The NIST Bugs Framework (BF)

Data Type Bugs Taxonomy: Integer Overflow, Juggling, and Pointer Arithmetics in Spotlight

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

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

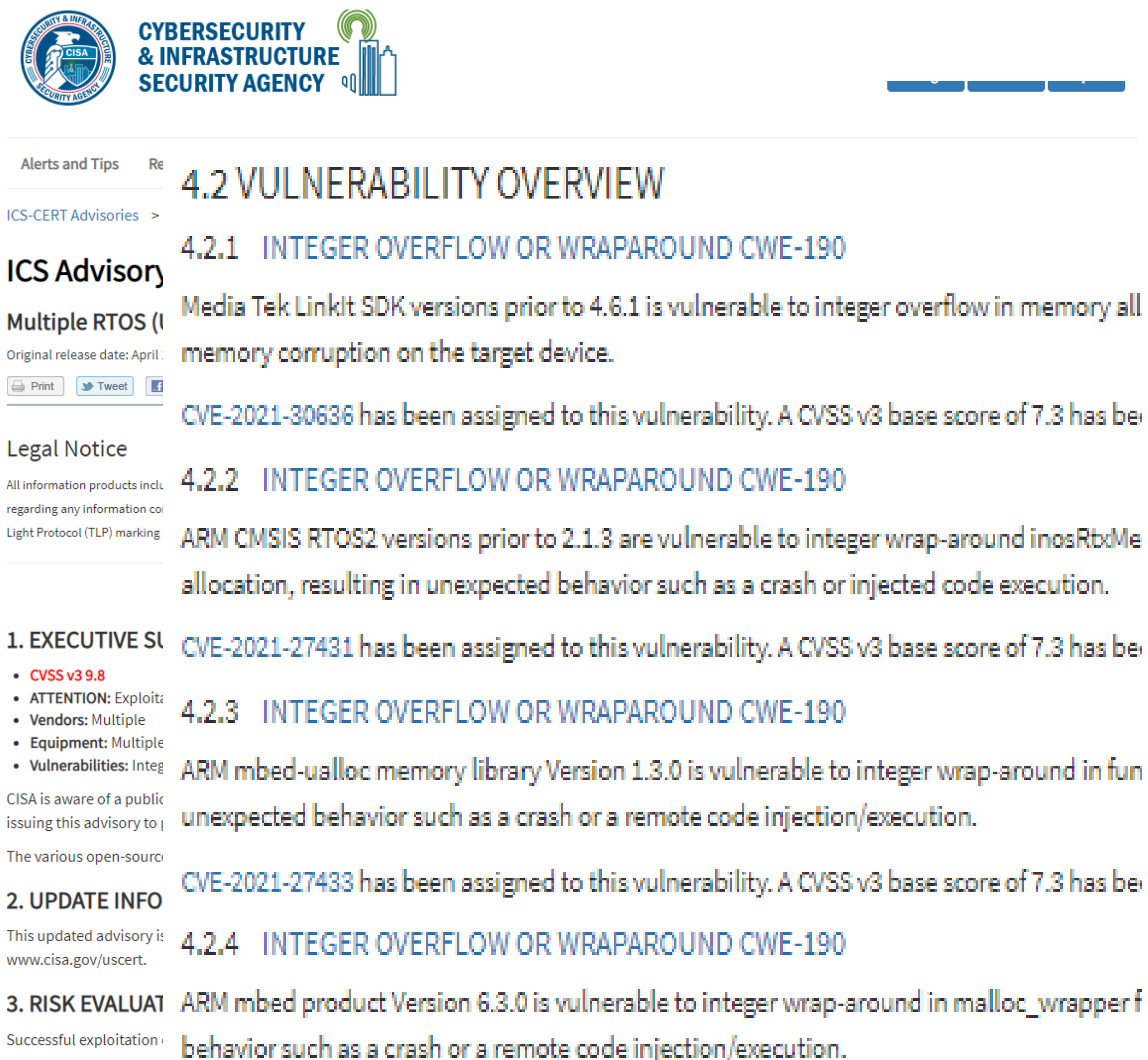
Irena Bojanova
Carlos Galhardo
Sara Moshtari

Agenda

- Introduction:
  - “Bad Alloc” Pattern
  - Terminology:
    - ✓ Bug
    - ✓ Weakness
    - ✓ Vulnerability
    - ✓ Failure
- Existing Repositories:
  - CWE
  - CVE
  - NVD
  - KEV
- The Bugs Framework (BF)
  - Goals
  - Features
- BF Taxonomy – of Data Type Bugs
- Validation towards CWE
- BF Hands On:
  - Bad Allocation Chain
  - Incorrect Pointer Scaling Chain
- Potential Impacts

Introduction

“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

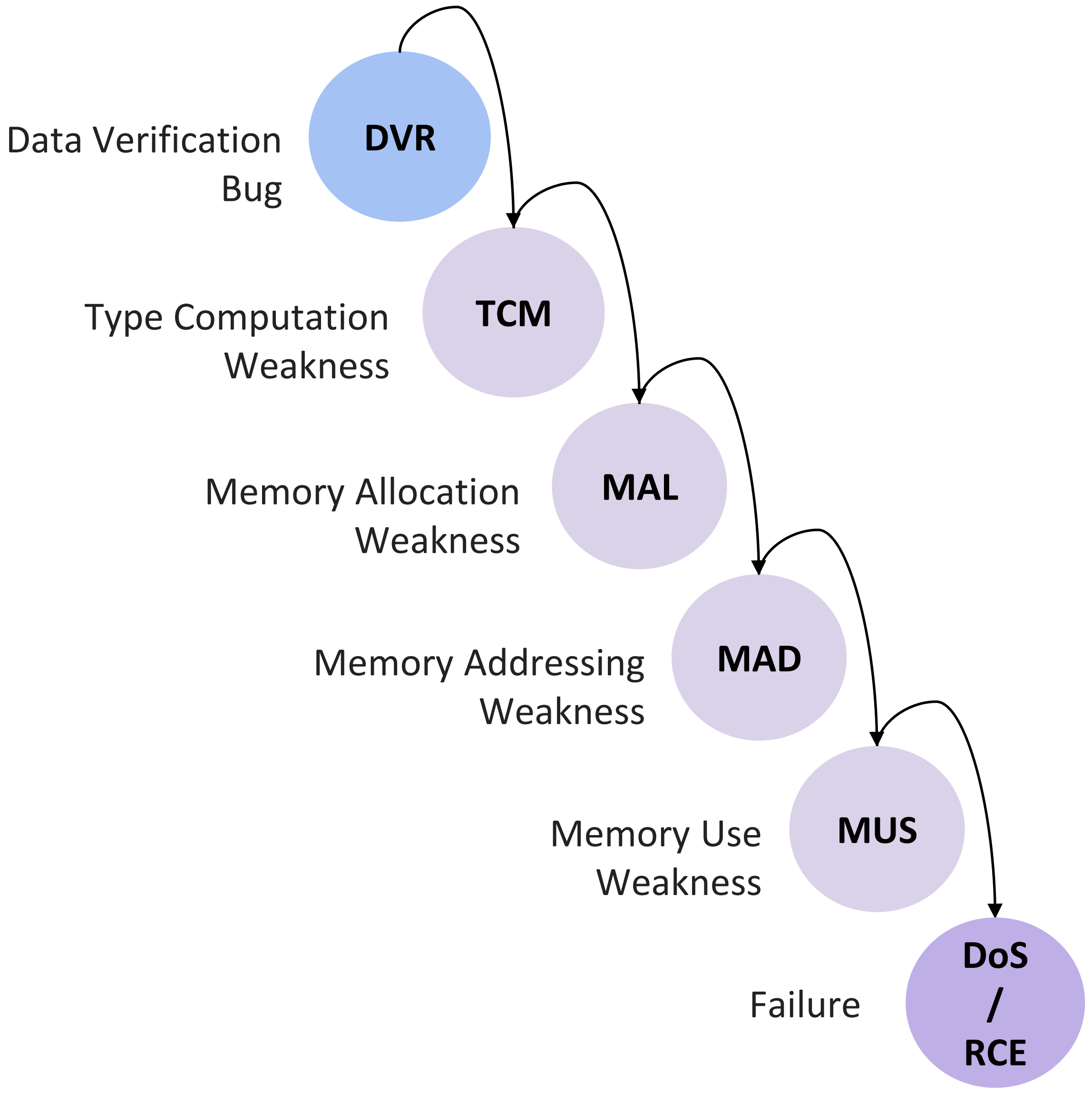
4.2.3 INTEGER OVERFLOW OR WRAPAROUND CWE-190

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.


