

# SARD: A Software Assurance Reference Dataset

Paul E. Black

SAMATE Project

Software Quality Group

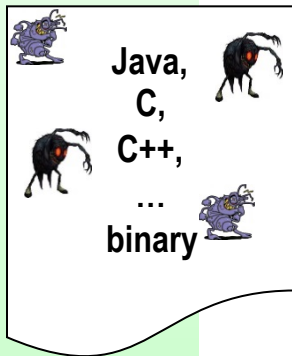
Software & Systems Division

NIST

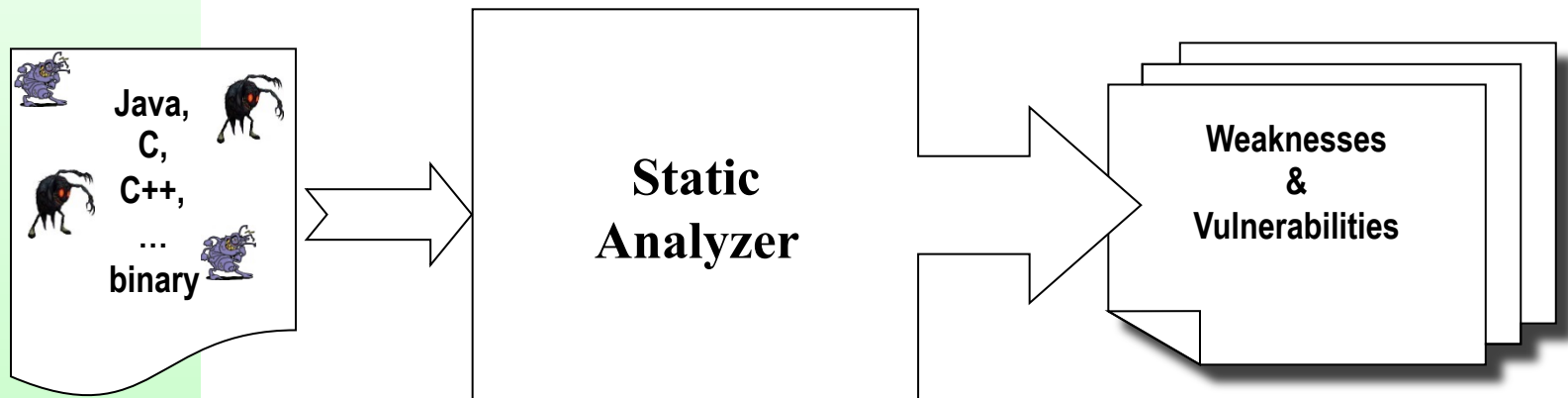


**Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology (NIST), nor does it imply that the products are necessarily the best available for the purpose.**

# What is Static Analysis?

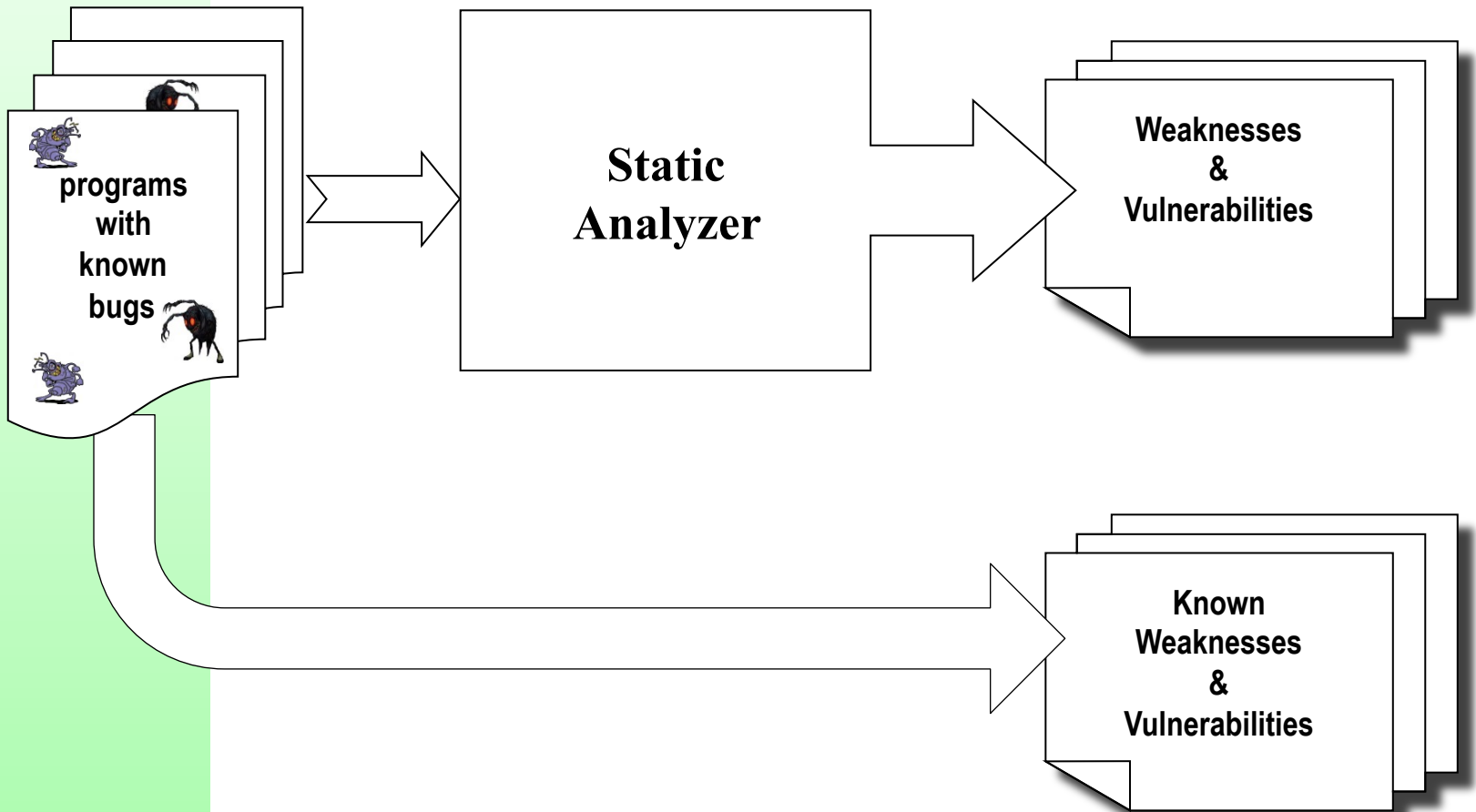


# What is Static Analysis?



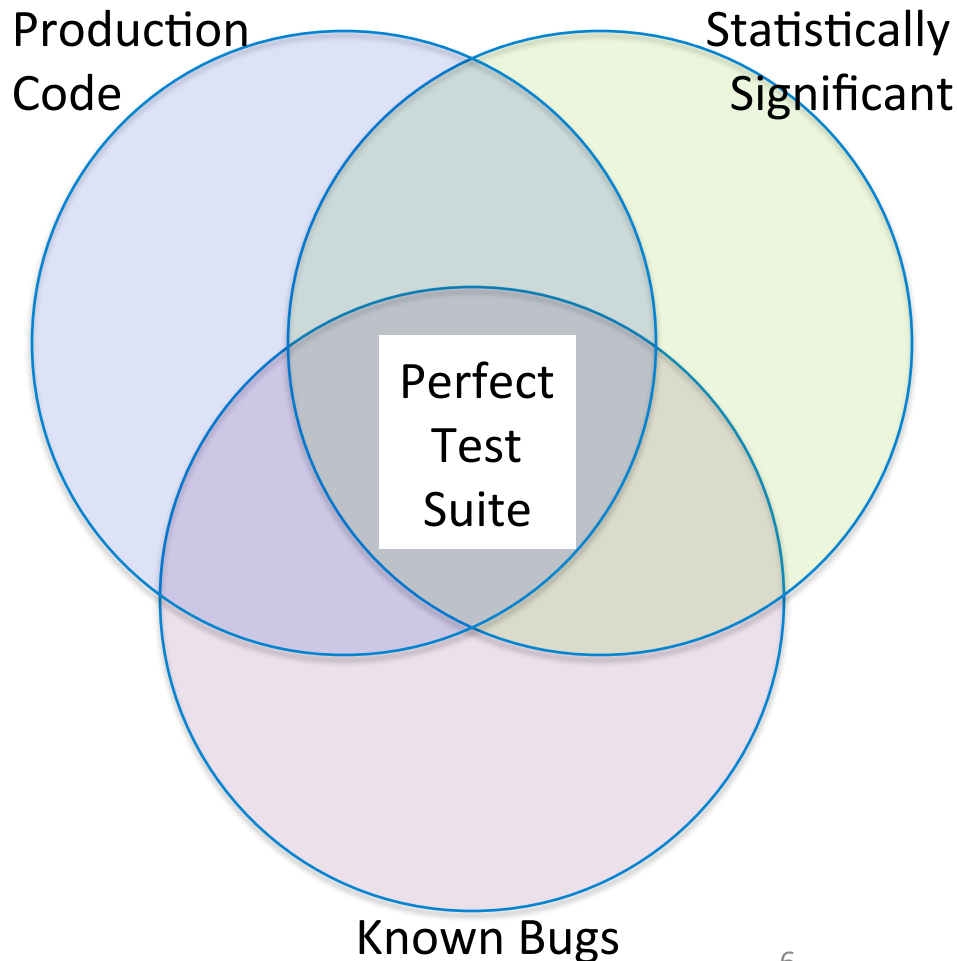
- Examine source code or binary for weaknesses, adherence to guidelines, etc.
- Level of formality may vary from program “proofs” to heuristics
- Level of automation may vary from analysis assistant to fully automated

