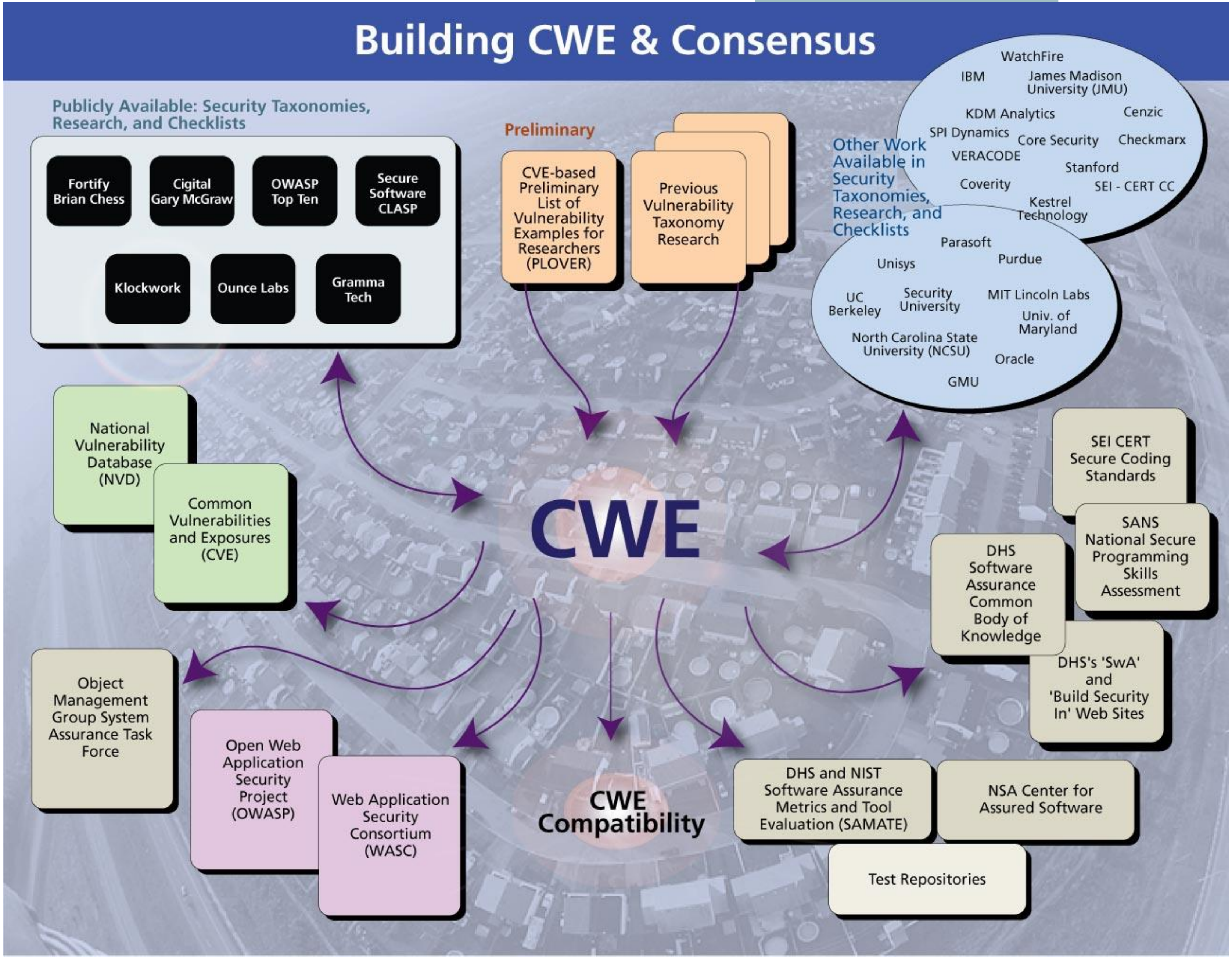
Formalizing Software Bugs

Irena Bojanova
UMUC, NIST

12/08/2014

Software Bugs

- “Software bug” – a concept that applies to:
  - CWE – Common Weakness Enumeration
  - CVE – Common Vulnerabilities and Exposures
  - CAPEC – Common Attack Pattern Enumeration and Classification.
- Related, but different:
  - Weakness – a static presence in software – might never cause problems;
  - only when an attacker finds the weakness and exploits it
  - the vulnerability of this software is exposed.
- CWE & CVE + CAPEC:
  - Considerable efforts, providing foundational knowledge
  - However, not sufficient, accurate, and precise enough.



Building CWE & Consensus

Publicly Available: Security Taxonomies, Research, and Checklists

Fortify Brian Chess

Cigital Gary McGraw

OWASP Top Ten

Secure Software CLASP

Klockwork

Ounce Labs

Gramma Tech

Preliminary

CVE-based Preliminary List of Vulnerability Examples for Researchers (PLOVER)

Previous Vulnerability Taxonomy Research

Other Work Available in Security Taxonomies, Research, and Checklists

WatchFire

IBM

James Madison University (JMU)

KDM Analytics

Cenzic

SPI Dynamics

Core Security

Checkmarx

VERACODE

Stanford

Coverity

SEI - CERT CC

Kestrel Technology

Parasoft

Unisys

Purdue

UC Berkeley

Security University

MIT Lincoln Labs

Univ. of Maryland

North Carolina State University (NCSU)

Oracle

GMU

National Vulnerability Database (NVD)

Common Vulnerabilities and Exposures (CVE)

CWE

SEI CERT Secure Coding Standards

SANS National Secure Programming Skills Assessment

DHS Software Assurance Common Body of Knowledge

DHS's 'SwA' and 'Build Security In' Web Sites

Object Management Group System Assurance Task Force

Open Web Application Security Project (OWASP)

Web Application Security Consortium (WASC)

CWE Compatibility

DHS and NIST Software Assurance Metrics and Tool Evaluation (SAMATE)

NSA Center for Assured Software

Test Repositories

Problems with CWE, CVE, & CAPEC

- CWE:
  - Not orthogonal, just an enumeration
  - Ambiguous definitions
  - Do not match well with classes reported by test tools.

Example: CWE-119:

Improper Restriction of Operations within the Bounds of a Memory Buffer

> "The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the **intended boundary** of the buffer."

Problems CWE, CVE, & CAPEC (cont.)

- CVE:
  - Ambiguous definitions
  - Do not fully and precisely describe involved CWEs.

Example: CVE-160: Heartbleed.

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

“Technical Details: Buffer Errors (CWE-119)”

→ Actual exploited is CWE-126, while only parent CWE-119 is listed.

→ CWE-130 is in CWE-126, but CWE-20 also can precede CWE-126.

Problems CWE, CVE, & CAPEC (cont.)

- CAPEC:
  - Do not fully show dynamics of activities leading to realization of attack
  - CVE, CWE, CAPEC inks could be confusing.

Example: Heartbleed – which CAPEC and CWEs?
From attack to weakness:

- Heartbleed exploits Buffer Overread (CWE-126)
  -> attack should be CAPEC 540: Overread Buffers
- CAPEC 540, “Rrelated Weakness” is CWE-125, parent of CWE-126, lists CVE-160 (Heartbleed).

Form weakness to attack:

- CVE-160 (Heartbleed), “Technical Details”— CWE-119.
- CWE-119, “Related Attack Patterns” – no CAPEC 540, closest CAPEC-47.

→ It is a fact that CVE-160 is listed in CAPEC 540, but not in CAPEC-47.

Need For

→Precise descriptions of attacks (CAPECs) that lead to realization of vulnerabilities (CVEs), exposed by software weaknesses (CWEs).

→Research to explore formalization of CWEs, CVEs, & CAPECs.

CAPEC 540 – CAPEC 47

CAPEC-540: Overread Buffers

“An adversary attacks a target by providing input that causes an application to read beyond the boundary of a defined buffer. This typically occurs when a value influencing where to start or stop reading is set to reflect positions outside of the valid memory location of the buffer. This type of attack may result in exposure of sensitive information, a system crash, or arbitrary code execution.”

