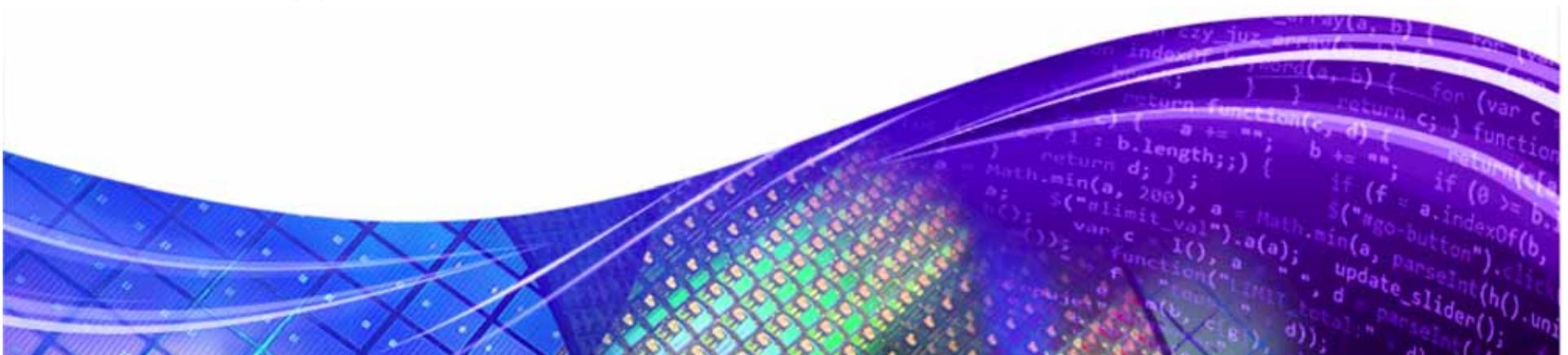


Integrated Analysis and Reporting In Multiple Tools

What cybersecurity and robustness testing tool manufactures should be building towards.

Mike Ahmadi



Agenda

Who should be testing and why

What tools do today

What tools should be moving towards

The challenges

If I had a wish

Who Should Be Testing and Why

Who: All Stakeholders In The Supply Chain

Why: Because all stakeholders are affected by failures in cybersecurity (but in different ways).

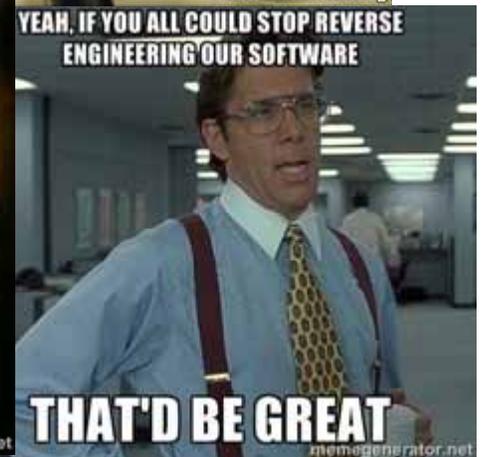
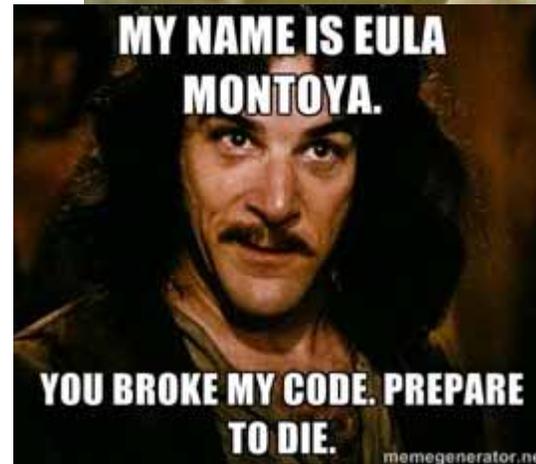


At some point someone (usually the end user) has to **trust...but verify.**

However, not all links in the chain are as well-suited to perform testing.

We Are The Vendor. Trust Us...Or Else!

- CSO of a large software company recently posted a blog admonishing organizations that analyze their code...or hire others to do so.
- This did not bode well with the security world.
- Fortunately, the company took down the blog post and stated that the sentiments expressed in the blog did not represent the organization's sentiment.



A Stopped Clock Is Right Twice A Day

- Despite the ranting tone of the posting, some important points were made.
- To Paraphrase: Finding a lot of vulnerabilities is far less useful than finding ways to
 - determine risks
 - create prioritized action plans.



The Risk Management Game

- *“One CISO told me that he performs risk assessment backwards. He says that he already knows what he needs to do for the next five years to develop adequate security. So he creates some risk numbers that support his contention. Then he works backwards to create types of loss incidents, frequencies, and impacts that produce those numbers. He then refines the input and output to make it all seem plausible. I suggested that his efforts are unethical since his input data and calculations are all fake.” – Donn Parker*
- Determining how to capture the right inputs for a risk calculation is a critical part of creating a useful output.

Undertaking Risk Assessments

- The formula to calculate risks is:
- **Likelihood x Severity**
- (Where likelihood = is it likely to happen)
- 4. Unlikely
- 5. Quite Likely
- 6. Very Likely

OPERATIONAL RISK MANAGEMENT

What is Operational Risk Management?

The risk formula : The risk formula attempts to capture the various components which influence the amount of risk which a hazard may produce for a community or population.

$$\text{Risk} = \text{hazard} \times \text{exposure} \times \frac{\text{vulnerability}}{\text{manageability}}$$