# Testing Static Analysis Tools



# Three Desired Characteristics of Test Suites

- **Needs**
  - Test cases applicable to production code
  - Statistically significant number of test cases
  - Test cases with ground truth: known bugs
- **Objective:**
  - Collect and develop test cases with those characteristics





# Material to Properly Test Tools

- Static analysis
- Dynamic bug detection



DRAGON TAILS - <http://www.Dragon-Tails.com>



Copyright Tim Dawson 2002



# Software Assurance Reference Dataset (SARD)

**SAMATE**  
**NIST**  
National Institute of Standards and Technology

SRD Home View / Download Search / Download More Downloads Submit Test Suites

Extended Search Source Code Search

Number (Test case ID):  
Description contains:  
Contributor/Author:  
Bad / Good: Any...  
Language: Any...  
Type of Artifact: Any...  
Status: Candidate  Approved   
Weakness: Any...  
Code complexity: Any...  
Date:  Any  Before  After  
(Format: M/d/Y)  
use the calendar (next icon).

**Weakness** **Code Complexity**

- Any...
- + CWE-485: Insufficient Encapsulation
- CWE-388: Error Handling
- + CWE-389: Error Conditions, Return
- + CWE-254: Security Features
- + CWE-227: Failure to Fulfill API Contract
- + CWE-019: Data Handling
- + CWE-361: Time and State
- CWE-398: Indicator of Poor Code Quality
- CWE-470: Use of Externally-Controlled Resources
- + CWE-465: Pointer Issues
- + CWE-411: Resource Locking Problems
- CWE-401: Failure to Release Memory
- CWE-415: Double Free
- CWE-416: Use After Free
- + CWE-417: Channel and Path Errors

## Need:

- Suites of programs with known bugs to calibrate software assurance tools

## Objective:

- Collect and develop sets of programs with known bugs in various languages, with bugs of various classes, and bugs woven into various code structures

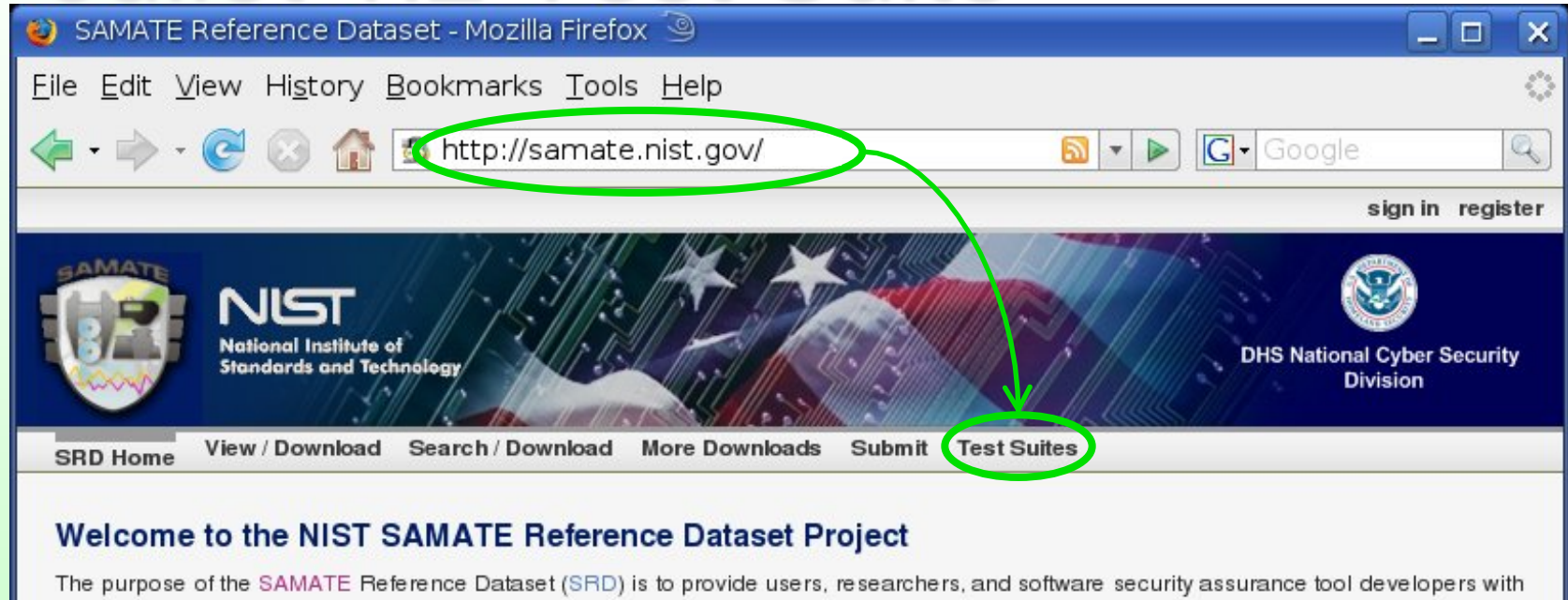
<http://samate.nist.gov/SARD/>

# Software Assurance Reference Dataset (SARD)

- Over 89 000 cases in C, C++, Java, C#, and PHP
- Contributions also from Fortify, Defence R&D Canada, Klocwork, Kratkiewicz, MIT Lincoln Laboratory, Secure Software, Praxis, etc.
- NSA Juliet 1.2 - over 86 000 small, synthetic test cases in C, C++, and Java covering 150 bug classes
- IARPA STONESOUP Phase 3 - 15 000 cases based on 12 web apps with injected bug from 25 classes
- 2000 PHP cases developed at TELECOM Nancy
- Users can search and download by language, weakness, size, content, etc.
- Test cases from Static Analysis Tool Exposition (SATE)



# Juliet 1.2 Test Suite



- **86864 small C/C++ and Java programs for almost two hundred weakness classes**
- **Each case is one or two pages of code**
- **Described in IEEE Computer, Oct 2012**

# IARPA STONESOUP Phase 3

## cases

- **7770 cases in Java and C**
- **Real programs with flaw inserted. Each case has inputs to trigger flaw and “safe” inputs**
- **Each case has inputs triggering the vulnerability, as well as “safe” inputs**
- **Cover weaknesses in Integer Overflow, Tainted Data, Command Injection, Buffer Overflow, Null Pointer, etc.**

# Kratkiewicz MIT cases

- **1164 cases in C for CWE-121 Stack-Based Buffer Overflow**
- **Created to investigate static analysis and dynamic detection methods**
- **Each case is one of four variants:**
  - access within bounds (ok)
  - access just outside bound (min)
  - somewhat outside bound (med)
  - far outside bound (large)
- **Code complexities: index, type, control, ...**