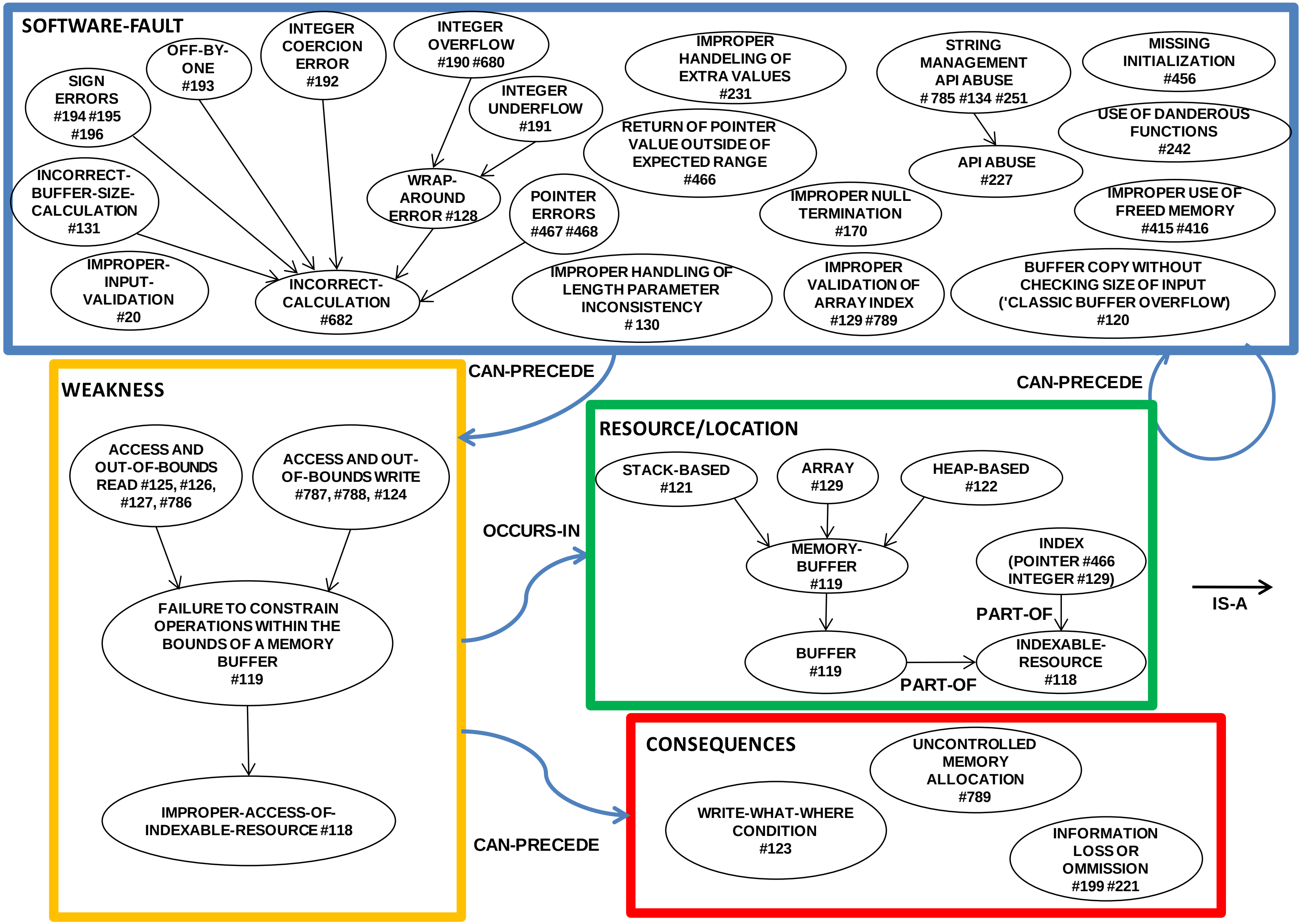
CAPEC-47: Buffer Overflow via Parameter Expansion

“In this attack, the target software is given input that the attacker knows will be modified and expanded in size during processing. This attack relies on the target software failing to anticipate that the expanded data may exceed some internal limit, thereby creating a buffer overflow.”

Previous Related Work

- Semantic Templates – Yan Wu's Dissertation
  - Classify
  - Extract commonalities
  - Reorganize information into four groups:
    - Software faults that lead to a weakness
    - 2) Resources that a weakness affects
    - 3) Weakness characteristics
    - 4) Consequences/failures resulting from the weakness.
  - Support by vulnerabilities and feedback to annotate vulnerabilities

- Software Fault Patterns – KDM Analytics
  - Classify
  - Identify patterns
  - Test cases generator.



SOFTWARE-FAULT

SIGN ERRORS #194 #195 #196

OFF-BY-ONE #193

INTEGER COERCION ERROR #192

INTEGER OVERFLOW #190 #680

INTEGER UNDERFLOW #191

IMPROPER HANDELING OF EXTRA VALUES #231

RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE #466

STRING MANAGEMENT API ABUSE # 785 #134 #251

API ABUSE #227

MISSING INITIALIZATION #456

USE OF DANDEROUS FUNCTIONS #242

INCORRECT-BUFFER-SIZE-CALCULATION #131

WRAP-AROUND ERROR #128

POINTER ERRORS #467 #468

IMPROPER NULL TERMINATION #170

IMPROPER USE OF FREED MEMORY #415 #416

IMPROPER-INPUT-VALIDATION #20

INCORRECT-CALCULATION #682

IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY # 130

IMPROPER VALIDATION OF ARRAY INDEX #129 #789

BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW') #120

CAN-PRECEDE

CAN-PRECEDE

WEAKNESS

ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786

ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124

FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119

IMPROPER-ACCESS-OF-INDEXABLE-RESOURCE #118

OCCURS-IN

RESOURCE/LOCATION

STACK-BASED #121

ARRAY #129

HEAP-BASED #122

MEMORY-BUFFER #119

INDEX (POINTER #466 INTEGER #129)