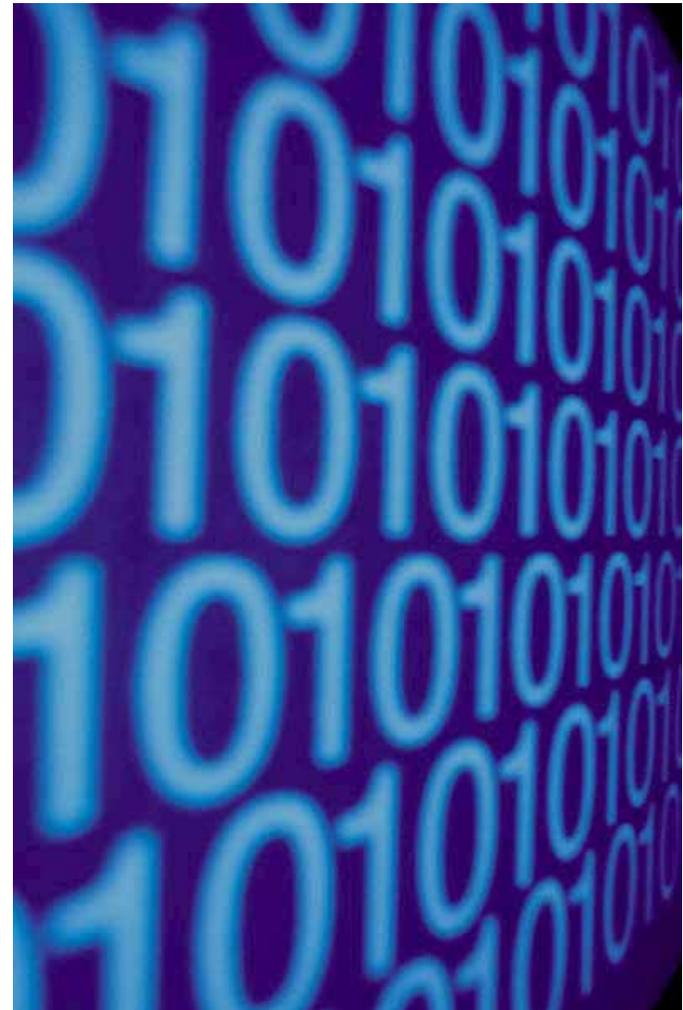
For Training Purpose Only



$$\text{Risk} = \left(\frac{\text{Vulnerability} \times \text{Threat}}{\text{Counter Measure Score}} \right) \times \text{Valuation}$$

Taking Action

- Software and applications have to ship. That is the bottom line. We need software to do things, regardless of the risk.
- Organizations need to sign off on security, and will do so regardless of the veracity of their information.
- True cyber assurance means having a sign off process that enables advancement in technologies and ultimately product features, rather than expending too many cycles reacting to big security challenges.



Types of Automated Tools Testing

And What They Find

- **Dynamic Runtime Analysis** – Finds security issues during runtime, which can be categorized as CWE's
 - **Malformed input testing** (fuzz testing, DoS testing) – Finds zero-days and robustness issues through negative testing.
 - **Behavioral analysis** – Finds exploitable weaknesses by analyzing how the code behaves during “normal” runtime.
- **Software Composition Analysis** – Finds known vulnerabilities and categorizes them as CVE's and via other means.
- **Static Code Analysis** – Finds defects in source code and categorizes them as Cyber Weakness Enumerators (CWE's) and other means
- **Known Malware Testing** – Finds known malware (e.g. viruses and other rogue code).

Generally speaking, all of these tests can be used to enumerate CVE's and CWE's, which can be (and should be) further categorized into prioritized lists.

Some Prioritized Lists To Consider

- SANS CWE Top 25 – A list of the top 25 most commonly encountered Cyber Weakness Enumerators (CWEs), found in (<https://www.sans.org/top25-software-errors/>)
- OWASP Top 10 Vulnerabilities – A list of the 10 Most Critical Web Application Security Risks compiled by OWASP (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)
- Verizon Report Top 10 CVEs – List of the 10 most commonly encountered Common Vulnerabilities and Exposures (CVEs) used in exploits (<http://news.verizonenterprise.com/2015/04/2015-data-breach-report-info/>)

Secure Interaction Resilient Components

Item	Category
1. Secure Interaction Resilient Components	Security
2. Secure Interaction Resilient Components	Security
3. Secure Interaction Resilient Components	Security
4. Secure Interaction Resilient Components	Security
5. Secure Interaction Resilient Components	Security
6. Secure Interaction Resilient Components	Security
7. Secure Interaction Resilient Components	Security
8. Secure Interaction Resilient Components	Security
9. Secure Interaction Resilient Components	Security
10. Secure Interaction Resilient Components	Security

Risky Resource Management

Item	Category
1. Risky Resource Management	Security
2. Risky Resource Management	Security
3. Risky Resource Management	Security
4. Risky Resource Management	Security
5. Risky Resource Management	Security
6. Risky Resource Management	Security
7. Risky Resource Management	Security
8. Risky Resource Management	Security
9. Risky Resource Management	Security
10. Risky Resource Management	Security

Perpetual Defense

Item	Category
1. Perpetual Defense	Security
2. Perpetual Defense	Security
3. Perpetual Defense	Security
4. Perpetual Defense	Security
5. Perpetual Defense	Security
6. Perpetual Defense	Security
7. Perpetual Defense	Security
8. Perpetual Defense	Security
9. Perpetual Defense	Security
10. Perpetual Defense	Security

OWASP Top 10

Item	Category
1. Broken Authentication	Security
2. Injection	Security
3. Sensitive Data Exposure	Security
4. XML External Entity (XXE)	Security
5. Broken Access Control	Security
6. Security Misconfiguration	Security
7. Cross-Site Scripting (XSS)	Security
8. Insecure Deserialization	Security
9. Unvalidated Redirects and Forwards	Security
10. Unvalidated Input	Security



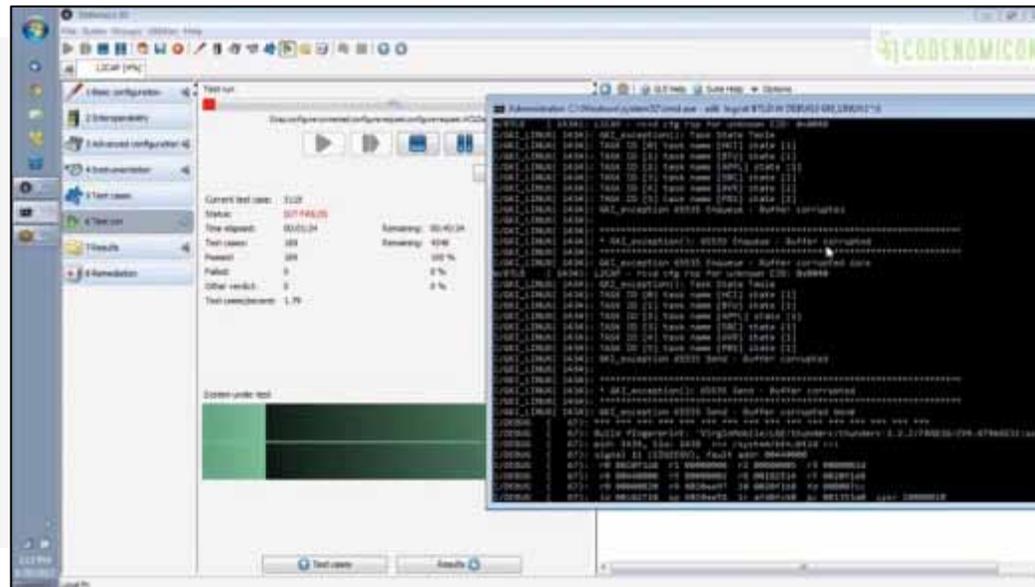
What Can't Be Easily Automated

- What the “dark wizards” of the world of security research find.
- Small numbers of researchers that are the “special forces” of the security world.
- Commonly referred to as hackers.



