Towards a “Periodic Table” of Bugs

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

April 9, 2015

NIST, BGSU

Agenda

I. Our Vision (Why Use the Term "periodic table“)

Our vision is a “natural” organization of a catalog or dictionary or taxonomy to describe software weaknesses and vulnerabilities. Such an organization will help the community to:

a) more closely explain the nature of vulnerabilities (e.g. Heartbleed, Shellshock, Ghost, Chrome WebCore, etc.) and eventually detect, mitigate, or prevent them
b) more closely describe the classes of weaknesses that tools warnings cover (e.g. buffer overflow, injection, etc.)
c) eliminate the need for an exhaustive Cartesian product of weakness classes as in CWEs [1].

It may also help:

d) predict new classes of weaknesses and vulnerabilities
e) improve existing classifications.

Definition: Vulnerability

According to NIST Special Publication 800-27 A:

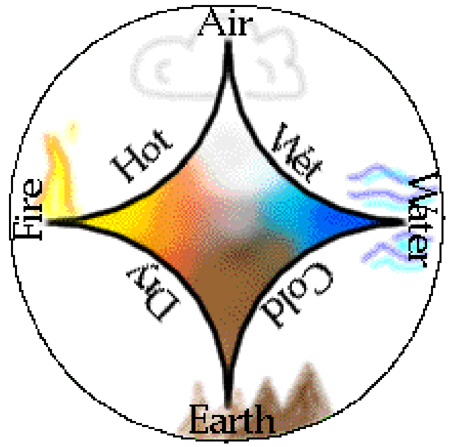
A *vulnerability* is "a weakness in system security requirements, design, implementation, or operation that could be accidentally triggered or intentionally exploited and result in a security failure" [2].

Towards Mendeleev’s Periodic Table

We use the term “periodic table” by analogy.

However obvious it seems today, it required extensive thought and investigation:

- Greeks used element and atom to name differences between materials and smallest parts of matter.
- In 330 BC, Aristotle proposed that everything is a mixture of “root elements”: Earth, Fire, Air, Water.
- In the Middle Ages, alchemists made lists of materials, such as alcohol, sulfur, mercury, and salt.
  - Lavoisier created a list of 33 elements – e.g. oxygen, nitrogen, hydrogen, phosphorus, mercury, zinc, sulfur, light, and caloric, and distinguished metals from non-metals.
  - Dalton realized “atoms of same element are identical in all respects, particularly weight."



Source: Reich Chemistry, http://reich-chemistry.wikispaces.com/Ancient%20Time%20LG

Mendeleev’s Periodic Table

Several tables of elements were developed in the 1800s (Fig. 1).

- De Chancourtois first noticed periodicity of elements. When ordered by their atomic weights, similar elements occur at regular intervals.
- Mendeleev’s Periodic Table in 1869 and his forecast of properties of missing elements reflected the century of growth in knowledge that reflects atomic structure.
  - ❖ Columns correspond to the number of electrons in the outer shell and the fundamental chemical properties
  - ❖ Rows correspond to the number of electron shells.

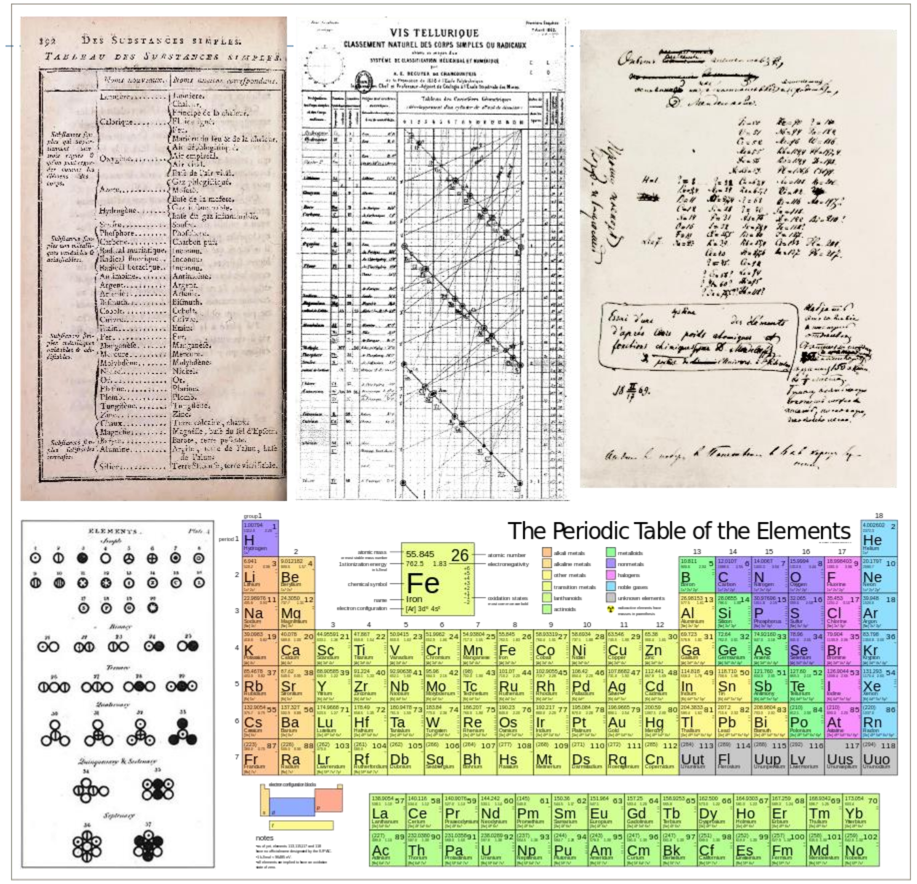

Des Substances simples.
Tableau des Substances simples.
VIS TELLURIQUE
The Periodic Table of the Elements
ELEMENTS.

Figure 1. Historic development documents of modern periodic table (clockwise from top left) - Lavoisier's 'Table of Simple substances'; de Chancourtois' 'Vis Tellurique'; Mendeleev's hand-written periodic table; a modern periodic table; John Dalton's list of atomic weights & symbols (Source: The History of the Periodic Table, http://allperiodictables.com/ClientPages/AAEpages/aaeHistory.html).

Other Organizational Structures in Science

Science has developed many different organizational structures:

- Linnaeus' Taxonomy –
  Categorizes living things into a hierarchy of:
  Domain (added recently), Kingdom, Phylum, Class, Order, Family, Genus, Species.

