Bug Type="_INP" Class="DVR">
        <Cause Type="Improper Operation">Missing</Cause>
        <Operation Comment="(u64)ptr-&gt;nb_entries &gt; (u64)SIZE_MAX/sizeof(u64)">Verify</Operation>
        <Consequence Comment="&gt; max 64-bit int" Type="Improper Data Value">Inconsistent Value</Consequence>
        <Attributes>...</Attributes>
    </Bug>
    <Weakness Type="_DTC" Class="TCM">
        <Cause Type="Improper Data Value">Wrong Argument Value
        <Operation Comment="ptr-&gt;nb_entries*sizeof(u64)">Cal
        <Consequence Type="Improper Data Value">Wrap Around</C
        <Attributes>
            <Operation>
                <Attribute Comment="Arithmetic: '*'" Type="Mech
                <Attribute Comment="Library box_code_base.c" Ty
            </Operation>
            <Operand Name="Data Value">
                <Attribute Type="Data Value Kind">Numeric</Att
            </Operand>
            <Operand Name="Data Type">
                <Attribute Type="Data Type Kind">Structure</Att
            </Operand>
        </Attributes>
    </Weakness>
    <Weakness Type="_MEM" Class="MAL">
        <Cause Comment="Size of memory to allocate" Type="Impro
        <Operation Comment="gf_malloc()">Allocate</Operation>
        <Consequence Type="Improper Object Size">Not Enough Me
        <Attributes>...</Attributes>
    </Weakness>
```

```
    <Weakness Type="_MEM" Class="MAD">
        <Cause Type="Improper Object Size">Not Enough Memory Allocated</Cause>
        <Operation>Reposition</Operation>
        <Consequence Type="Improper Object Address">Over Bounds Pointer</Consequen
        <Attributes>...</Attributes>
    </Weakness>
    <Weakness Type="_MEM" Class="MUS">
        <Cause Type="Improper Object Address">Over Bounds Pointer</Cause>
        <Operation>Write</Operation>
        <Consequence Type="Memory Error">Buffer Overflow</Consequence>
        <Attributes>
            <Operation>
                <Attribute Type="Mechanism">Sequential</Attribute>
                <Attribute Comment="Library box_code_base.c" Type="Source Code">Th
                <Attribute Type="Execution Space">Userland</Attribute>
            </Operation>
            <Operand Name="Object Address">
                <Attribute Type="Span">Huge</Attribute>
                <Attribute Type="Location">Heap</Attribute>
            </Operand>
        </Attributes>
    </Weakness>
    <Failure Type="_FLR" Class="DOS">
        <Cause Type="Memory Error">Buffer Overflow</Cause>
```

CWE mapped to BF – BFCWE.xml

BFCWE.xml*

```xml
<!--@author Irena Bojanova(ivb)-->
<!--@date - 07/09/2021-->
<BFCWE>
    <Cluster Name="_ALL">
        <showClassCWEs>...</showClassCWEs>
    </Cluster>
    <Cluster Name="_INP">
        <!--fig 1-->
        <showClassCWEs>
            <ClassOperation name="DVL Validate">
                <CWE>41</CWE>
                <CWE>42</CWE>
                <CWE>43</CWE>
                <CWE>44</CWE>
                ...
            </ClassOperation>
            <ClassOperation name="DVL Sanitize">...</ClassOperation>
            <ClassOperation name="DVR Check">...</ClassOperation>
            <ClassOperation name="DVR Verify">...</ClassOperation>
            <ClassOperation name="DVL Validate an">...</ClassOperation>
        </showClassCWEs>
        <showConsequenceCWEs>
            <Consequence name="Query Injection">...</Consequence>
            <Consequence name="Command Injection">
                <CWE>77</CWE>
                <CWE>78</CWE>
                <CWE>114</CWE>
                <CWE>624</CWE>
            </Consequence>
            <Consequence name="Source Code Inj">...</Consequence>
            <Consequence name="Parameter Injec">...</Consequence>
            <Consequence name="File Injection">...</Consequence>
```

