Risk Analysis and Measurement with CWRAF

IT Security Automation Conference

October 31st 2011

Richard Struse
DHS

Steve Christey
MITRE

Software Assurance

The level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidently inserted at anytime during its life cycle and that the software functions as intended. *Derived From: CNSSI-4009*

The level of **confidence that software** is **free from vulnerabilities** and **functions as intended**

“Making Security Measureable”:
measurablesecurity.mitre.org

Sponsored by DHS

Resources provided for voluntary adoption

Open, community efforts that are *free* to use

XML-based

Some important things to note

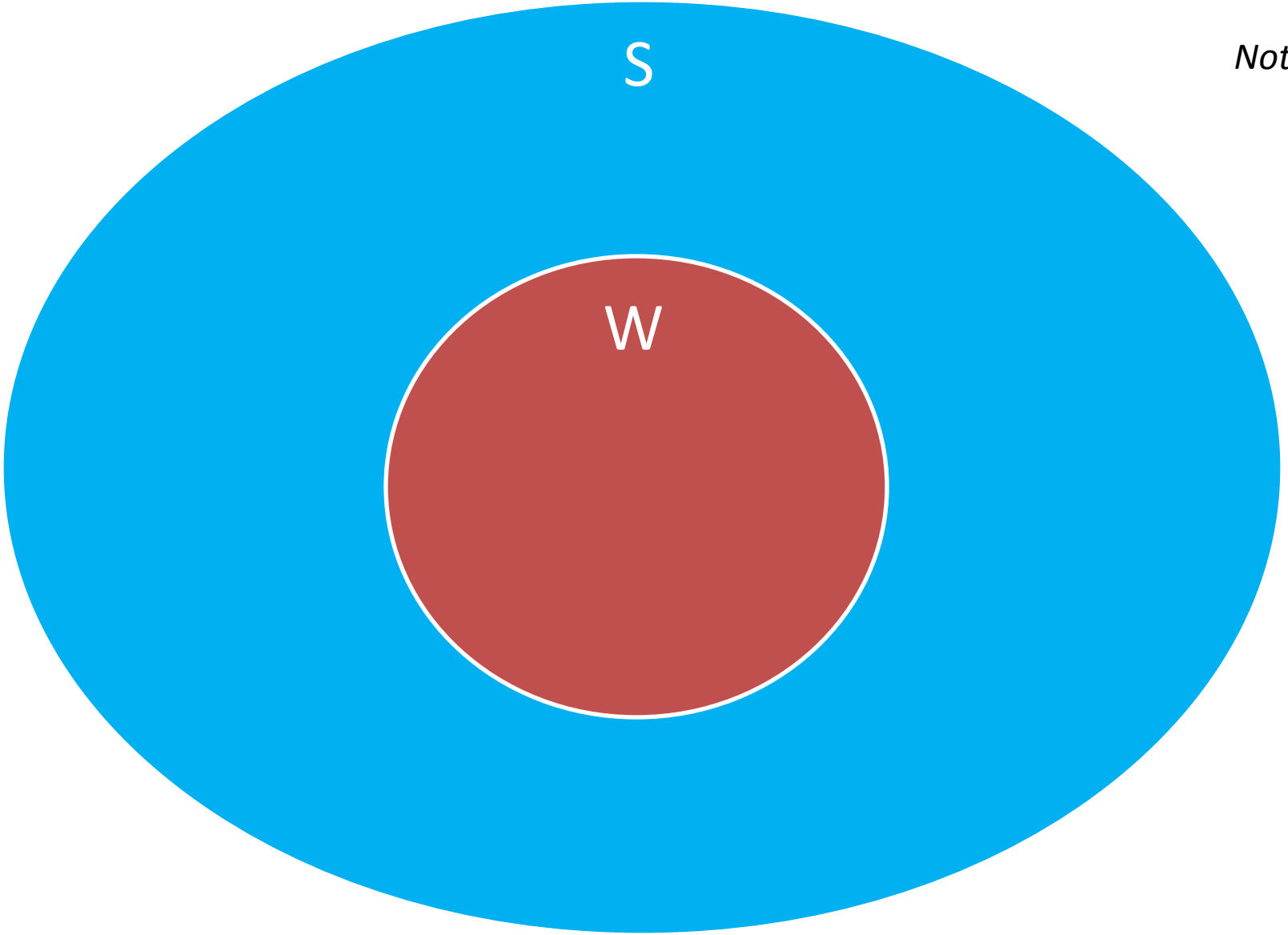
What is the context?

Where can automation help - *today*?

What problems are we trying to solve?

Where do we start?

S: The set of all software in existence at some point in time



Notional

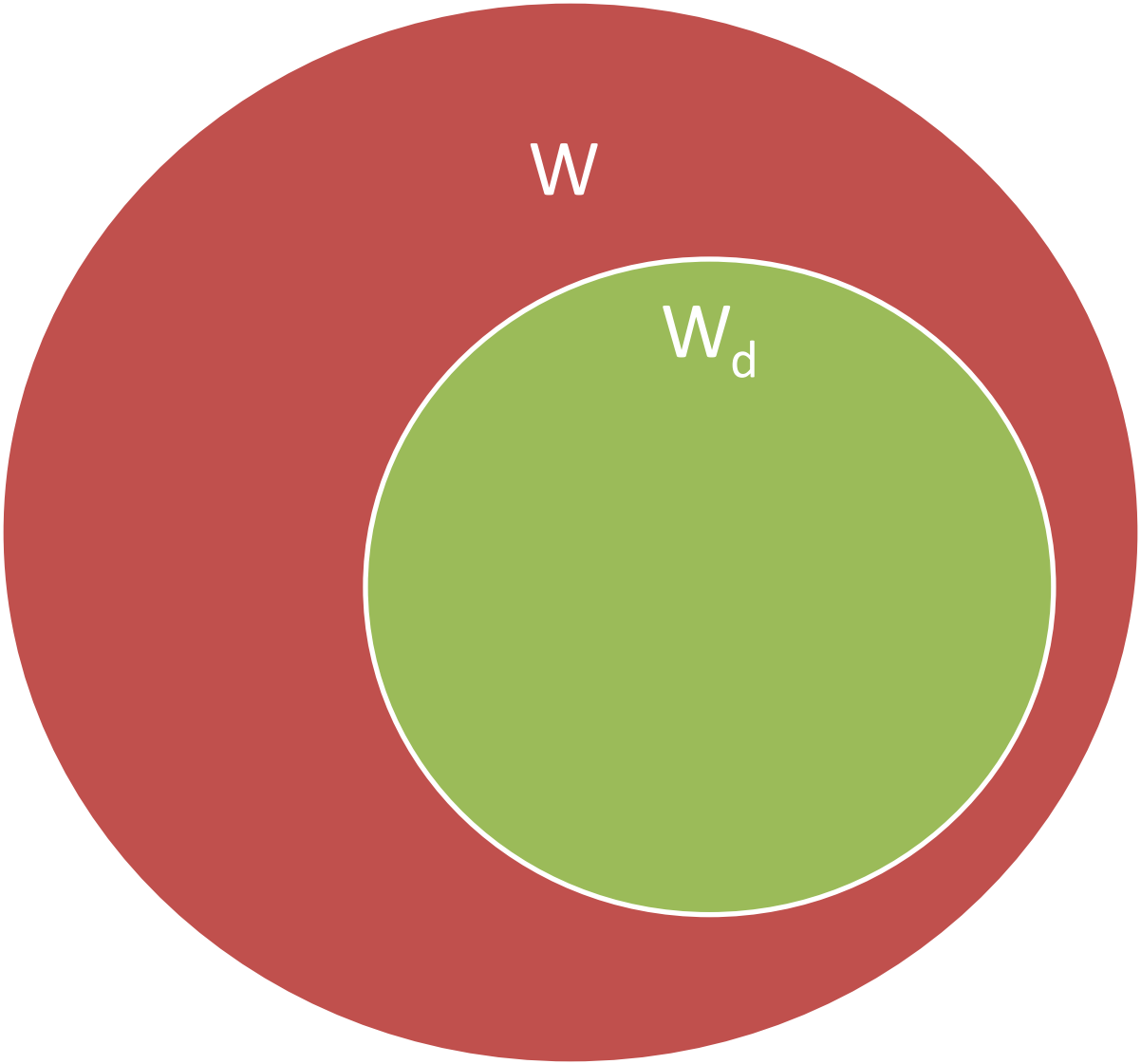
W: The set of all instances of software weaknesses in S

SIDEBAR

There are many definitions of “weakness.”
What do we mean by weakness *in this context*?

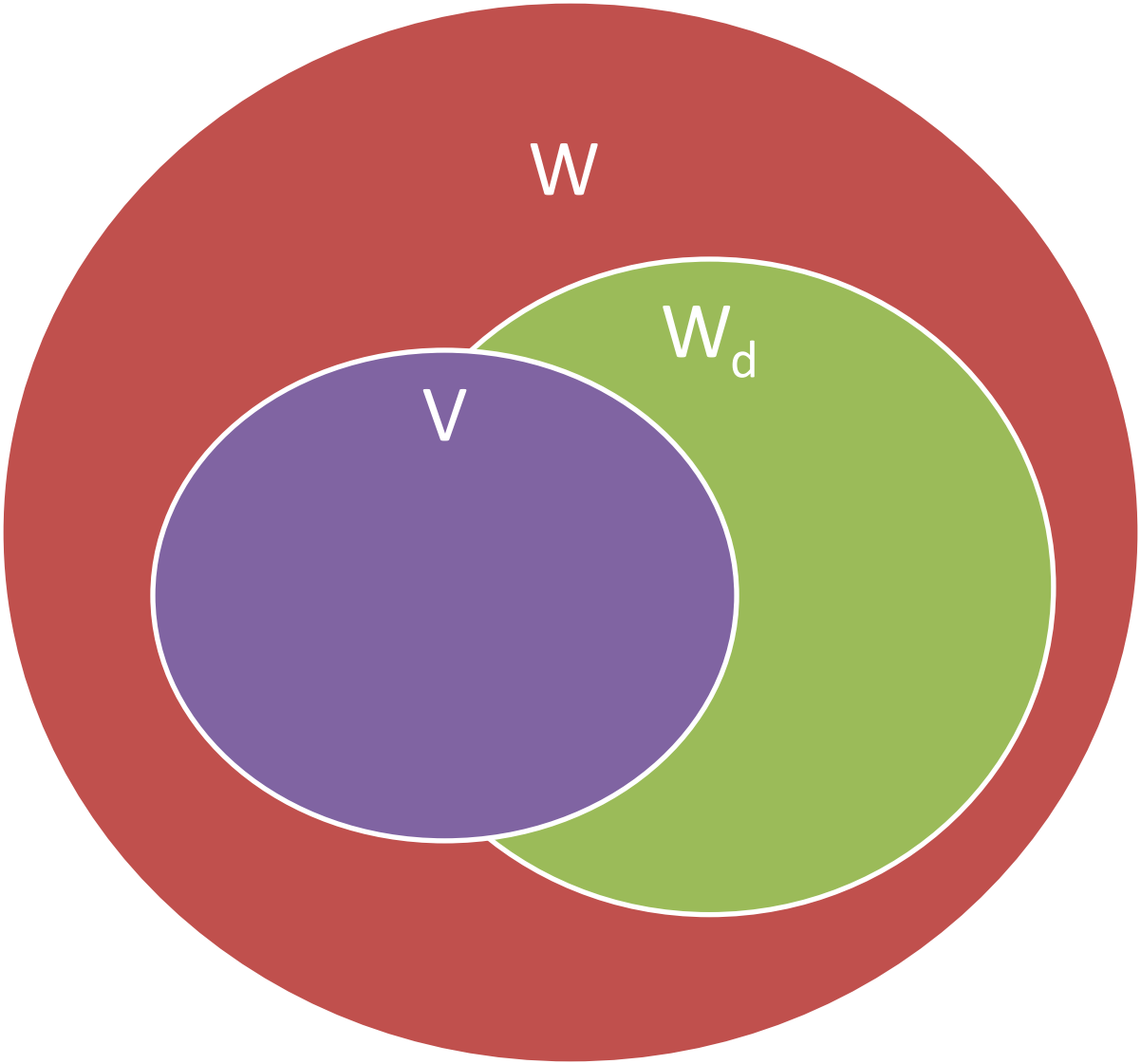
A *(software) weakness* is a property of software/systems that, under the right conditions, may permit unintended / unauthorized behavior.