# Other SRD Content

- **Zitser, Lippmann, & Leek MIT cases**
  - 28 slices from BIND, Sendmail, WU-FTP, etc.
- **Fortify benchmark 112 C and Java cases**
- **Klocwork benchmark 40 C cases**
- **25 cases from Defence R&D Canada**
- **Robert Seacord, “Secure Coding in C and C++” 69 cases**
- **Comprehensive, Lightweight Application Security Process (CLASP) 25 cases**
- **329 cases from our static analyzer suite**

# Common Nomenclature

## Common Weakness Enumeration (CWE)

- A “dictionary” of every *class* of bug or flaw in software
- More than 600 distinct classes, e.g., buffer overflow, directory traversal, OS injection, race condition, cross-site scripting, hard-coded password, and insecure random numbers

<http://cwe.mitre.org/>

## Common Vulnerability Enumeration (CVE)

- A list of *instances* of security vulnerabilities in software
- More than 9000 CVEs were assigned in 2014  
Heartbleed is CVE-2014-0160
- NIST’s National Vulnerability Database (NVD) has fixes, severity ratings, etc. for CVEs

<https://cve.mitre.org/>

# Common Weakness Enumeration (CWE) is a Mess

- **CWE is widely used - by far the best dictionary of software weaknesses. Many tools, projects, etc. are based on CWE.**
- **But definitions are imprecise and inconsistent.**
- **CWEs are “coarse grained”: they bundle lots of stuff, like consequences and likely attacks.**
- **The coverage is uneven, with some combinations well represented and others not represented at all.**
- **No mobile weaknesses, eg., battery drain, physical sensors (GPS, gyro, microphone, hi-res camera), unencrypted wireless communication, etc.**



# Definitions are Imprecise

- **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.”**

- ***Note that “read from or write to a memory location” is not tied to the buffer!***

# Overflow Has Gaps in Coverage

- **CWE-124: Buffer Underwrite ('Buffer Underflow')** and **CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')** vs.
- **CWE-121: Stack-based Buffer Overflow** and **CWE-122: Heap-based Buffer Overflow**
- **CWE-127: Buffer Under-read** and **CWE-126: Buffer Over-read**
- *but no read-stack and read-heap versions.*

# ... and a buncha' others, too

- **CWE-123: Write-what-where Condition**
- **CWE-125: Out-of-bounds Read**
- **CWE-787: Out-of-bounds Write**
- **CWE-786: Access of Memory Location Before Start of Buffer**
- **CWE-788: Access of Memory Location After End of Buffer**
- **CWE-805: Buffer Access with Incorrect Length Value**
- **CWE-823: Use of Out-of-range Pointer Offset**

# Path Traversal is too Detailed

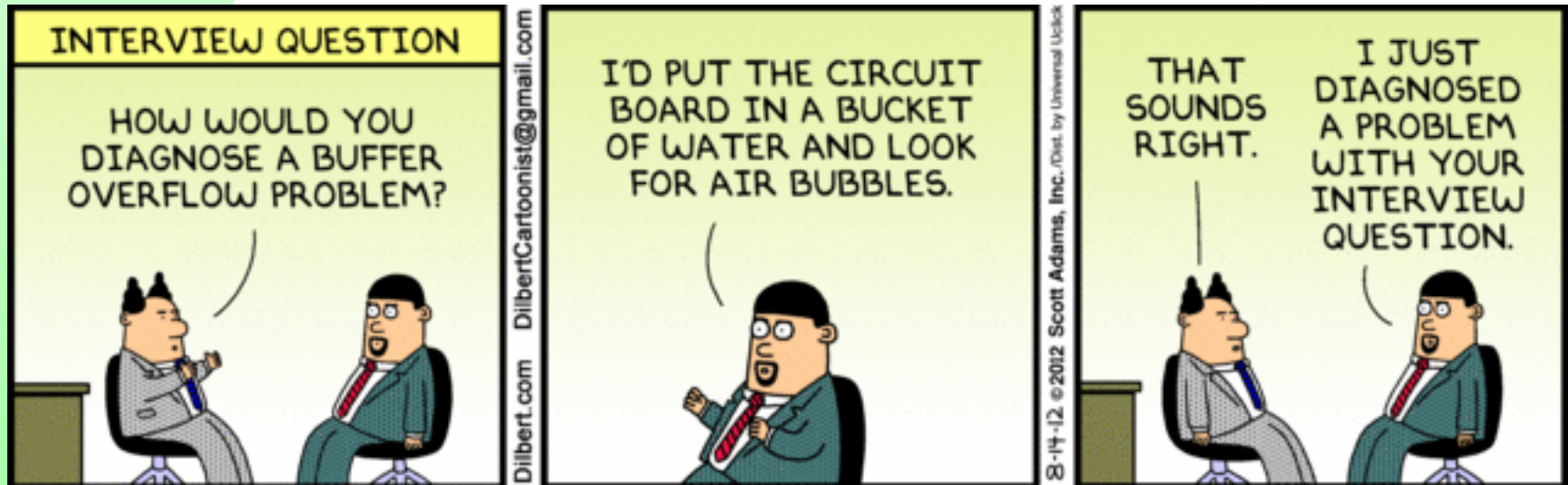
- **CWE-23: Relative Path Traversal**
- **CWE-24: Path Traversal: '../filedir'**
- **CWE-25: Path Traversal: '/../filedir'**
- **CWE-26: Path Traversal: '/dir/../filename'**
- **CWE-27: Path Traversal: 'dir/../../filename'**
- **CWE-28: Path Traversal: '..\filedir'**
- **CWE-29: Path Traversal: '..\filename'**
- **CWE-30: Path Traversal: '\dir..\filename'**
- **CWE-31: Path Traversal: 'dir\..\filename'**
- **CWE-32: Path Traversal: '...' (Triple Dot)**
- **CWE-33: Path Traversal: '....' (Multiple Dot)**
- **CWE-34: Path Traversal: '....//'**
- **CWE-35: Path Traversal: '.../...//'**

# Other Bug Descriptions Have Problems, Too.

- **Software Fault Patterns (SFP)**
  - “factor” weaknesses into parameters, but
  - don’t include upstream causes or consequences,
  - and are based solely on CWEs.
- **Semantic Templates**
  - collect CWEs into four general areas
    - Software-fault
    - Weakness
    - Resource/Location
    - Consequences
  - but are guides to aid human comprehension.

# We Need Better Vocabulary

- **Finer grained, common vocabulary to describe bugs**
  - **Common Weakness Enumeration (CWE) is widely-used, but does not match well the classes that tools report. Tools' classes are precise, but are hard to match to other tools.**



# Precise Medical Vocabulary

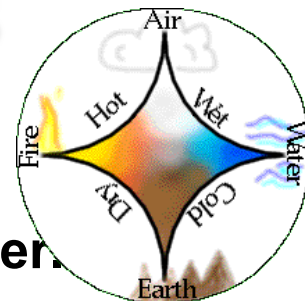
- Medical professionals have terms to precisely name muscles, bones, organs, conditions, diseases, and so forth.



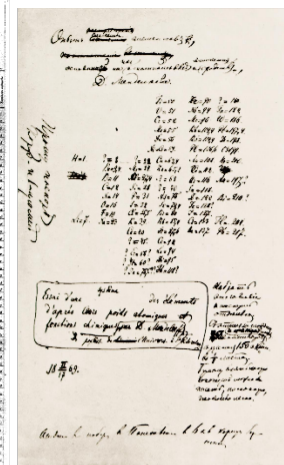
Figure 2: Computed tomography of a comatose patient with a left temporal epidural haematoma, right parenchymal temporal lobe haematoma, and a right convexity subdural haematoma before and after craniotomy and evacuation of haematomas



# Periodic Table Took Centuries



- Greeks used the terms *element* and *atom*.
- Aristotle: everything is a mix of Earth, Fire, Air, or Water.
- Alchemists in the Middle Ages cataloged materials like alcohol, sulfur, mercury, and salt.
- Lavoisier listed 33 elements and distinguished metals and non-metals.
  - including oxygen, nitrogen, hydrogen, phosphorus, mercury, zinc, sulfur, *light*, and *caloric*.
- Dalton realized “atoms of same element are identical in all respects, particularly weight.”
- Mendeleev’s table embodied centuries of knowledge that reflects atomic structure and forecast properties of missing elements.