PART-OF

BUFFER #119

INDEXABLE-RESOURCE #118

PART-OF

IS-A

CAN-PRECEDE

CONSEQUENCES

UNCONTROLLED MEMORY ALLOCATION #789

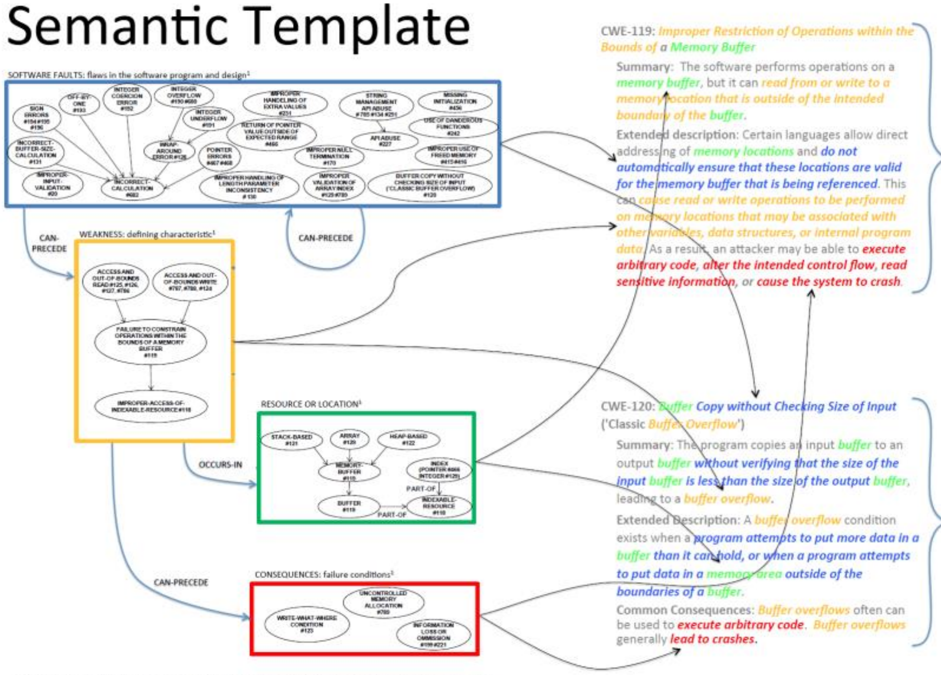
WRITE-WHAT-WHERE CONDITION #123

INFORMATION LOSS OR OMMISSION #199 #221

Buffer Overflow Semantic Template

Previous Related Work

Semantic Template


SOFTWARE FAULTS: flaws in the software program and design¹
SIGN ERRORS #194 #195 #196
OFF-BY-ONE #193
INTEGER COERCION ERROR #192
INTEGER OVERFLOW #190 #680
INTEGER UNDERFLOW #191
INCORRECT-BUFFER-SIZE-CALCULATION #131
WRAP-AROUND ERROR #128
POINTER ERRORS #467 #468
IMPROPER-INPUT-VALIDATION #20
INCORRECT-CALCULATION #682
IMPROPER HANDELING OF EXTRA VALUES #231
RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE #466
STRING MANAGEMENT API ABUSE #785 #134 #251
API ABUSE #227
MISSING INITIALIZATION #456
USE OF DANGEROUS FUNCTIONS #242
IMPROPER NULL TERMINATION #170
IMPROPER USE OF FREED MEMORY #415 #416
IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY # 130
IMPROPER VALIDATION OF ARRAY INDEX #129 #789
BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW) #120
CAN-PRECEDE
CAN-PRECEDE
WEAKNESS: defining characteristic¹
ACCESS AND OUT-OF-BOUNDS READ #125, #126, #127, #786
ACCESS AND OUT-OF-BOUNDS WRITE #787, #788, #124
FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER #119
IMPROPER-ACCESS-OF-INDEXABLE-RESOURCE #118
OCCURS-IN
RESOURCE OR LOCATION¹
STACK-BASED #121
ARRAY #129
HEAP-BASED #122
MEMORY-BUFFER #119
INDEX (POINTER #466 INTEGER #129)
PART-OF
BUFFER #119
PART-OF
INDEXABLE-RESOURCE #118
CAN-PRECEDE
CONSEQUENCES: failure conditions¹
UNCONTROLLED MEMORY ALLOCATION #789
WRITE-WHAT-WHERE CONDITION #123
INFORMATION LOSS OR OMMISSION #199 #221

CWE-119: *Improper Restriction of Operations within the Bounds of* a *Memory Buffer*

Summary: The software performs operations on a *memory buffer*, but it can *read from or write to a memory location that is outside of the intended boundary of the buffer.*

Extended description: Certain languages allow direct addressing of *memory locations* and *do not automatically ensure that these locations are valid for the memory buffer that is being referenced*. This can *cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data*. As a result, an attacker may be able to ***execute arbitrary code, alter the intended control flow, read sensitive information***, or ***cause the system to crash***.

CWE-120: *Buffer Copy without Checking Size of Input* ('Classic *Buffer Overflow*')

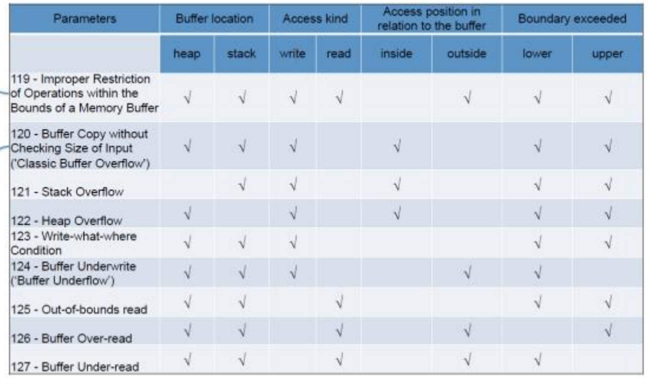
Summary: The program copies an input *buffer* to an output *buffer without verifying that the size of the input buffer is less than the size of the output buffer*, leading to a *buffer overflow*.

Extended Description: A *buffer overflow* condition exists when a *program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer.*

Common Consequences: *Buffer overflows* often can be used to ***execute arbitrary code***. *Buffer overflows* generally ***lead to crashes***.

¹Yan Wu, "Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories", 2011

Software Fault Pattern



Parameters	Buffer location		Access kind		Access position in relation to the buffer		Boundary exceeded	
	heap	stack	write	read	inside	outside	lower	upper
119 - Improper Restriction of Operations within the Bounds of a Memory Buffer	√	√	√	√		√	√	√
120 - Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	√	√	√		√		√	√
121 - Stack Overflow		√	√		√		√	√
122 - Heap Overflow	√		√		√		√	√
123 - Write-what-where Condition	√	√	√				√	√
124 - Buffer Underwrite ('Buffer Underflow')	√	√	√			√	√	
125 - Out-of-bounds read	√	√		√			√	√
126 - Buffer Over-read	√	√		√		√		√
127 - Buffer Under-read	√	√		√		√	√	

Note: chart is adapted

Tried to formally describe :

- Vulnerabilities as chains of exploited weaknesses
- Involved CWEs utilizing the corresponding SFPs.

Failed midway – realized need to specify also the dynamic aspect of the attack.

Formalization of CWE, CVE & CAPEC

- Formalization of CWEs will:
  - Clarify their definitions
  - Remove ambiguities among them.

  => As a result, reclassification of CWEs will be possible.
- Same reasoning applies to CVEs and CAPECs formalization.

- Important:
  - Base of CVEs are CWEs.
    Each CVE is a result of CAPEC(s) exploiting software CWEs.

  - Each CWE is a static software property.
  - Each CVE is a dynamic software property that occurs as a result of some CAPEC (s).

Considered Notations

- Static software properties: formal software specification models, languages, notations, and tools.
  - Z notation - standardized by ISO.
- Dynamic software properties – those used for process specification: process algebra, Petri nets, and state machines.
  - CSP – Communicating Sequential Processes
  - UML & OCL – Unified Modeling Language with Object Constraint Language.

- Z notation and CSP – strongly based on the mathematical notation without introducing strange constructions
- UML – provides good visual representation.

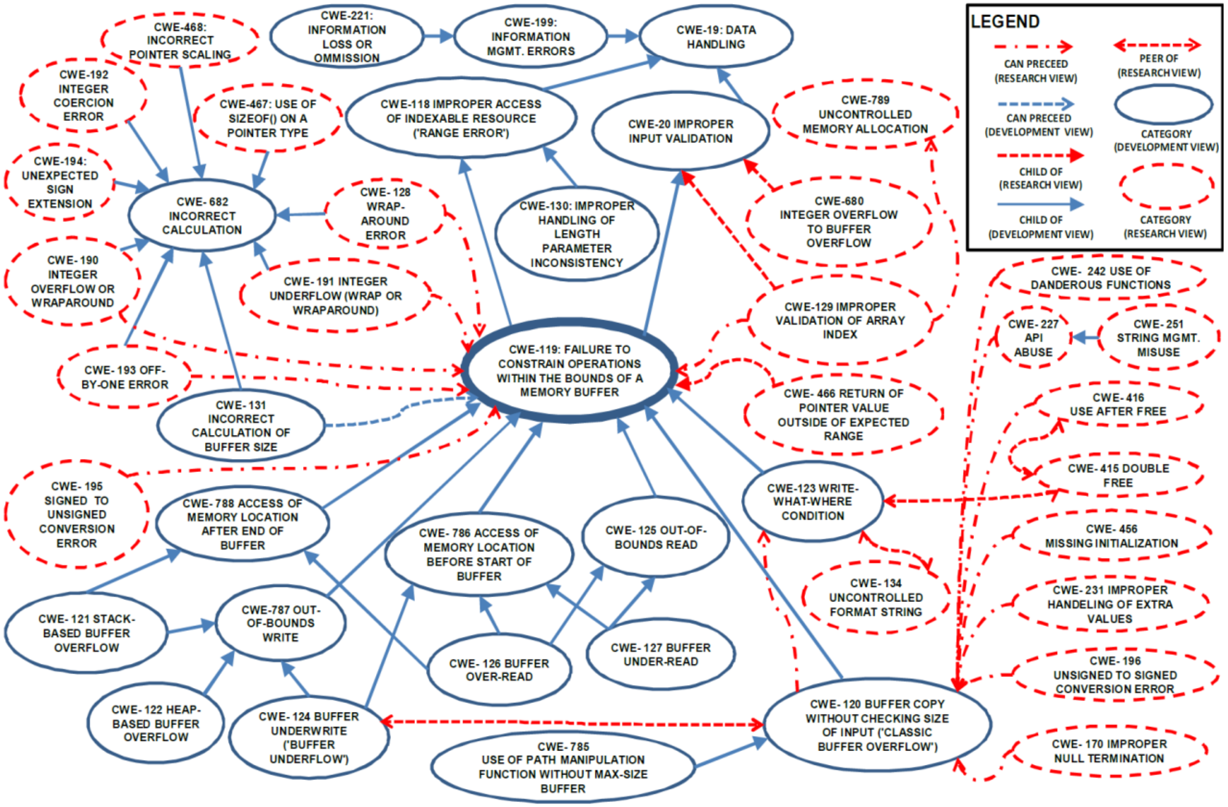
CWEs "Mental Models" (Step 1)

- CWE & CVE databases – represented in XML
- CWEs & CVEs text presentations – verbose and hardly navigable.