Notional



W_d: The set of all *discovered* software weaknesses in W

Notional



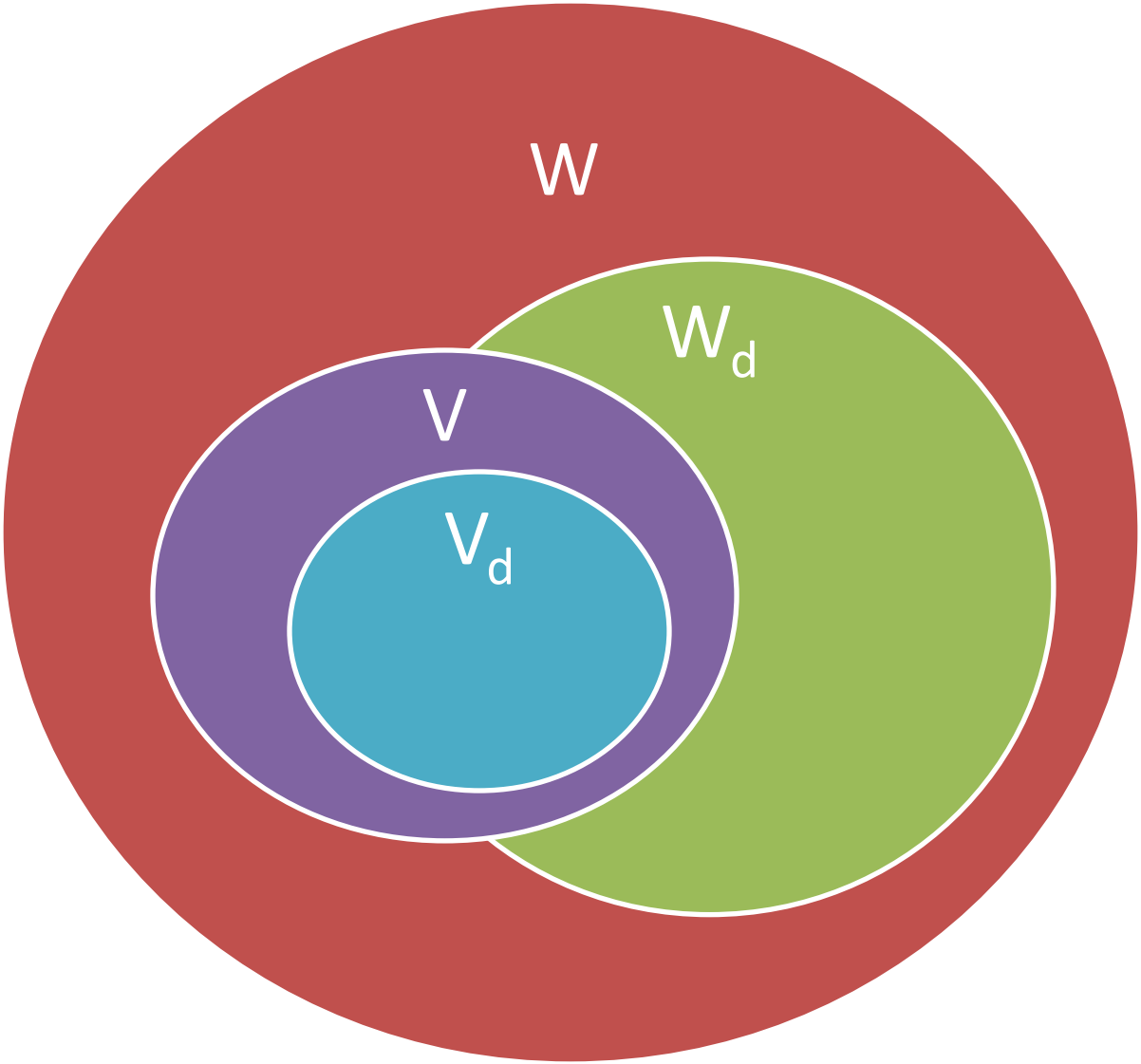
V: The set of all vulnerabilities in W

SIDEBAR

There are many definitions of “vulnerability.” What do we mean by vulnerability *in this context*?

A *(software) vulnerability* is a collection of one or more weaknesses that contain the right conditions to permit unauthorized parties to force the software to perform unintended behavior (a.k.a. “is exploitable”)

Notional



V_d: The set of all *discovered* vulnerabilities in V

Weakness vs. Vulnerability: Code Example

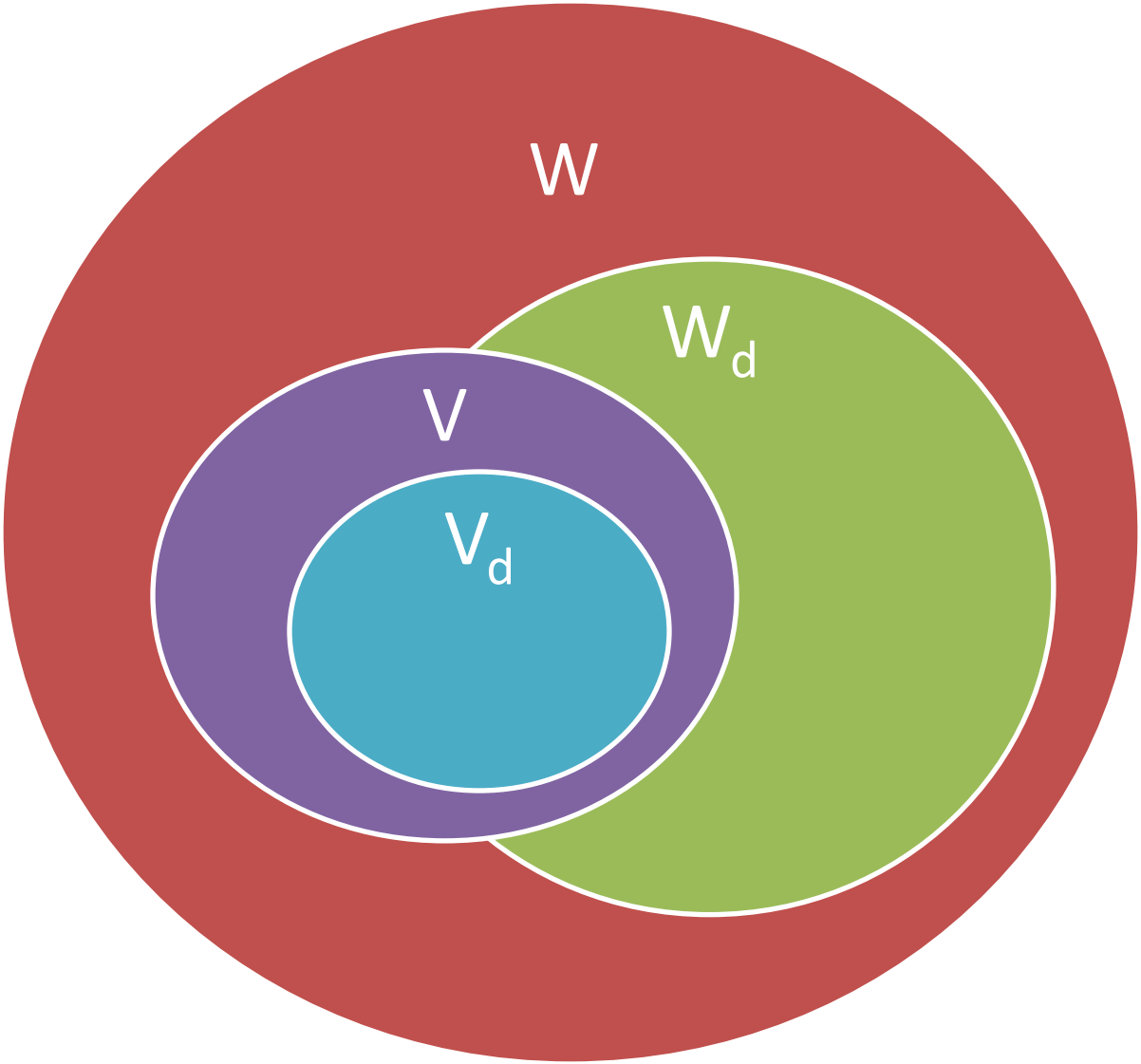
```
char *copyUserName (int nameSize, char *name) {
  int bufferSize;
  char *dupeName;

/* CWE-20: Improper input validation */
 bufferSize = (nameSize * sizeof(char));
 /* Potential integer overflow (CWE-190) and incorrect buffer size calculation (CWE-131). */
 dupeName = malloc(bufferSize);
 /* CWE-252: Unchecked Return Value */
 strcpy(dupeName, name);
 /* Potential heap-based buffer overflow (CWE-122), NULL pointer dereference (CWE-476) */
 return(dupeName);
}
```

```
copyUserName(6, "Steve");

copyUserName(atoi(argv[1]), argv[2]);

Packet = ReadNetworkPacket(20);
Size = ParseInteger(Packet);
Name = ParseString(Packet);
copyUserName(Size, Name);
```