Data Verification Bug
DVR
Type Computation Weakness
TCM
Memory Allocation Weakness
MAL
Memory Addressing Weakness
MAD
Memory Use Weakness
MUS
Failure
DoS / RCE

Terminology

- 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

Existing Repositories

Commonly Used Repositories

● Weaknesses: CWE – Common Weakness Enumeration	https://cwe.mitre.org/
● Vulnerabilities: CVE – Common Vulnerabilities and Exposures →over 18 000 documented in 2020	https://cve.mitre.org/
● 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 → links also to KEV	https://nvd.nist.gov/

Repository Problems

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

- Example:

  CWEs coverage of buffer overflow by:

  ✓ Read/ Write

  ✓ Over/ Under

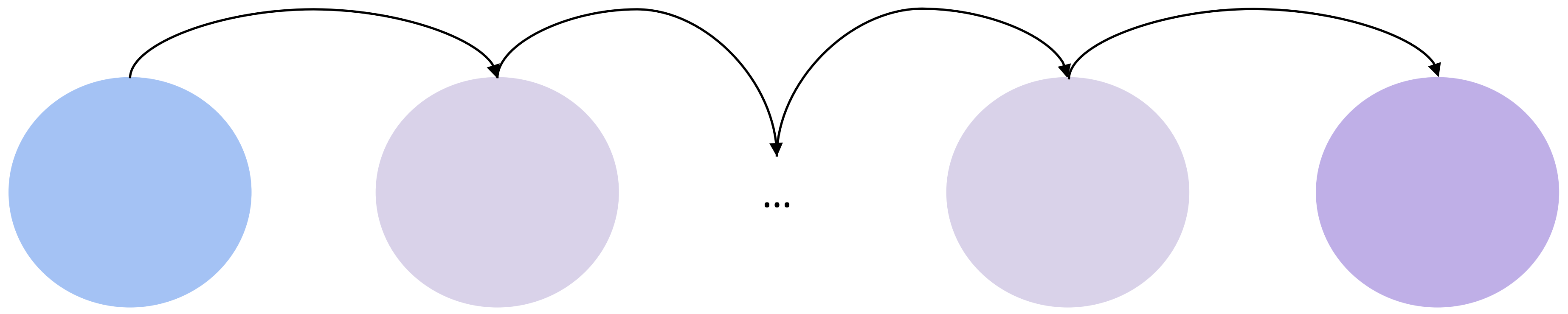
  ✓ Stack/ Heap

	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

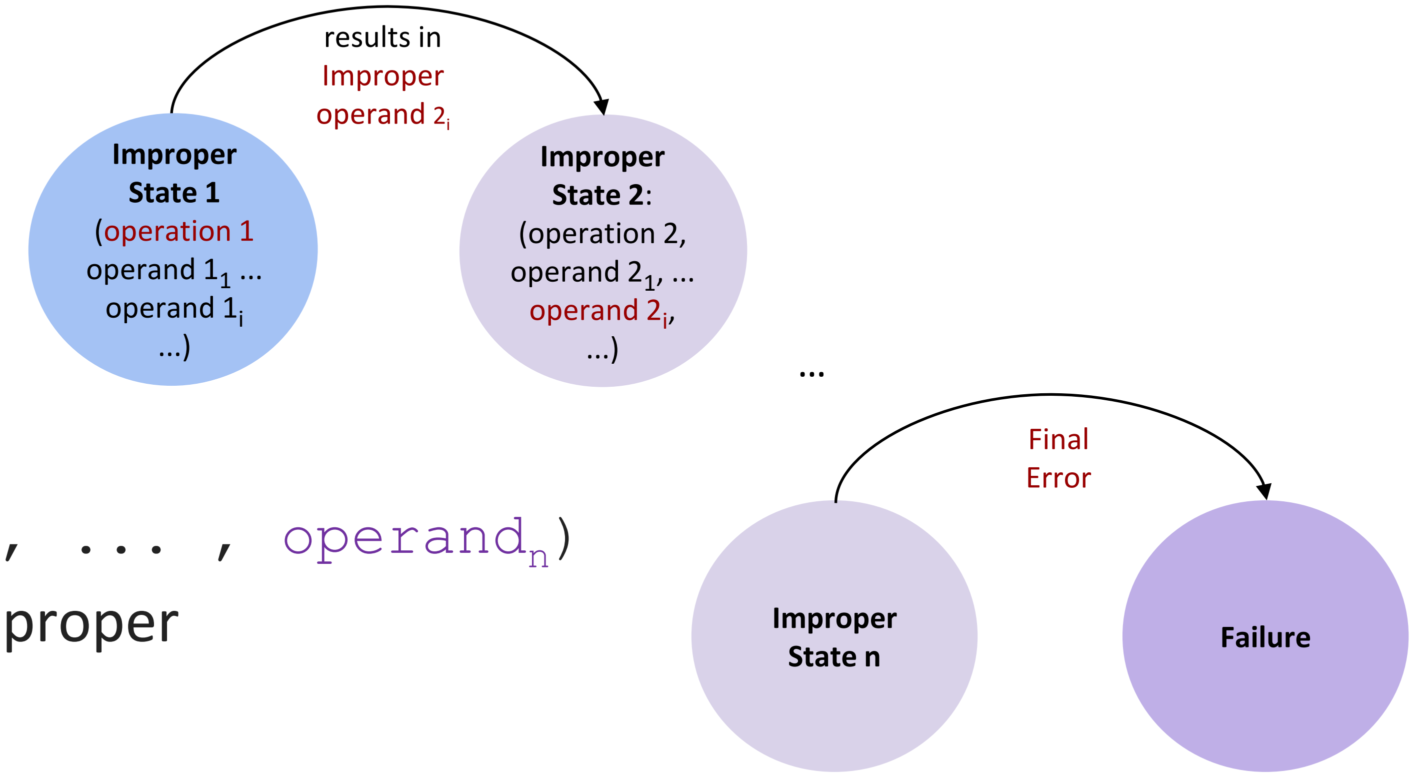
- Improper State –

  a tuple (`operation`, `operand`$_1$, ... , `operand`$_n$)

  , where at least one element is improper

- Transition –

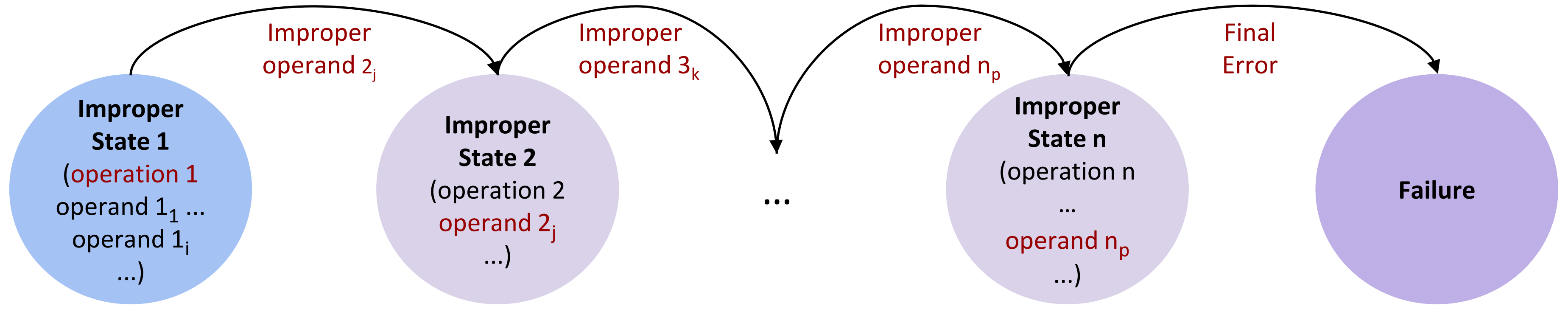
  the result of the `operation` over the `operands`

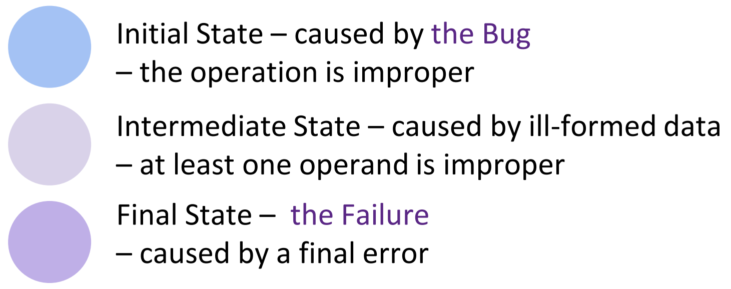

results in
Improper operand 2_i
Improper State 1 (operation 1 operand 1_1 ... operand 1_i ...)
Improper State 2: (operation 2, operand 2_1, ... operand 2_i, ...)
...
Final Error
Improper State n
Failure

🔵	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 – Chaining Weaknesses

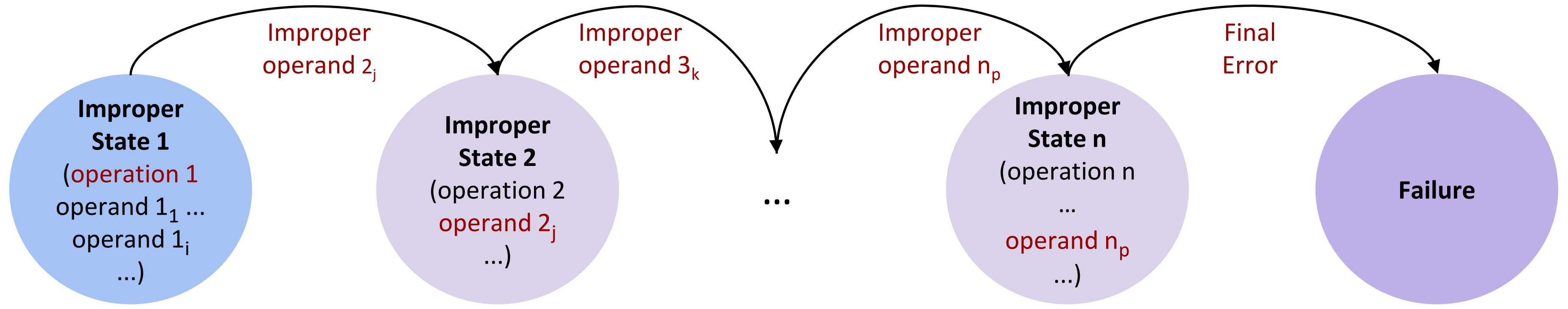
- BF describes a vulnerability as:
  - A chain of improper states and their transitions
  - States change until a failure is reached