What Malformed Input Testing Finds

- Essentially, ways to get a system or application to misbehave or fail through misuse (intentional or otherwise).
- This can be as simple as a single bad packet.
- Once failure modes occur they can lead to ways to take down a system or introduce malware (or both).



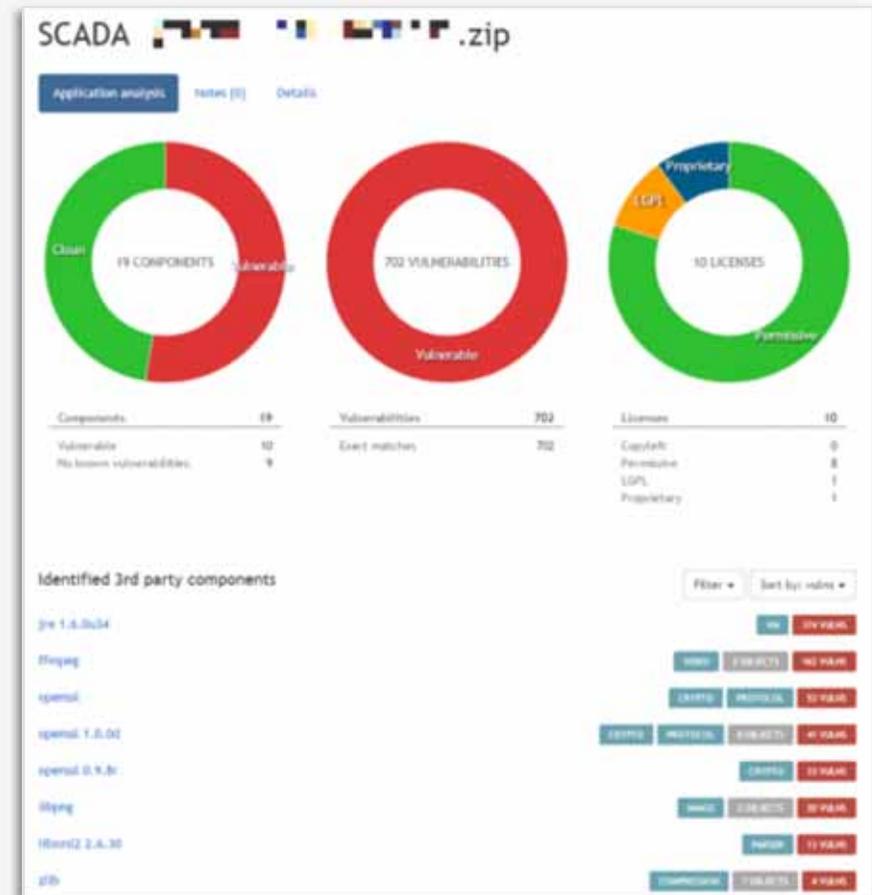
What Behavioral Analysis Finds

- Watches what code is doing while it is running.
- Can be a useful way to eliminate false positives, as long as every possible state is executed during analysis.
- Can determine contextual risk of an exploit.
- Works well with web services and traditional IT systems, more complex for embedded systems and RTOS environments.



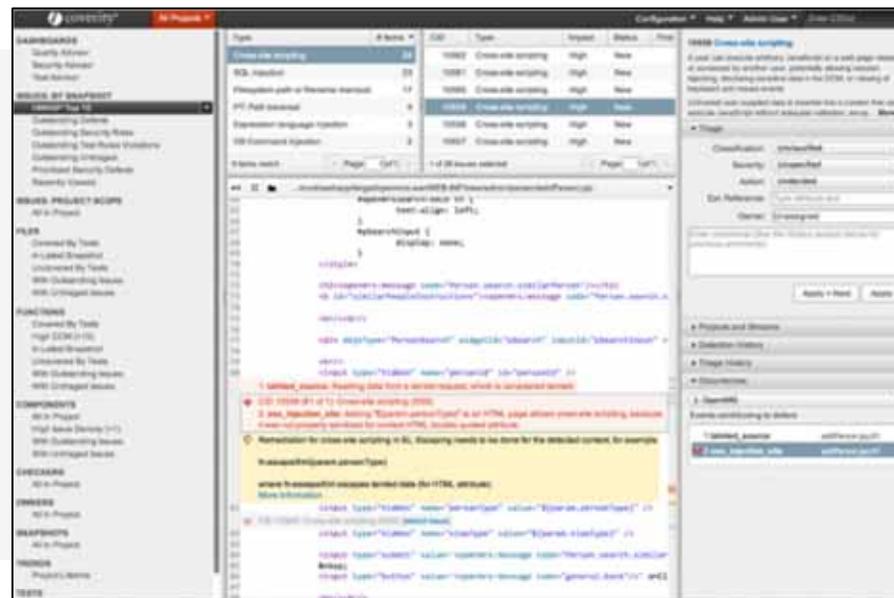
What Software Composition Analysis Finds

- Looks at compiled code and determines what third-party (or proprietary) components it is built from.
- Queries databases of known vulnerabilities for identified components and lists them out. **Finds CVEs.**
- Controversial because all identified vulnerabilities are not necessarily exposed.
- Can automatically track vulnerabilities in a software package over time.



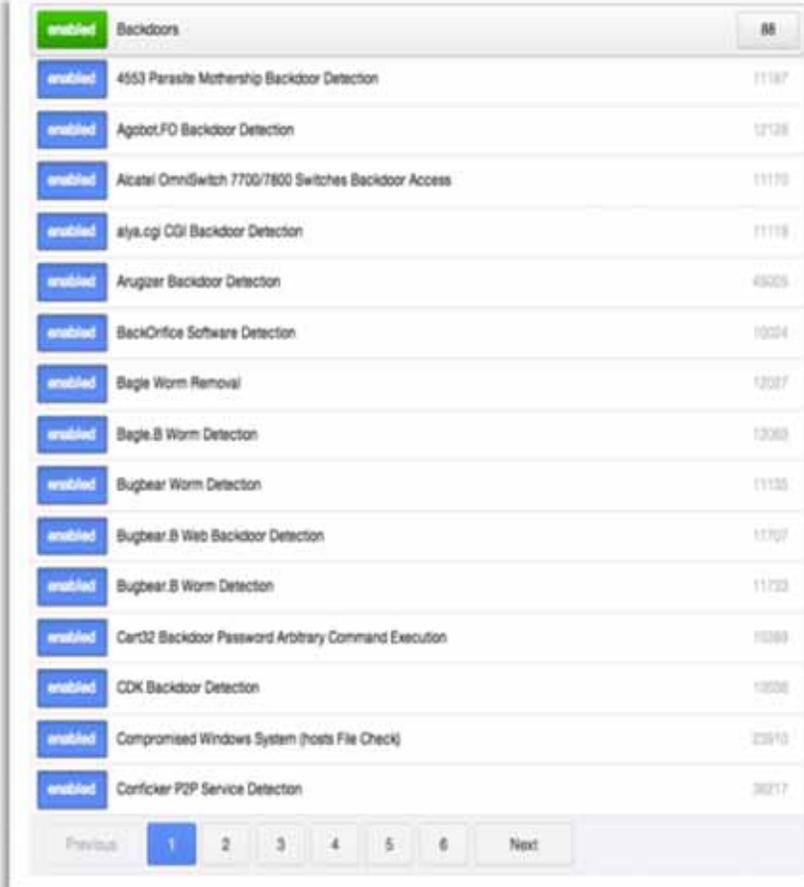
What Static Code Analysis Finds

- Identifies defects in source code.
- Identifies CWEs
- Like software composition analysis, can be controversial because identified defects can range from trivial (low or no real risk) to critical (high risk).