Chain

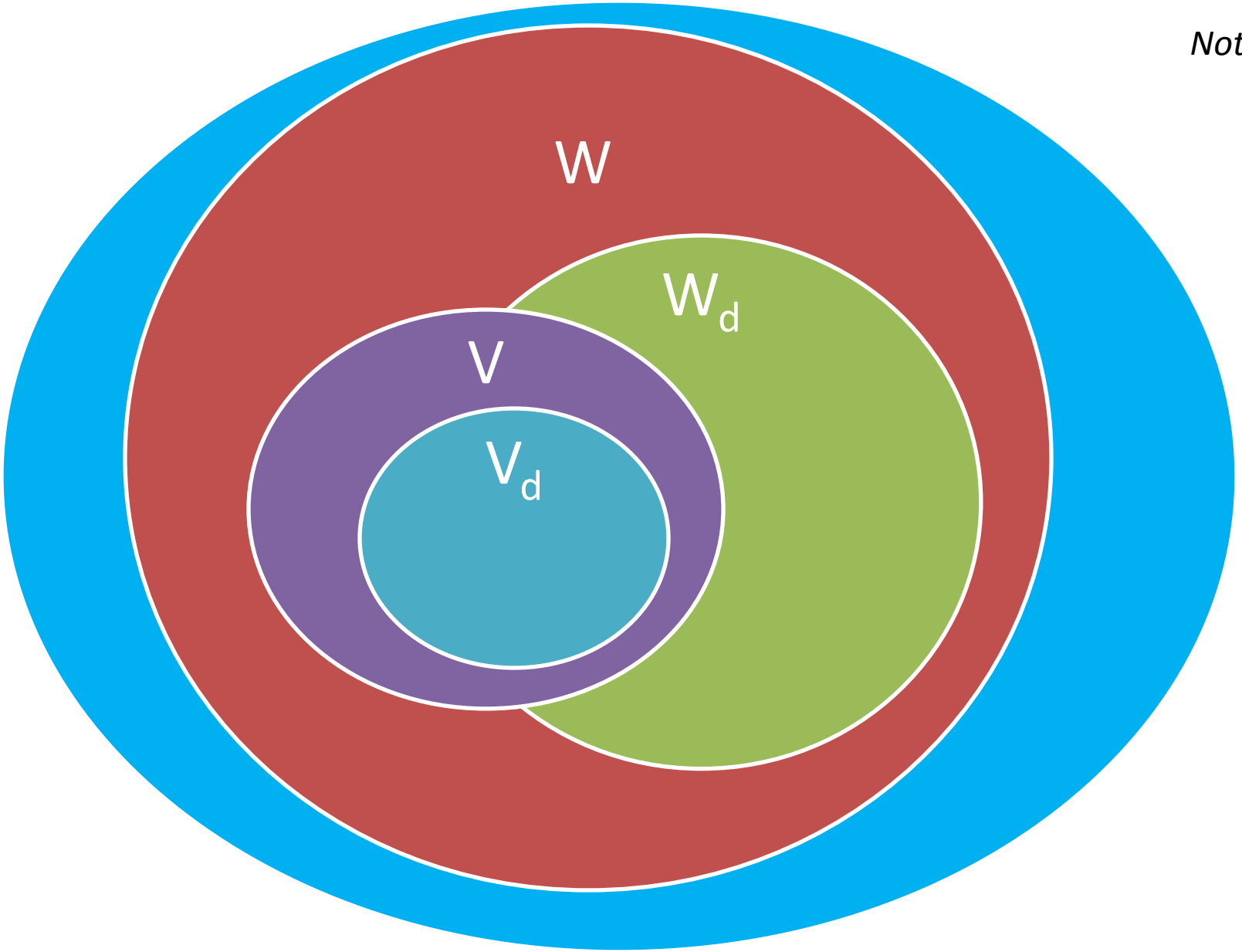
CWE-20 -> CWE-190 -> CWE-122

Notional



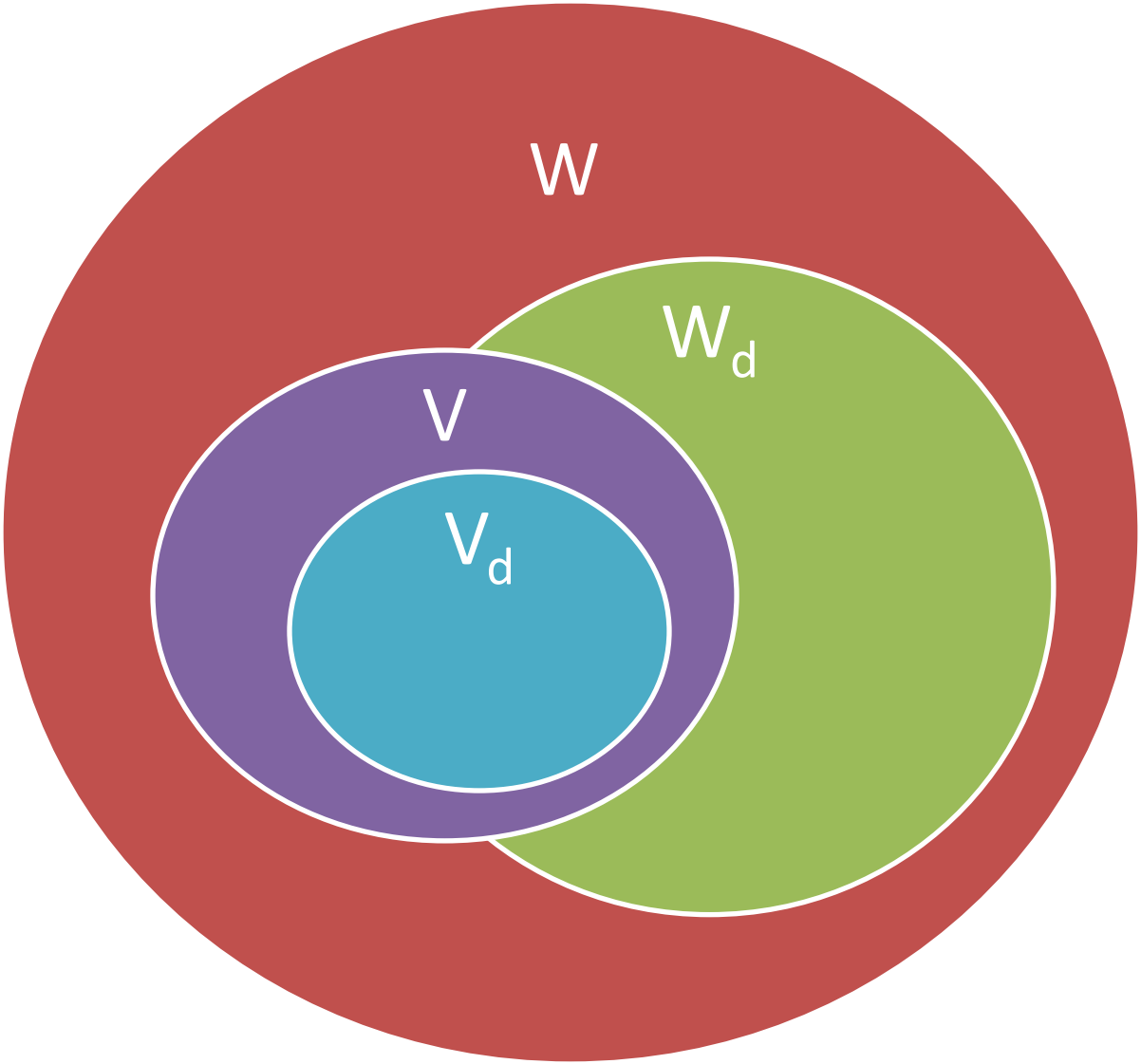
What does the future hold?

Notional



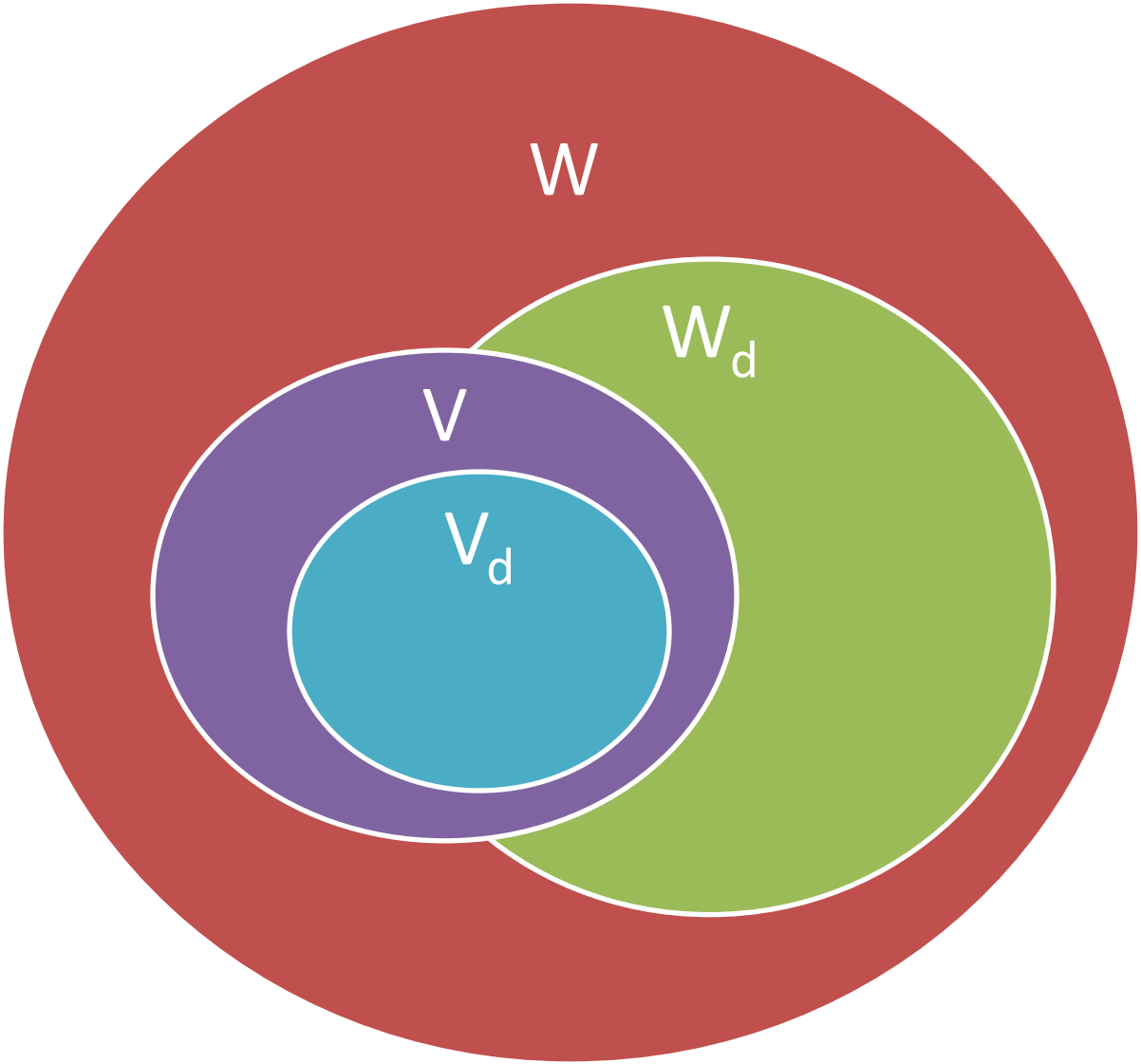
We know it's *not* this, at least not in the near-term