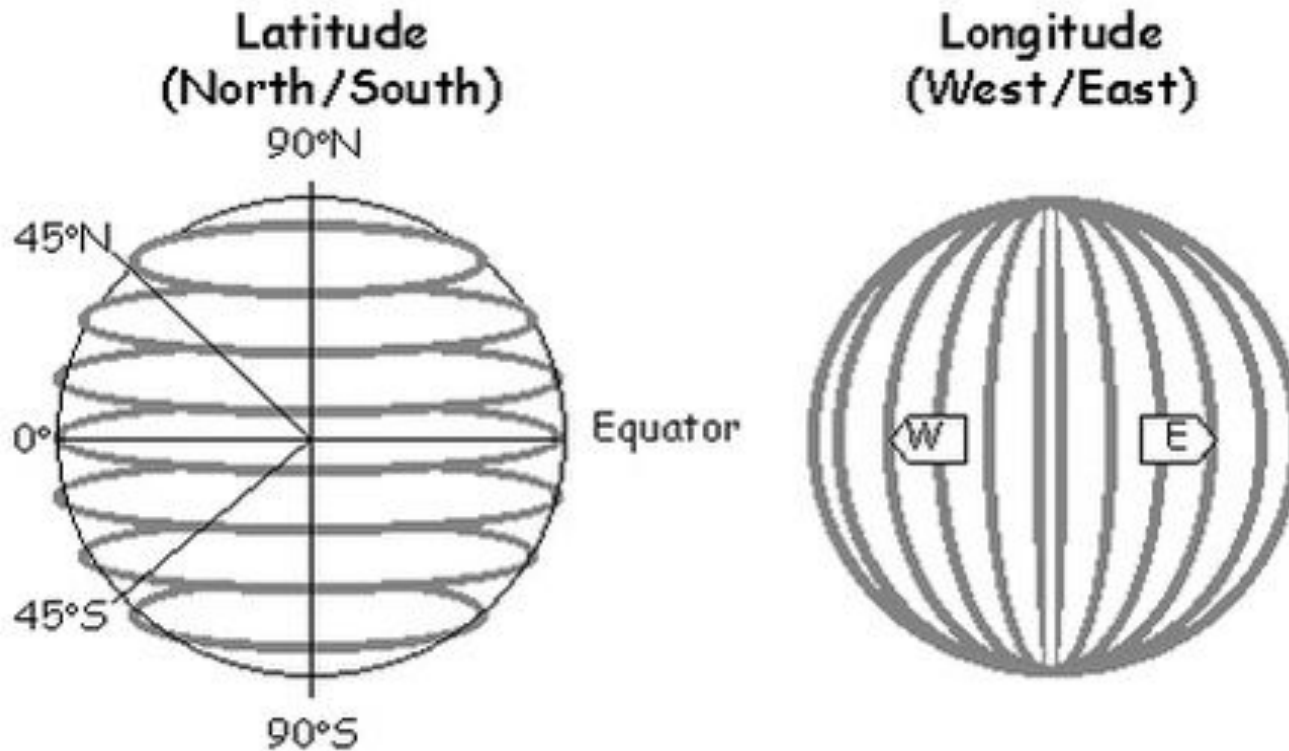


The Periodic Table of the Elements

1	2											18											
H	He																						
3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18								
Li	Be	B	C	N	O	F	Ne																
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36						
Na	Mg	Al	Si	P	S	Cl	Ar																
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54						
K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr						
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72						
Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe						
87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104						
Cs	Ba	La	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn						
119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136						
Fr	Ra	Lr	Rf	Db	Sg	Bh	Hs	Mt	Ds	Rg	Cn	Uut	Fl	Uup	Lv	Uus	Uuo						
												108	109	110	111	112	113	114	115	116	117	118	
												119	120	121	122	123	124	125	126	127	128	129	130
												131	132	133	134	135	136	137	138	139	140	141	142
												143	144	145	146	147	148	149	150	151	152	153	154
												155	156	157	158	159	160	161	162	163	164	165	166
												167	168	169	170	171	172	173	174	175	176	177	178
												179	180	181	182	183	184	185	186	187	188	189	190
												191	192	193	194	195	196	197	198	199	200	201	202
												203	204	205	206	207	208	209	210	211	212	213	214
												215	216	217	218	219	220	221	222	223	224	225	226
												227	228	229	230	231	232	233	234	235	236	237	238
												239	240	241	242	243	244	245	246	247	248	249	250
												251	252	253	254	255	256	257	258	259	260	261	262
												263	264	265	266	267	268	269	270	271	272	273	274
												275	276	277	278	279	280	281	282	283	284	285	286
												287	288	289	290	291	292	293	294	295	296	297	298
												299	300	301	302	303	304	305	306	307	308	309	310
												311	312	313	314	315	316	317	318	319	320	321	322
												323	324	325	326	327	328	329	330	331	332	333	334
												335	336	337	338	339	340	341	342	343	344	345	346
												347	348	349	350	351	352	353	354	355	356	357	358
												359	360	361	362	363	364	365	366	367	368	369	370
												371	372	373	374	375	376	377	378	379	380	381	382
												383	384	385	386	387	388	389	390	391	392	393	394
												395	396	397	398	399	400	401	402	403	404	405	406
												407	408	409	410	411	412	413	414	415	416	417	418
												419	420	421	422	423	424	425	426	427	428	429	430
												431	432	433	434	435	436	437	438	439	440	441	442
												443	444	445	446	447	448	449	450	451	452	453	454
												455	456	457	458	459	460	461	462	463	464	465	466
												467	468	469	470	471	472	473	474	475	476	477	478
												479	480	481	482	483	484	485	486	487	488	489	490
												491	492	493	494	495	496	497	498	499	500	501	502
												503	504	505	506	507	508	509	510	511	512	513	514
												515	516	517	518	519	520	521	522	523	524	525	526
												527	528	529	530	531	532	533	534	535	536	537	538
												539	540	541	542	543	544	545	546	547	548	549	550
												551	552	553	554	555	556	557	558	559	560	561	562
												563	564	565	566	567	568	569	570	571	572	573	574
												575	576	577	578	579	580	581	582	583	584	585	586
												587	588	589	590	591	592	593	594	595	596	597	598
												599	600	601	602	603	604	605	606	607	608	609	610
												611	612	613	614	615	616	617	618	619	620	621	622
												623	624	625	626	627	628	629	630	631	632	633	634
												635	636	637	638	639	640	641	642	643	644	645	646
												647	648	649	650	651	652	653	654	655	656	657	658
												659	660	661	662	663	664	665	666	667	668	669	670
												671	672	673	674	675	676	677	678	679	680	681	682
												683	684	685	686	687	688	689	690	691	692	693	694
												695	696	697	698	699	700	701	702	703	704	705	706
												707	708	709	710	711	712	713	714	715	716	717	718
												719	720	721	722	723	724	725	726	727	728	729	730
												731	732	733	734	735	736	737	738	739	740	741	742
												743	744	745	746	747	748	749	750	751	752	753	754
												755	756	757	758	759	760	761	762	763	764	765	766
												767	768	769	770	771	772	773	774	775	776	777	778
												779	780	781	782	783	784	785	786	787	788	789	790
												791	792	793	794	795	796	797	798	799	800	801	802
												803	804	805	806	807	808	809	810	811	812	813	814
												815	816	817	818	819	820	821	822	823	824	825	826
												827	828	829	830	831	832	833	834	835	836	837	838
												839	840	841	842	843	844	845	846	847	848	849	850
												851	852	853	854	855	856	857	858	859	860	861	862
												863	864	865	866	867	868	869	870	871	872	873	874
												875	876	877	878	879	880	881	882	883	884	885	886
												887	888	889	890	891	892	893	894	895	896	897	898
												899	900	901	902	903	904	905	906	907	908	909	910
												911	912	913	914	915	916	917	918	919	920	921	922
												923	924	925	926	927	928	929	930	931	932	933	934
												935	936	937	938	939	940	941	942	943	944	945	946
												947	948	949	950	951	952	953	954	955	956	957	958
												959	960	961	962	963	964	965	966	967	968	969	970
												971	972	973	974	975	976	977	978	979	980	981	982
												983	984	985	986	987	988	989	990	991	992	993	994
												995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006
												1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018
												1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030
												1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042
												1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054
												1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066
												1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078
												1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090
												1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102
												1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114
												1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126
												1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138