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 – Backtracking

- 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

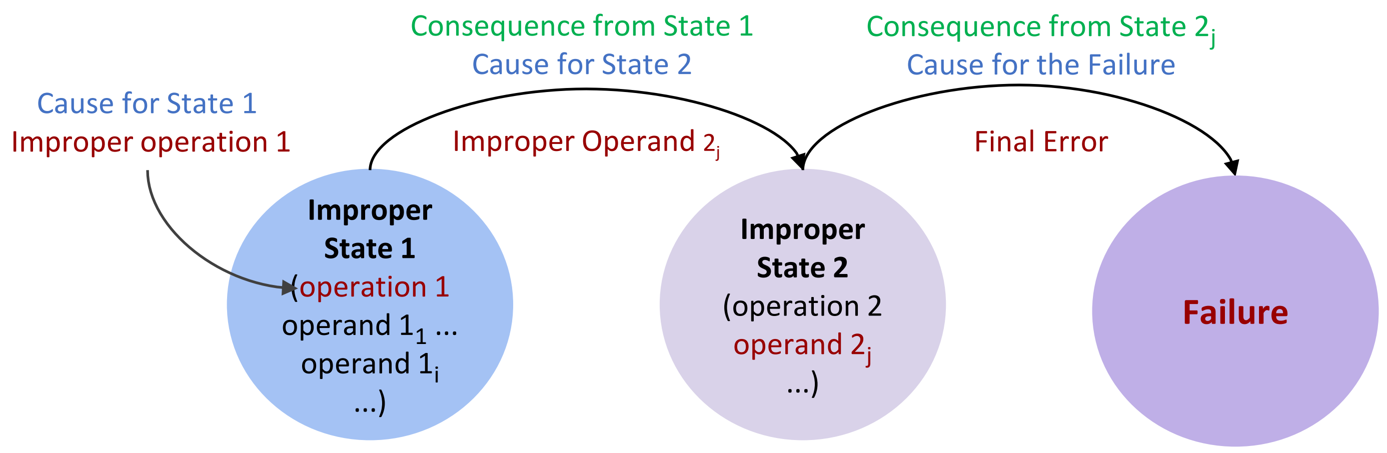
● Initial State – caused by the Bug – the operation is improper

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

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

BF Features – 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

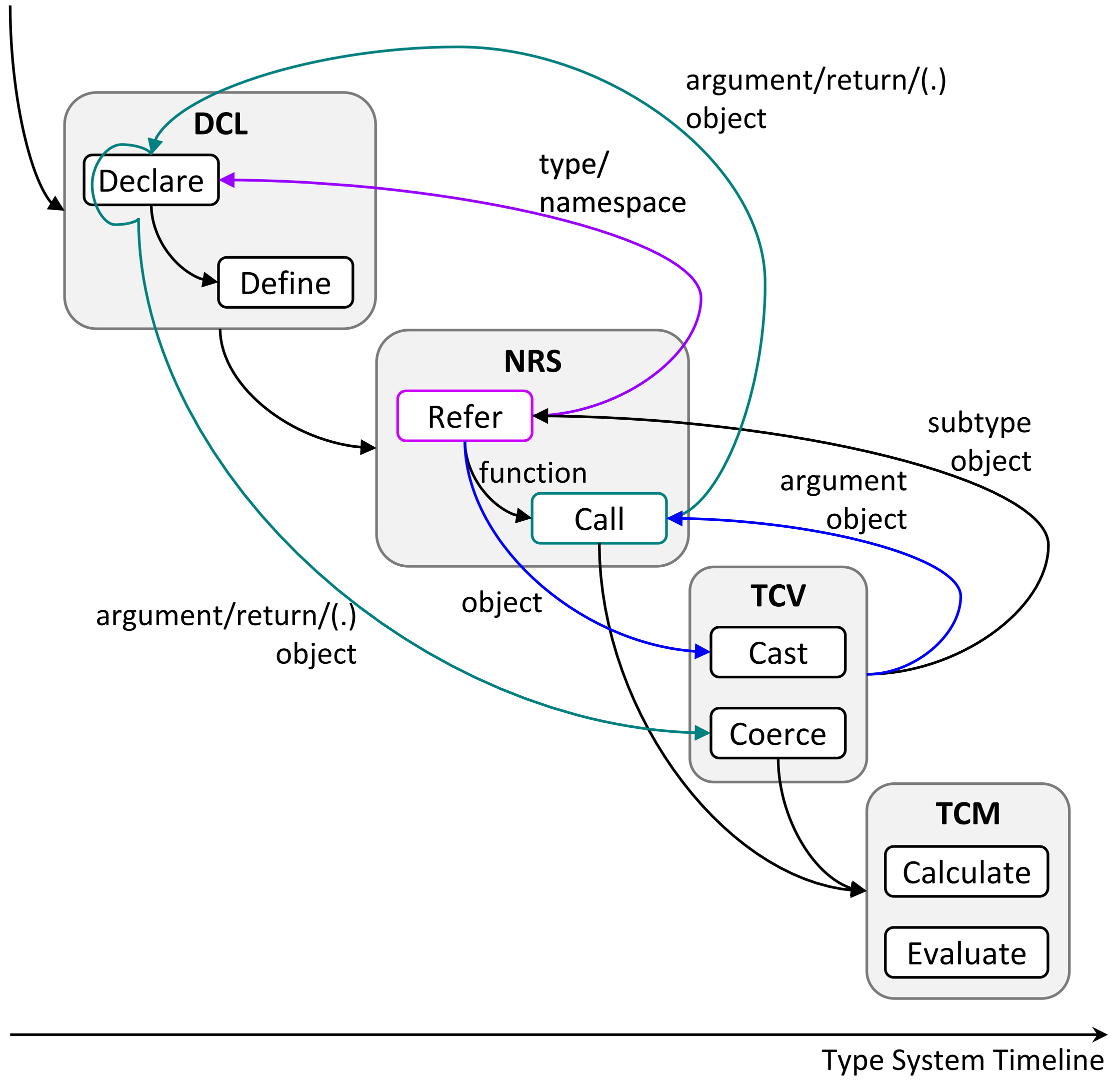
- BF vulnerability description –
  - chain of BF classes instances
  - consequence–cause transitions.

BF Taxonomy – Data Type Bugs

BF Data Type Bugs Model

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

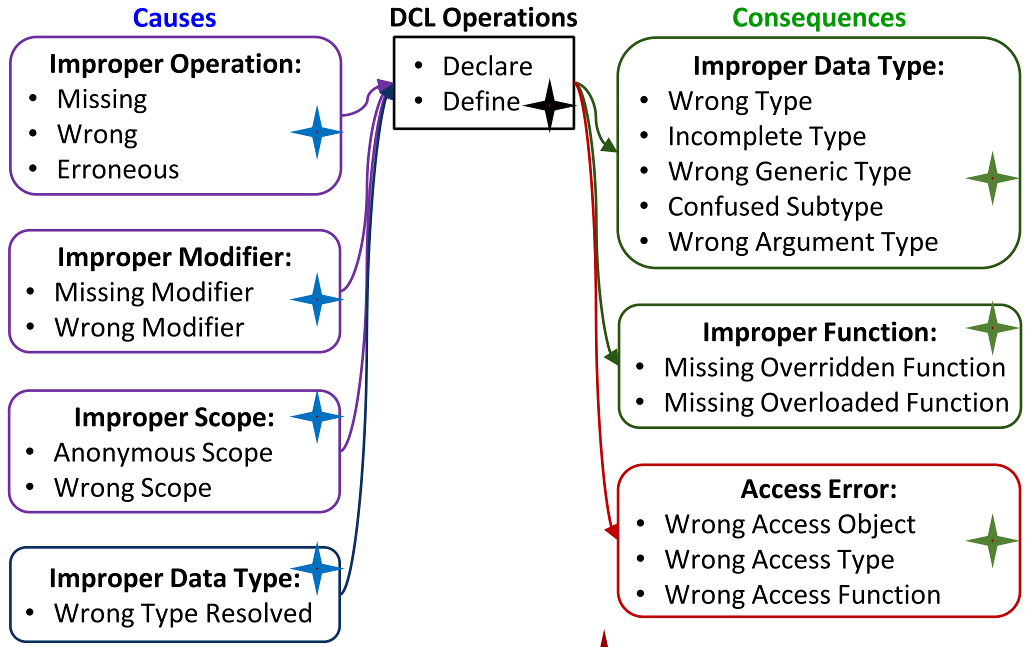
- Data Type operations flow



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

BF Data Type Bugs Classes: DCL & NRS

Declaration Bugs (DCL) – *An object, a function, a data type, or a namespace is declared or defined improperly.*