Notional



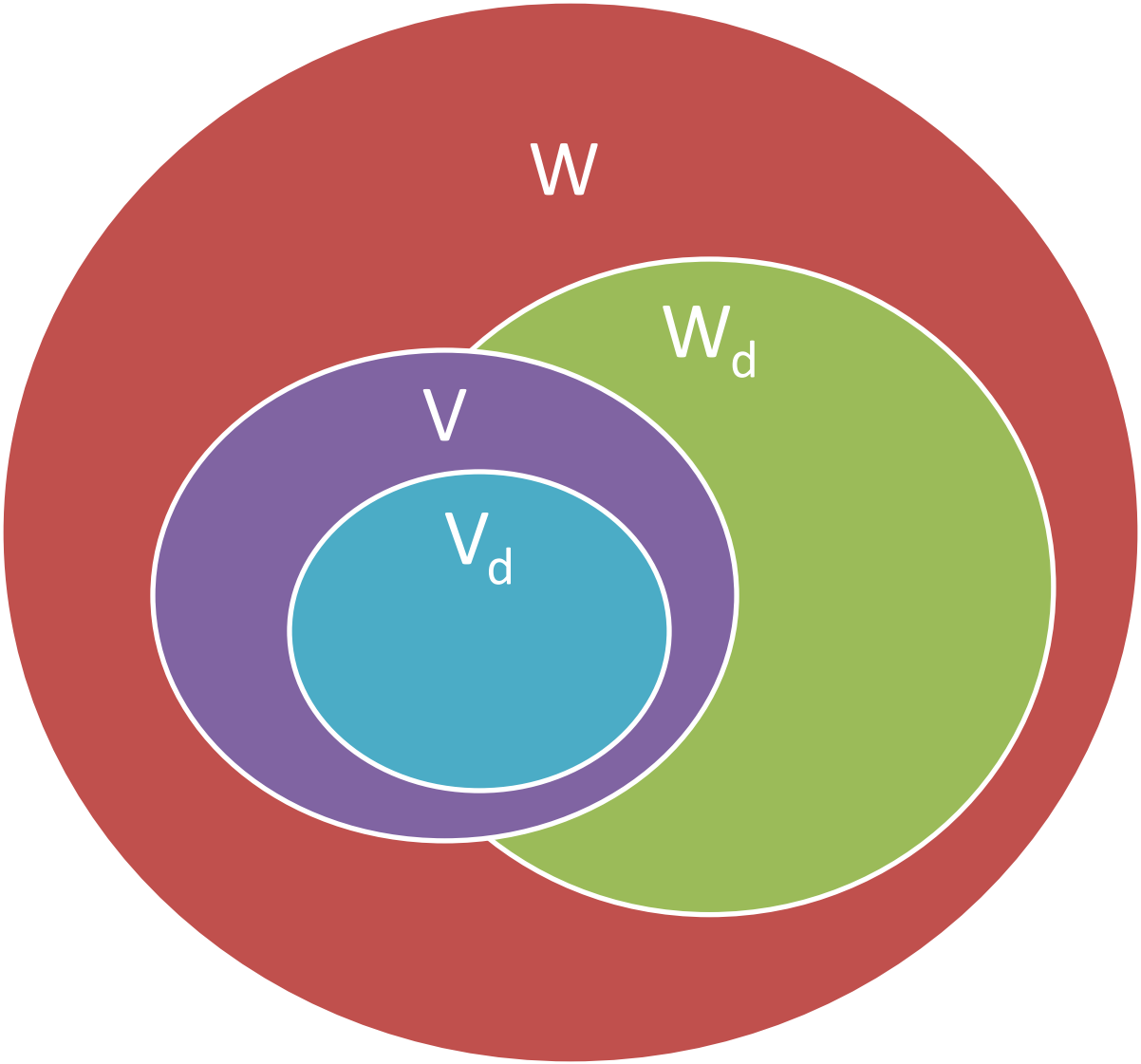
Maybe the problem grows unbounded?

Notional



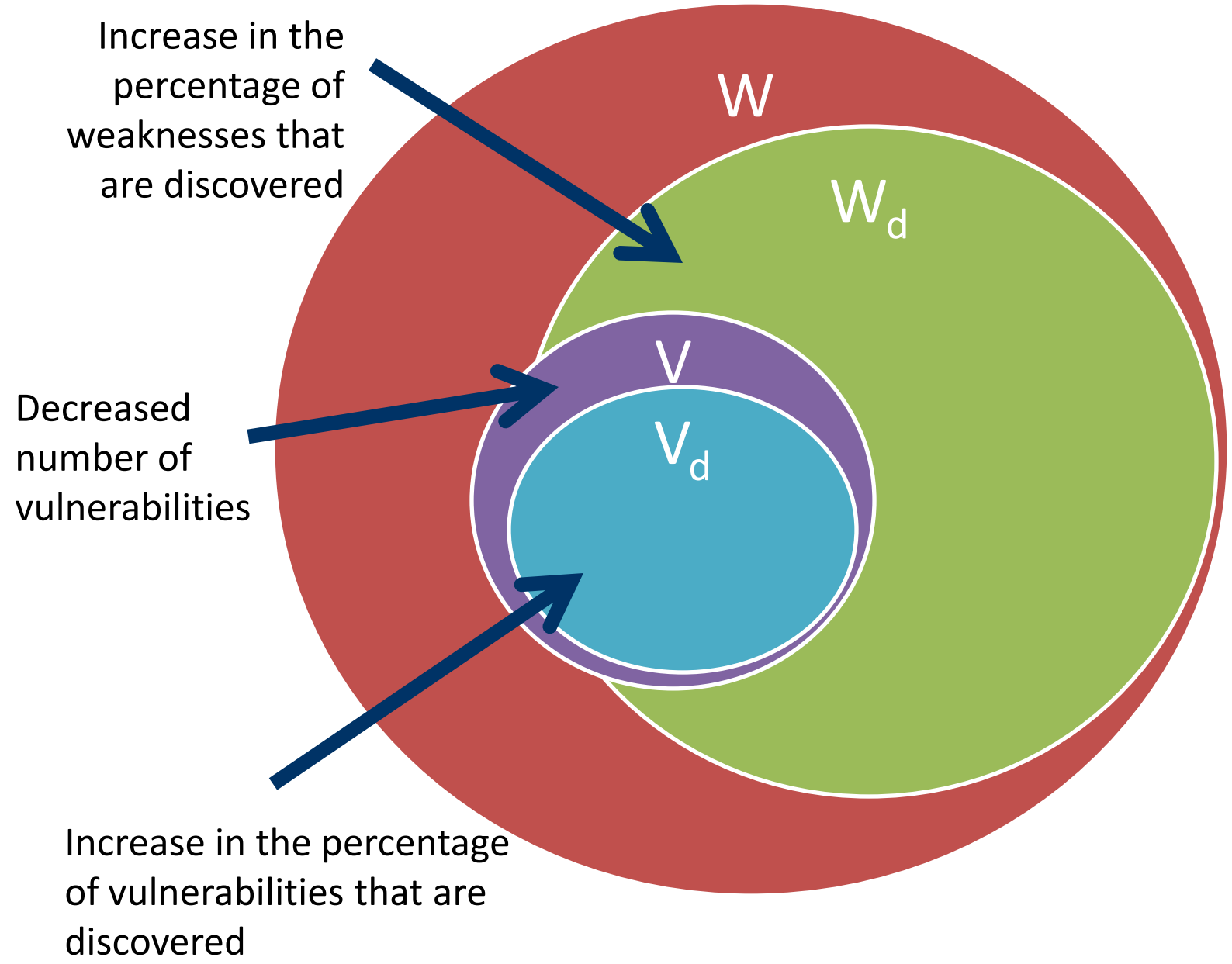
Maybe just some things get worse?

Notional



One reasonable near-term goal

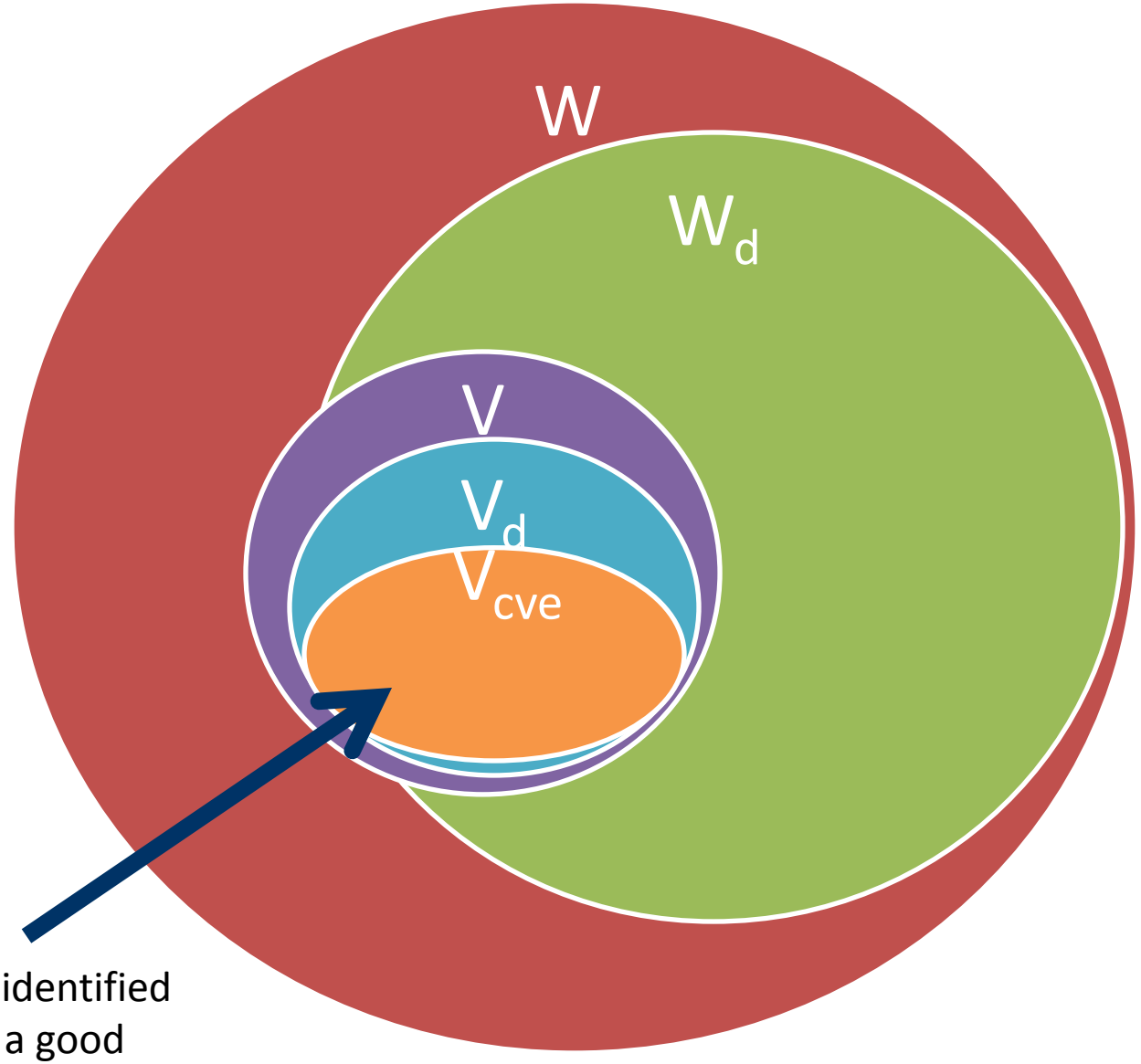
Notional



Is this really better? **Yes**

For the software we're responsible for

Notional



Vulnerabilities identified with a CVE are a good starting point

where should we start?