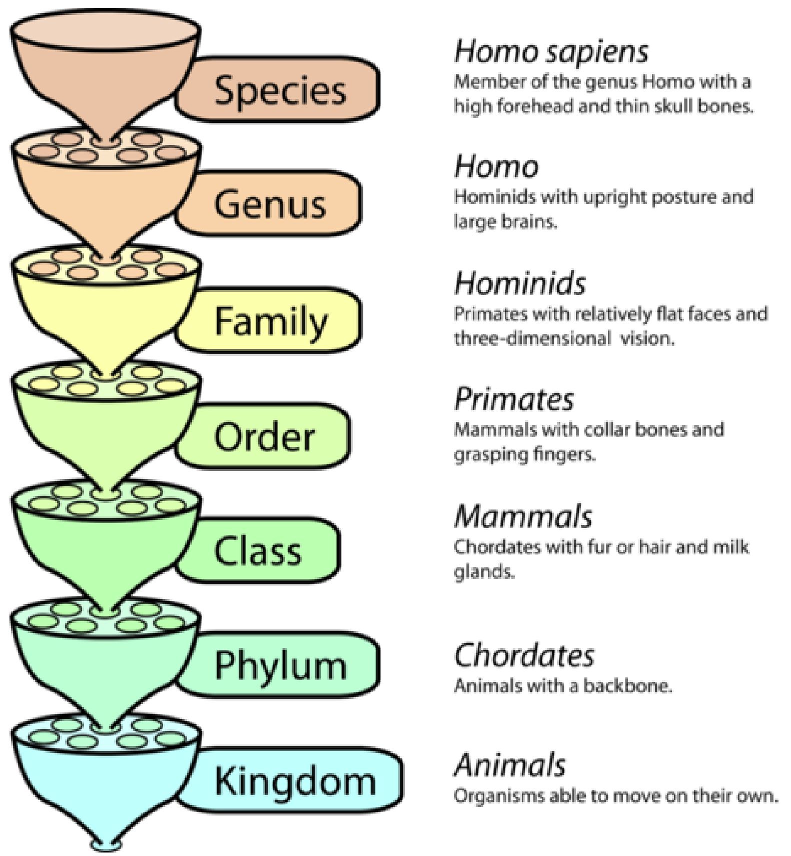
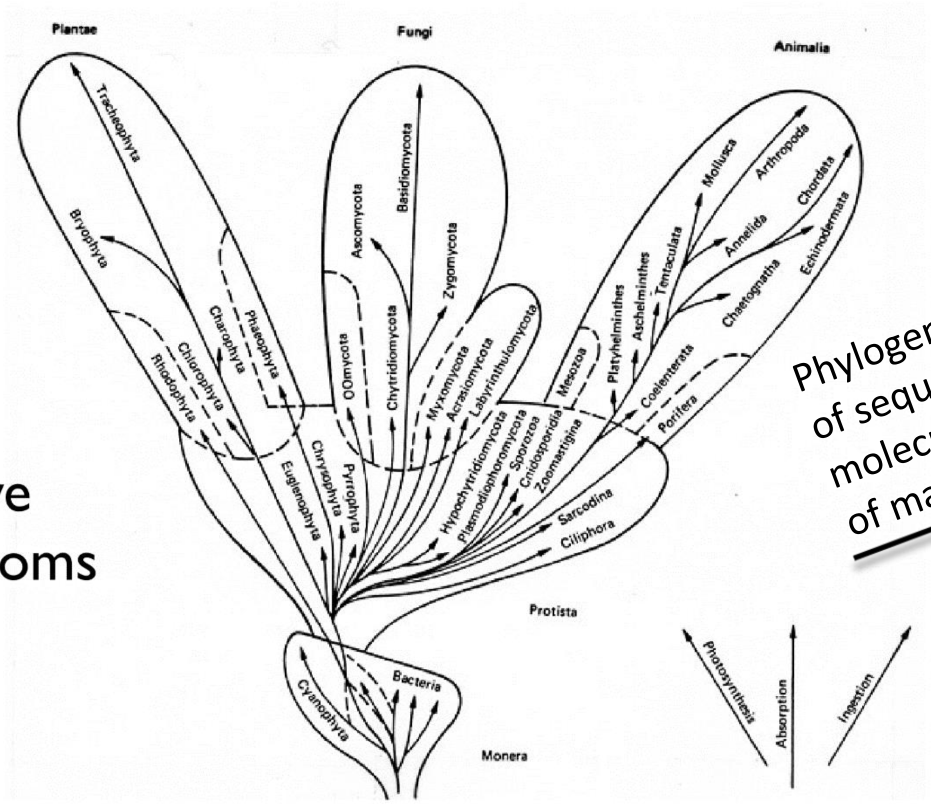

Species
Homo sapiens
Member of the genus Homo with a high forehead and thin skull bones.
Genus
Homo
Hominids with upright posture and large brains.
Family
Hominids
Primates with relatively flat faces and three-dimensional vision.
Order
Primates
Mammals with collar bones and grasping fingers.
Class
Mammals
Chordates with fur or hair and milk glands.
Phylum
Chordates
Animals with a backbone.
Kingdom
Animals
Organisms able to move on their own.

Figure 2. Applying Linnaean system to classify our own species, Homo sapiens.
(Source: c-K12, http://www.ck12.org/book/CK-12-Life-Science-For-Middle-School/section/2.3/)

Other Organizational Structures in Science (Cont.)

- Tree of Life –
  Division of life into three domains:
  Bacteria, Archaea, and Eukaryotes.