Causes

Improper Operation:
- Missing
- Wrong
- Erroneous

Improper Modifier:
- Missing Modifier
- Wrong Modifier

Improper Scope:
- Anonymous Scope
- Wrong Scope

Improper Data Type:
- Wrong Type Resolved

DCL Operations
- Declare
- Define

Consequences

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

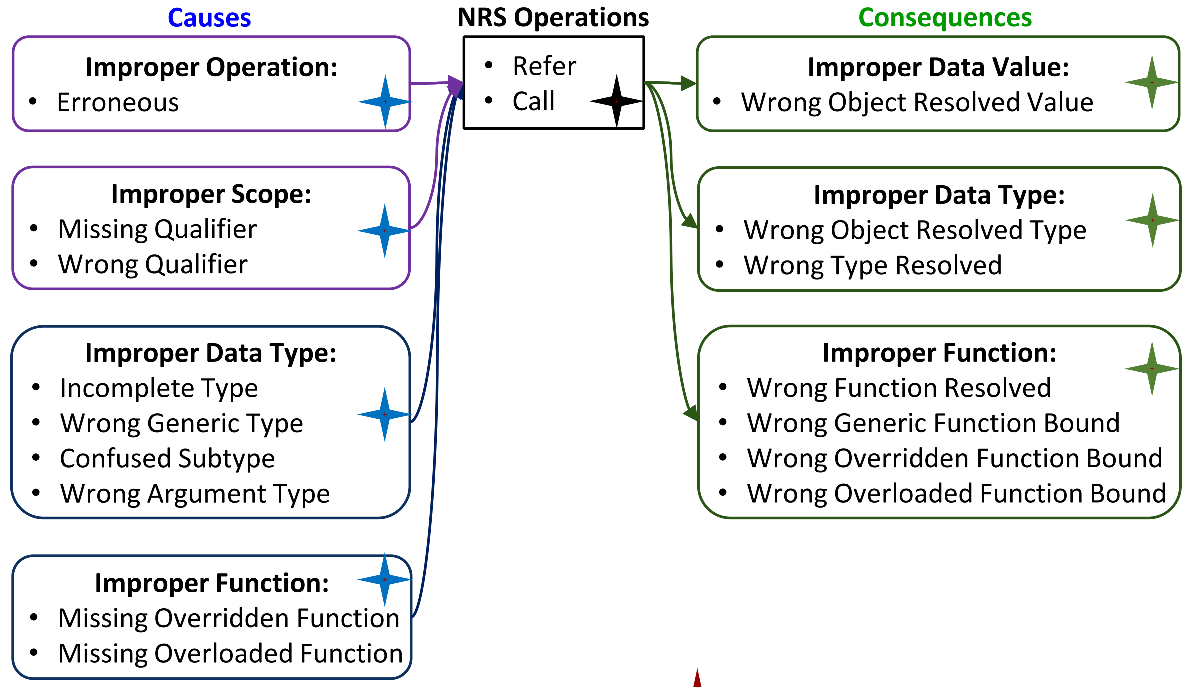
Improper Function:
- Missing Overridden Function
- Missing Overloaded Function

Access Error:
- Wrong Access Object
- Wrong Access Type
- Wrong Access Function

Attributes

Mechanism:	Source Code:	Entity:	Data Type Kind:
• Simple	• Codebase	• Object	• Primitive
• Generics	• Third Party	• Function	• Structured
• Overriding	• Standard Library	• Data Type	
• Overloading	• Compiler/Interpreter	• Namespace	

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	• Codebase	• Object	• Primitive
• Bind	• Third Party	• Function	• Structured
• Early Bind	• Standard Library	• Data Type	
• Late Bind	• Compiler/Interpreter	• Namespace	
• Ad-hoc Bind			

BF Data Type Bugs Classes: TCV & TCM

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

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

Other BF Classes – DVR, MAD

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

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

Attributes

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

Other BF Classes – MAL, MUS

NIST

Memory Allocation Bugs (MAL)

Causes

Improper Operation:
- Missing
- Mismatched
- Erroneous

Improper Data Value:
- Hardcoded Address
- Forbidden Address
- Single Owner of Object Address
- Wrong Size Used

Improper Object Address:
- Wild Pointer
- Dangling Pointer
- Wrong Position Pointer

MAL Operations

- Allocate
- Extend
- Reallocate-Extend

Consequences

Improper Object Address:
- NULL Pointer
- Wild Pointer

Improper Object Size:
- Not Enough Memory Allocated

Memory Error:
- Memory Overflow
- Memory Leak
- Double Free
- Object Corruption

Attributes

Operation			Object	
Mechanism:	**Source Code:**	**Execution Space:**	**Ownership:**	**Location:**
• Implicit • Explicit	• Codebase • Third Party • Standard Library • Compiler/ Interpreter	• Userland • Kernel • Bare-Metal	• None • Single • Shared	• Stack • Heap • ...

Memory Use Bugs (MUS)