Dictionary of software weakness *types*

- CWE ID
- Name
- Description
- Alternate Names
- Applicable Platforms
- Applicable Languages
- **Technical Impacts**
- Potential Mitigations
- **Observed Instances (CVE's)**
- **Related Attack Patterns (CAPEC's)**
- Examples

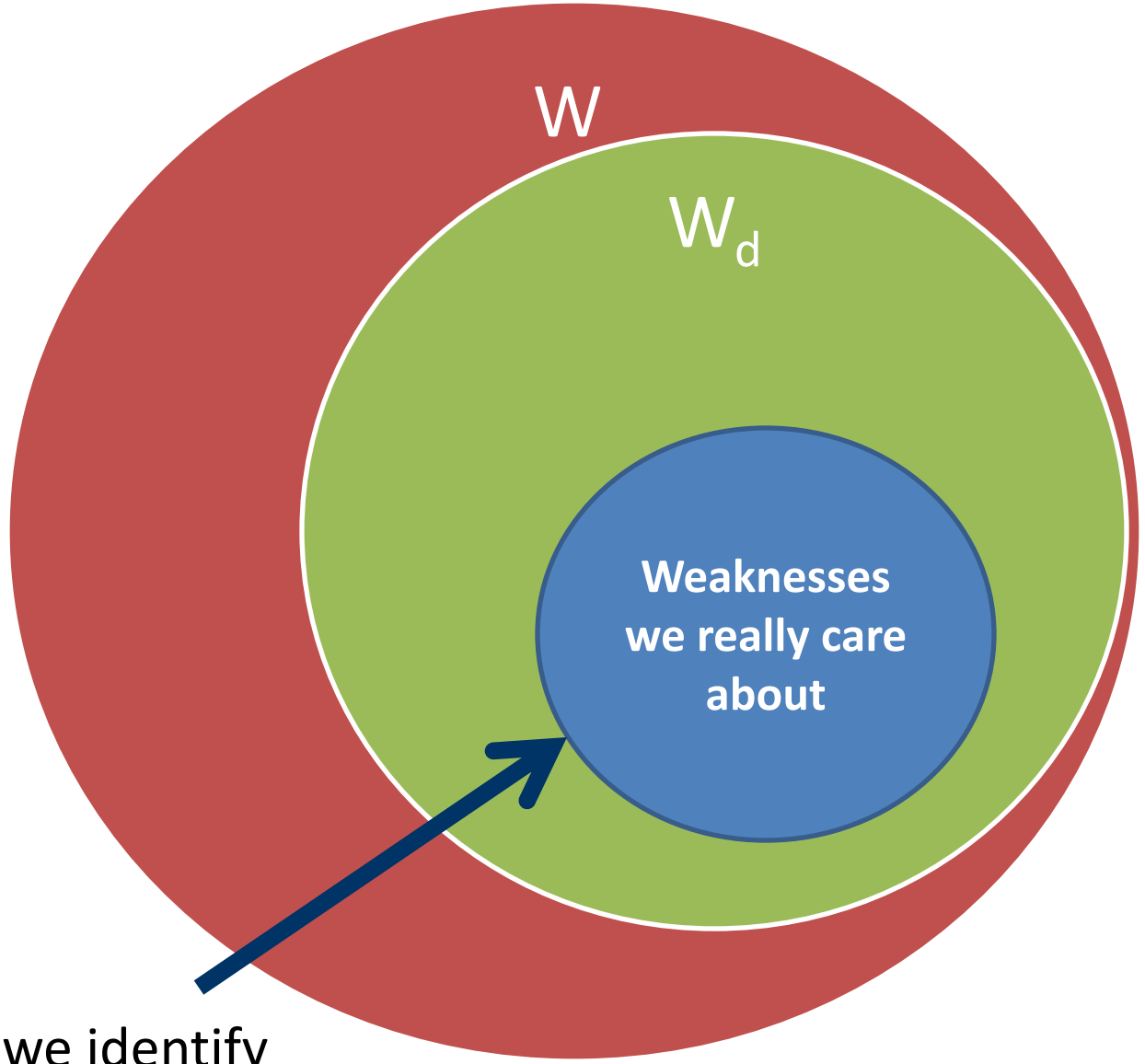
Plus much, much more

860+ entries in a tree-structure

Common Weakness Enumeration (CWE)

For the software we're responsible for

Notional



How do we identify these?

which weaknesses are most important?

Prioritizing weaknesses to be mitigated

OWASP
The Open Web Application Security Project
http://www.owasp.org

OWASP Top 10

CWE/SANS Top 25

Lists are a good start but they are designed to be broadly applicable

We would like a way to specify priorities based on business/mission risk

Common Weakness Risk Analysis Framework (CWRAF)

*How do I **identify** which of the 800+ CWE's are most important for my specific business domain, technologies and environment?*

Common Weakness Scoring System (CWSS)

*How do I **rank** the CWE's I care about according to my specific business domain, technologies and environment?*

How do I identify and score weaknesses important to my organization?

Common Weakness Risk Analysis Framework (CWRAF)

Technical Impacts

1. Modify data
2. Read data
3. DoS: unreliable execution
4. DoS: resource consumption
5. Execute unauthorized code or commands
6. Gain privileges / assume identity
7. Bypass protection mechanism
8. Hide activities

Weightings

W1=0
W2=0
W3=10
W4=4
W5=10
W6=0
W7=0
W8=0

Layers

1. System
2. Application
3. Network
4. Enterprise

Technical Impact Scorecard

Multiple pieces – we'll focus on "Vignettes"

CWRAF: Technical Impact Scorecard

and each technical impact

	MD	RD	UE	RC	EA	GP	BP	HA
Application								
System		**8**						
Network								
Enterprise								**3**

For each layer

assign a weighting from 0 to 10

CWRAF: Technical Impact Scorecard

	MD	RD	UE	RC	EA	GP	BP	HA
Application	9	7	3	2	10	8	7	2
System	8	8	4	2	10	9	5	1
Network	9	5	6	2	10	5	7	1
Enterprise	4	7	6	2	10	6	4	3

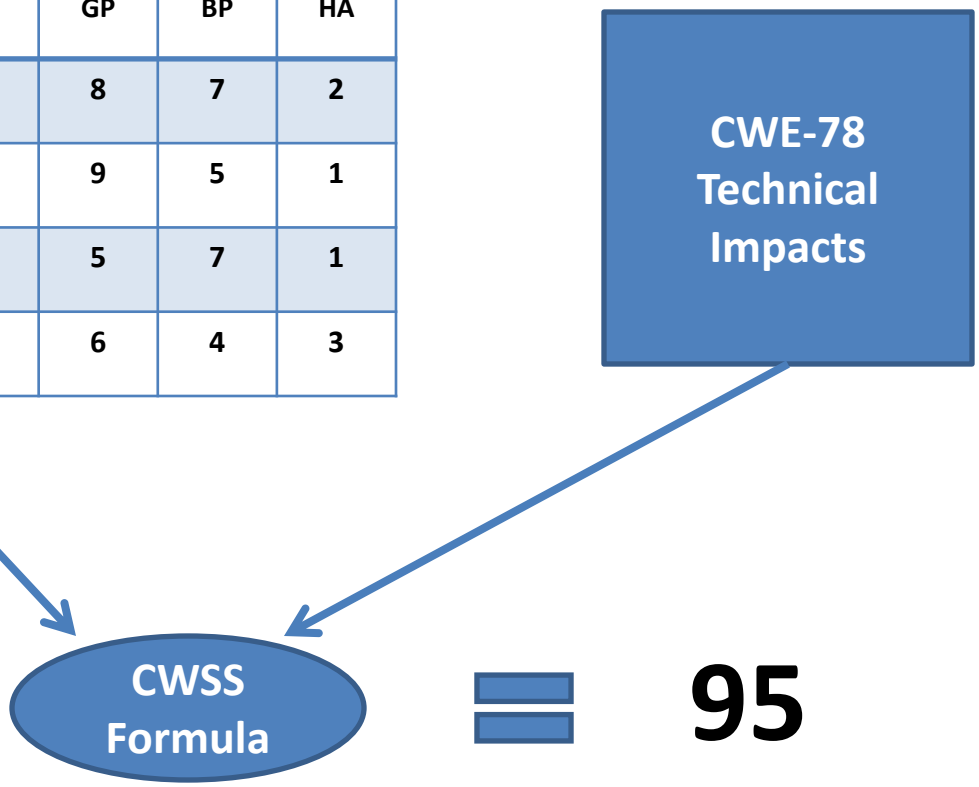
These weightings can now be used to evaluate individual CWE's based on each CWE's Technical Impacts

Note: Values for illustrative purposes only

Note: Values for illustrative purposes only

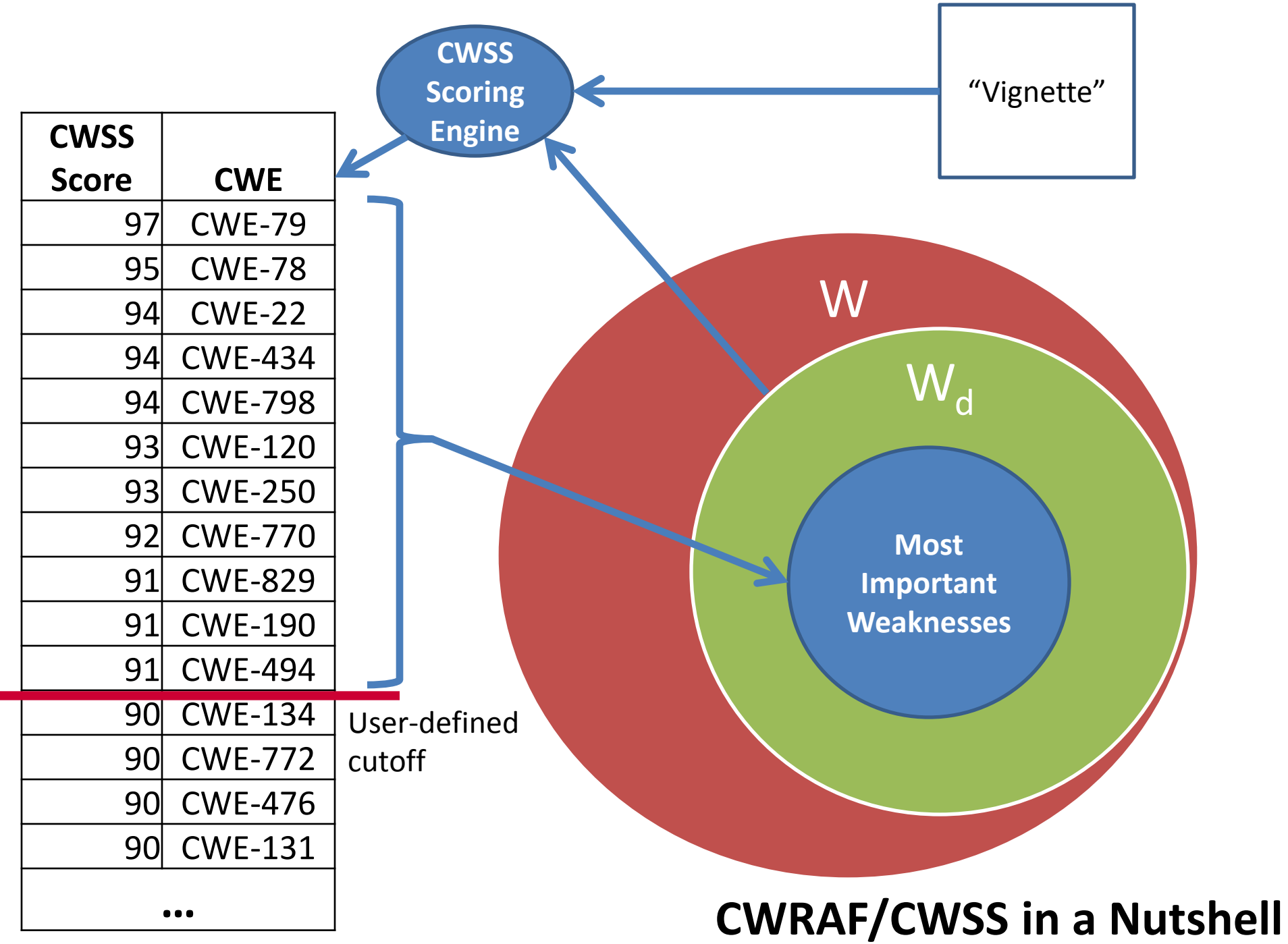
Notional

	MD	RD	UE	RC	EA	GP	BP	HA
Application	9	7	3	2	10	8	7	2
System	8	8	4	2	10	9	5	1
Network	9	5	6	2	10	5	7	1
Enterprise	4	7	6	2	10	6	4	3



CWSS Score for CWE-78 ***for this vignette***

Common Weakness Scoring System (CWSS)


CWSS Score
CWE
97 CWE-79
95 CWE-78
94 CWE-22
94 CWE-434
94 CWE-798
93 CWE-120
93 CWE-250
92 CWE-770
91 CWE-829
91 CWE-190
91 CWE-494
90 CWE-134
90 CWE-772
90 CWE-476
90 CWE-131
...
User-defined cutoff
CWSS Scoring Engine
"Vignette"
W
W_d
Most Important Weaknesses
CWRAF/CWSS in a Nutshell

Organizations that have declared plans to work on CWRAF Vignettes and Technical Scorecards to help evolve CWRAF to meet their customer's and the community's needs for a scoring system for software errors.