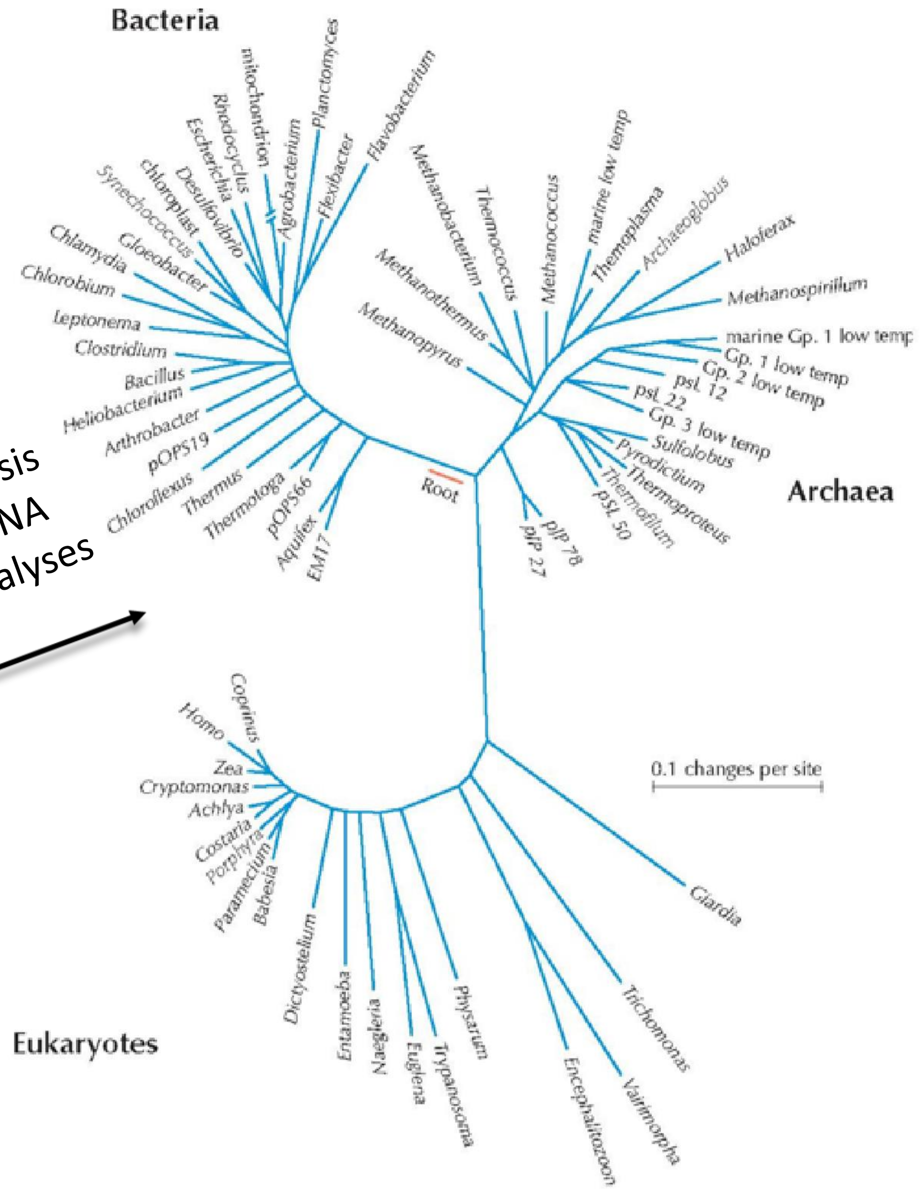
Five kingdoms



Whittaker 1967

Phylogenetic tree, based on analysis of sequences of small-subunit rRNA molecules and supported by analyses of many other characteristics.

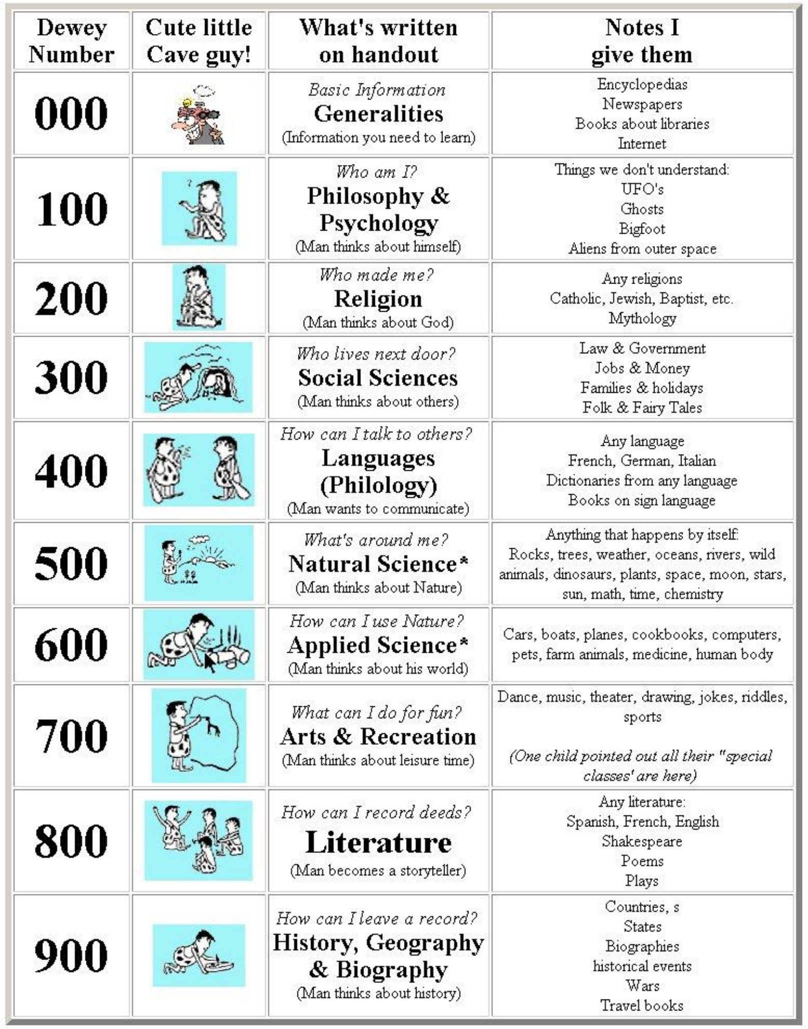
Three domains



Other Organizational Structures in Science (Cont.)

- Dewey Decimal Classification system – Allows new books and whole new subjects to be placed in reasonable locations in a library, for easy retrieval based on subject (Fig. 3).
- Fingerprints are classified and retrieved using: loops, whorls, and arches as basic patterns.