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

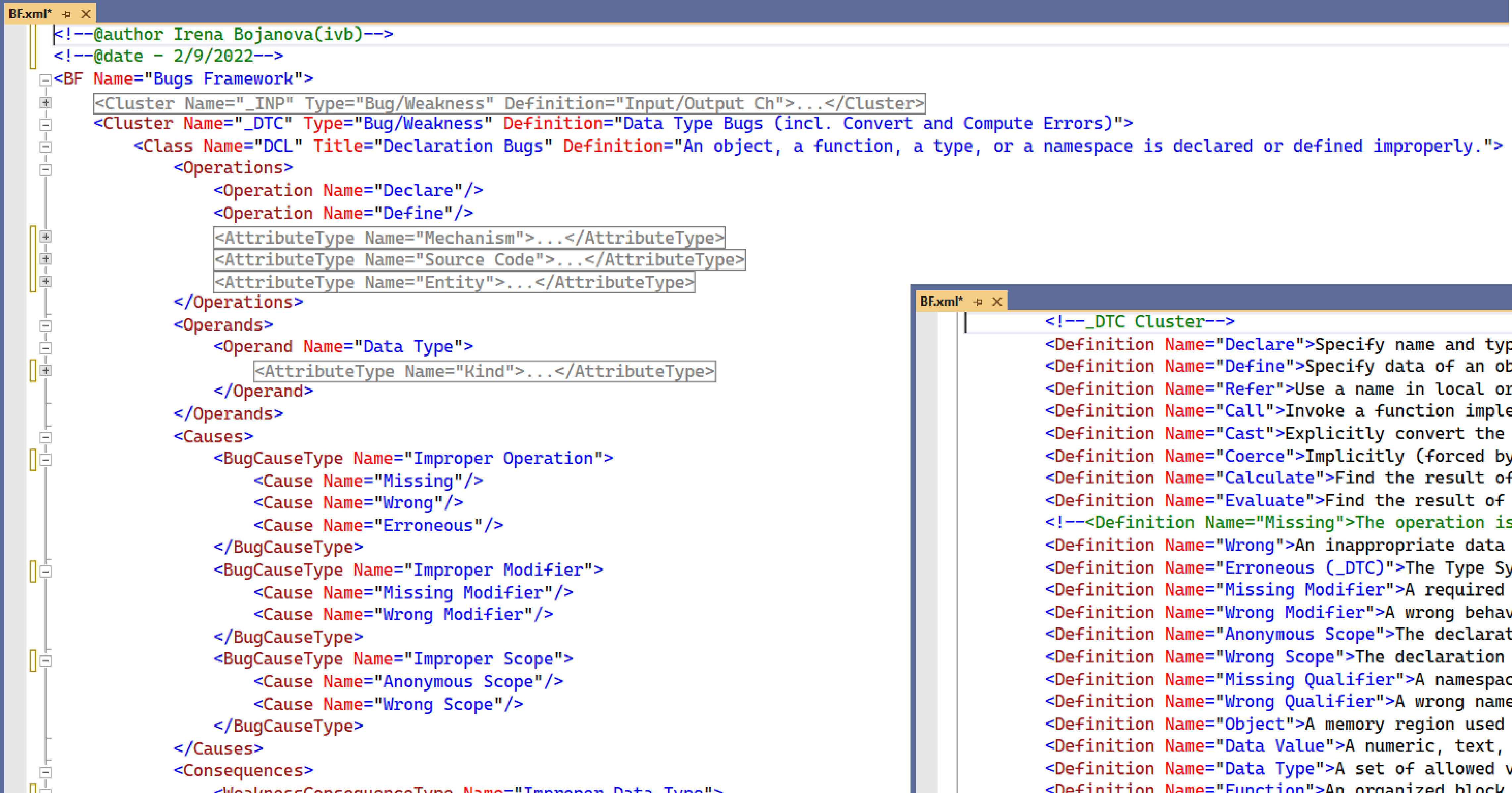
Consequences

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

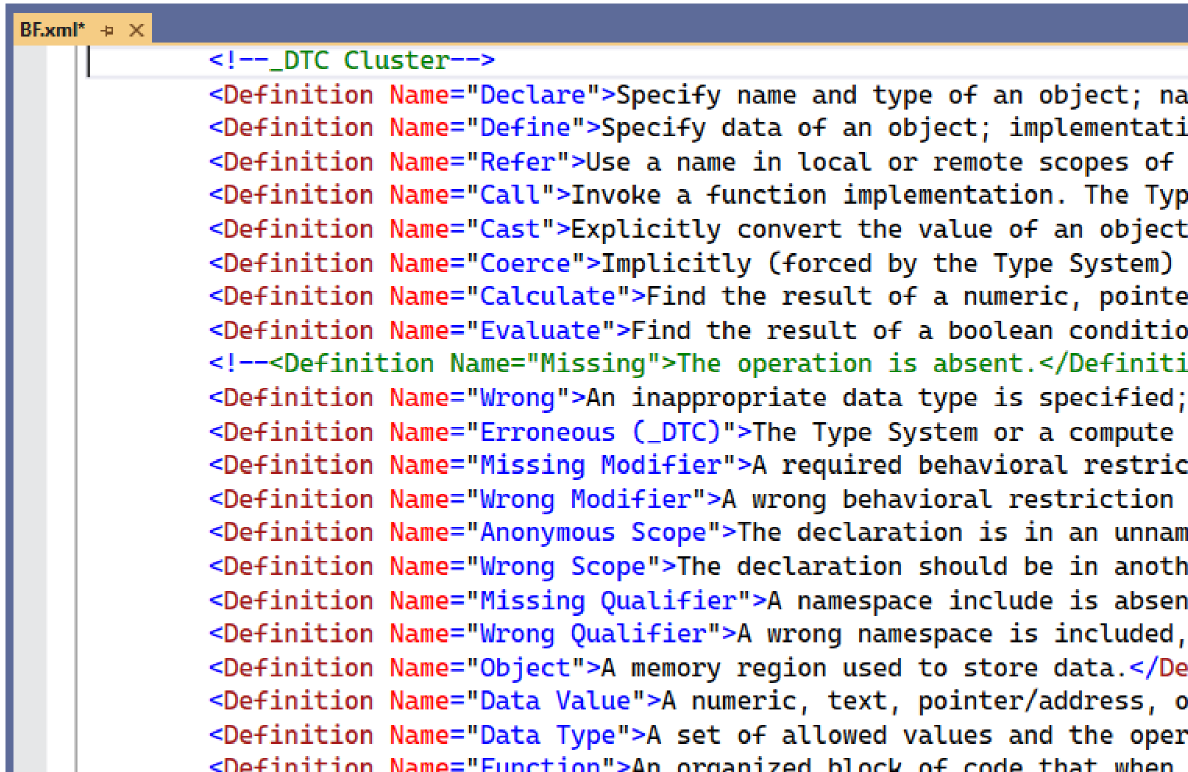
Attributes

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

BF in XML Format



```
<!--@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">
```



```
<!--_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
```

BF – Defined

- BF is a …
  - ➢ Structured
  - ➢ Complete
  - ➢ Orthogonal
  - ➢ Language independent

Classification System of software bugs and weaknesses.

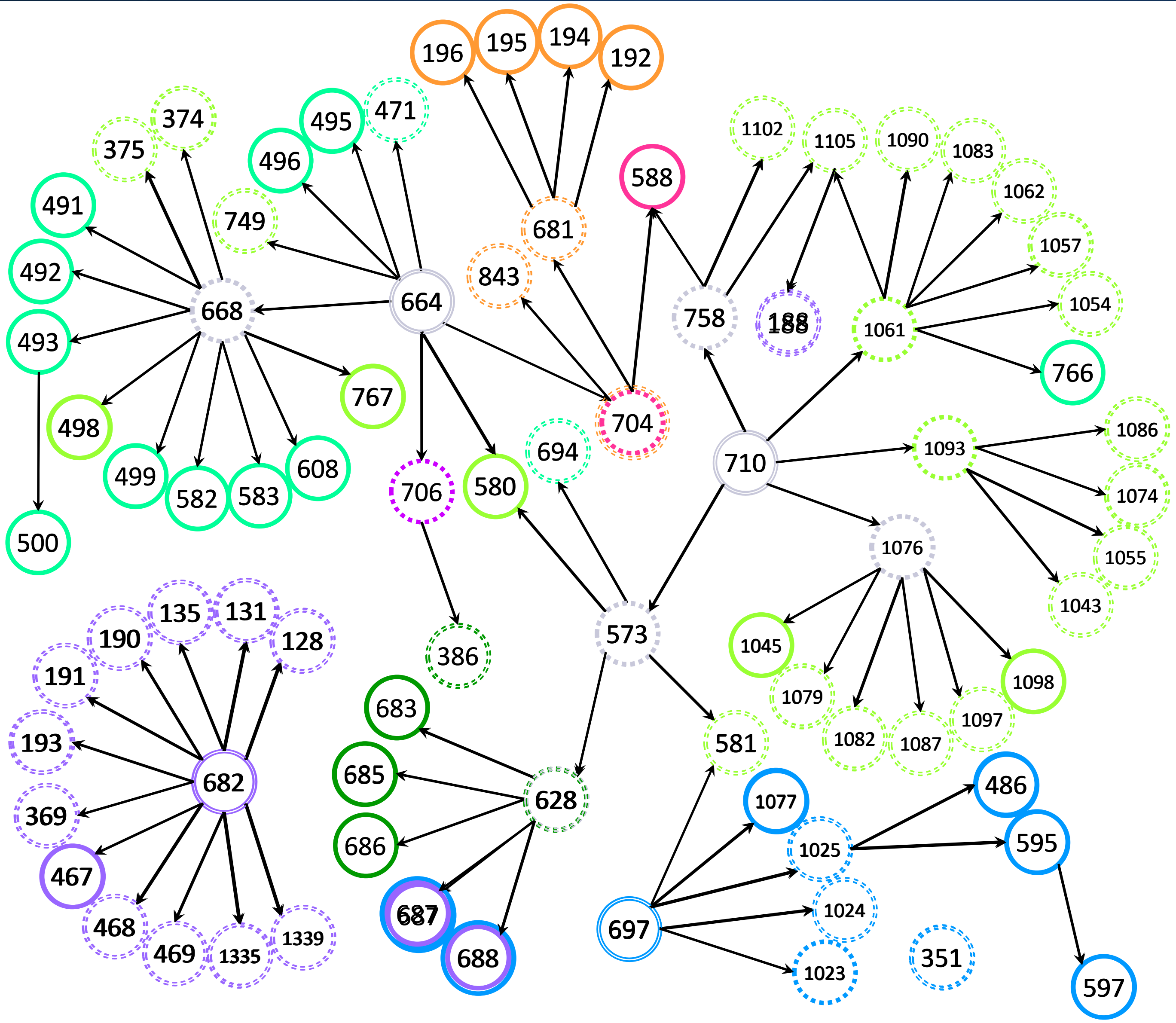
Validation towards CWE

Data Type Related CWEs

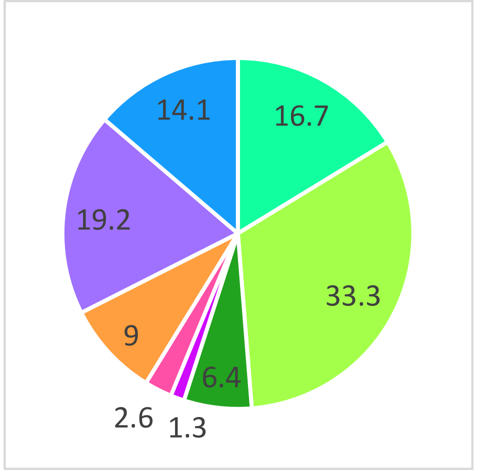
- Identifying CWEs:
  1. CWE Filtering
  2. Automated Extraction
  3. Manual Review

- 84 CWEs:
  - 78 data type related – incl.:
    - integer overflow (wrap around)
    - juggling (argument coercion)
    - pointer arithmetics
  - six others – kept for parent-child completeness.

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
◯	DCL Define	◯	NRS Call
◯	TCV Cast	◯	TCM Calculate
◯	TCV Coerce	◯	TCM Evaluate

CWEs by Abstraction:

◯	Pillar	◌	Base
◌	Class	◯	Variant

BF Hands-on:
Bad Alloc

"BadAlloc"(CVE-2021-21834)