DTCC®

CISQ

EC-Council

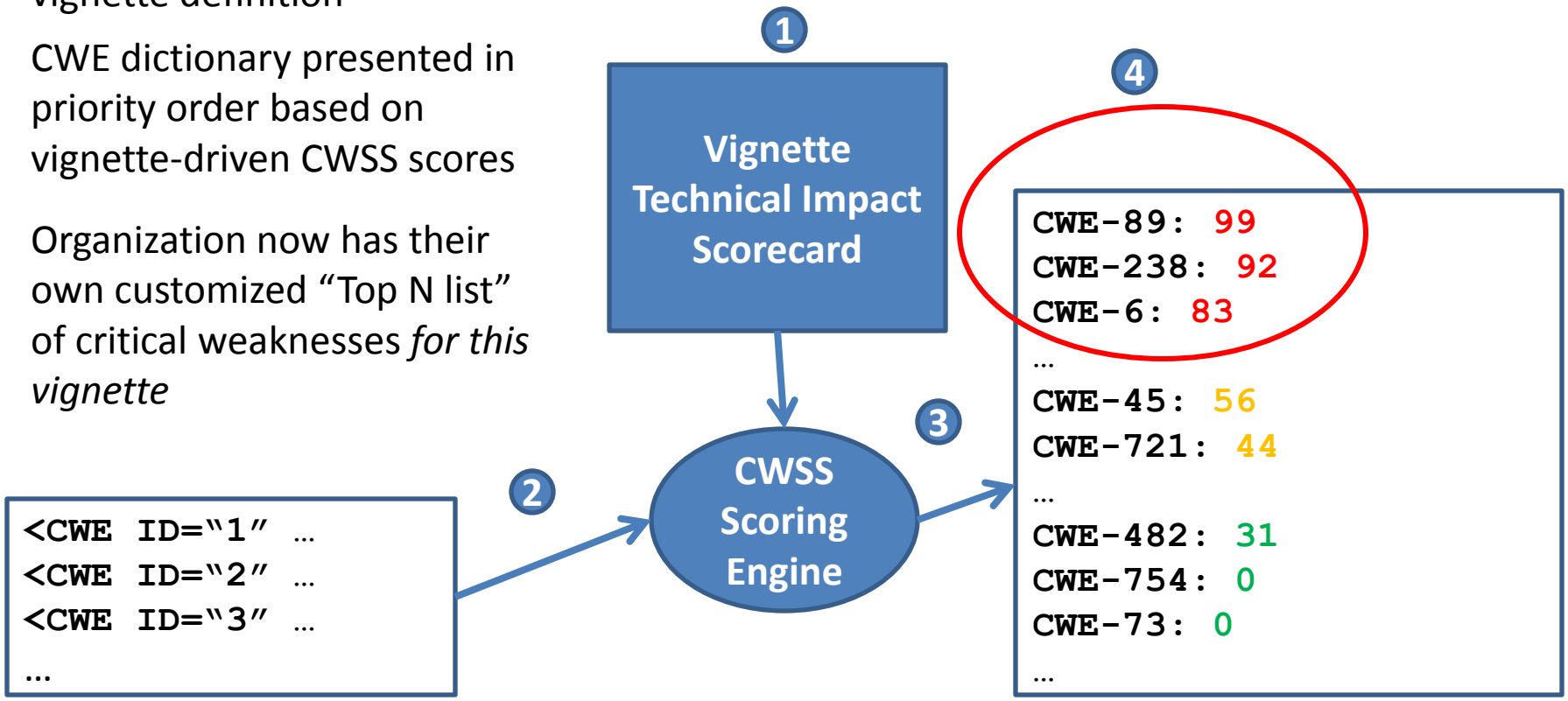
OWASP
The Open Web Application Security Project

SAIC®

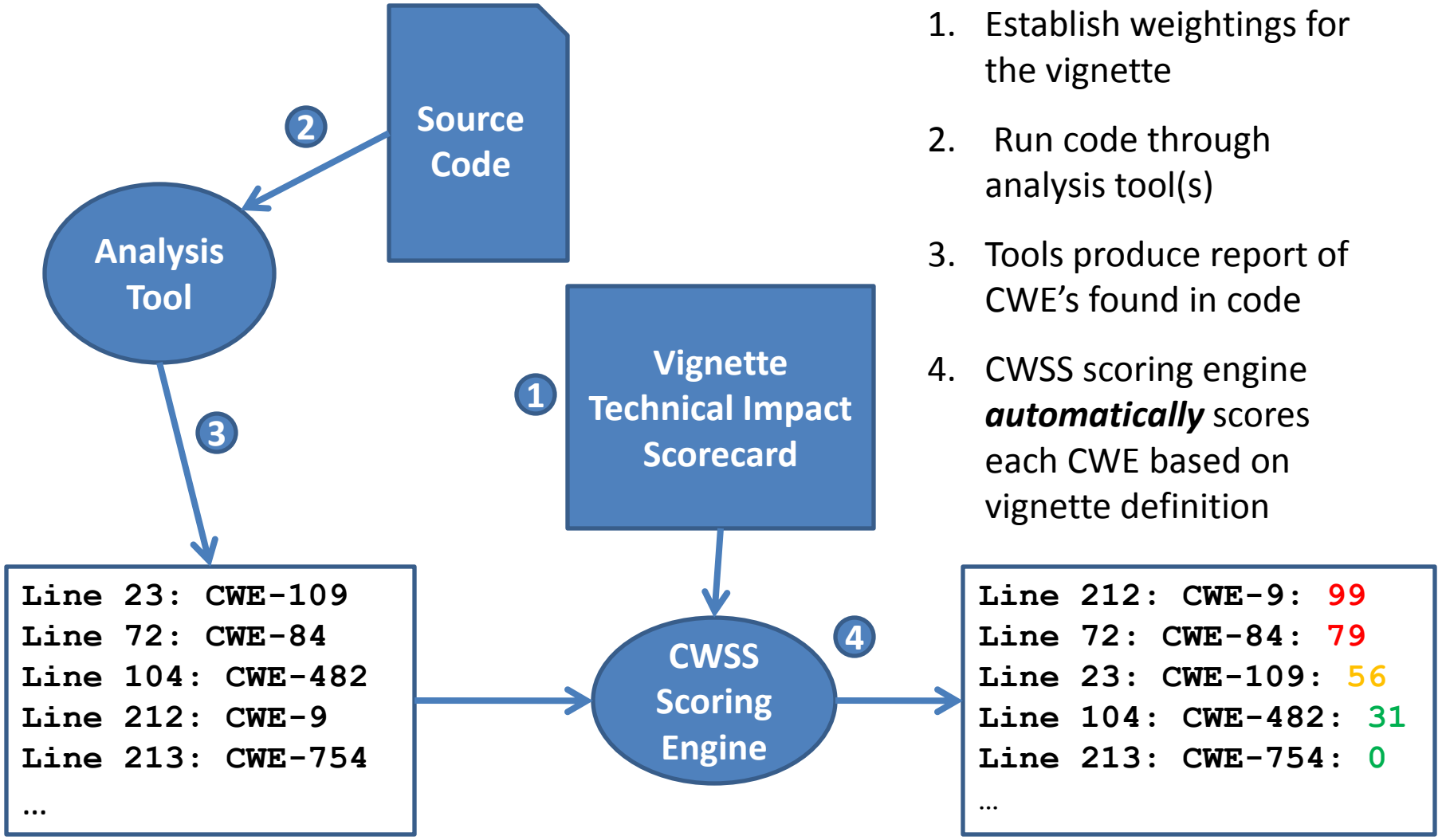
How do you score weaknesses using CWSS?

1. Establish weightings for the vignette
2. CWSS scoring engine processes each relevant CWE entry and ***automatically*** scores each CWE based on vignette definition
3. CWE dictionary presented in priority order based on vignette-driven CWSS scores
4. Organization now has their own customized "Top N list" of critical weaknesses *for this vignette*

Step 1 is only done once – the rest is automatic



How do you score weaknesses discovered in code using CWSS?



1. Establish weightings for the vignette
2. Run code through analysis tool(s)
3. Tools produce report of CWE's found in code
4. CWSS scoring engine ***automatically*** scores each CWE based on vignette definition

Step 1 is only done once – the rest is automatic

Organizations that have declared plans to support CWSS in their future offerings and are working to help evolve CWSS to meet their customer's and the community's needs for a scoring system for software errors.

More Technical Details for CWSS and the Top 25

CWSS Metric Groups

Base Finding Group

- Technical Impact
- Acquired Privilege
- Acquired Privilege Layer
- Internal Control Effectiveness
- Finding Confidence

Attack Surface Group

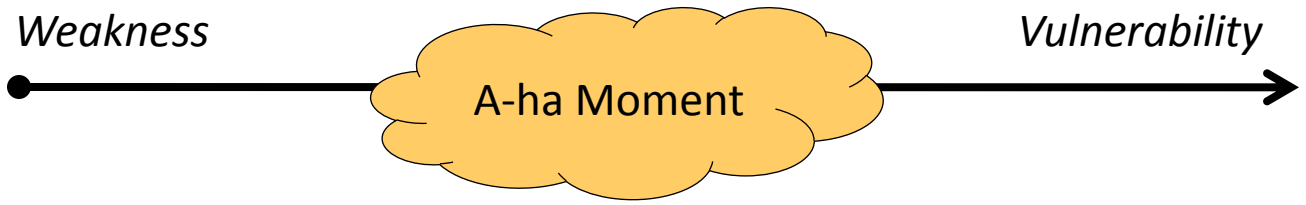
- Required Privilege
- Required Privilege Layer
- Access Vector
- Authentication Strength
- Authentication Instances
- Level of Interaction
- Deployment Scope

Environmental Group

- Business Impact
- Likelihood of Discovery
- Likelihood of Exploit
- External Control Effectiveness
- Remediation Cost
- Prevalence