Source: Wikipedia, https://en.wikipedia.org/?title=Fingerprint

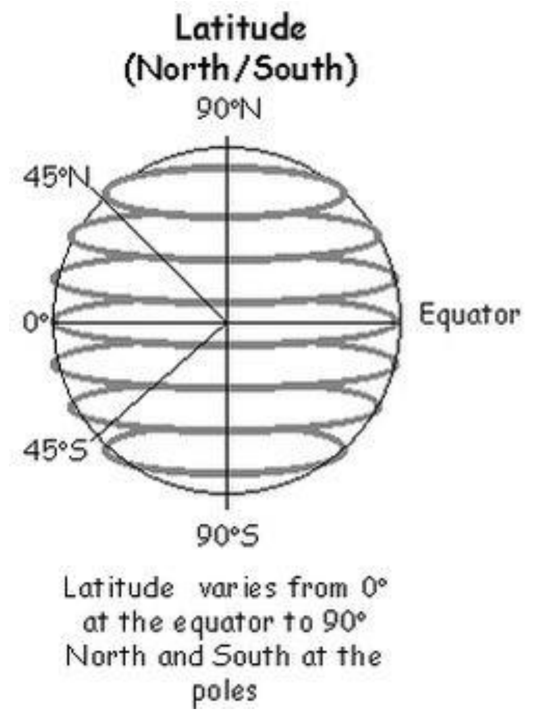


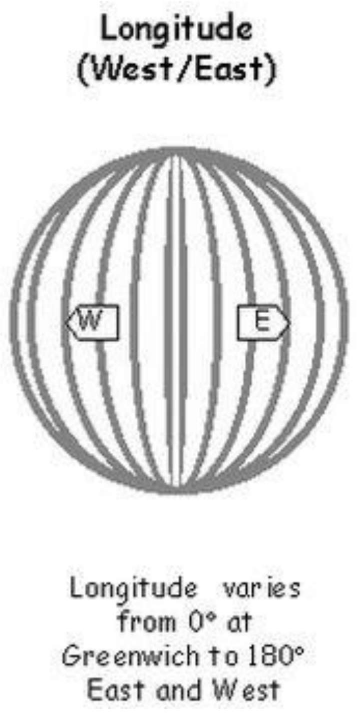
Dewey Number	Cute little Cave guy!	What's written on handout	Notes I give them
000		*Basic Information* **Generalities** (Information you need to learn)	Encyclopedias Newspapers Books about libraries Internet
100		*Who am I?* **Philosophy & Psychology** (Man thinks about himself)	Things we don't understand: UFO's Ghosts Bigfoot Aliens from outer space
200		*Who made me?* **Religion** (Man thinks about God)	Any religions Catholic, Jewish, Baptist, etc. Mythology
300		*Who lives next door?* **Social Sciences** (Man thinks about others)	Law & Government Jobs & Money Families & holidays Folk & Fairy Tales
400		*How can I talk to others?* **Languages (Philology)** (Man wants to communicate)	Any language French, German, Italian Dictionaries from any language Books on sign language
500		*What's around me?* **Natural Science*** (Man thinks about Nature)	Anything that happens by itself: Rocks, trees, weather, oceans, rivers, wild animals, dinosaurs, plants, space, moon, stars, sun, math, time, chemistry
600		*How can I use Nature?* **Applied Science*** (Man thinks about his world)	Cars, boats, planes, cookbooks, computers, pets, farm animals, medicine, human body
700		*What can I do for fun?* **Arts & Recreation** (Man thinks about leisure time)	Dance, music, theater, drawing, jokes, riddles, sports *(One child pointed out all their "special classes' are here)*
800		*How can I record deeds?* **Literature** (Man becomes a storyteller)	Any literature: Spanish, French, English Shakespeare Poems Plays
900		*How can I leave a record?* **History, Geography & Biography** (Man thinks about history)	Countries, s States Biographies historical events Wars Travel books

Figure 3. Categories of Dewey Decimal Classification System.
(Source: AIS, http://www.ais.up.ac.za/vet/infomania/infomania14/dewey14.htm)

Other Organizational Structures in Science (Cont.)

- Geographic Coordinate System –
  Specifies any location on Earth using: Latitude, Longitude, and Elevation.(Fig.4).


Latitude
(North/South)
90°N
45°N
0°
Equator
45°S
90°S
Latitude varies from 0° at the equator to 90° North and South at the poles



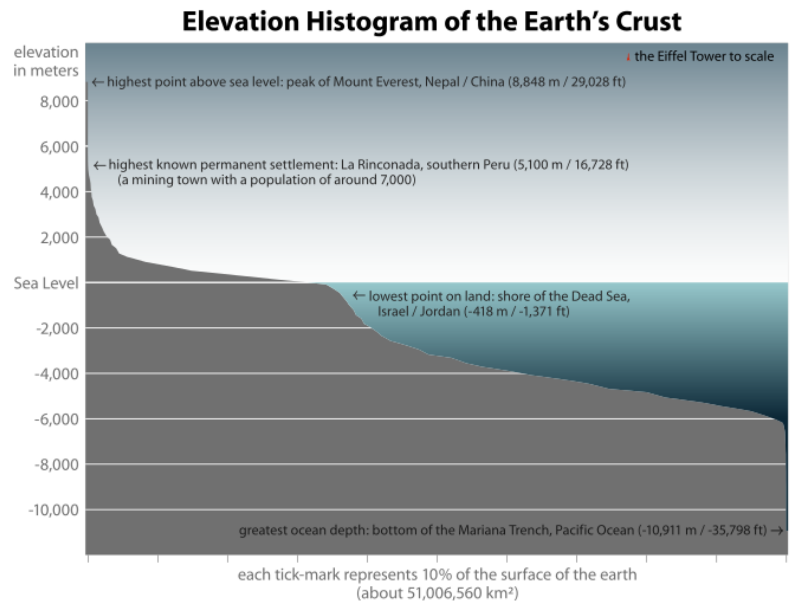


Figure 4. Longitude lines are perpendicular and latitude lines are parallel to the equator.
(Sources: Wikipedia, http://en.wikipedia.org/wiki/Geographic_coordinate_system ; http://en.wikipedia.org/wiki/Elevation)

Other Organizational Structures in Science (Cont.)

- Medical professionals have extension vocabulary to name all muscles, bones, and organs, and conditions and diseases, so they can communicate clearly.

  For instance, the image caption uses some obscure/precise medical terminology.

  - They are not trying to obfuscate.
  - They are "painting a picture" (adding arrows and circles) with words.

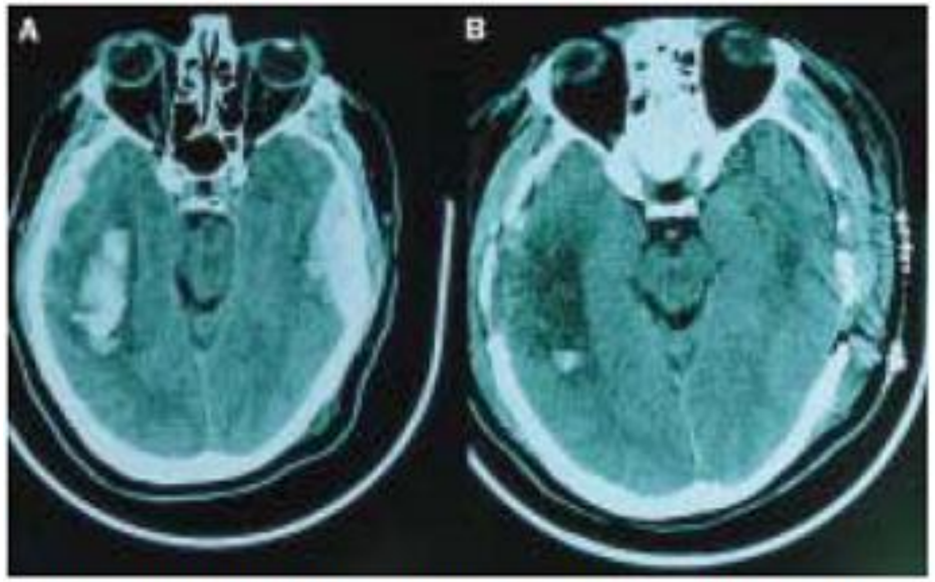


Figure 2: **Computed tomography of a comatose patient with a left temporal epidural haematoma, right parenchymal temporal lobe haematoma, and a right convexity subdural haematoma before and after craniotomy and evacuation of haematomas**

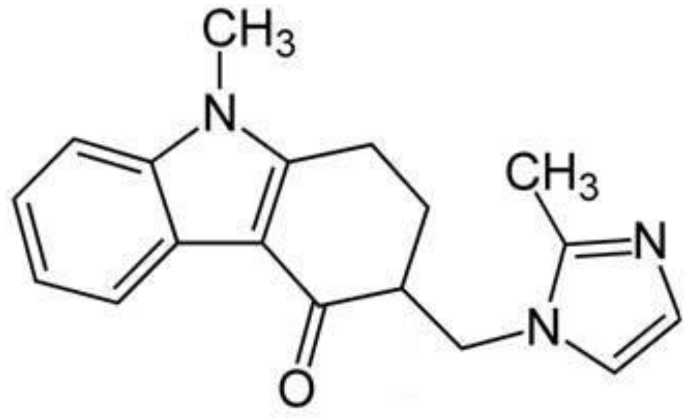
(Source: http://i.stack.imgur.com/uLH9P.jpg)

Other Organizational Structures in Science (Cont.)