```xml
<classStyles>
    <Caption n="CWEs by DVL and/or DVR operation:" u="sng" x="10
    <ClassOperation n="DVL Validate" c="0099FF"/>
    <ClassOperation n="DVL Sanitize" c="339966"/>
    <ClassOperation n="DVR Verify" c="FF3399"/>
    <ClassOperation n="DVL Validate and DVR Verify" c="9966FF"/>
</classStyles>
<consequenceStyles>
    <Caption n="CWEs by DVL Injection Error:" u="sng" x="1076556
    <Consequence n="Query Injection" c="99FF66"/>
    <Consequence n="Command Injection" c="6699FF"/>
    <Consequence n="Source Code Injection" c="FF9966"/>
    <Consequence n="Parameter Injection" c="66FFCC"/>
    <Consequence n="File Injection" c="CC99FF"/>
    <Caption n="CWEs by DVL or DVR Wrong Data:" u="sng" x="17278
    <Consequence n="DVL Invalid Data" c="FF0000"/>
    <Consequence n="DVR Wrong Value, Inconsistent Value, and Wro
    <OnlyCause n="No consequence" c="C8C8DA" fill="F3F3F3"/>
</consequenceStyles>
```

BF in ML & AI

Machine readable formats of:

- BF taxonomy
- BF vulnerability descriptions
- CWEs to BF mappings

→ Query and analyze sets of BF descriptions
→ NLP, ML, and AI projects related to software bugs/weaknesses, failures and risks.

BF in ML & AI

- JHU APL – Automated Vulnerability Testing via Executable Attack Graphs:
  - Chain vulnerabilities via logical directed graphs
  - Determine most mitigation "paths" with least changes
  - Detect user behavior prior to malicious effect

The lack of formal, precise descriptions of known vulnerabilities and software weaknesses in the current National Vulnerability Database (NVD) has become an increasingly limiting factor in vulnerability research, mitigation research, and expression of software systems in low level modeling form.

We were thrilled to hear that a researcher at NIST was undertaking the needed improvement to make such descriptions more formal and machine-readable. Such an endeavor will greatly enhance the ability of cyber researchers to explore more complex attacks via computational methods. This will be a huge boost to the U.S.'s ability to defend its networks, military systems, and critical infrastructure, and will lead the way to better mitigation designs, improved software development practices, and automated cyber testing capabilities.

- RIT Secure and Trustworthy Cyberspace (SaTC):

The NIST Bugs Framework (BF) has made significant advances in creating first-of-its-kind classification of software weaknesses that has enabled the community to express vulnerabilities using a precise description.

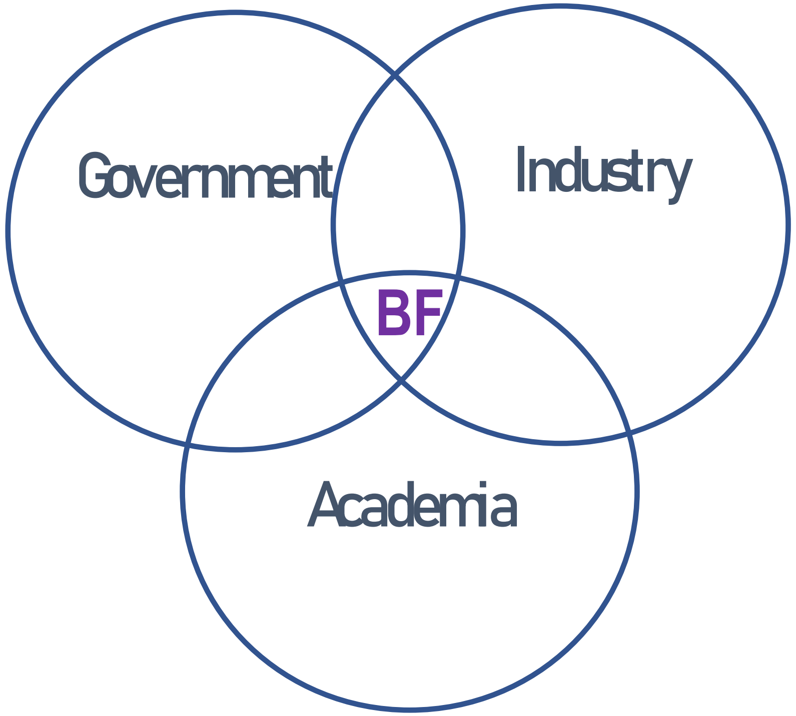
allowing us to obtain a fine-grained understanding of security bugs and their root causes. Additionally, the taxonomies and root causes in each bug class will provide us valuable data to guide and enhance our static program analysis techniques and achieve higher accuracy.

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

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