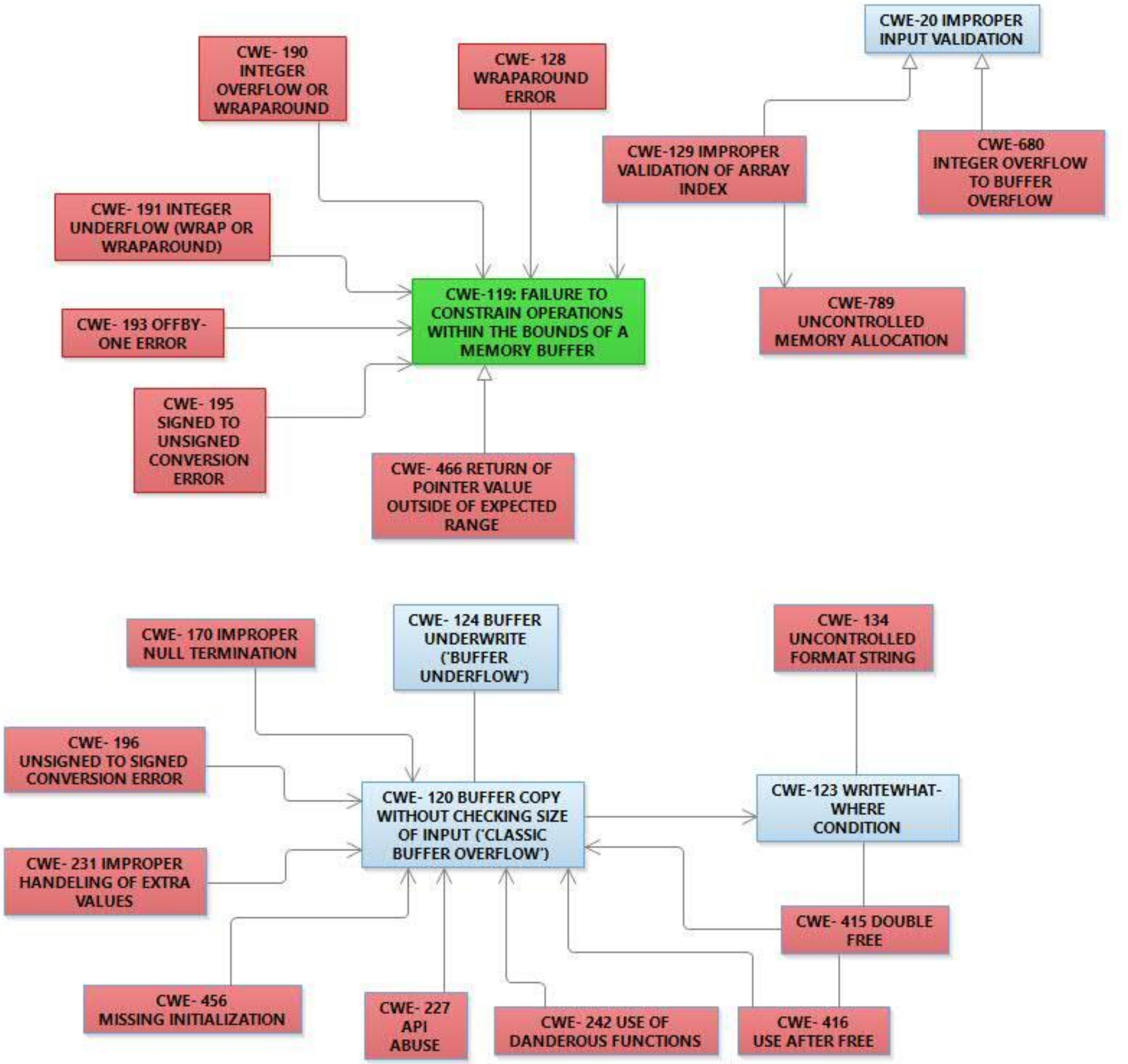
First Research Step:

- Formal graph representation of these databases
- "Mental models" of base CWEs – UML class diagrams:
  - CWEs –classes
  - Relationships between CWEs – associations

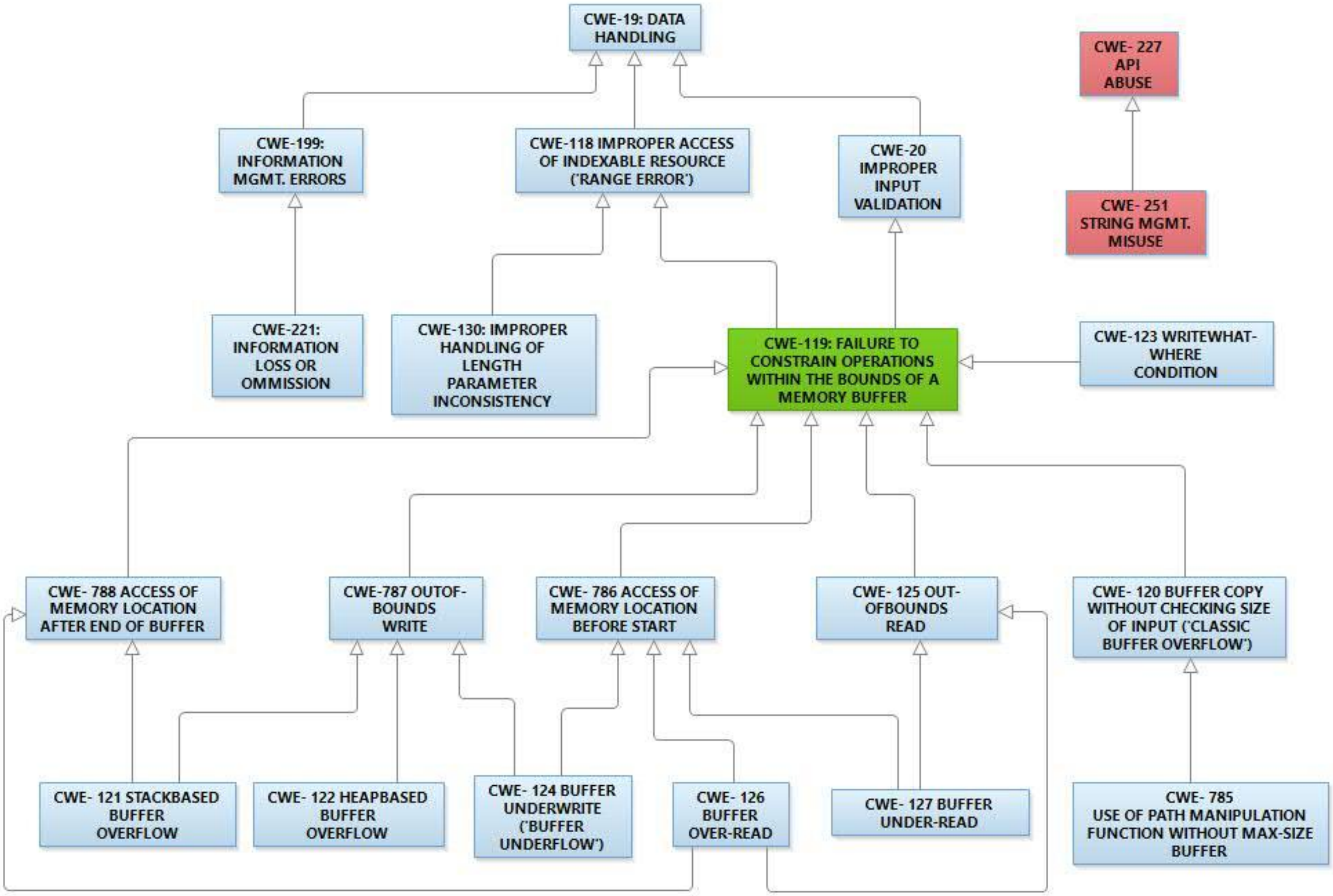
➔Later incorporate an OCL specification for each CWE.