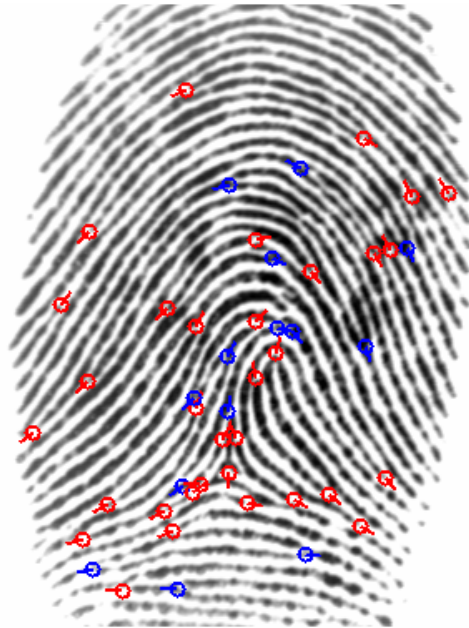


# Specify Location with Latitude, Longitude, and Elevation



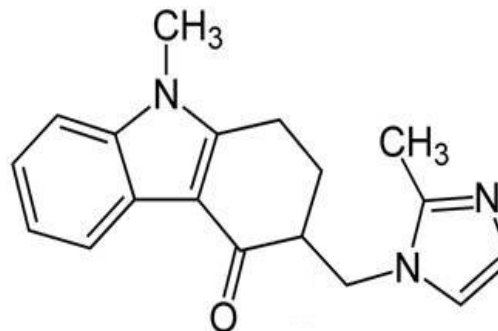
# Fingerprints

- **Classified as loop, whorl, or arch.**
- **Retrieved by minutia**



# Chemists Have Detailed Systems to Describe Chemicals

Zofran ODT is:  $C_{18}H_{19}N_3O$



(±) 1, 2, 3, 9-tetrahydro-9-methyl-3-[(2-methyl-1H-imidazol-1-yl)methyl]-4H-carbazol-4-one

# Integers Have Prime Factors

$$6 = 2 \times 3$$

$$70 = 2 \times 5 \times 7$$

$$47,298,756 = 2 \times 2 \times 7 \times 641 \\ \times 1471 \times 1657$$

Our vision is to have  
a precise descriptive language for bugs  
organized in a “natural” way.  
(e.g., vocabulary, grammar, taxonomy, ontology, etc.)

# Outline

- The “Science” of Weaknesses
- **Our Nomenclature**
- Examples of Applying Our Approach
- Using This

# We Start With Buffer Overflow

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

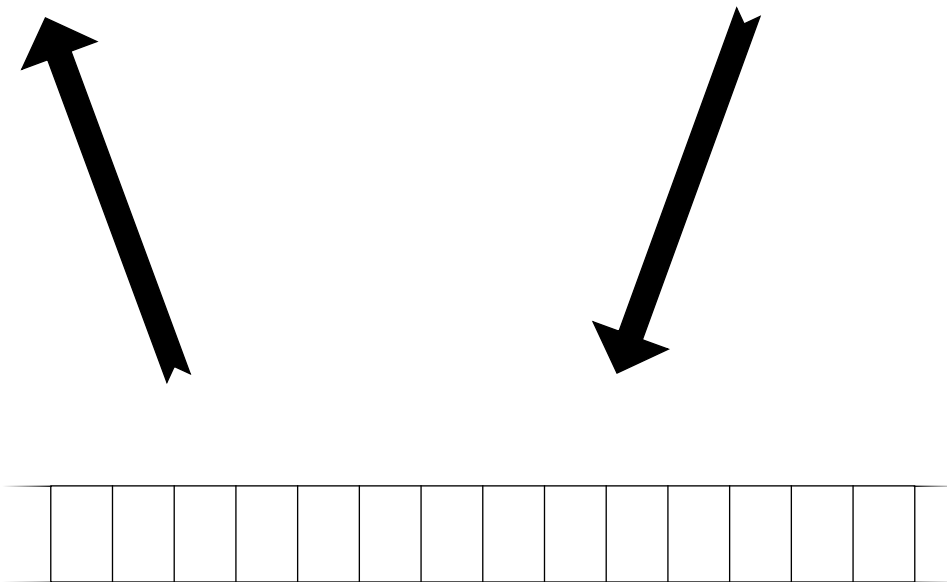
# Buffer Overflow: Attributes





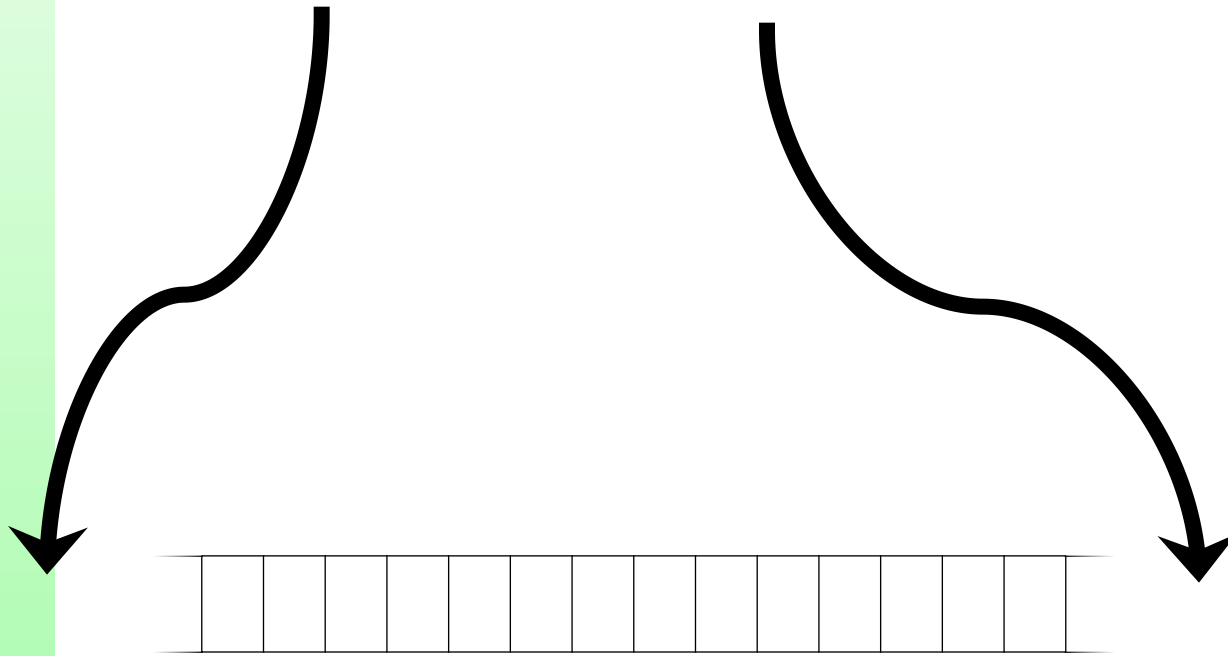
# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.



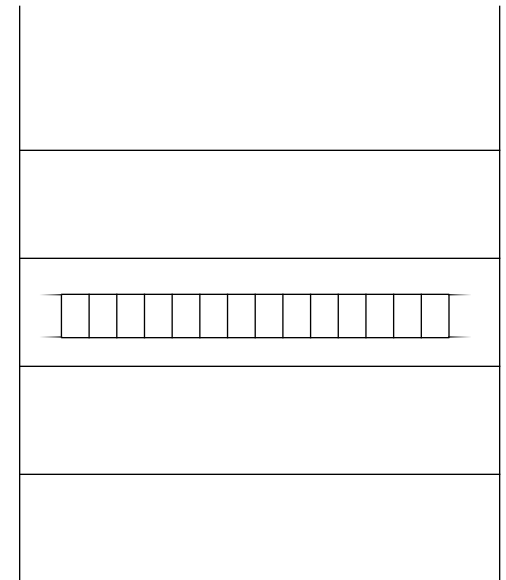
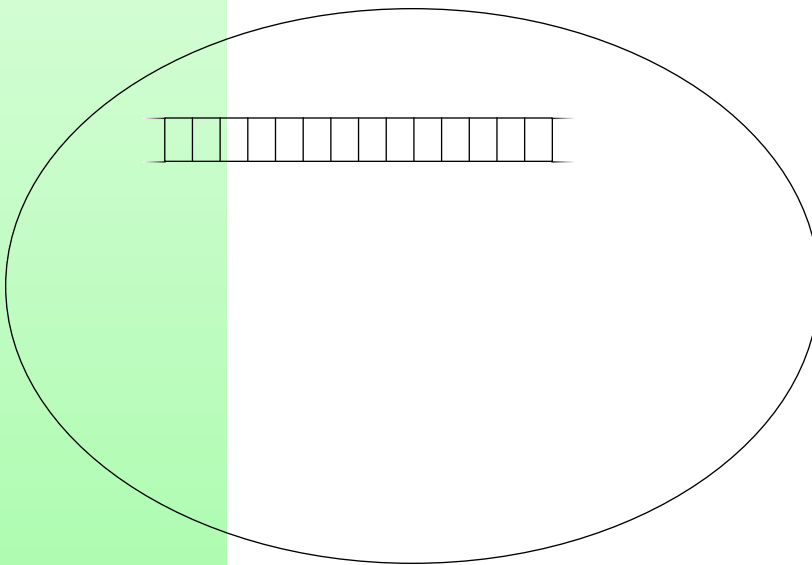
# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.
- **Side:**
  - Below (before, under, or lower), Above (after, over, or upper).



# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.
- **Side:**
  - Below (before, under, or lower), Above (after, over, or upper).
- **Segment (memory area):**
  - Heap, Stack, BSS (uninitialized data), Data (initialized), Code (text).

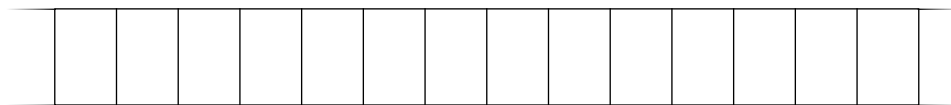


# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.
- **Side:**
  - Below (before, under, or lower), Above (after, over, or upper).
- **Segment (memory area):**
  - Heap, Stack, BSS (uninitialized data), Data (initialized), Code (text).
- **Method:**
  - Indexed, (bare) Pointer.

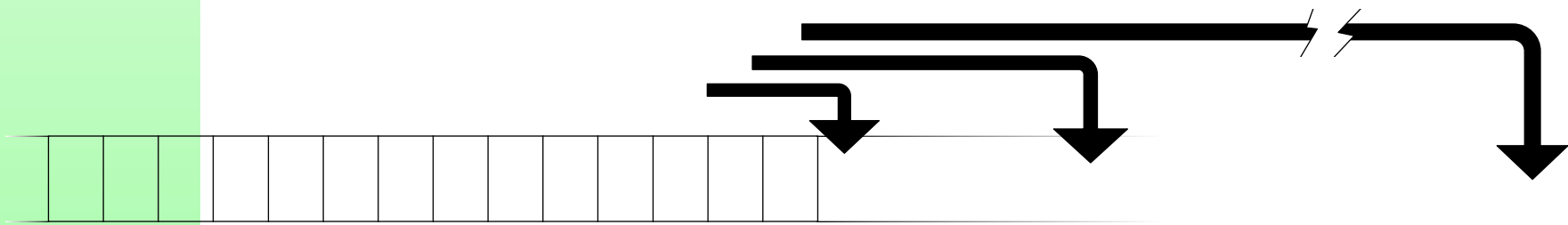
```
t = buf[j];
```

```
*buf = mind();
```



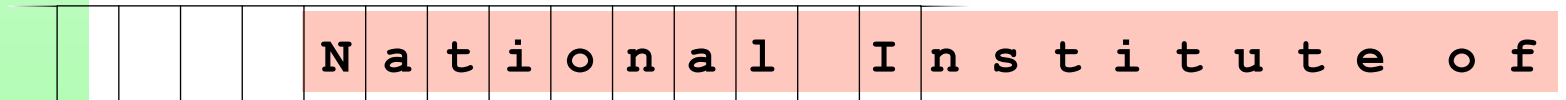
# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.
- **Side:**
  - Below (before, under, or lower), Above (after, over, or upper).
- **Segment (memory area):**
  - Heap, Stack, BSS (uninitialized data), Data (initialized), Code (text).
- **Method:**
  - Indexed, (bare) Pointer.
- **Magnitude (how far outside):**
  - Minimal (just barely outside), Moderate, Far (e.g. 4000).

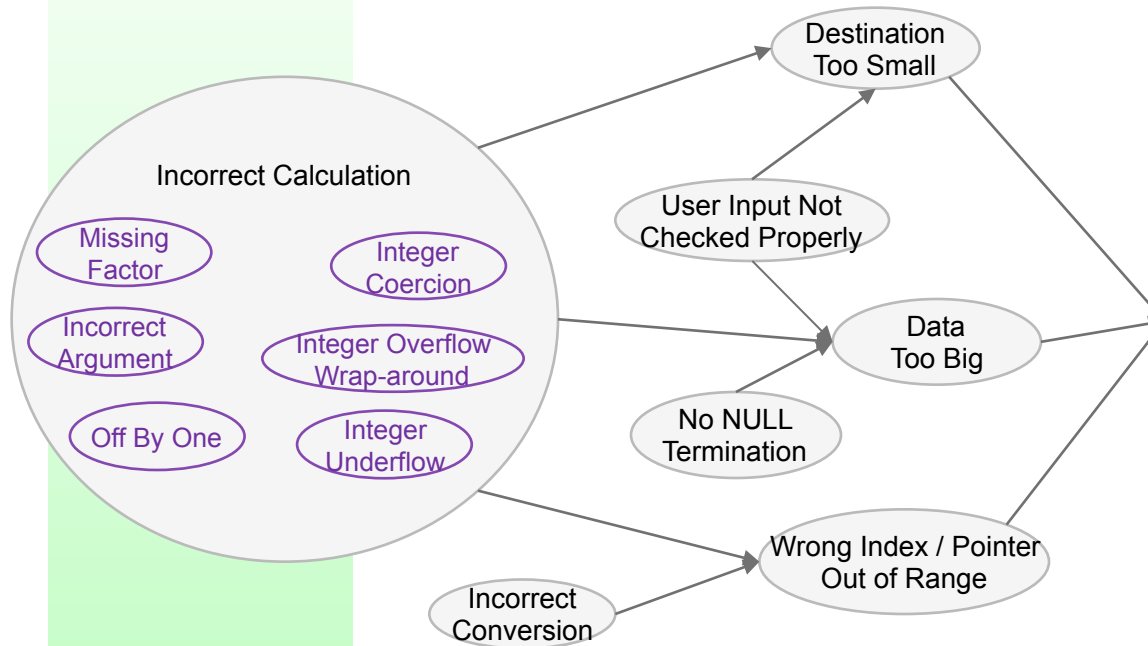


# Buffer Overflow: Attributes

- **Access:**
  - Read, Write.
- **Side:**
  - Below (before, under, or lower), Above (after, over, or upper).
- **Segment (memory area):**
  - Heap, Stack, BSS (uninitialized data), Data (initialized), Code (text).
- **Method:**
  - Indexed, (bare) Pointer.
- **Magnitude (how far outside):**
  - Minimal (just barely outside), Moderate, Far (e.g. 4000).
- **Data Size (how much is outside):**
  - Minimal, Some (e.g. half dozen), Gazillion.



# Buffer Overflow: Causes



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.
- Those 3 have preceding causes that may lead to them.