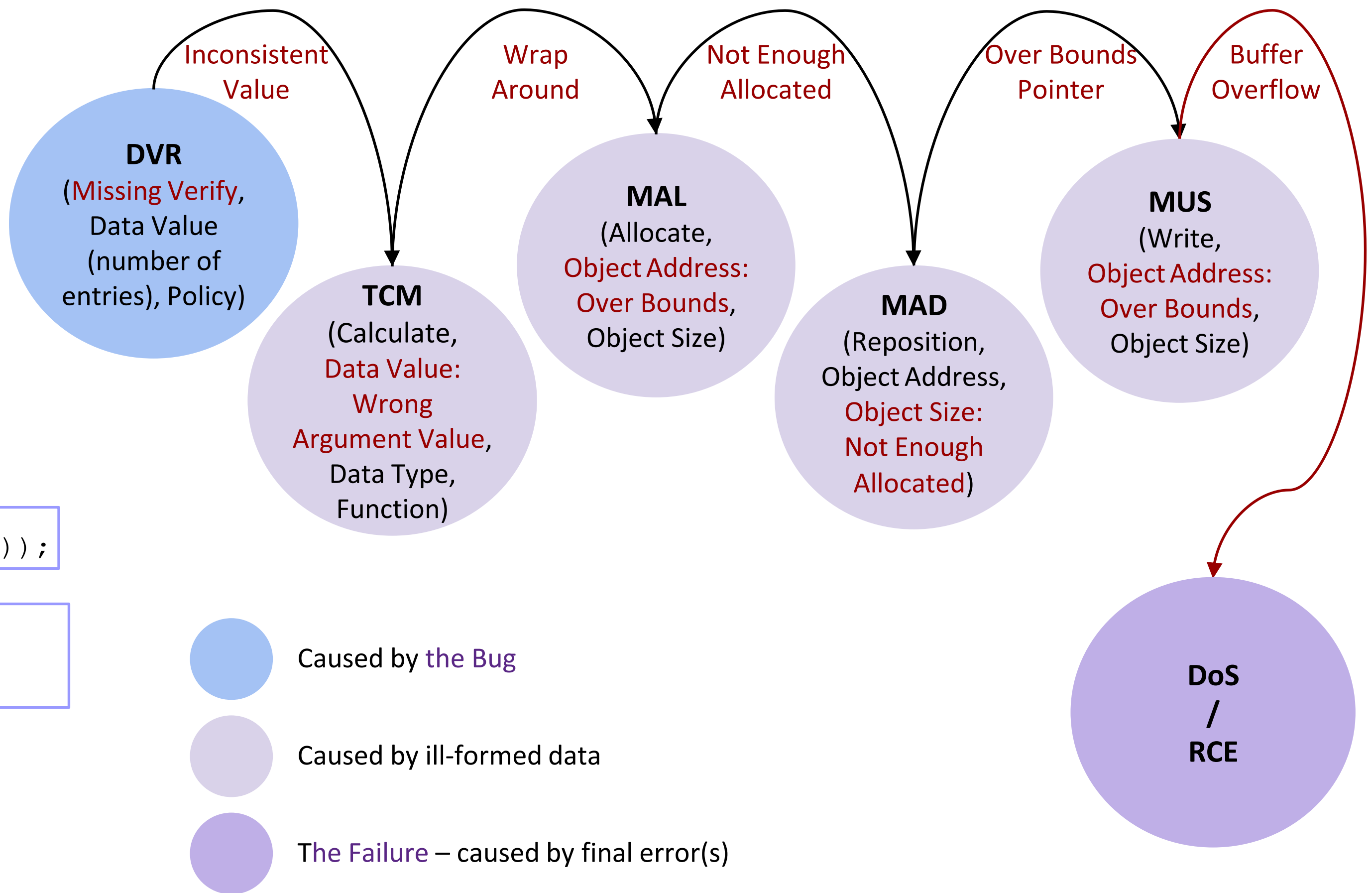
CVE-2021-21834 An exploitable integer overflow vulnerability exists within the MPEG-4 decoding functionality of the GPAC Project on Advanced Content library v1.0.1. A specially crafted MPEG-4 input when decoding the atom for the &#8220;co64&#8221; FOURCC can cause an integer overflow due to unchecked 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;
}
```



Inconsistent Value
Wrap Around
Not Enough Allocated
Over Bounds Pointer
Buffer Overflow
DVR (Missing Verify, Data Value (number of entries), Policy)
TCM (Calculate, Data Value: Wrong Argument Value, Data Type, Function)
MAL (Allocate, Object Address: Over Bounds, Object Size)
MAD (Reposition, Object Address, Object Size: Not Enough Allocated)
MUS (Write, Object Address: Over Bounds, Object Size)
DoS / RCE
Caused by the Bug
Caused by ill-formed data
The Failure – caused by final error(s)

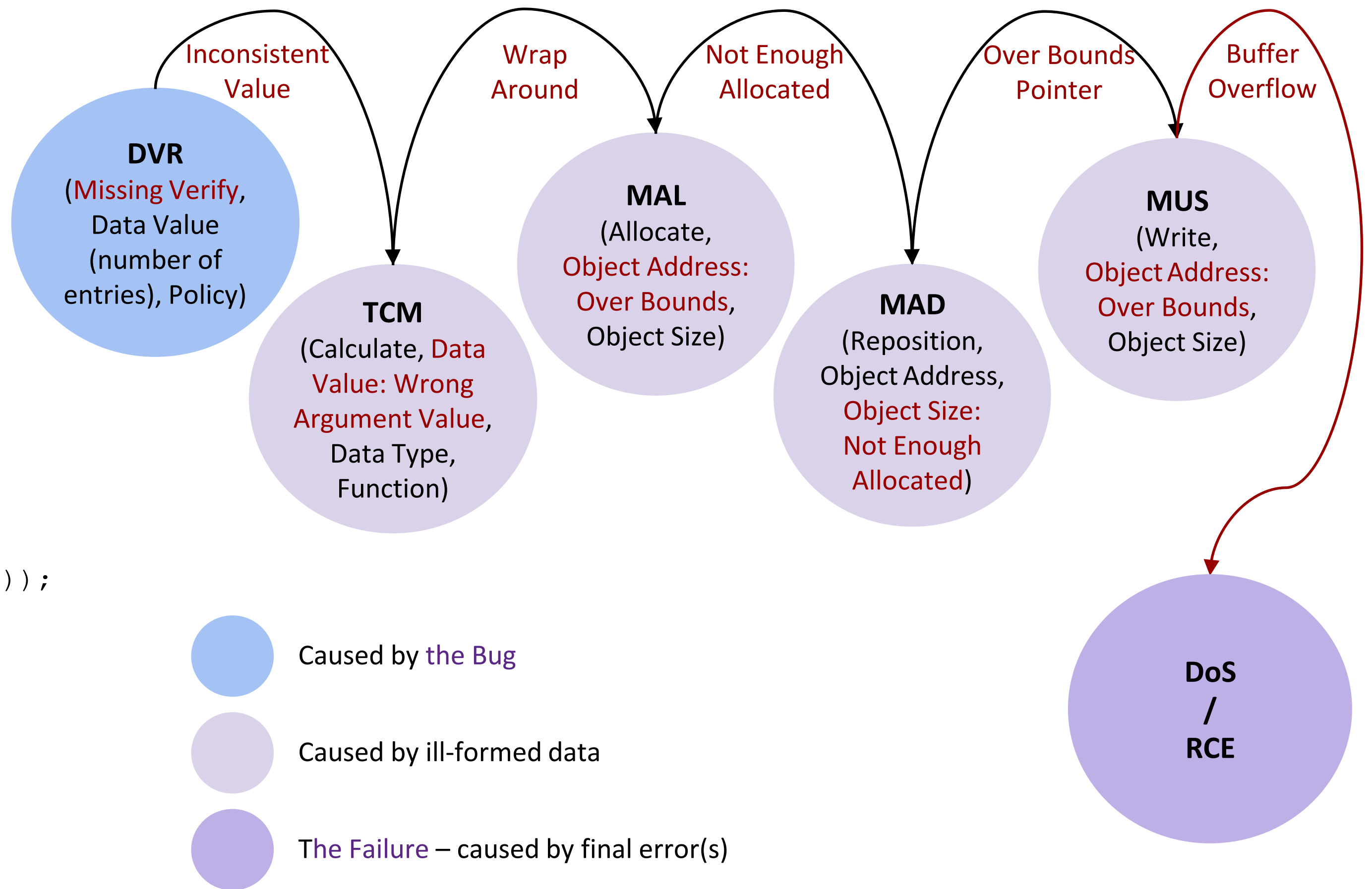
“BadAlloc” – the Fix

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 &#8220;co64&#8221; 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;
}
```



Inconsistent Value
Wrap Around
Not Enough Allocated
Over Bounds Pointer
Buffer Overflow
DVR (Missing Verify, Data Value (number of entries), Policy)
TCM (Calculate, Data Value: Wrong Argument Value, Data Type, Function)
MAL (Allocate, Object Address: Over Bounds, Object Size)
MAD (Reposition, Object Address, Object Size: Not Enough Allocated)
MUS (Write, Object Address: Over Bounds, Object Size)
DoS / RCE
Caused by the Bug
Caused by ill-formed data
The Failure – caused by final error(s)

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

Denial of Service / Remote Code Execution

Attributes

Mechanism:
- Sequential

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

Execution Space:
- Userland

Span:
- Huge

Location:
- Heap

BF Hands-on: Incorrect Pointer Scaling

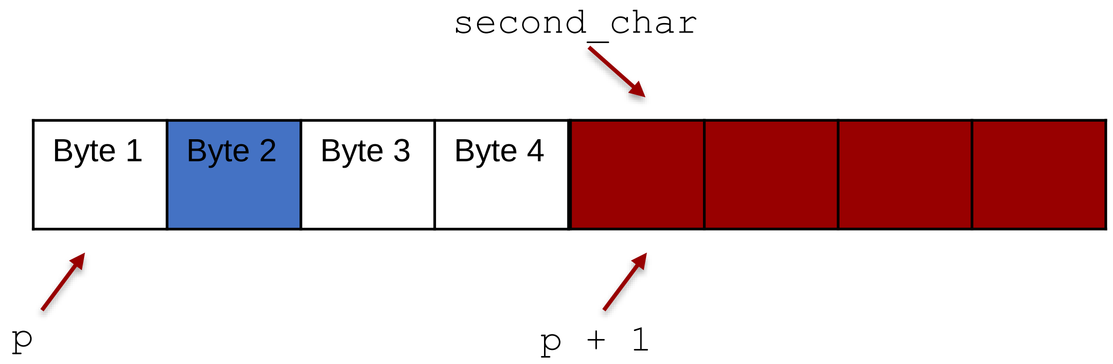
Incorrect Pointer Scaling (CWE-468, Ex. 1)