LEGEND
CAN PRECEED (RESEARCH VIEW)
PEER OF (RESEARCH VIEW)
CAN PRECEED (DEVELOPMENT VIEW)
CATEGORY (DEVELOPMENT VIEW)
CHILD OF (RESEARCH VIEW)
CHILD OF (DEVELOPMENT VIEW)
CATEGORY (RESEARCH VIEW)
CWE-468: INCORRECT POINTER SCALING
CWE-221: INFORMATION LOSS OR OMMISSION
CWE-199: INFORMATION MGMT. ERRORS
CWE-19: DATA HANDLING
CWE-192 INTEGER COERCION ERROR
CWE-467: USE OF SIZEOF() ON A POINTER TYPE
CWE-118 IMPROPER ACCESS OF INDEXABLE RESOURCE ('RANGE ERROR')
CWE-20 IMPROPER INPUT VALIDATION
CWE-789 UNCONTROLLED MEMORY ALLOCATION
CWE-194: UNEXPECTED SIGN EXTENSION
CWE- 682 INCORRECT CALCULATION
CWE- 128 WRAP-AROUND ERROR
CWE-130: IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY
CWE-680 INTEGER OVERFLOW TO BUFFER OVERFLOW
CWE- 190 INTEGER OVERFLOW OR WRAPAROUND
CWE- 191 INTEGER UNDERFLOW (WRAP OR WRAPAROUND)
CWE- 242 USE OF DANDEROUS FUNCTIONS
CWE-129 IMPROPER VALIDATION OF ARRAY INDEX
CWE- 227 API ABUSE
CWE- 251 STRING MGMT. MISUSE
CWE-119: FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER
CWE- 193 OFF-BY-ONE ERROR
CWE- 466 RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE
CWE- 416 USE AFTER FREE
CWE- 131 INCORRECT CALCULATION OF BUFFER SIZE
CWE- 415 DOUBLE FREE
CWE- 195 SIGNED TO UNSIGNED CONVERSION ERROR
CWE- 788 ACCESS OF MEMORY LOCATION AFTER END OF BUFFER
CWE-123 WRITE-WHAT-WHERE CONDITION
CWE- 786 ACCESS OF MEMORY LOCATION BEFORE START OF BUFFER
CWE- 125 OUT-OF-BOUNDS READ
CWE- 456 MISSING INITIALIZATION
CWE- 134 UNCONTROLLED FORMAT STRING
CWE- 231 IMPROPER HANDELING OF EXTRA VALUES
CWE- 121 STACK-BASED BUFFER OVERFLOW
CWE-787 OUT-OF-BOUNDS WRITE
CWE- 126 BUFFER OVER-READ
CWE- 127 BUFFER UNDER-READ
CWE- 196 UNSIGNED TO SIGNED CONVERSION ERROR
CWE- 122 HEAP-BASED BUFFER OVERFLOW
CWE- 124 BUFFER UNDERWRITE ('BUFFER UNDERFLOW')
CWE- 120 BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW')
CWE- 170 IMPROPER NULL TERMINATION
CWE- 785 USE OF PATH MANIPULATION FUNCTION WITHOUT MAX-SIZE BUFFER

Buffer Overflow-related CWEs Extracted in the Preparation and Collection Phase
(Figure 5 from Yan Wu's dissertation)


CWE- 190 INTEGER OVERFLOW OR WRAPAROUND
CWE- 128 WRAPAROUND ERROR
CWE-20 IMPROPER INPUT VALIDATION
CWE-129 IMPROPER VALIDATION OF ARRAY INDEX
CWE-680 INTEGER OVERFLOW TO BUFFER OVERFLOW
CWE- 191 INTEGER UNDERFLOW (WRAP OR WRAPAROUND)
CWE-119: FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER
CWE-789 UNCONTROLLED MEMORY ALLOCATION
CWE- 193 OFFBY-ONE ERROR
CWE- 195 SIGNED TO UNSIGNED CONVERSION ERROR
CWE- 466 RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE
CWE- 170 IMPROPER NULL TERMINATION
CWE- 124 BUFFER UNDERWRITE ('BUFFER UNDERFLOW')
CWE- 134 UNCONTROLLED FORMAT STRING
CWE- 196 UNSIGNED TO SIGNED CONVERSION ERROR
CWE- 120 BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW')
CWE-123 WRITEWHAT-WHERE CONDITION
CWE- 231 IMPROPER HANDELING OF EXTRA VALUES
CWE- 415 DOUBLE FREE
CWE- 456 MISSING INITIALIZATION
CWE- 227 API ABUSE
CWE- 242 USE OF DANDEROUS FUNCTIONS
CWE- 416 USE AFTER FREE

CWE-119 Research View – Inheritance and Precedence Relationships



CWE-19: DATA HANDLING

CWE- 227 API ABUSE

CWE-199: INFORMATION MGMT. ERRORS

CWE-118 IMPROPER ACCESS OF INDEXABLE RESOURCE ('RANGE ERROR')

CWE-20 IMPROPER INPUT VALIDATION

CWE- 251 STRING MGMT. MISUSE

CWE-221: INFORMATION LOSS OR OMMISSION

CWE-130: IMPROPER HANDLING OF LENGTH PARAMETER INCONSISTENCY

CWE-119: FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER

CWE-123 WRITEWHAT-WHERE CONDITION

CWE- 788 ACCESS OF MEMORY LOCATION AFTER END OF BUFFER

CWE-787 OUTOF-BOUNDS WRITE

CWE- 786 ACCESS OF MEMORY LOCATION BEFORE START

CWE- 125 OUT-OFBOUNDS READ

CWE- 120 BUFFER COPY WITHOUT CHECKING SIZE OF INPUT ('CLASSIC BUFFER OVERFLOW')

CWE- 121 STACKBASED BUFFER OVERFLOW

CWE- 122 HEAPBASED BUFFER OVERFLOW

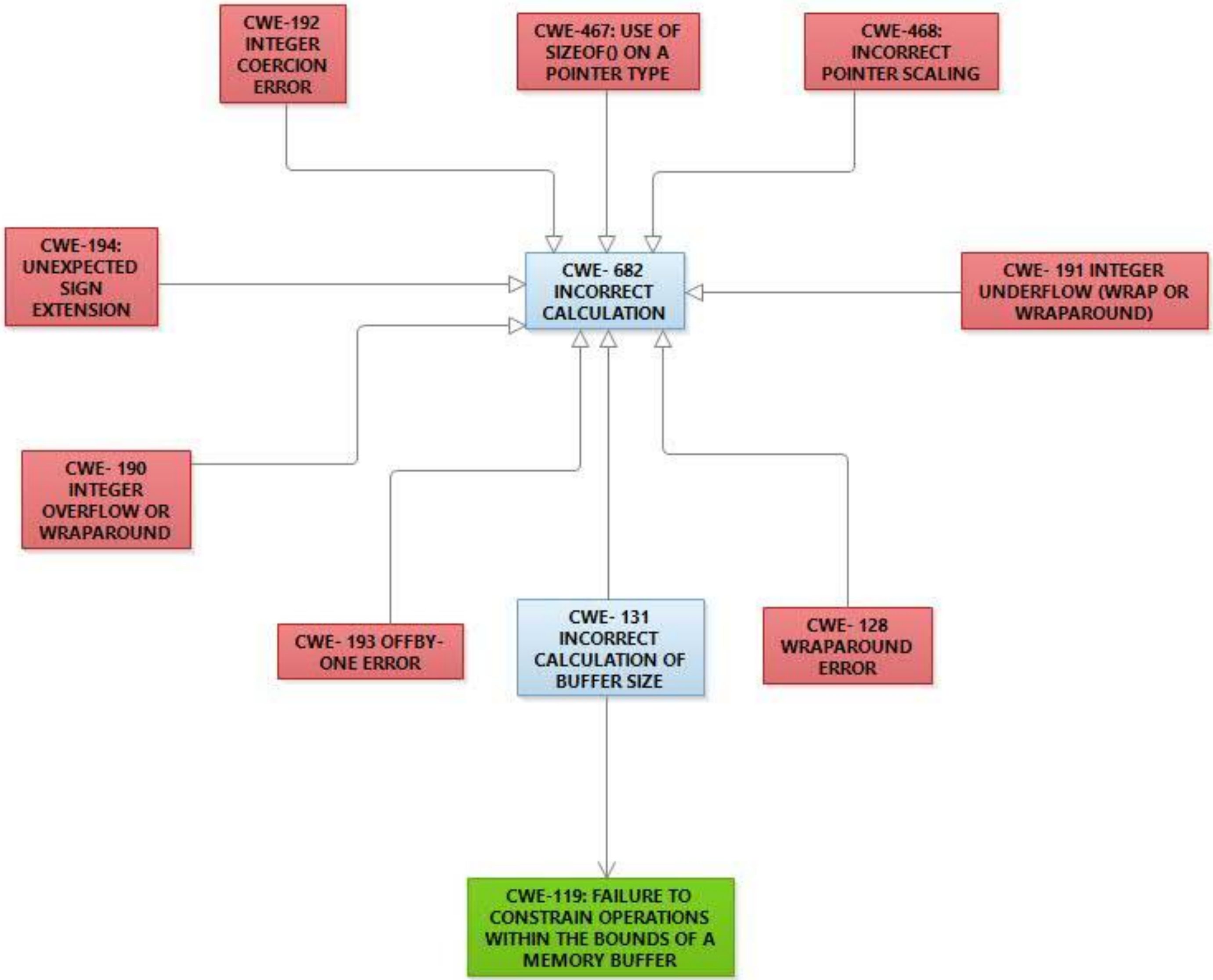
CWE- 124 BUFFER UNDERWRITE ('BUFFER UNDERFLOW')