## Buffer Overflow

### Attributes:

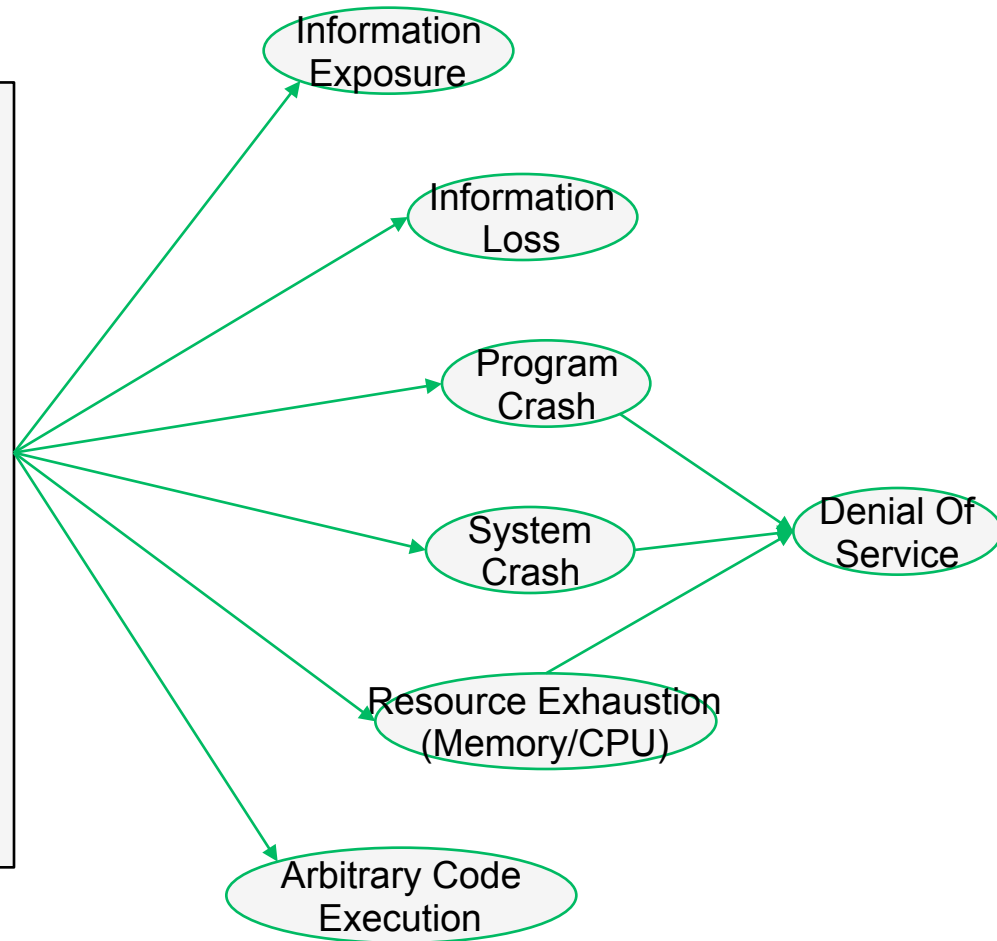
- Access:
  - ✓ *Read, Write.*
- Side:
  - ✓ *Below* (before or under), *Above* (after or over)
- Segment (memory area):
  - ✓ *Heap, Stack, BSS, Data* (initialized), *Code* (text)
- Method:
  - ✓ *Indexed*, (bare) *Pointer.*
- Magnitude (how far outside):
  - ✓ *Minimal* (just barely), *Moderate*, *Far* (e.g. 4000).
- Data Size (how much data) :
  - ✓ *Minimal, Some, Gazillion.*

# Buffer Overflow: Consequences

**Buffer Overflow**

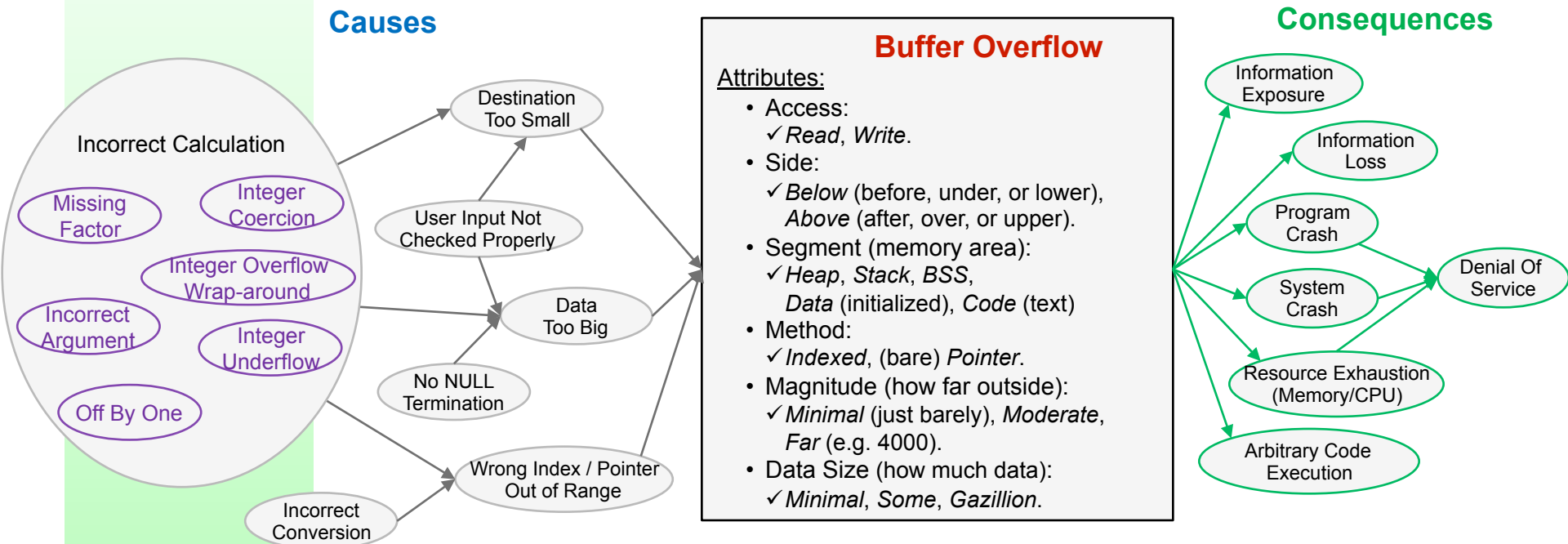
Attributes:

- Access:
  - ✓ *Read, Write.*
- Side:
  - ✓ *Below* (before, under, or lower), *Above* (after, over, or upper).
- Segment (memory area):
  - ✓ *Heap, Stack, BSS, Data* (initialized), *Code* (text)
- Method:
  - ✓ *Indexed, (bare) Pointer.*
- Magnitude (how far outside):
  - ✓ *Minimal* (just barely), *Moderate, Far* (e.g. 4000).
- Data Size (how much data) :
  - ✓ *Minimal, Some, Gazillion.*





# Buffer Overflow: Causes, Attributes, and Consequences



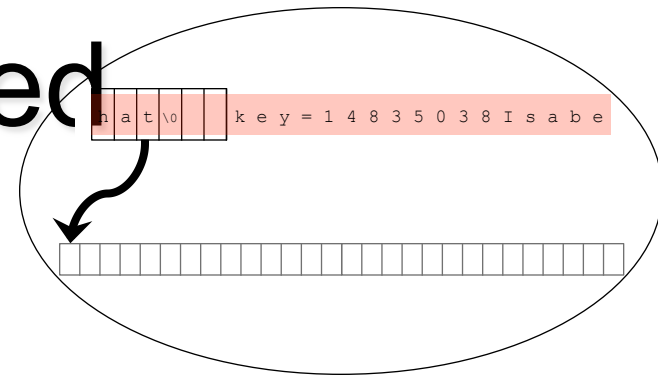
The graph of causes shows:

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

# Outline

- The “Science” of Weaknesses
- Our Nomenclature
- **Examples of Applying Our Approach**
- Using This