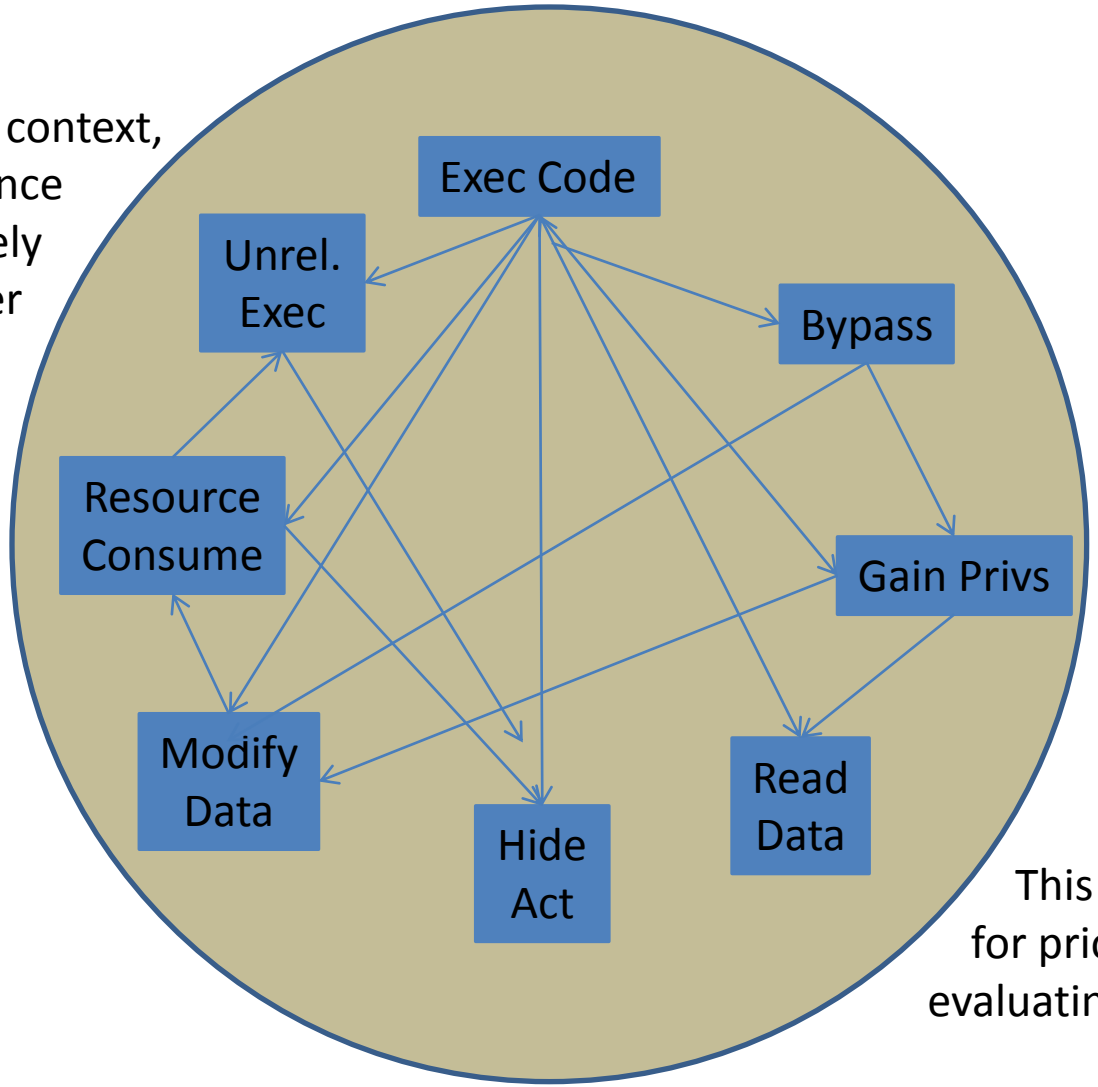
CWSS vs. CVSS

CVSS	CWSS
Mature	New
Focuses on impact to system	Considers impact to System, Application, Network, or Enterprise (SANE)
Used after vulnerability is discovered	Applied the moment there is suspicion
1-50 per software package	Thousands of “findings” per package
Must be manually performed	Can be partially automated
Discrete-yet-numeric values	Finer-grained “quantitative” support
Environment rarely considered	Environment/business considerations built-in
Applied once for Base score	Refined iteratively
Difficult to apply with incomplete information	Explicitly supports incomplete information



A Hard Problem: The Circle of Technical Impacts

Depending on context, one consequence can immediately lead to another



This is a classic problem for prioritizing issues and evaluating multi-step attacks

CWE/SANS Top 25

- 3 years running
- Latest version published in June 2011
- Survey results from over 25 organizations
- 41 CWE entries nominated
- CWSS 0.8 used to rank results
  - Technical Impact, Prevalence, Likelihood of Exploit
- Coming: pocket guide for mitigating the Top 25 (and other weaknesses, too)

MITRE

SANS

Insecure Interaction Between Components

Rank	CWE ID	Name
[1]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[4]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[9]	CWE-434	Unrestricted Upload of File with Dangerous Type
[12]	CWE-352	Cross-Site Request Forgery (CSRF)
[22]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')

Risky Resource Management

Rank	CWE ID	Name
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[13]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	CWE-494	Download of Code Without Integrity Check
[16]	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[18]	CWE-676	Use of Potentially Dangerous Function
[20]	CWE-131	Incorrect Calculation of Buffer Size
[23]	CWE-134	Uncontrolled Format String
[24]	CWE-190	Integer Overflow or Wraparound

Porous Defenses

Rank	CWE ID	Name
[5]	CWE-306	Missing Authentication for Critical Function
[6]	CWE-862	Missing Authorization
[7]	CWE-798	Use of Hard-coded Credentials
[8]	CWE-311	Missing Encryption of Sensitive Data
[10]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	CWE-250	Execution with Unnecessary Privileges
[15]	CWE-863	Incorrect Authorization
[17]	CWE-732	Incorrect Permission Assignment for Critical Resource
[19]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[21]	CWE-307	Improper Restriction of Excessive Authentication Attempts
[25]	CWE-759	Use of a One-Way Hash without a Salt

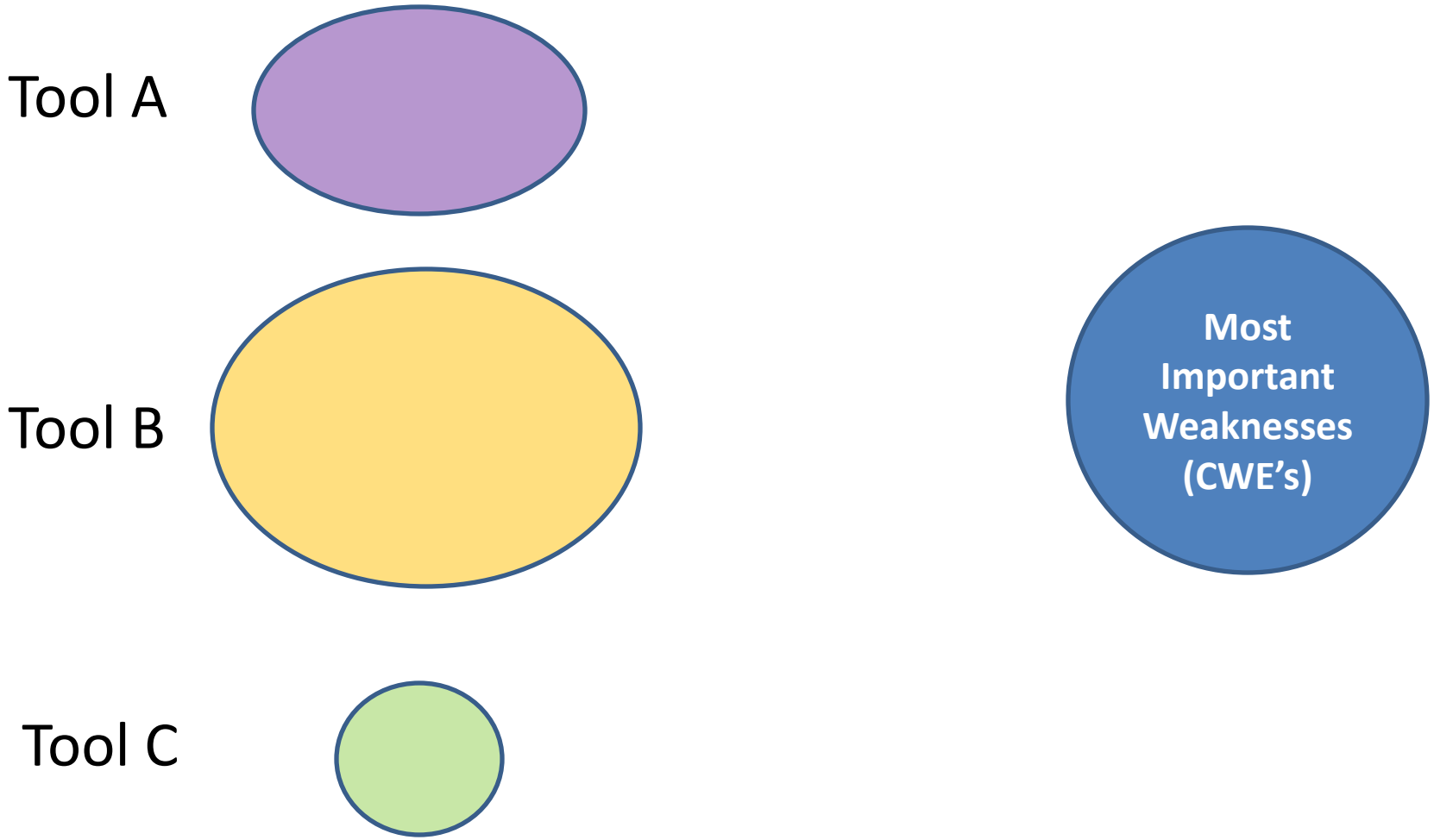
TOP 25

Direct Contributors to the 2011 CWE/SANS Top 25

• Mark J. Cox	Red Hat Inc.
• Carsten Eiram	Secunia (Denmark)
• Pascal Meunier	CERIAS, Purdue University
• Razak Ellafi & Bonsignour	CAST Software
• David Maxwell	NetBSD
• Cassio Goldschmidt & Mahesh Saptarshi	Symantec Corporation
• Chris Eng	Veracode, Inc.
• Paul Anderson	Grammatech Inc.
• Masato Terada	Information-Technology Promotion Agency (IPA) (Japan)
• Bernie Wong	IBM
• Dennis Seymour	Ellumen, Inc.
• Kent Landfield	McAfee
• Hart Rossman	SAIC
• Jeremy Epstein	SRI International
• Matt Bishop	UC Davis
• Adam Hahn & Sean Barnum	MITRE
• Jeremiah Grossman	White Hat Security
• Kenneth van Wyk	KRvW Associates
• Bruce Lowenthal	Oracle Corporation
• Jacob West	Fortify Software, an HP Company
• Frank Kim	ThinkSec
• Christian Heinrich (Australia)	
• Ketan Vyas	Tata Consultancy Services (TCS)
• Joe Baum	Motorola Solutions
• Matthew Coles, Aaron Katz & Nazira Omuralieva	RSA, the Security Division of EMC
• National Security Agency (NSA) Information Assurance Division	
• Department of Homeland Security (DHS) National Cyber Security Division	

CWE Coverage Claims Representation

Set of CWE's tool *claims* to cover



Which static analysis tools find the CWE's I care about?