CWE- 126 BUFFER OVER-READ

CWE- 127 BUFFER UNDER-READ

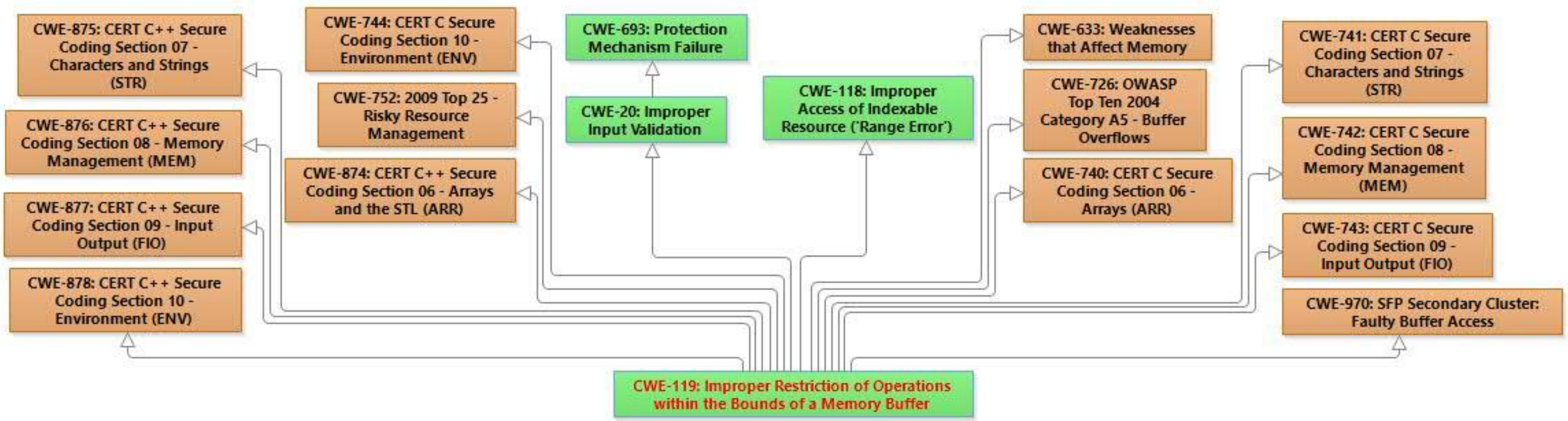
CWE- 785 USE OF PATH MANIPULATION FUNCTION WITHOUT MAX-SIZE BUFFER

CWE-119 Development View – Inheritance Relationships


CWE-192 INTEGER COERCION ERROR
CWE-467: USE OF SIZEOF() ON A POINTER TYPE
CWE-468: INCORRECT POINTER SCALING
CWE-194: UNEXPECTED SIGN EXTENSION
CWE- 682 INCORRECT CALCULATION
CWE- 191 INTEGER UNDERFLOW (WRAP OR WRAPAROUND)
CWE- 190 INTEGER OVERFLOW OR WRAPAROUND
CWE- 193 OFFBY-ONE ERROR
CWE- 131 INCORRECT CALCULATION OF BUFFER SIZE
CWE- 128 WRAPAROUND ERROR
CWE-119: FAILURE TO CONSTRAIN OPERATIONS WITHIN THE BOUNDS OF A MEMORY BUFFER

CWE-119 Development View – Precedence Relationships

CWE-119 Mental Model



CWE-875: CERT C++ Secure Coding Section 07 - Characters and Strings (STR)

CWE-876: CERT C++ Secure Coding Section 08 - Memory Management (MEM)

CWE-877: CERT C++ Secure Coding Section 09 - Input Output (FIO)

CWE-878: CERT C++ Secure Coding Section 10 - Environment (ENV)

CWE-744: CERT C Secure Coding Section 10 - Environment (ENV)

CWE-752: 2009 Top 25 - Risky Resource Management

CWE-874: CERT C++ Secure Coding Section 06 - Arrays and the STL (ARR)

CWE-693: Protection Mechanism Failure

CWE-20: Improper Input Validation

CWE-118: Improper Access of Indexable Resource ('Range Error')

CWE-633: Weaknesses that Affect Memory

CWE-726: OWASP Top Ten 2004 Category A5 - Buffer Overflows

CWE-740: CERT C Secure Coding Section 06 - Arrays (ARR)

CWE-741: CERT C Secure Coding Section 07 - Characters and Strings (STR)

CWE-742: CERT C Secure Coding Section 08 - Memory Management (MEM)

CWE-743: CERT C Secure Coding Section 09 - Input Output (FIO)

CWE-970: SFP Secondary Cluster: Faulty Buffer Access

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

CWE-119 Relationships - not complete yet
(extracted from CWE database XML representation)

CWEs Formal Specification (Step 2)

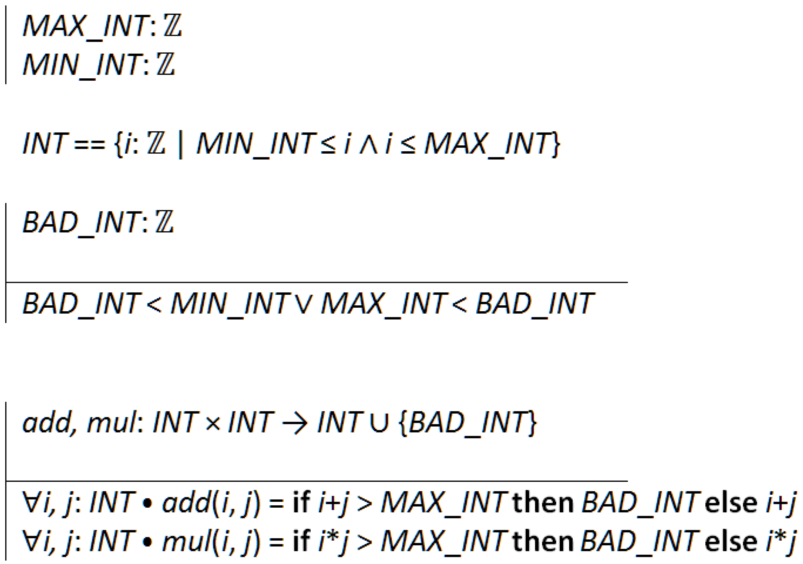
Second Research Step :

- CWEs formalization in Z notation
  → will remove ambiguities in their definitions
  → can lead to CWEs reclassification.
- Conversion of these Z notations in OCL specifications.

- UML representations – could be source to industry tools for code verification and tests generation (RSA, MS Visio).
- However, humans will not be able to use efficiently OCL, because of its complexity.
- Formalization in Z notation – open to theorem solvers and eventually to analyzers (e.g Alloy Analyzer).
- Z Word can be used for specification purposes and Z/Eves for verification.

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.”