What Known Malware Analysis Finds

- This is the new generation of antivirus type tools with a lot of additional capabilities and features.
- Malware is created to exploit vulnerabilities, or simply run “uninvited” as privileged applications in an environment that allows such actions.
- Tools need to check for existence of malware against a known database. Some tools use heuristics.



The screenshot displays a web-based interface for malware detection. At the top, there is a header bar with a green 'enabled' button, the title 'Backdoors', and a page number '88'. Below this is a list of 16 items, each with a blue 'enabled' button, a description, and a numerical value. The items are:

Enabled	Signature Name	Value
enabled	4553 Parasite Mothership Backdoor Detection	11187
enabled	Agbot.FD Backdoor Detection	12128
enabled	Alcatel OmniSwitch 7700/7800 Switches Backdoor Access	11170
enabled	alya.cgi CGI Backdoor Detection	11119
enabled	Arugzer Backdoor Detection	49005
enabled	BackOffice Software Detection	10204
enabled	Bagle Worm Removal	12027
enabled	Bagle.B Worm Detection	12060
enabled	Bugbear Worm Detection	11135
enabled	Bugbear.B Web Backdoor Detection	11707
enabled	Bugbear.B Worm Detection	11723
enabled	Cert32 Backdoor Password Arbitrary Command Execution	10389
enabled	CDK Backdoor Detection	10206
enabled	Compromised Windows System (hosts File Check)	23910
enabled	Conficker P2P Service Detector	39217

At the bottom of the interface, there is a navigation bar with a 'Previous' button, a series of numbered buttons (1, 2, 3, 4, 5, 6), and a 'Next' button. The number '1' is highlighted in blue.

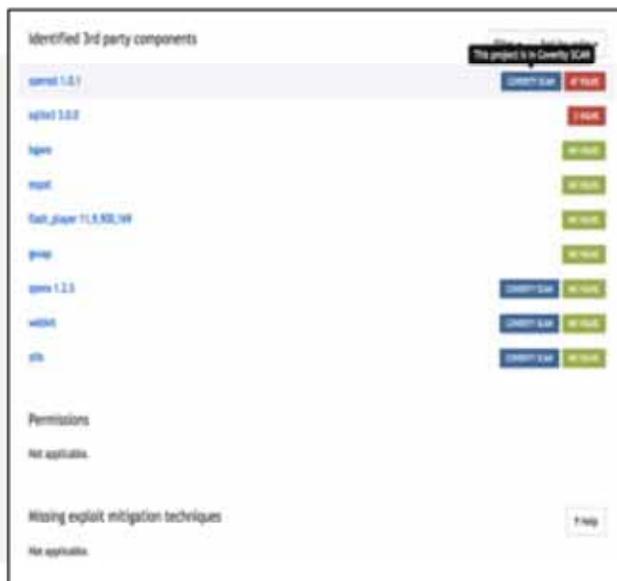
What Security Researchers Find

- Some of what was previously mentioned
- A lot that cannot be easily discovered with automated tools:
 - Physical ports and interfaces
 - Undocumented and hidden services
 - Hidden back doors and passwords
 - Configuration errors
 - Failures in process
- True experts are few and far between and very expensive.
- No real formal training exists
- Tend to stop or ease up on testing once a big exploit emerges...or once a specific target is reached.



Inference Through Multiple Data Points

- Knowing CVEs, CWEs, and defect density is more useful than knowing only one of these.
- Knowing how often a codebase is maintained for defects is more useful than a single scan result.
- Multiple data points draw a better picture.



An Ingredient List

Supplement Facts			
Serving Size		2 fl. oz.	
	Amount per Serving	% Daily Value*	Amount per Serving
Calories	20		40
Sodium	18mg	1%	35mg
Potassium	35mg	1%	70mg
Total Carbohydrate	5g	3%	10g
Dietary Fiber	less than 1g	2%	1g
Sugars	4g		8g
Other Carbohydrate	less than 1g		1g
Vitamin B₃ (niacin, niacinamide)	4mg	20%	8mg
Vitamin B₆ (pyridoxine HCl)	4mg	200%	8mg
Vitamin B₁₂ (cyanocobalamin)	15mcg	250%	30mcg

* Percent Daily Values are based on a 2,000 calorie diet.
† Daily Value not established.

Other Ingredients: Linux Kernel, Zlib, Glibc, OpenSSL

Software bill of materials

Component:	Version	License
bind	9.5.0	ISC
commons-lang	2.4	Apache
openssl	0.9.6f †	Apache
	0.9.7a †	
	0.9.8g †	
	1.0.0j †	
pcre	7.6	BSD
rsync	2.6.9	GPL
tcl	8.5.0	BSD
zlib	1.2.1.2	zlib

† Daily Value not established

Other Ingredients:

Simply knowing software “ingredients” arms a user with an enormous resource for determining risk.

Understanding Context

- Let's use Heartbleed as an example:
 - Scored a 5 CVSS score (not considered critical)
 - Yet, if found in a server application, it is indeed very critical.
 - Not nearly as critical if found in a client application.



If Only...Finding Weaknesses in Software was as Easy as...