- Chemists have a detailed system beyond the periodic table to describe chemicals.

For instance, Zofran ODT is:

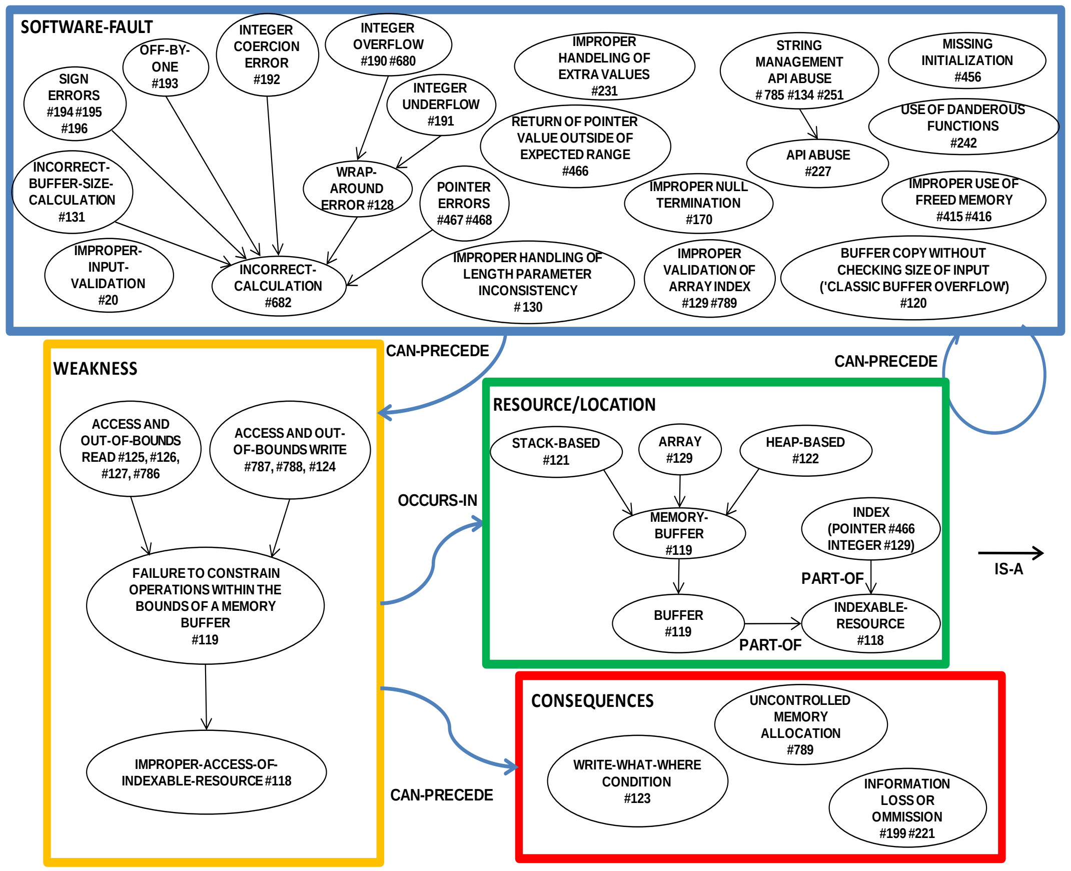
$C_{18}H_{19}N_3O$
or
(±) 1, 2, 3, 9-tetrahydro-9-methyl-3-[(2-methyl-1H-imidazol-1-yl)methyl]-4H-carbazol-4-one.



Analogously, we seek to:
Factor software weaknesses into their constituent components.

Problem: Existing Classifications Must Be Improved

- Common Weakness Enumerations (CWE) [1] are:
  - not orthogonal
  - coarse-grained.
- Software Fault Patterns (SFP) [3] don't include:
  - attacks, upstream influences, or consequences
- Semantic Templates (ST) [4] are:
  - only general interactions.



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
STRING MANAGEMENT API ABUSE # 785 #134 #251
MISSING INITIALIZATION #456
USE OF DANDEROUS FUNCTIONS #242
RETURN OF POINTER VALUE OUTSIDE OF EXPECTED RANGE #466
API ABUSE #227
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

Figure 5. Buffer Overflow ST. (Source: Yan Wu's dissertation.)

Solution: A Formal Orthogonal "Periodic Table" of Bugs

A "natural" organization of a catalog or dictionary or taxonomy to describe software weaknesses and vulnerabilities. It will help the community to:

a) more closely explain the nature of vulnerabilities (e.g. Heartbleed, Shellshock, Ghost, Chrome WebCore, etc.) and eventually detect, mitigate, or prevent them

b) more closely describe the classes of weaknesses that tools warnings cover (e.g. buffer overflow, injection, etc.)

c) eliminate the need for an exhaustive Cartesian product of weakness classes as in CWEs [1].

It may also help:

d) predict new classes of weaknesses and vulnerabilities

e) improve existing classifications.

II. Taxonomy and (Formal) Meanings

We refined and extended the structures based on:

- Common Weaknesses Enumeration (CWEs) & the notions of chains and composites
- Software Fault Patterns (SFPs)
- Semantic Templates.

Focus First On: Buffer Overflow

- 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.

  → "Read from or write to a memory location" is not tied to the buffer. Our definition clarifies that access is through the same buffer to which the intended boundary pertains. Our definition also accurately, precisely, and concisely describes violation of memory safety.