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

$INT == \{i: \mathbb{Z} \mid MIN\_INT \leq i \land i \leq MAX\_INT\}$

$BAD\_INT: \mathbb{Z}$

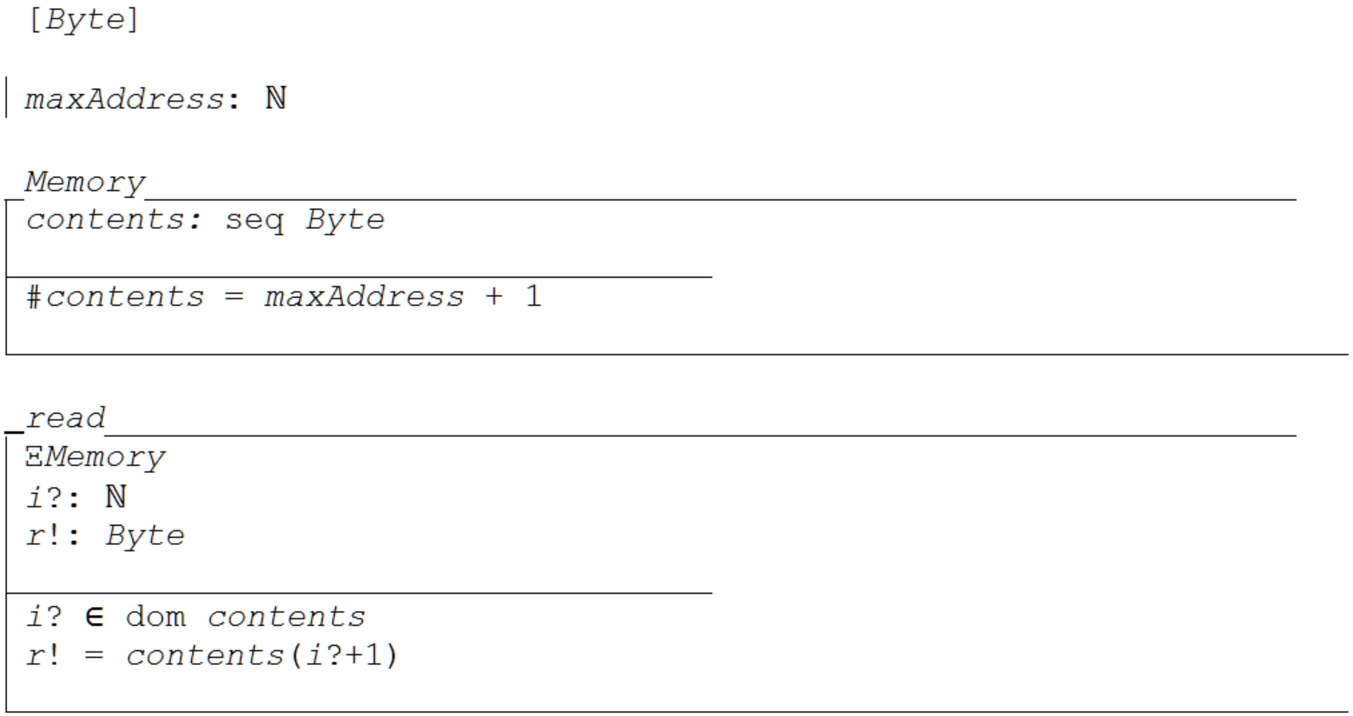
$BAD\_INT < MIN\_INT \lor MAX\_INT < BAD\_INT$

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

$\forall i, j: INT \bullet add(i, j) = \textbf{if } i+j > MAX\_INT \textbf{ then } BAD\_INT \textbf{ else } i+j$
$\forall i, j: INT \bullet mul(i, j) = \textbf{if } i*j > MAX\_INT \textbf{ then } BAD\_INT \textbf{ else } i*j$

CWE-119 in Z notation

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer: “The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.”



```
[Byte]

| maxAddress: ℕ

_Memory____________________________________________
| contents: seq Byte
|___________________________
| #contents = maxAddress + 1
|__________________________________________________

_read______________________________________________
| ΞMemory
| i?: ℕ
| r!: Byte
|___________________________
| i? ∈ dom contents
| r! = contents(i?+1)
|__________________________________________________
```

write

$\Delta Memory$

$i?: \mathbb{N}$

$w?: Byte$

$i? \in \mathrm{dom}\ contents$

$contents' = (1..i?) \lhd contents \frown \langle w? \rangle \frown ((i?+2)..\#contents) \lhd contents$

Program

$Memory$

$m, n: \mathbb{N}$

$BufferSpace: \mathbb{P}\ \mathbb{N}$

$m \leq n \leq maxAddress$

$BufferSpace = m..n$

badRun

$\Delta Program$

$m?, n?: \mathbb{N}$

$m = m? \land n = n?$

$\exists i: \mathbb{N} \bullet i \notin BufferSpace \land$

$((\exists r: Byte \bullet read[i/i?, r/r!]) \lor (\exists w: Byte \bullet write[i/i?, w/w?]))$

CVE/ CAPECs Formal Specification (Step 3)

Third Research Step:

- CVEs/CAPECs specification in CSP.
- CVEs/CAPECs specification in UML activity diagrams or BPPMN diagrams

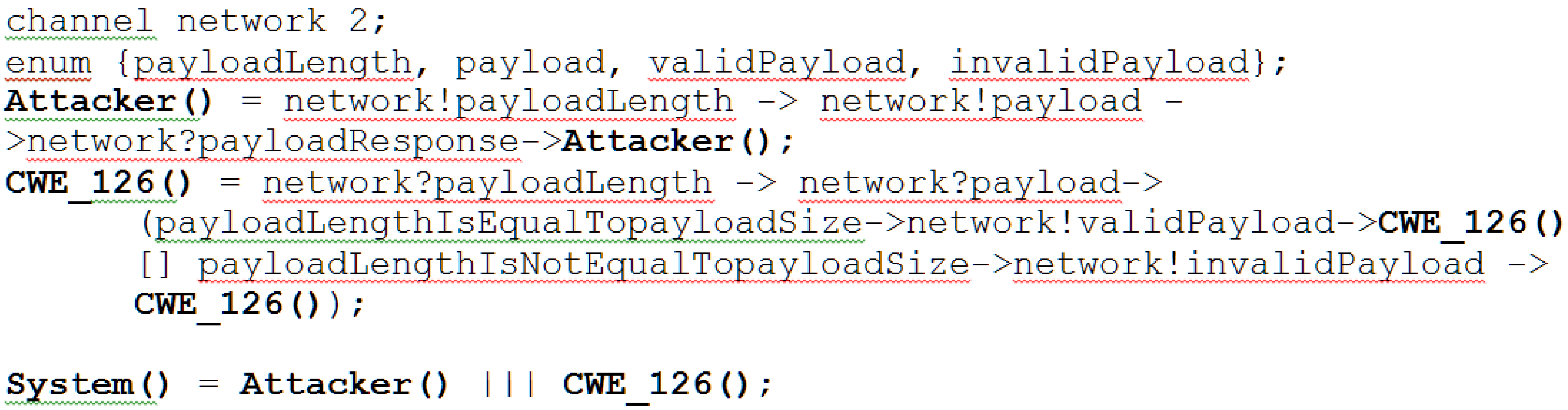
→ will remove ambiguities in CVEs definitions

→ can lead to CWEs reclassification

→ CAPECs could be assembled on higher levels.