Robert A. Martin

Senior Principal Engineer

Cyber Security Center

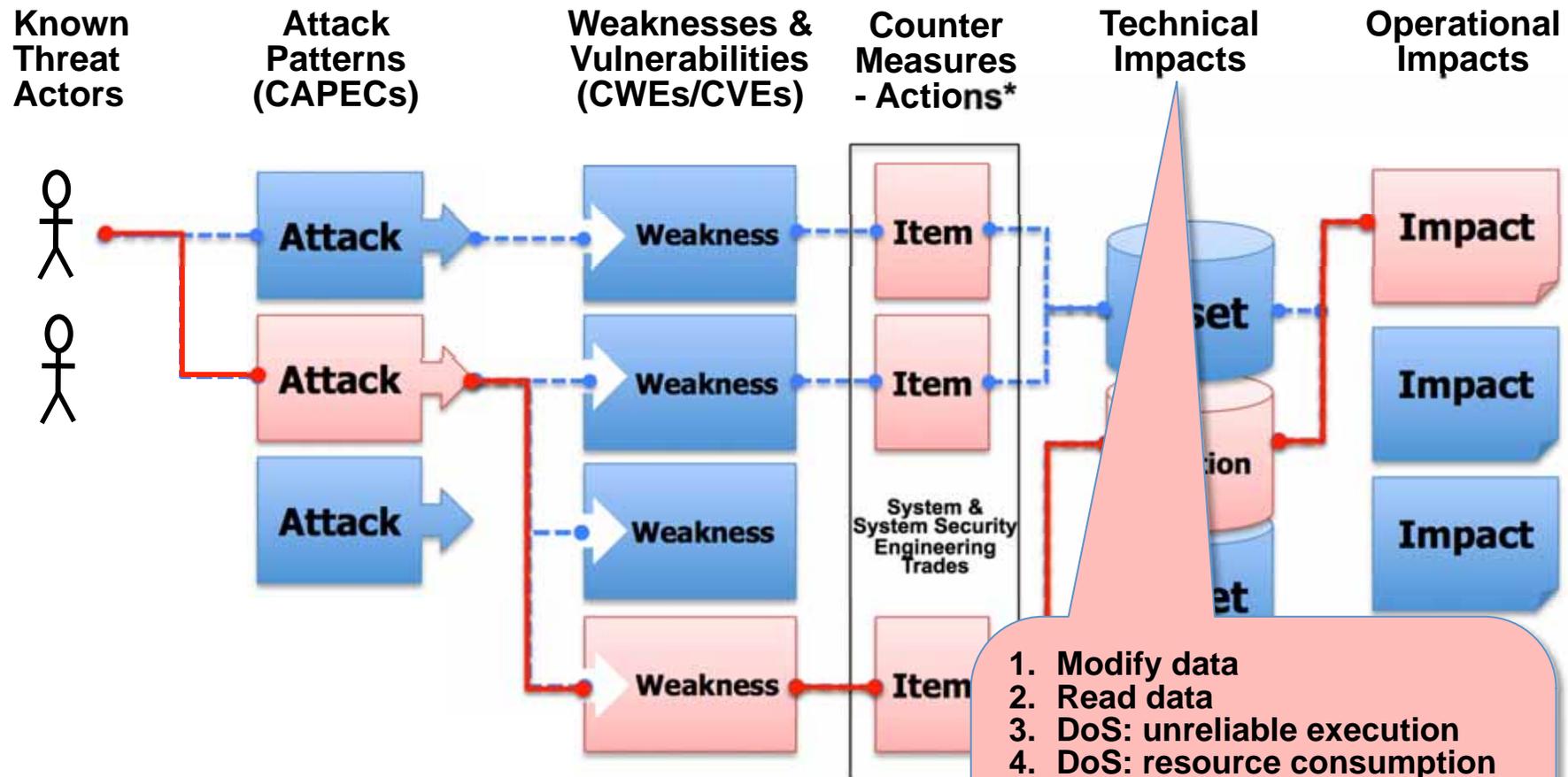
Center for National Security

The MITRE Corporation



MITRE

Assurance About Mitigating the Attacks That Can Impact Operations

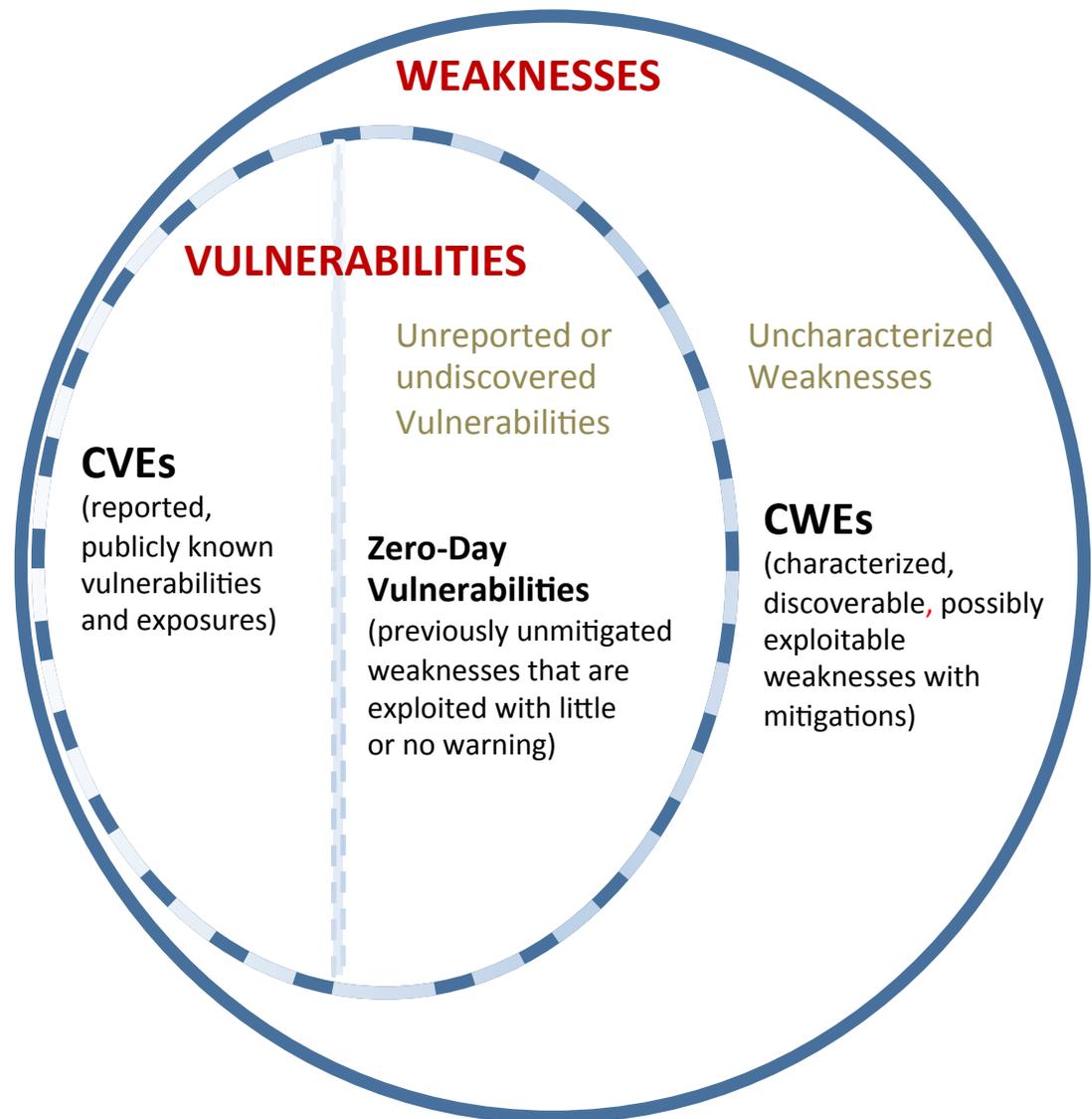


* “Counter Measures - Actions” include: architecture choices; design choices; added security functions, activities & processes; protection schemes; physical decomposition choices; static & dynamic code assessments; design reviews; dynamic testing; and pen testing