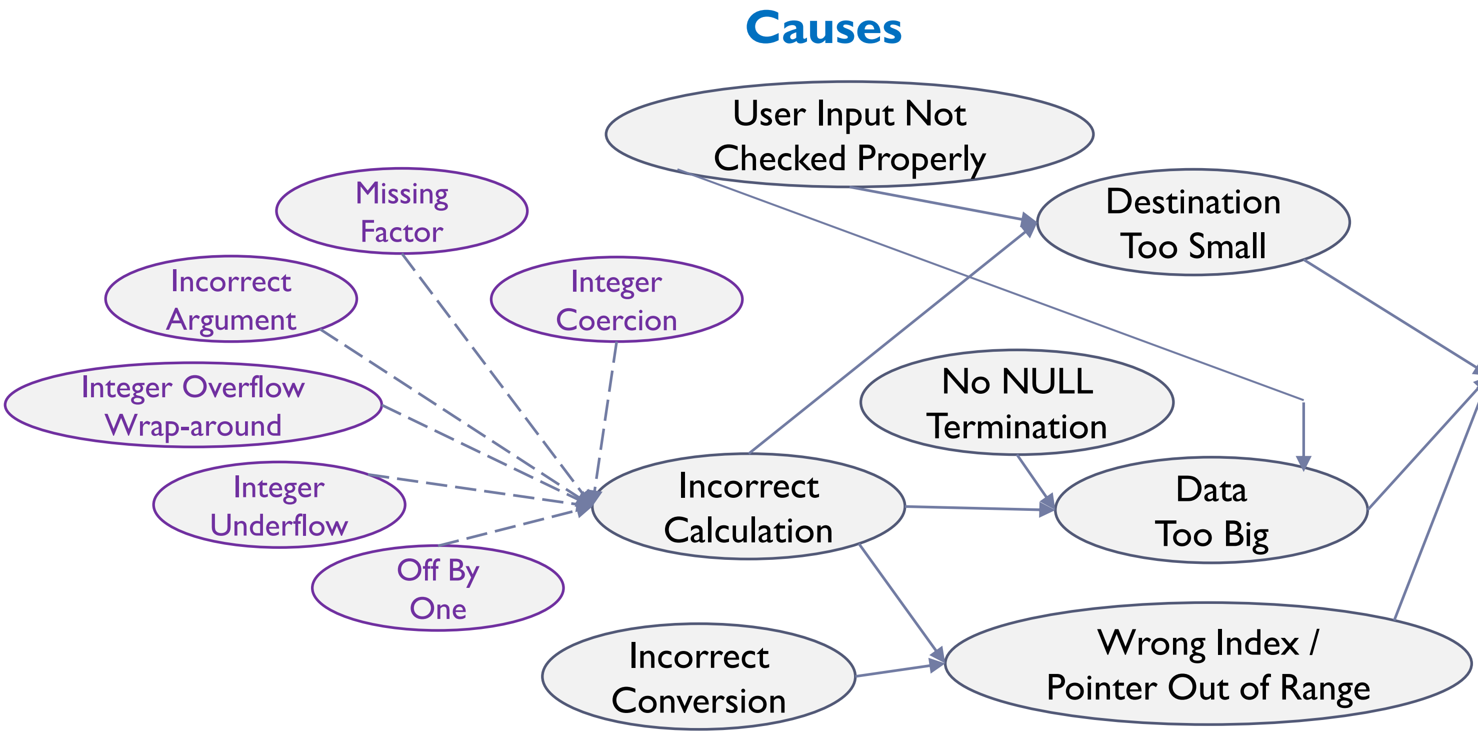
- *Our Definition*: The software can access through a buffer a memory location that is not allocated to that buffer.

Buffer Overflow: Attributes

- Segment (memory area):
  - ➢ Heap, Stack, BSS (uninitialized data), Data (initialized), Code (text) [5,6,3].
- Access:
  - ➢ Read, Write. [5,3].
- Side:
  - ➢ Below (before or under), Above (after or over) [5].
- Method:
  - ➢ Indexed, (bare) Pointer [5,3].
- Magnitude (how far outside):
  - ➢ Minimal (just barely outside), Moderate, Far (e.g. 4000) [5].
- Data Size (base may be inside, but large chunk of data extends outside).

Note: Any of these attributes may be "Unknown", "Any", or "Don't Care".

Buffer Overflow: Causes



Buffer Overflow

Attributes:

- Access:
  - ✓*Read*, *Write*.
- Side:
  - ✓*Below* (before or under), *Above* (after or over)
- Segment (memory area):
  - ✓*Heap*, *Stack*, *BSS* (uninitialized data), *Data* (initialized), *Code* (text
- Method:
  - ✓*Indexed*, (bare) *Pointer*.
- Magnitude (how far outside):
  - ✓*Minimal* (just barely), *Moderate*, *Far* (e.g. 4000).
- Data Size (base may be inside, but large chunk of data extends outside).

➢ There are only 3 proximate causes of buffer overflows:

- Destination is too small
- Data is too big
- Wrong index / pointer out of range.

➢ Some of the preceding causes that may lead to those.

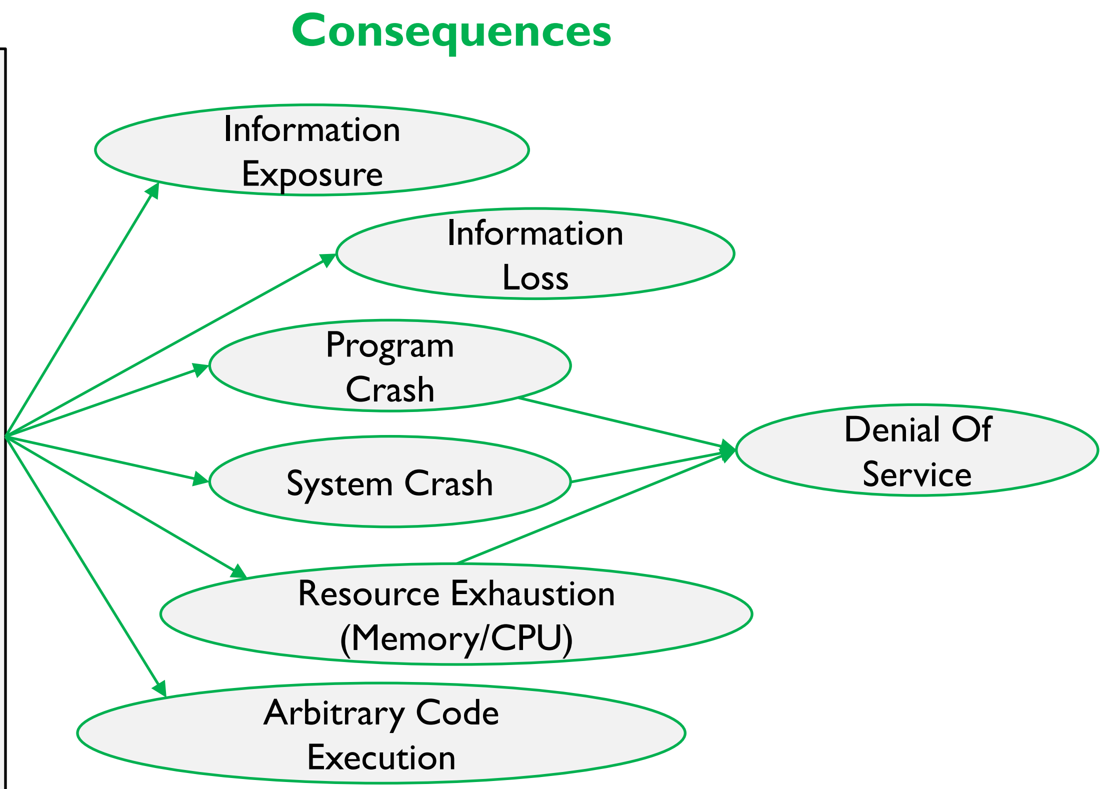
- - ▸	means "is-a"
──▸	means "can precede".