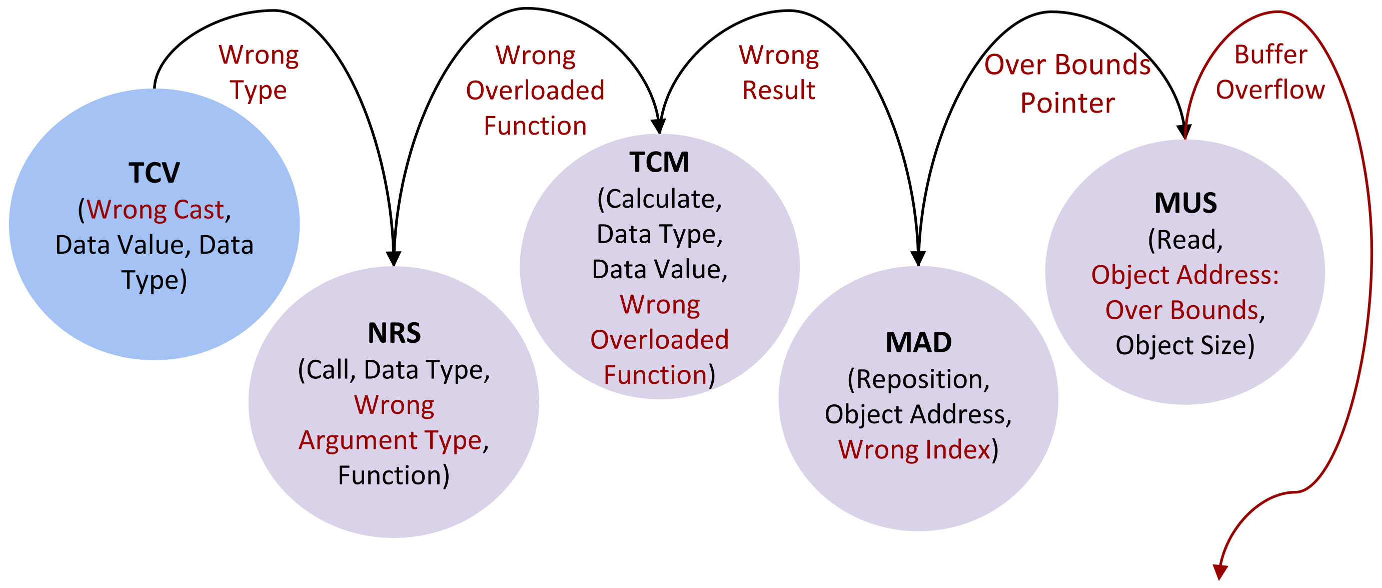
CWE-468, Example 1: This example attempts to calculate the position of the second byte of a pointer.

Example Language: **C**

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

moving 4 bytes




TCV (Wrong Cast, Data Value, Data Type)
Wrong Type
NRS (Call, Data Type, Wrong Argument Type, Function)
Wrong Overloaded Function
TCM (Calculate, Data Type, Data Value, Wrong Overloaded Function)
Wrong Result
MAD (Reposition, Object Address, Wrong Index)
Over Bounds Pointer
MUS (Read, Object Address: Over Bounds, Object Size)
Buffer Overflow