SwA Working Groups – Next meeting: Week of Nov 28 @ MITRE in McLean, VA

All SwA Program events are free and open to the public

SwA Forum – Next Forum: Week of March 26, 2012 @ MITRE in McLean, VA

SwA Websites: www.us-cert.gov/swa

Making Security Measureable: measurablesecurity.mitre.org

Email: software.assurance@dhs.gov

Software Assurance Resources

Questions?

thank you.

Extra Slides

me:

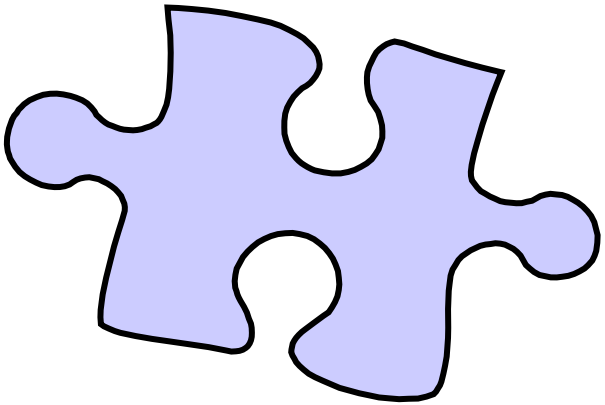
Richard Struse
Deputy Director
Software Assurance
National Cyber Security Div.

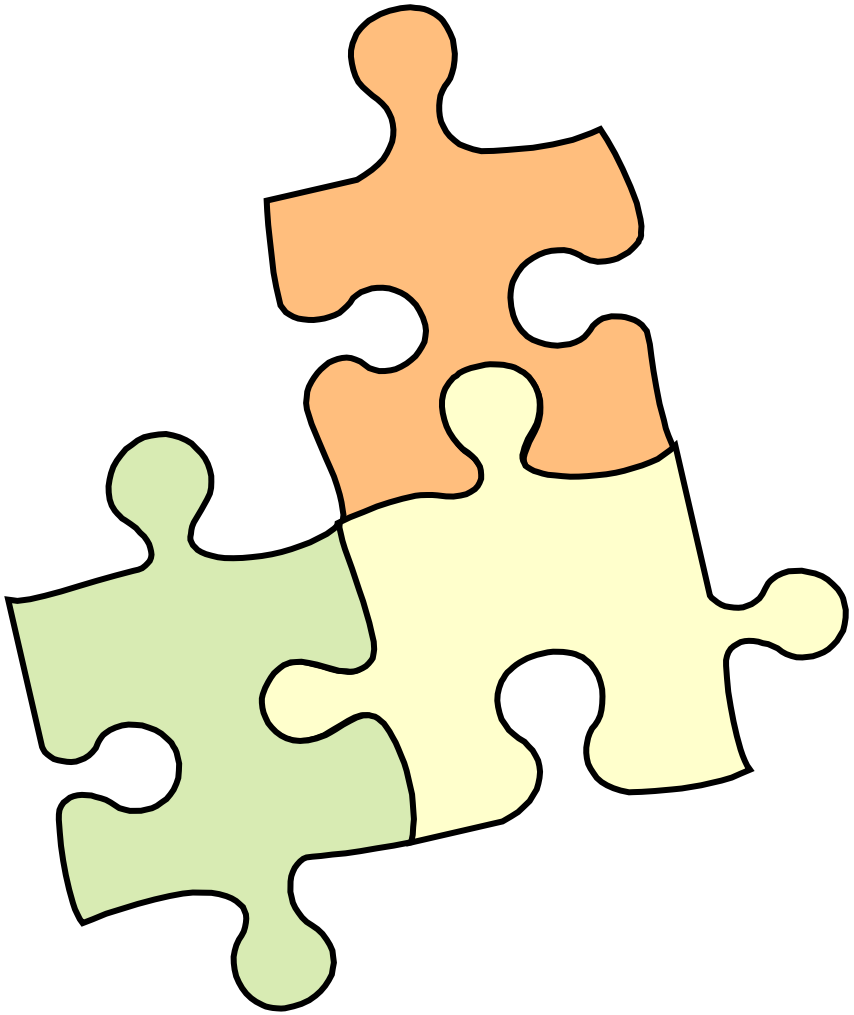
U.S. Department of Homeland Security
National Protection & Programs Directorate

richard.struse@dhs.gov

Homeland Security

Automation is *one piece*





of the SwA puzzle.

automation can help…

- Common Weakness Enumeration (**CWE**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Common Weakness Enumeration (**CWE**)
- Common Weakness Risk Analysis Framework (**CWRAF**)
- Common Weakness Scoring System (**CWSS**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Security Content Automation Protocol (**SCAP**) Components, including:
- Common Vulnerabilities and Exposures (**CVE**)
- Open Vulnerability Assessment Language (**OVAL**)

Differing levels of maturity…

Effort	Maturity
CVE	Very Mature
OVAL	Very Mature
CWE	Mature
CAPEC	Somewhat Mature
CWE CCR	Brand-new
CWSS	Brand-new
CWRAF	Brand-new

We encourage you to get involved in these communities

Dictionary of publicly-disclosed vulnerabilities with unique identifiers

- CVE ID
- Status
- Description
- References

Note: Each CVE entry is the result of expert analysis to verify, de-conflict and de-duplicate public vulnerability disclosures

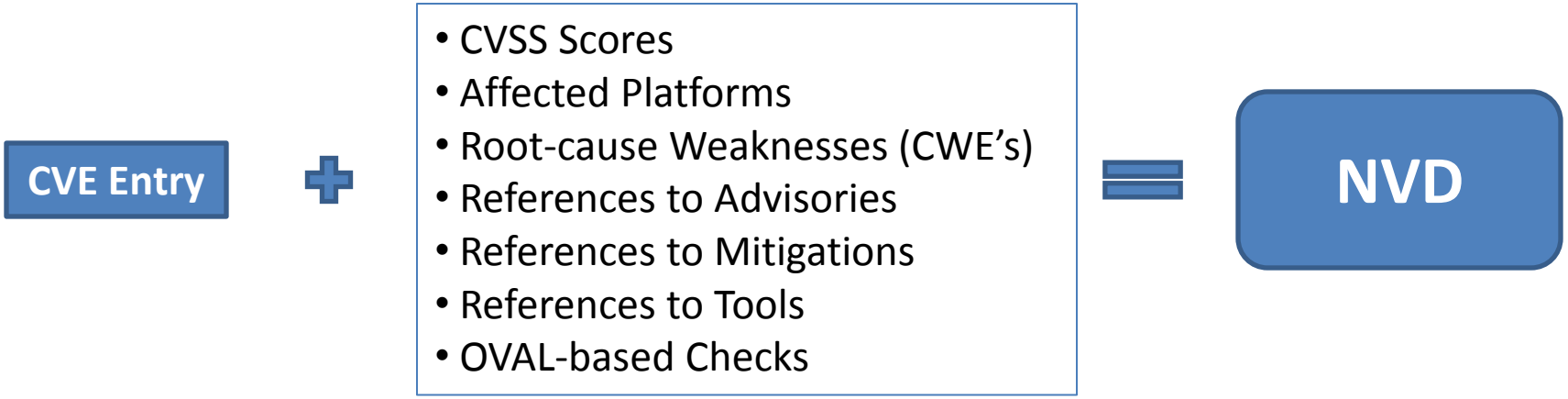
CVE entries feed into *NVD*

assert(CVE != Bug_Database);

47,258 entries (as of last week)

Common Vulnerabilities and Exposures (CVE)

National Vulnerability Database (NVD)



U.S. government repository of standards-based vulnerability management data

website: **nvd.nist.gov**

Common Attack Pattern Enumeration and Classification (CAPEC)

Dictionary of attack types (mostly software)

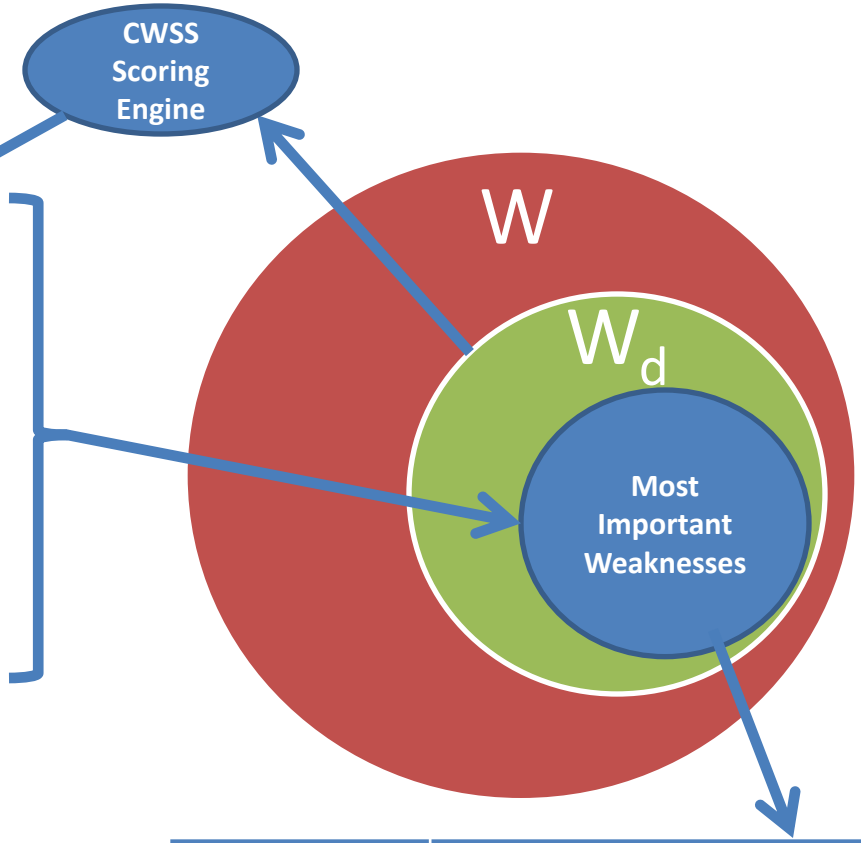
- CAPEC ID
- Name
- Description
- Attack Prerequisites
- Indicators of Attack
- Examples
- **Related Weaknesses (CWE's)**
- Mitigations

Plus much, much more

386 patterns, organized by categories, with views

What types of attacks should I test my system against?

CWSS Score	CWE
97	CWE-79
95	CWE-78
94	CWE-22
94	CWE-434
94	CWE-798
93	CWE-120
93	CWE-250
92	CWE-770
91	CWE-829
91	CWE-190
91	CWE-494
90	CWE-134
90	CWE-772
90	CWE-476
90	CWE-131
...	



CWE	Related CAPEC ID's
CWE-79	CAPEC-232, CAPEC-106, CAPEC-19, …
CWE-78	CAPEC-108, CAPEC-15, CAPEC-43, CAPEC-6, …
…	…

Common Attack Pattern Enumeration and Classification

automation can help - *today*…

- Common Weakness Enumeration (**CWE**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Common Weakness Enumeration (**CWE**)
- Common Weakness Risk Analysis Framework (**CWRAF**)
- Common Weakness Scoring System (**CWSS**)
- Common Attack Pattern Enumeration and Classification (**CAPEC**)
- CWE Coverage Claims Representation (**CCR**)

- Security Content Automation Protocol (**SCAP**) Components, including:
- Common Vulnerabilities and Exposures (**CVE**)
- Open Vulnerability Assessment Language (**OVAL**)