Buffer Overflow: Consequences

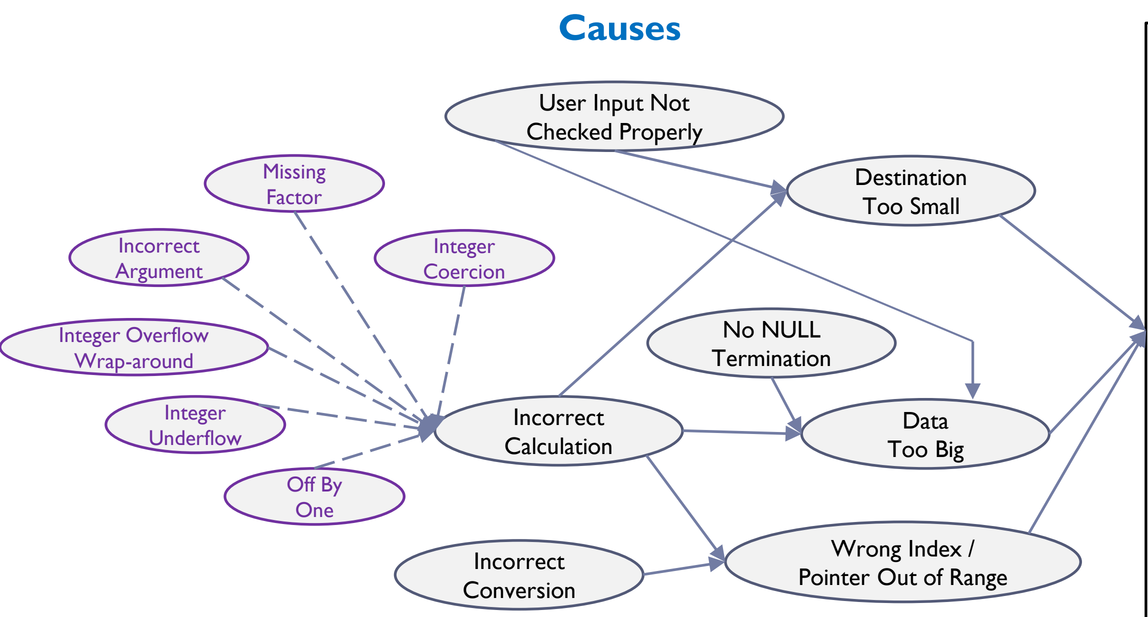
Buffer Overflow

Attributes:

- Access:
  - ✓ *Read*, *Write*.
- Side:
  - ✓ *Below* (before or under), *Above* (after or over)
- Segment (memory area):
  - ✓ *Heap*, *Stack*, *BSS* (uninitialized data), *Data* (initialized), *Code* (text)
- Method:
  - ✓ *Indexed*, (bare) *Pointer*.
- Magnitude (how far outside):
  - ✓ *Minimal* (just barely), *Moderate*, *Far* (e.g. 4000).
- Data Size (base may be inside, but large chunk of data extends outside).



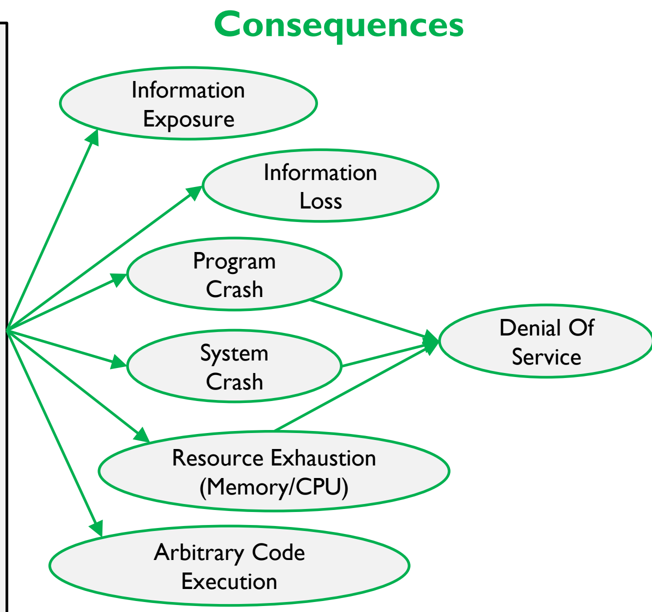
Buffer Overflow: Attributes, Causes & Consequences



Buffer Overflow

Attributes:

- Access:
  - ✓*Read*, *Write*.
- Side:
  - ✓*Below* (before or under), *Above* (after or over)
- Segment (memory area):
  - ✓*Heap*, *Stack*, *BSS* (uninitialized data), *Data* (initialized), *Code* (text)
- Method:
  - ✓*Indexed*, (bare) *Pointer*.
- Magnitude (how far outside):
  - ✓*Minimal* (just barely), *Moderate*, *Far* (e.g. 4000).
- Data Size (base may be inside, but large chunk of data extends outside).



The graph of causes shows:

- ➢ There are only 3 proximate causes of buffer overflows:
  - Destination is too small
  - Data is too big
  - Wrong index / pointer out of range.
- ➢ Some of the preceding causes that may lead to those.

Note: In the graph of causes:

$\dashrightarrow$ means "is-a"

$\longrightarrow$ means "can precede".

III. Examples on Applying Our Techniques

- CVE (Common Vulnerabilities and Exposures,) is a dictionary of security vulnerabilities.
  → We will demonstrate the use of our techniques for describing some CVEs.

- CppCheck is a static analysis tool [7]
  → We will demonstrate the use of our techniques for analysis of cppCheck warning classes.

- We will also demonstrate characterization of buffer overflow CWEs.

Example 1: Ghost (CVE-2015-0235)

CVE-2015-0235 is "Heap-based buffer overflow in the __nss_hostname_digits_dots function in glibc 2.2, and other 2.x versions before 2.18, allows context-dependent attackers to execute arbitrary code via vectors related to the (1) gethostbyname or (2) gethostbyname2 function, aka "GHOST."" [8,9].

Applying our techniques, we obtain:

Ghost — glibc gethostbyname buffer overflow is

- caused by a *Destination Too Small*
- because of an *Incorrect Calculation* specifically *Missing Factor*
- where there was a *Write* that was *After* the end by a *Moderate* number of bytes
- of a buffer in the *Heap*
- which may be exploited for *Arbitrary Code Execution*.