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

Exploitable Weaknesses, Vulnerabilities & Exposures

- **Weakness:** mistake or flaw condition in ICT architecture, design, code, or process that, if left unaddressed, could under the proper conditions contribute to a [cyber-enabled capability](#) being vulnerable to exploitation; represents potential source vectors for zero-day exploits -- Common Weakness Enumeration (CWE) <https://cwe.mitre.org/>
- **Vulnerability:** mistake in software that can be directly used by a hacker to gain access to a system or network; **Exposure:** configuration issue of a mistake in logic that allows unauthorized access or exploitation – Common Vulnerability and Exposure (CVE) <https://cve.mitre.org/>
- **Exploit:** take advantage of a weakness (or multiple weaknesses) to achieve a [negative technical impact](#) -- attack approaches from the set of known exploits are used in the Common Attack Pattern Enumeration and Classification (CAPEC) <https://capec.mitre.org>
- The existence (even if only theoretical) of an exploit designed to take advantage of a [weakness](#) (or multiple weaknesses) and achieve a [negative technical impact](#) is what makes a weakness a [vulnerability](#).



Assurance Comes From Managing Weaknesses and the Supporting Evidence



Common Weakness Scoring System (5 Sep 2014)

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
- Level of Interaction
- Deployment Scope

Common Weakness Risk Analysis Framework (CWRAF)