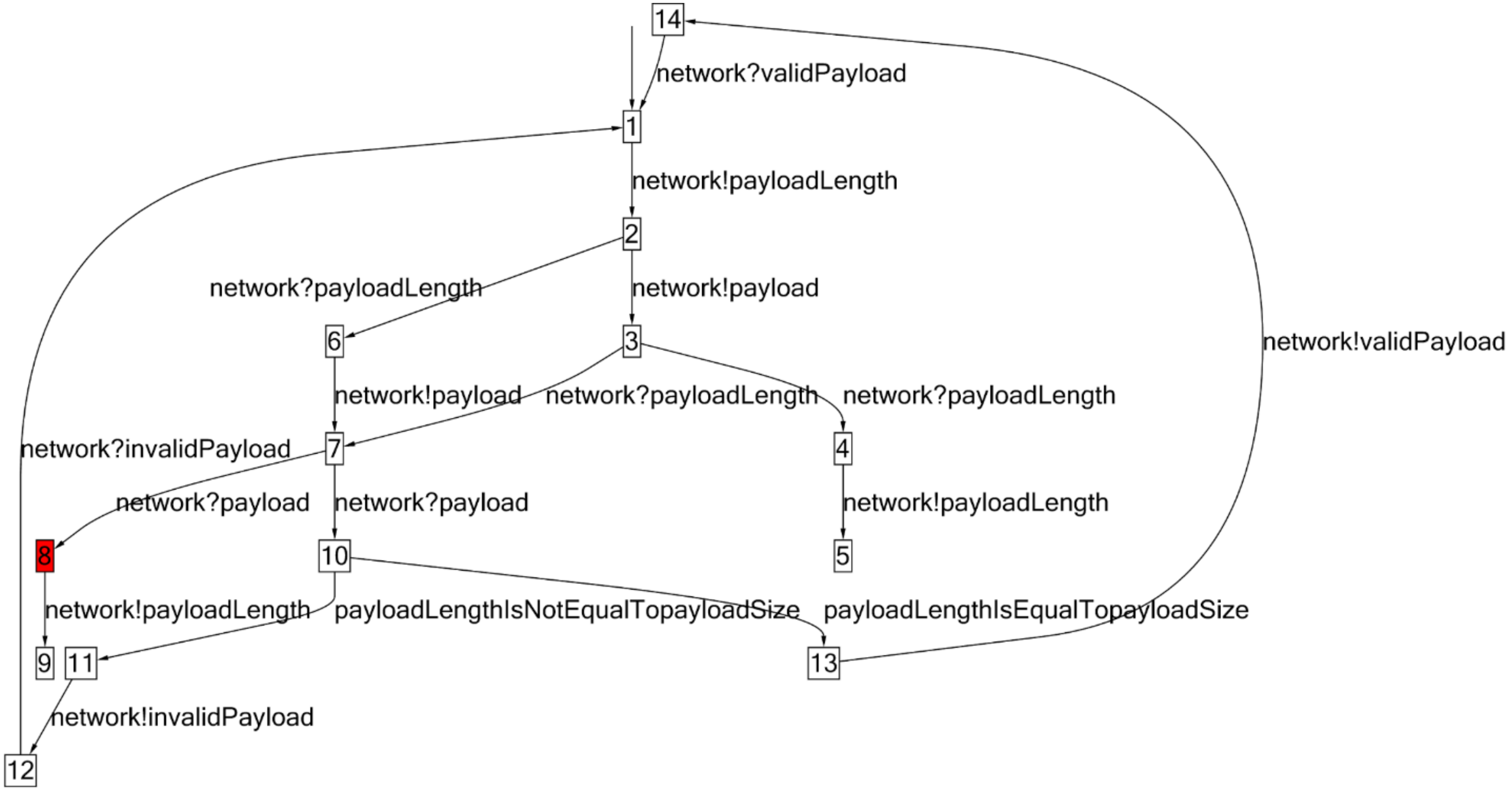
- Humans accept UML and BPMN diagrams better
- But UML and BPMN diagrams could be not too precise,
  as they are open to many platforms and programming languages.
- UML activity diagrams with OCL or to BPMN diagrams can be used for tests code generation.
- Z notation and CSP specifications can be used in theorem solvers for verification of dynamic software properties.

CVE-2014-160/CAPEC-540 in CSP



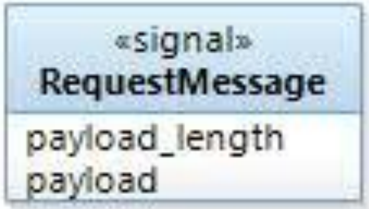
```
channel network 2;
enum {payloadLength, payload, validPayload, invalidPayload};
Attacker() = network!payloadLength -> network!payload -
>network?payloadResponse->Attacker();
CWE_126() = network?payloadLength -> network?payload->
      (payloadLengthIsEqualTopayloadSize->network!validPayload->CWE_126()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

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

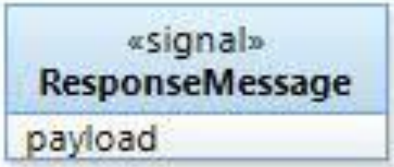
CVE-160/ CAPEC 540 SCP Simulation



CVE 160/CAPEC 540 in UML & OCL

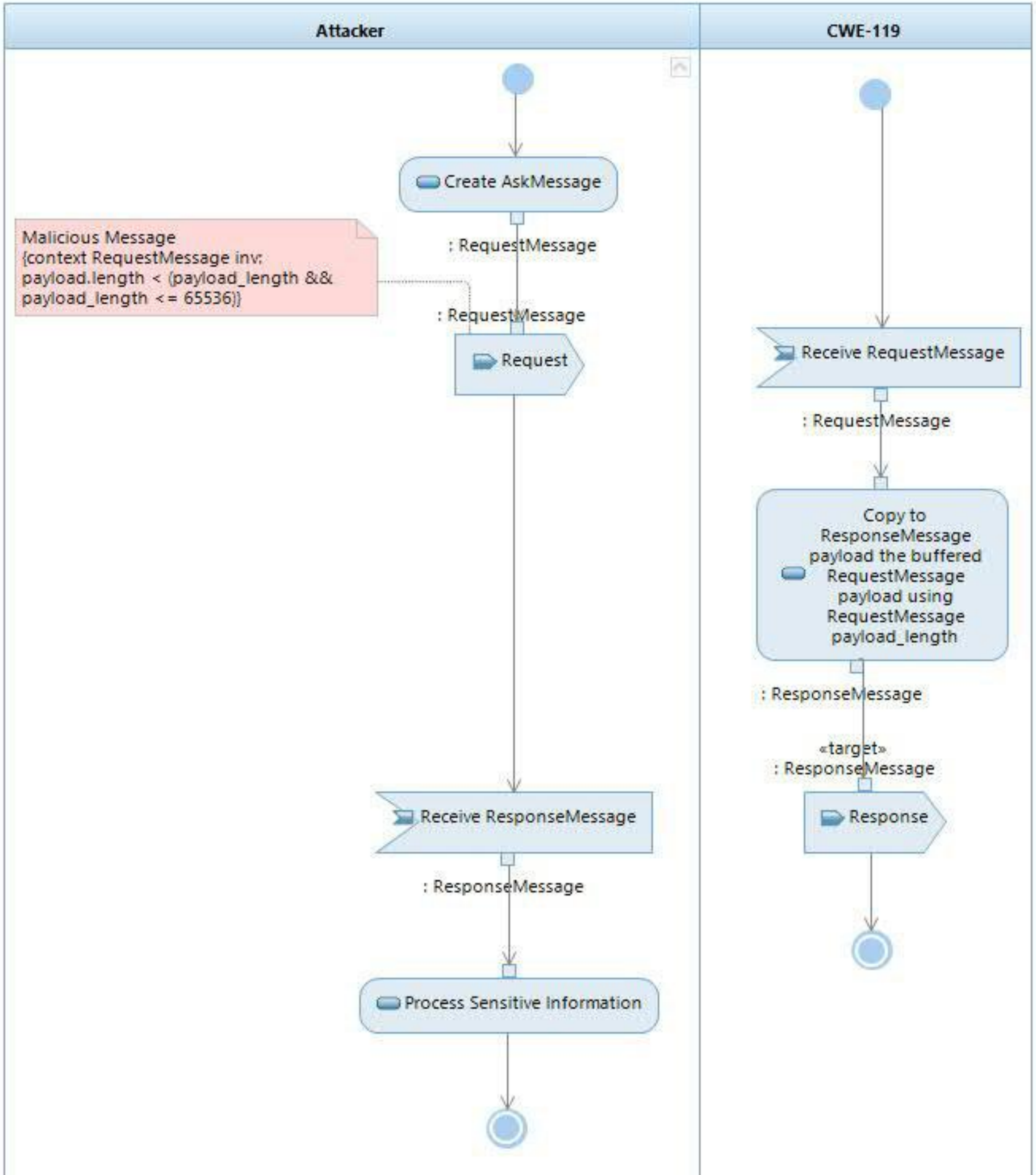
First, the two signals for a “heart beat” are defined in a class diagram.





The attribute types are not defined, because nowhere in corresponding CWEs or CVEs and CAPECs is mentioned anything about their types. They have to be C types as follows, independently of the attacker software:

```
payload_length: short;
payload: string;
```



Malicious Message
{context RequestMessage inv:
payload.length < (payload_length &&
payload_length <= 65536)}

Final Notes

- Pending:
  - Explore how OCL specifications can be used to connect the CWE Z-specification to the UML class diagram and the UML activity diagrams. → converting Z specifications into OCL.
  - Demonstrate how chains of CWEs can be specified with CAPECs in UML.

- There are industry efforts on legacy code reengineering. Their target are UML code representation for reuse and integration. UML is the point where CWEs, CVEs and CAPECs specifications can meet.