Example 2: Chrome WebCore (CVE-2010-1773)

CVE-2010-1773 is "Off-by-one error in the toAlphabetic function in rendering/RenderListMarker.cpp in WebCore in WebKit before r59950, as used in Google Chrome before 5.0.375.70, allows remote attackers to obtain sensitive information, cause a denial of service (memory corruption and application crash), or possibly execute arbitrary code via vectors related to list markers for HTML lists, aka rdar problem 8009118." [10]

Applying our techniques we obtain:

Chrome WebCore — toAlphabetic render buffer overflow is

- caused by a *Wrong Index*
- because of an *Incorrect Calculation* specifically *Off by One*
- where there was a *Read* that was *Below* the start by a *Minimal* amount
- of a buffer in the *Heap*
- which leads to use of *User Input Not Checked Properly*
- which may be exploited for *Information Exposure*, *Arbitrary Code Execution*, or *Program Crash* leading to *Denial of Service*.

Example 3: Heartbleed (CVE-2014-0160)

- CVE-2014-0160 is: "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."[11].
- Applying our techniques, we obtain:
- 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*

Example 3: Heartbleed (CVE-2014-0160) (cont.)

- Information Exposure is also enabled by CWE-244: Improper Clearing of Heap Memory Before Release [14], and CWE-908: Use of Uninitialized Resource [15].

Example 4: cppCheck Warning Classes

CppCheck is a static analysis tool [7]. Table 1 provides descriptions of its warning classes.

Table 1. Analysis of cppCheck warning classes.

Warning\Attribute	Access	Side	Indexed	Size	Magnitude
array Index Out Of Bounds	-	-	Yes	-	-
buffer Access Out Of Bounds	-	-	-	-	-
out Of Bounds	-	-	-	-	-
negative Index	-	Below	Yes	-	-
insecure Cmd Line Args	-	-	-	-	-
write Outside Buffer Size	Write	-	-	-	-
invalid Scanf	Write	Above	-	Variable	Moderately outside

Example 5: Refactoring CWEs

Applying our definition and attributes, Buffer Overflow CWEs can be categorized as follows.

Buffer Overflow CWEs:

CWE 120: Write beyond buffer end.
CWE 121: Write outside buffer that is on stack.
CWE 122: Write outside buffer that is on heap.
CWE 123: Write outside buffer.
CWE 124: Write before start of buffer.
CWE 125: Read outside buffer.
CWE 126: Read after end of buffer.
CWE 127: Read before start of buffer.
CWE 786: Access before start of buffer.
CWE 787: Write outside buffer.
CWE 788: Access after end of buffer.

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

IV. Next Steps

- Provide more examples of applying our techniques
- Define more "vocabulary" – add terms, more formal, refine
- Focus on other CWEs – for example:
  - Improper Restriction of Excessive Authentication Attempts (CWE-307)
  - OS Command Injection (CWE-78 Improper Neutralization of Special Elements used in an OS Command).

Focus On: Injection

- CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'):
  The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.

  → "Using input", "intended command", and "correctly neutralizing" are imprecise. Our definition precisely defines "using input" and "intended command". We do not include "correctly neutralizing", because it simply means that intended OS command cannot be modified.

- *Our Definition*: For a common trusted input and two untrusted inputs, the sub-sequences of code symbols in the output program differ in a way that is not included in a description of a given syntax of allowed different sequences.

Focus On: Authentication

- CWE-307: Improper Restriction of Excessive Authentication Attempts:
  The software does not implement sufficient measures to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks.

- → "Multiple" and "short" are vague. Our definition recognizes that CWE-307 actually represents a set of weaknesses, each of which satisfies particular institution-specific definitions of "multiple" and "short".

- *Our Definition*: The software does not limit the number of failed authentication attempts or may allow more than a specified number of failed authentication attempts within a specified time period.

V. Conclusion

- This presentation outlined the progress we have made towards better understanding of software weaknesses and their:
  - definitions
  - causes
  - consequences.
- We hope that such progress will result in being able to:
  - write more secure software
  - improve tools that find weaknesses in code.

References

[1] The MITRE Corporation, CWE Common Weakness Enumeration, https://cwe.mitre.org/, viewed 2 April 2015.

[2] Stoneburner, G. et al., "Engineering Principles for Information Systems Security (A Baseline for Achieving Security)", Revision A, NIST Special Publication 800-27 Rev A, June 2004.

[3] Nikolai Mansourov, DoD Software Fault Patterns, https://buildsecurityin.us-cert.gov/sites/default/files/Mansourov-SoftwareFaultPatterns.pdf viewed on June 3, 2015, listed in CWE, Common Weakness Enumeration, Sources, https://cwe.mitre.org/about/sources.html viewed on June 3, 2015.

[4] Yan Wu, Robin A Gandhi, and Harvey Siy, "Using semantic templates to study vulnerabilities recorded in large software repositories", Proc. 2010 ICSE Workshop on Software Engineering for Secure Systems, pp 22-28.

[5] Kendra Kratkiewicz and Richard Lippmann, Using a Diagnostic Corpus of C Programs to Evaluate Buffer Overflow Detection by Static Analysis Tools. 2005 Workshop on the Evaluation of Software Defect Detection Tools 2005,June 12, Chicago, IL. https://www.cs.umd.edu/~pugh/BugWorkshop05/papers/62-kratkiewicz.pdf viewed on April 23, 2015

References (cont.)

[6] WIKIPEDIA, The Free Encyclopedia, Data segment https://en.wikipedia.org/wiki/Data_segment viewed on April 27, 2015

[7] ccpcheck.xml, SAMATE, NIST.

[8] CVE-2015-0235, MITRE, CVE, Common Vulnerabilities and Exposures, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0235 accessed April 23, 2015.

[9] Qualys Security Advisory, Qualys Security Advisory CVE-2015-0235 - GHOST: glibc gethostbyname buffer overflow, Openwall, bringing security into open environments, http://www.openwall.com/lists/oss-security/2015/01/27/9, viewed 9 April 2015.

[10] CVE-2010-1773, MITRE, CVE, Common Vulnerabilities and Exposures, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1773 accessed April 27, 2015.

References (cont.)

[11] CVE-2014-0160, MITRE, Common Vulnerabilities and Exposures, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160 viewed on July 16, 2015.

[12] Kupsch & Miller https://continuousassurance.org/swamp/SWAMP-Heartbleed.pdf viewed on July 16, 2015.

[13] http://securityintelligence.com/heartbleed-openssl-vulnerability-what-to-do-protect/#.VagHQXbD-fA viewed on July 16, 2015.

[14] https://cwe.mitre.org/data/definitions/244.html viewed on July 20, 2015

[15] https://cwe.mitre.org/data/definitions/908.html viewed on July 20, 2015