● Caused by the Bug

● Caused by ill-formed data

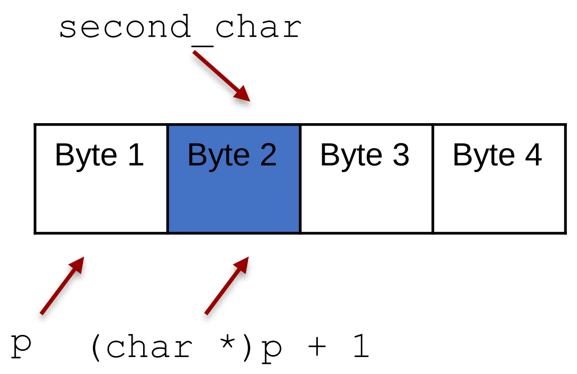
Incorrect Pointer Scaling – the Fix

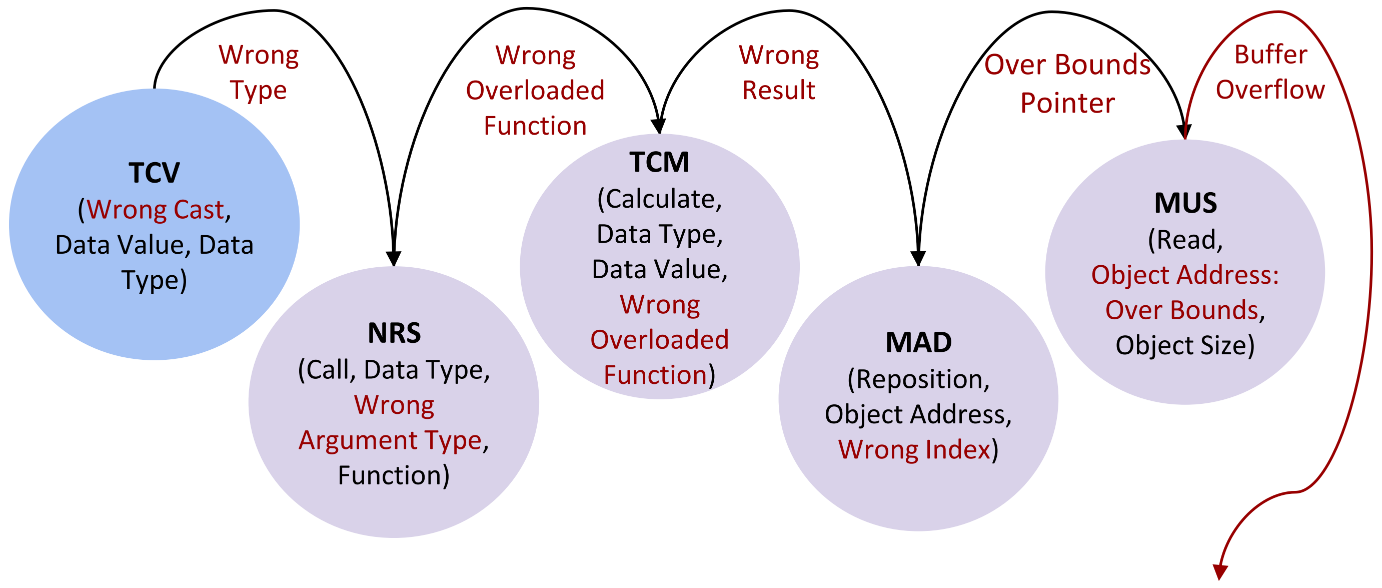
CWE-468 Example 1

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;
```



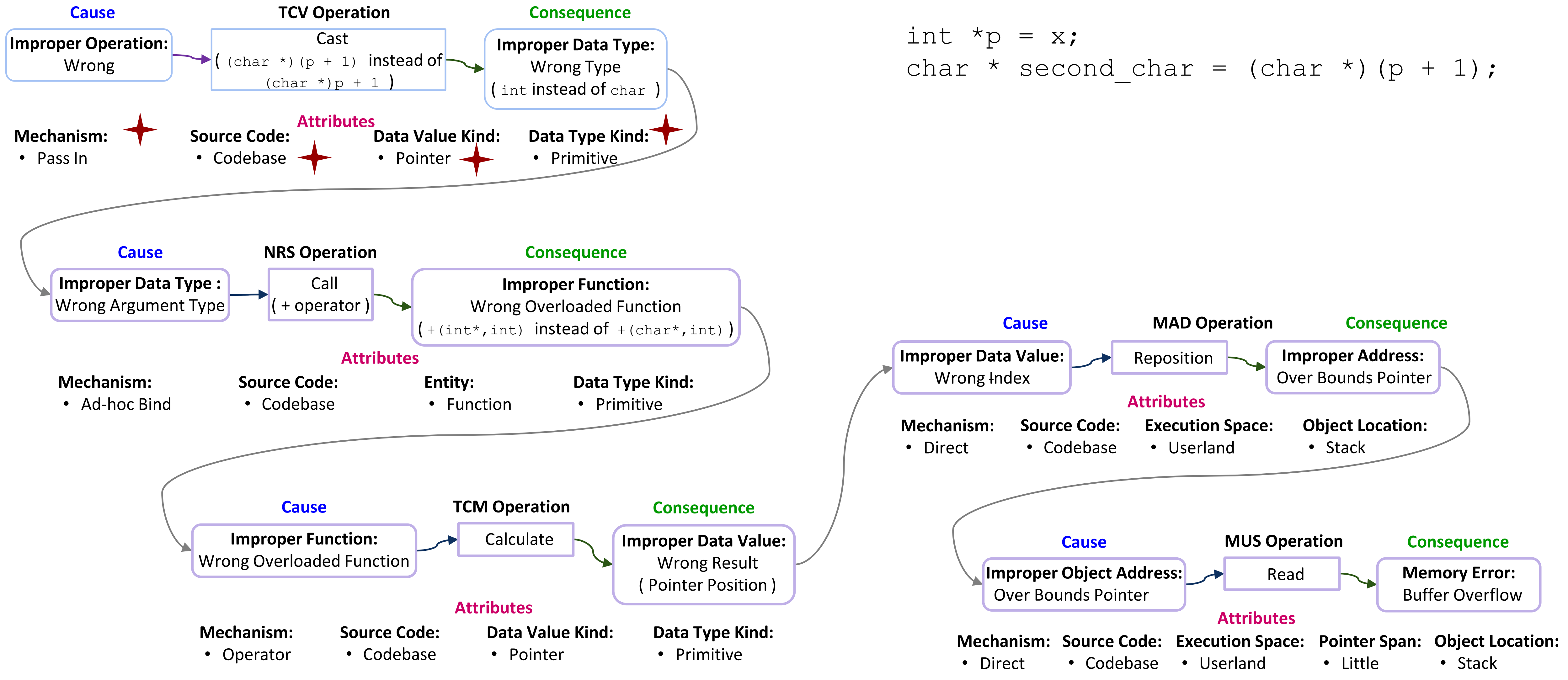


TCV (Wrong Cast, Data Value, Data Type)
Wrong Type
NRS (Call, Data Type, Wrong Argument Type, Function)
Wrong Overloaded Function
TCM (Calculate, Data Type, Data Value, Wrong Overloaded Function)
Wrong Result
MAD (Reposition, Object Address, Wrong Index)
Over Bounds Pointer
MUS (Read, Object Address: Over Bounds, Object Size)
Buffer Overflow

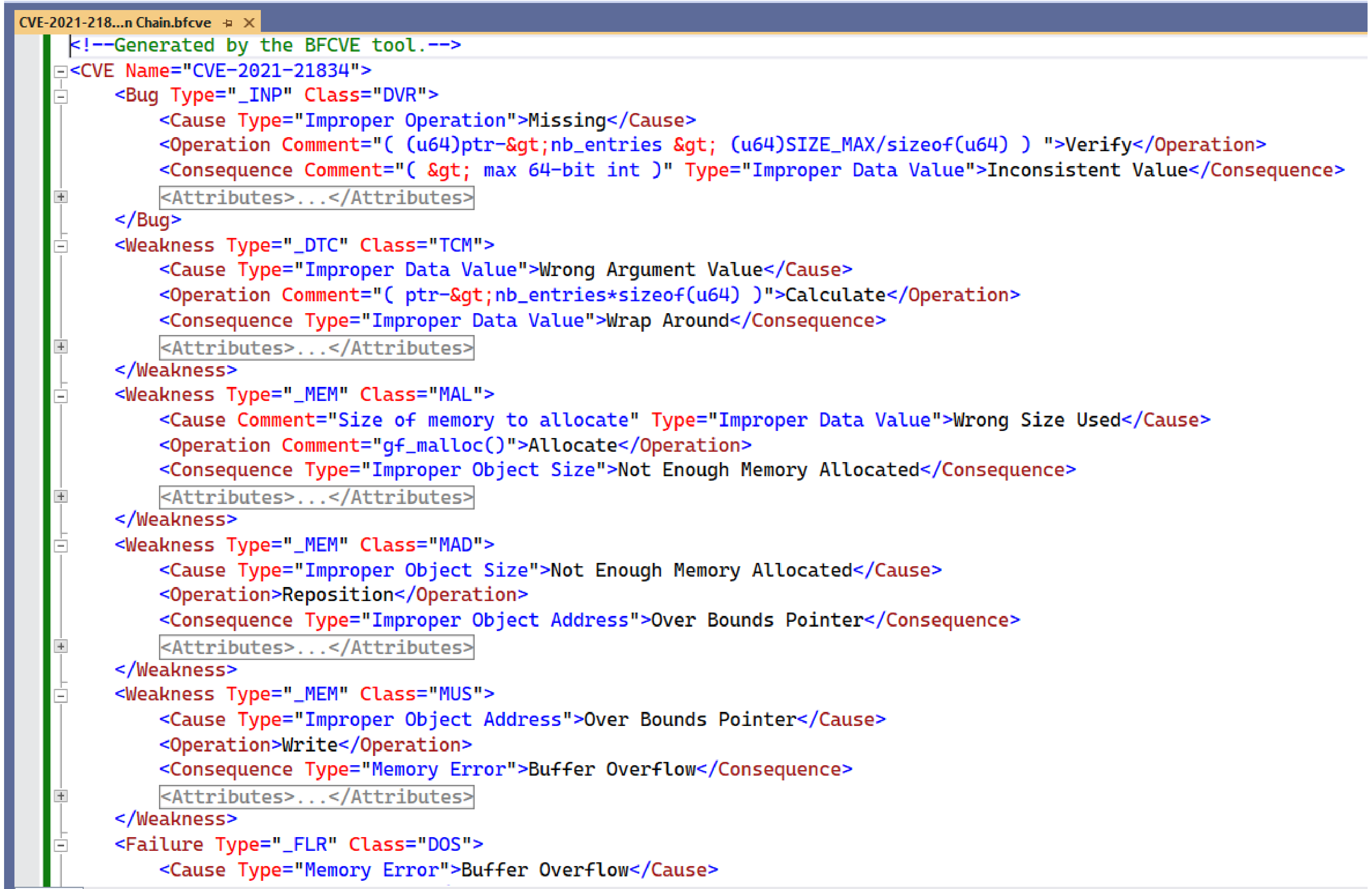
●	Caused by the Bug
●	Caused by ill-formed data

BF Description of CWE-468, Example 1

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


Cause
Improper Operation: Wrong
TCV Operation
Cast ((char *)(p + 1) instead of (char *)p + 1)
Consequence
Improper Data Type: Wrong Type (int instead of char)
Attributes
Mechanism: • Pass In
Source Code: • Codebase
Data Value Kind: • Pointer
Data Type Kind: • Primitive
Cause
Improper Data Type : Wrong Argument Type
NRS Operation
Call (+ operator)
Consequence
Improper Function: Wrong Overloaded Function (+(int*,int) instead of +(char*,int))
Attributes
Mechanism: • Ad-hoc Bind
Source Code: • Codebase
Entity: • Function
Data Type Kind: • Primitive
Cause
Improper Function: Wrong Overloaded Function
TCM Operation
Calculate
Consequence
Improper Data Value: Wrong Result (Pointer Position)
Attributes
Mechanism: • Operator
Source Code: • Codebase
Data Value Kind: • Pointer
Data Type Kind: • Primitive
Cause
Improper Data Value: Wrong Index
MAD Operation
Reposition
Consequence
Improper Address: Over Bounds Pointer
Attributes
Mechanism: • Direct
Source Code: • Codebase
Execution Space: • Userland
Object Location: • Stack
Cause
Improper Object Address: Over Bounds Pointer
MUS Operation
Read
Consequence
Memory Error: Buffer Overflow
Attributes
Mechanism: • Direct
Source Code: • Codebase
Execution Space: • Userland
Pointer Span: • Little
Object Location: • Stack

BF Descriptions in XML Format



CVE-2021-218...n Chain.bfcve

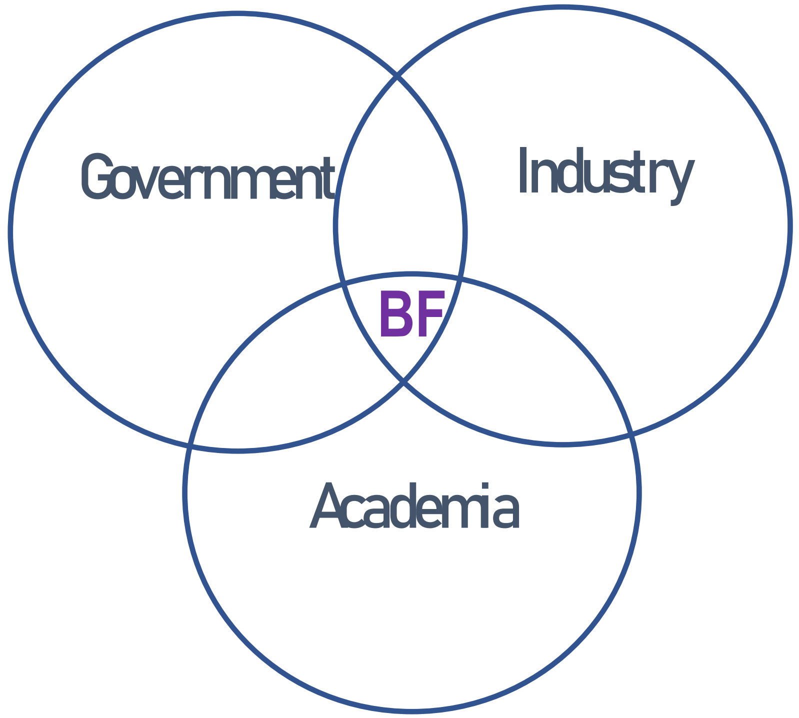
```
<!--Generated by the BFCVE tool.-->
<CVE Name="CVE-2021-21834">
    <Bug Type="_INP" Class="DVR">
        <Cause Type="Improper Operation">Missing</Cause>
        <Operation Comment="( (u64)ptr-&gt;nb_entries &gt; (u64)SIZE_MAX/sizeof(u64) ) ">Verify</Operation>
        <Consequence Comment="( &gt; 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</Cause>
        <Operation Comment="( ptr-&gt;nb_entries*sizeof(u64) )">Calculate</Operation>
        <Consequence Type="Improper Data Value">Wrap Around</Consequence>
        <Attributes>...</Attributes>
    </Weakness>
    <Weakness Type="_MEM" Class="MAL">
        <Cause Comment="Size of memory to allocate" Type="Improper Data Value">Wrong Size Used</Cause>
        <Operation Comment="gf_malloc()">Allocate</Operation>
        <Consequence Type="Improper Object Size">Not Enough Memory Allocated</Consequence>
        <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</Consequence>
        <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>...</Attributes>
    </Weakness>
    <Failure Type="_FLR" Class="DOS">
        <Cause Type="Memory Error">Buffer Overflow</Cause>
```

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

Sara Moshtari: sm2481@rit.edu

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