- Vignettes
- Technical Impact Scorecard

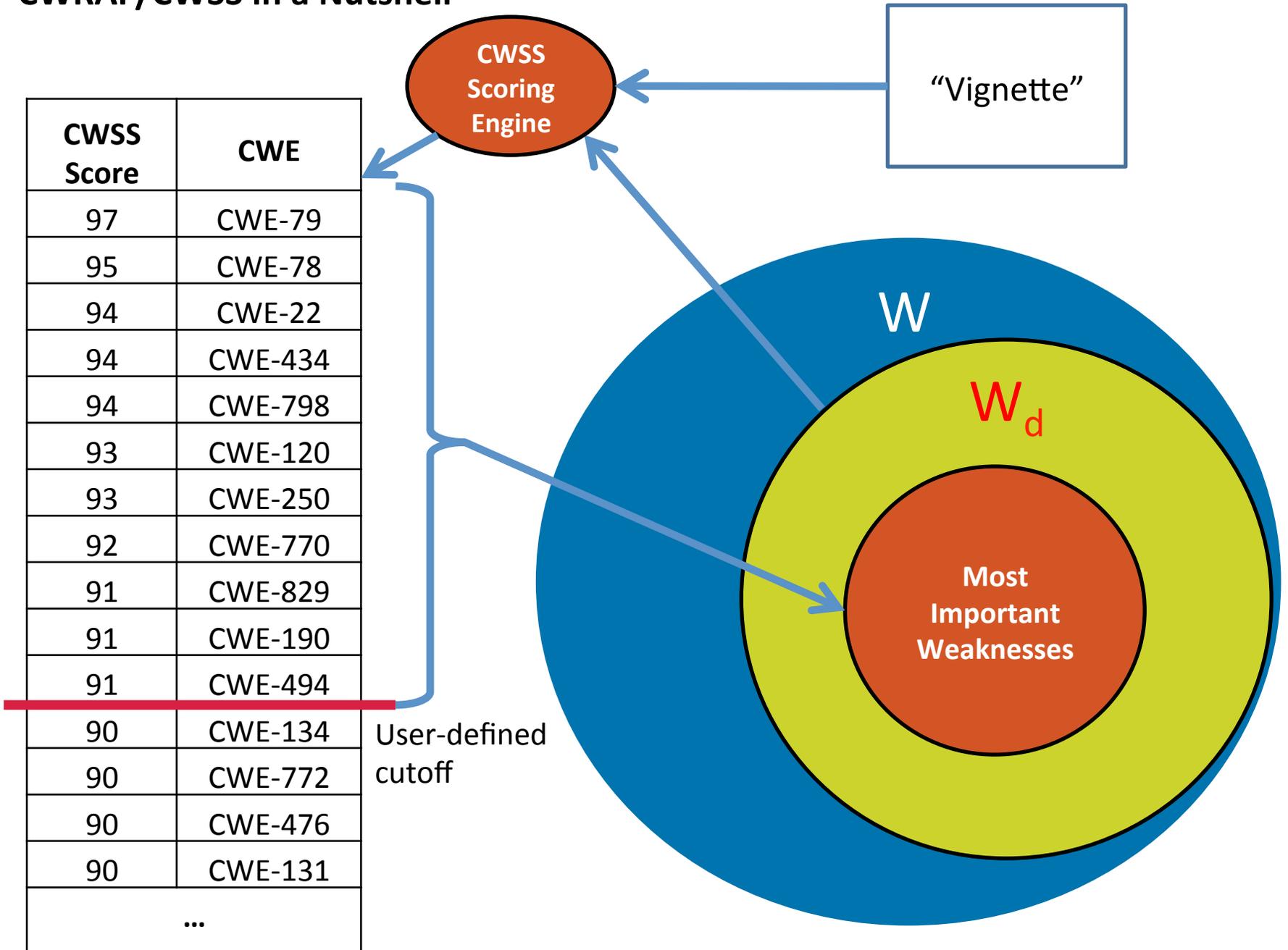
Environmental Group

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

CWSS



CWRAF/CWSS in a Nutshell



User-defined cutoff

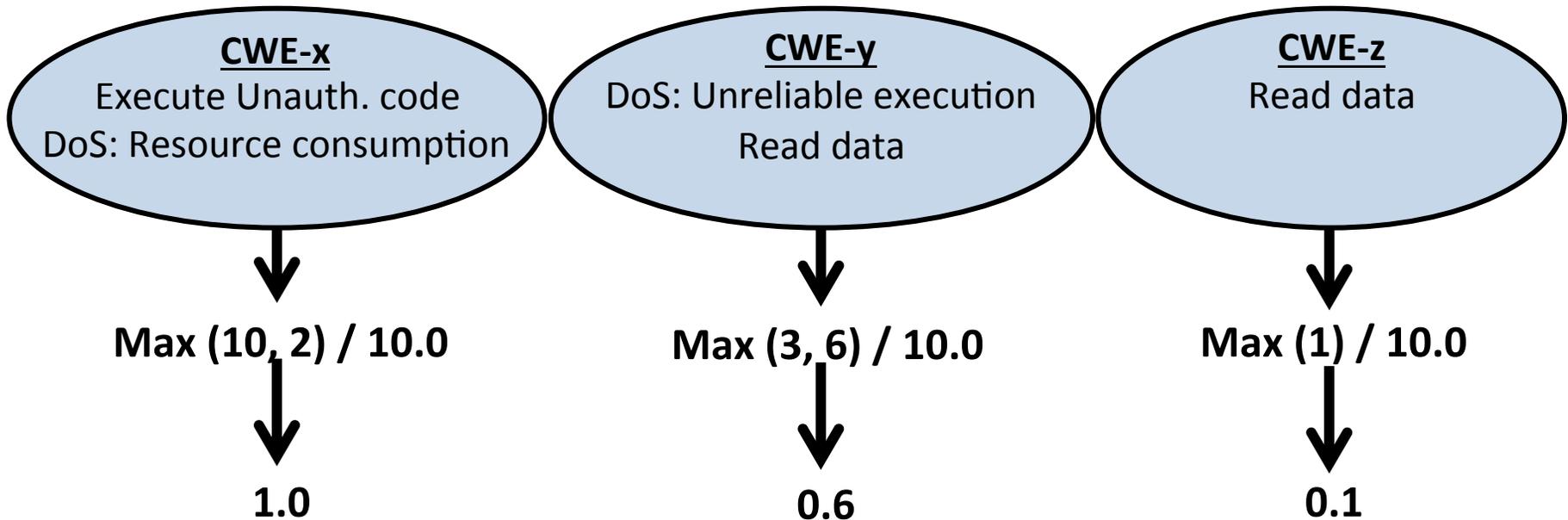
W is all possible weaknesses

W_d is all known weaknesses (CWE)

Calculating CWSS Impact Weights

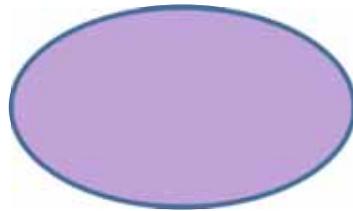
10 – Execute unauthorized code or commands
6 – Read data
3 – DoS: unreliable execution
2 – DoS: resource consumption

Technical
Impact
Scorecard



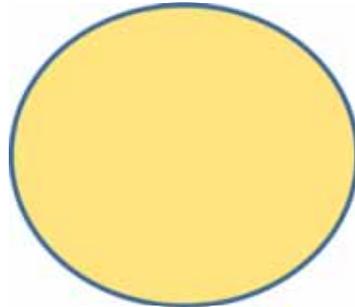
Making Use of the Prioritized List of Weaknesses to Identify Assessment Techniques

Code Review



CWEs a capability *claims* to cover

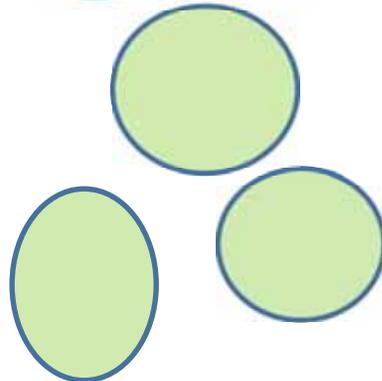
Static Analysis Tool A



Static Analysis Tool B

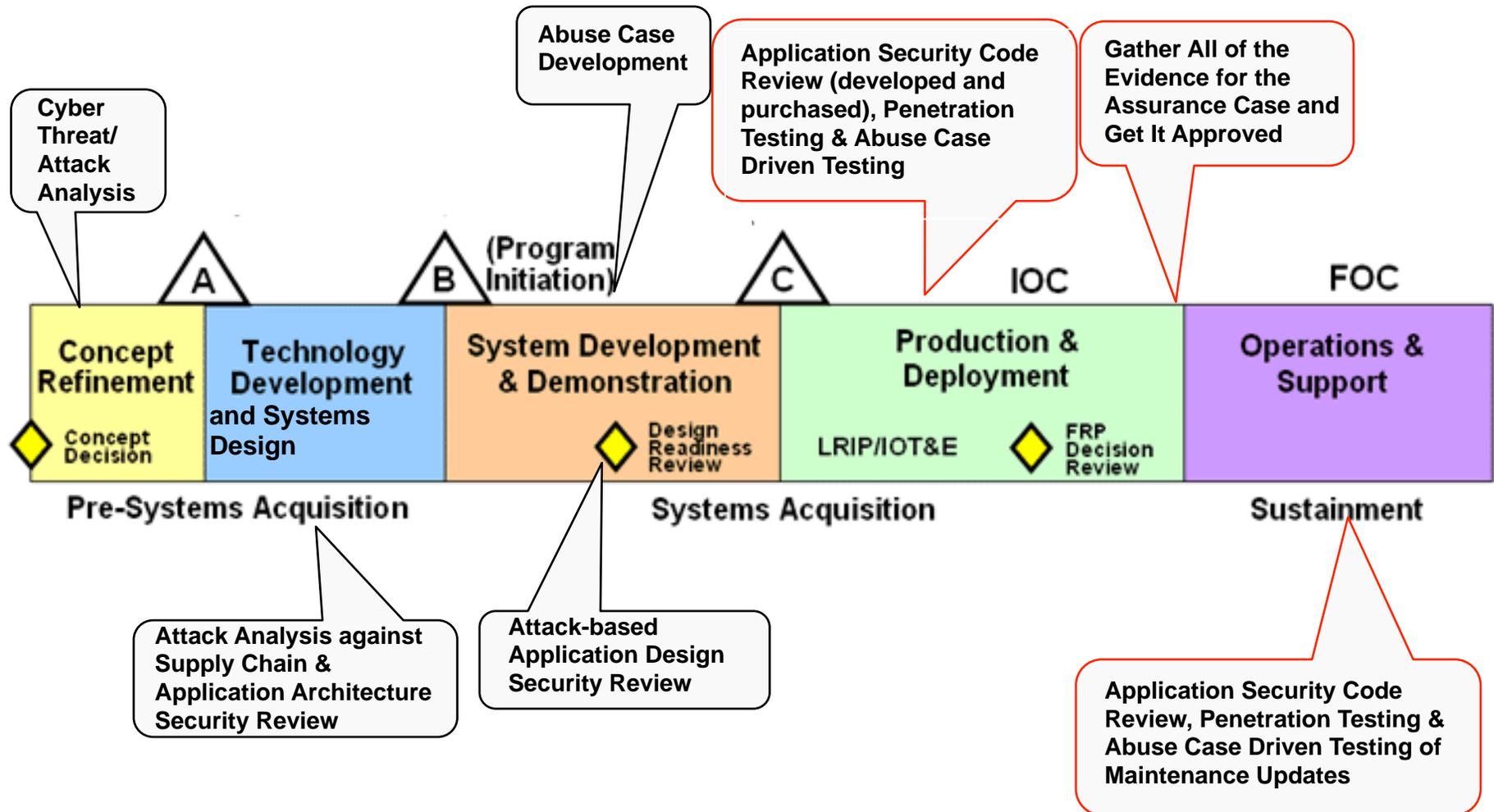


Pen Testing Services



Which static analysis tools, reviews, and Pen Testing services find the CWEs I care about?