# Example 1: Heartbleed CVE-2014-0160

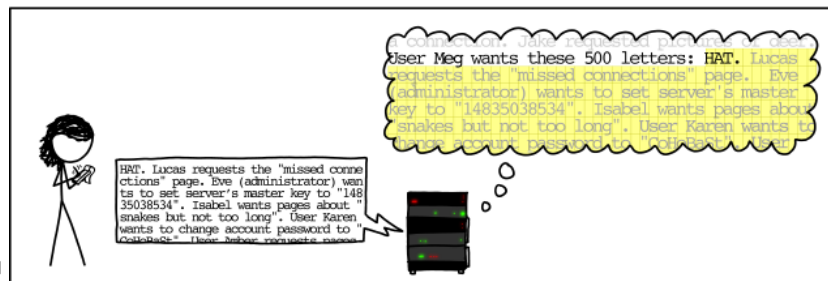
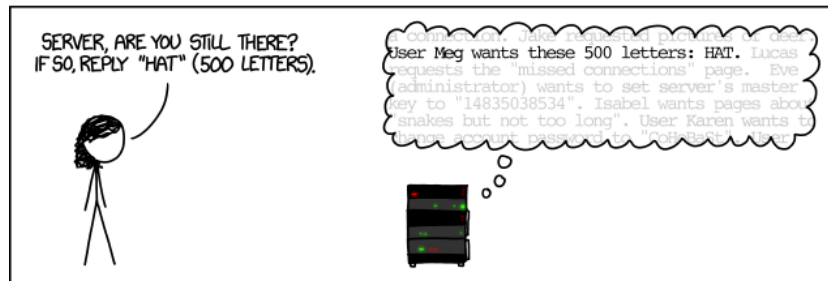
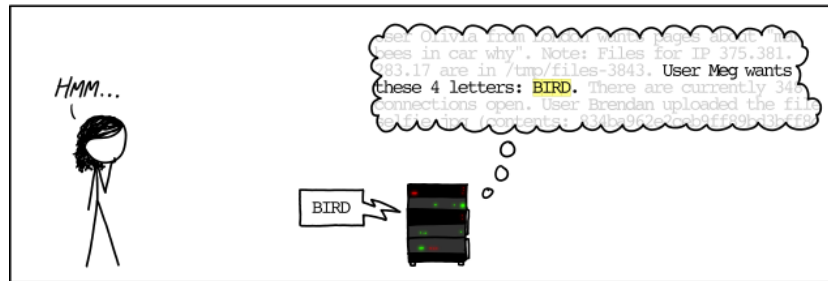


Heartbleed buffer overflow is:

- caused by *Data Too Big*
- because of *User Input not Checked Properly*
- where there was a *Read* that was *After* the end, *Far* outside
- reading a *Gazillion* bytes
- from a buffer in the *Heap*
- that may be exploited for *Information Exposure*
- when enabled by *Sensitive Information Uncleared Before Release (CWE-226)*.

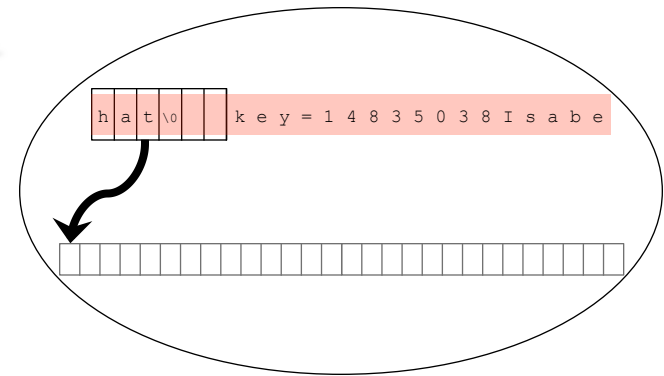
The (1) TLS and (2) DTLS implementations ... do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, ...

# Example 1: Heartbleed



from  
<http://xkcd.com/1354/>

# Example 1: Heartbleed CVE-2014-0160



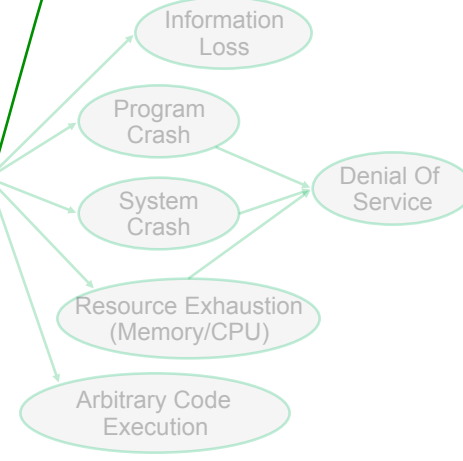
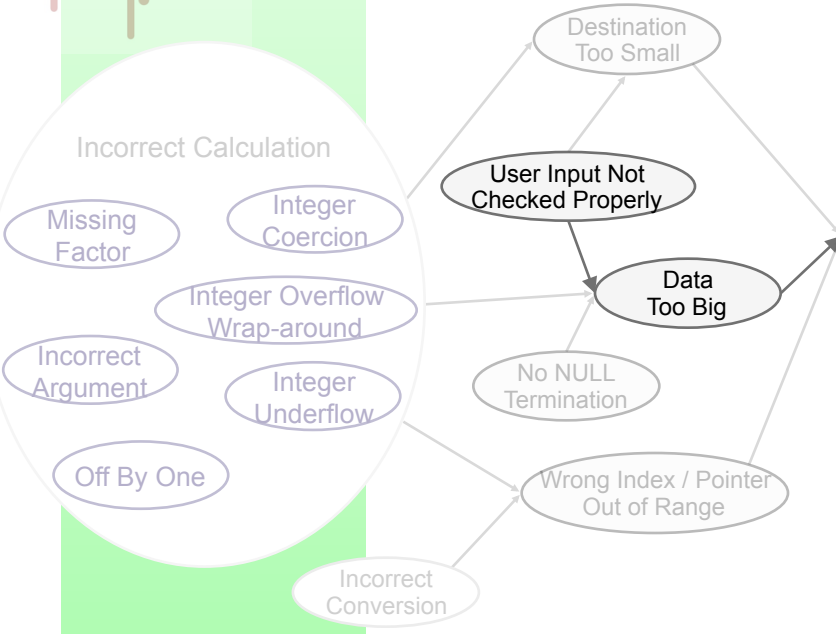
Sensitive Info Uncleared Before Release

**Buffer Overflow**

Attributes:

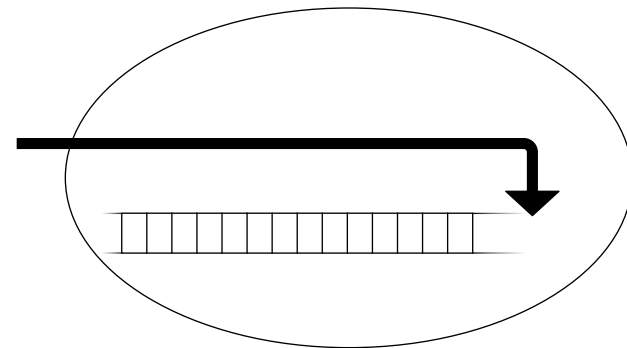
- Access:
  - ✓ Read, Write.
- Side:
  - ✓ Below (before, under, or lower), Above (after, over, or upper).
- Segment (memory area):
  - ✓ Heap, Stack, BSS, Data (initialized), Code (text)
- Method:
  - ✓ Indexed, (bare) Pointer.
- Magnitude (how far outside):
  - ✓ Minimal (just barely), Moderate, Far (e.g. 4000).
- Data Size (how much data):
  - ✓ Minimal, Some, Gazillion.

Information Exposure



# Example 2: Ghost

## CVE-2015-0235

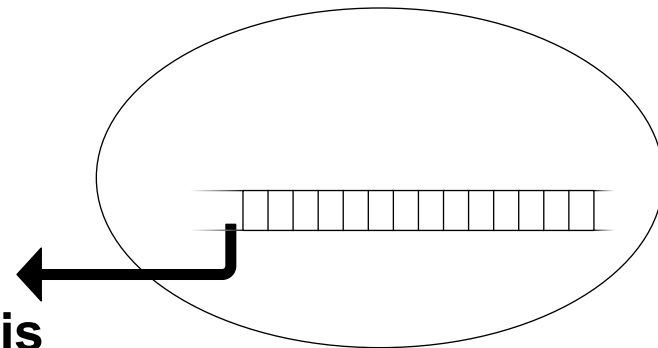


**Ghost — gethostbyname buffer overflow is**

- 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*
- that may be exploited for *Arbitrary Code Execution*.

**Heap-based buffer overflow in the `__nss_hostname_digits_dots` function ... allows context-dependent attackers to execute arbitrary code via vectors related to the (1) `gethostbyname` or (2) `gethostbyname2` function, aka “GHOST.”**

# Example 3: Chrome CVE-2010-1773



**Chrome WebCore — 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*
- that leads to use of *User Input Not Checked Properly*
- that may be exploited for *Information Exposure*, *Arbitrary Code Execution*, or *Program Crash* leading to *Denial of Service*.

**Off-by-one error in the toAlphabetic function ..., 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, ...**

# Example 4: Refactoring CWEs

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

Table 2. Buffer Overflow CWEs Organized by Attribute.

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