Assurance & the Systems Dev. Life-Cycle...



* Ideally Insert SwA before RFP release in Analysis of Alternatives

Leveraging and Managing to take Advantage of the Multiple Detection Methods

- Different assessment methods are effective at finding different types of weaknesses
- Some are good at finding the cause and some at finding the effect

	Static Code Analysis	Penetration Test	Data Security Analysis	Code Review	<i>Architecture Risk Analysis</i>
Cross-Site Scripting (XSS)	X	X		X	
SQL Injection	X	X		X	
Insufficient Authorization Controls		X	X	X	X
Broken Authentication and Session Management		X	X	X	X
Information Leakage		X	X		X
Improper Error Handling	X				
Insecure Use of Cryptography		X		X	X
Cross Site Request Forgery (CSRF)		X		X	
Denial of Service	X	X	X		X
<i>Poor Coding Practices</i>	X			X	

Technical Impacts – Common Consequences Detection Methods

CWE Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (2.5) Search by ID: 78 Go

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

▼ Applicable Platforms

Languages
All

Technology Classes
Database-Server

▼ Modes of Introduction

This weakness typically appears in

▼ Common Consequences

Scope	Effect
Confidentiality	Technical Impact: Read Since SQL databases SQL injection vulner
Access Control	Technical Impact: Bypa If poor SQL comman to a system as anoth
Access Control	Technical Impact: Bypa If authorization infor through the success
Integrity	Technical Impact: Modifi Just as it may be pos delete this informati

▼ Likelihood of Exploit

▼ Detection Methods

Automated Static Analysis

This weakness can often be detected using automated static analysis tools. Many modern tools use data flow analysis or constraint-based techniques to minimize the number of false positives. Automated static analysis might not be able to recognize when proper input validation is being performed, leading to false positives - i.e., warnings that do not have any security consequences or do not require any code changes. Automated static analysis might not be able to detect the usage of custom API functions or third-party libraries that indirectly invoke SQL commands, leading to false negatives - especially if the API/library code is not available for analysis.

This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Automated Dynamic Analysis

This weakness can be detected using dynamic tools and techniques that interact with the software using large test suites with many diverse inputs, such as fuzz testing (fuzzing), robustness testing, and fault injection. The software's operation may slow down, but it should not become unstable, crash, or generate incorrect results.

Effectiveness: Moderate

Manual Analysis

Manual analysis can be useful for finding this weakness, but it might not achieve desired code coverage within limited time constraints. This becomes difficult for weaknesses that must be considered for all inputs, since the attack surface can be too large.

▼ Demonstrative Examples

Example 1

In 2008, a large number of web servers were compromised using the same SQL injection attack string. This single

Detection Methods – updated with SOAR

Home > CWE List > CWE- Individual Dictionary Definition (2.5) Search by ID: 78 Go

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Applicable Platforms

Languages

Automated Static Analysis - Binary / Bytecode

According to SOAR, the following detection techniques may be useful:

Highly cost effective:

- Bytecode Weakness Analysis - including disassembler + source code
- Binary Weakness Analysis - including disassembler + source code

Effectiveness: SOAR High

Dynamic Analysis with automated results interpretation

According to SOAR, the following detection techniques may be useful:

Highly cost effective:

- Database Scanners

Cost effective for partial coverage:

- Web Application Scanner
- Web Services Scanner

Effectiveness: SOAR High

Dynamic Analysis with manual results interpretation

According to SOAR, the following detection techniques may be useful:

Cost effective for partial coverage:

- Fuzz Tester
- Framework-based Fuzzer

Effectiveness: SOAR Partial

ing automated static analysis tools. Many modern tools use data flow

Manual Static Analysis - Source Code

According to SOAR, the following detection techniques may be useful:

Highly cost effective:

- Manual Source Code Review (not inspections)

Cost effective for partial coverage:

- Focused Manual Spotcheck - Focused manual analysis of source code

Effectiveness: SOAR High

Automated Static Analysis - Source Code

According to SOAR, the following detection techniques may be useful:

Highly cost effective:

- Source code Weakness Analyzer
- Context-configured Source Code Weakness Analyzer

Effectiveness: SOAR High

Architecture / Design Review

According to SOAR, the following detection techniques may be useful:

Highly cost effective:

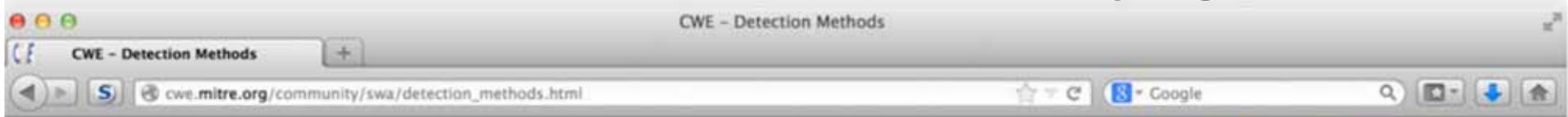
- Formal Methods / Correct-By-Construction

Cost effective for partial coverage:

- Inspection (IEEE 1028 standard) (can apply to requirements, design, source code, etc.)

Effectiveness: SOAR High

Detection Methods web page



Home > Community > Software Assurance > Detection Methods

Search by ID: Go

- CWE List**
- Full Dictionary View
- Development View
- Research View
- Reports
- Mapping & Navigation
- About**
- Sources
- Process
- Documents
- FAQs
- Community**
- Use & Citations
- SwA On-Ramp
- Discussion List
- Discussion Archives
- Contact Us
- Scoring**
- Prioritization
- CWSS
- CWRAP
- CWE/SANS Top 25
- Compatibility**
- Requirements
- Coverage Claims
- Representation
- Compatible Products
- Make a Declaration
- News**
- Calendar
- Free Newsletter
- Search the Site**

Detection Methods

The "Detection Methods" field within many CWE entries conveys information about what types of assessment activities that weakness can be found by. Increasing numbers of CWE entries will have this field filled in over time. The recent Institute of Defense Analysis (IDA) State of the Art Research report conducted for DoD provides additional information for use across CWE in this area. Labels for the Detection Methods being used within CWE are:

- Automated Analysis
- Automated Dynamic Analysis
- Automated Static Analysis
- Black Box
- Fuzzing
- Manual Analysis
- Manual Dynamic Analysis
- Manual Static Analysis
- White Box

With this type of information (shown in the table below), we can see which of the specific CWEs that can lead to a specific type of technical impact are detectable by dynamic analysis, static analysis, and fuzzing evidence and which ones are not.

This table is incomplete, because many CWE entries do not have a detection method listed.

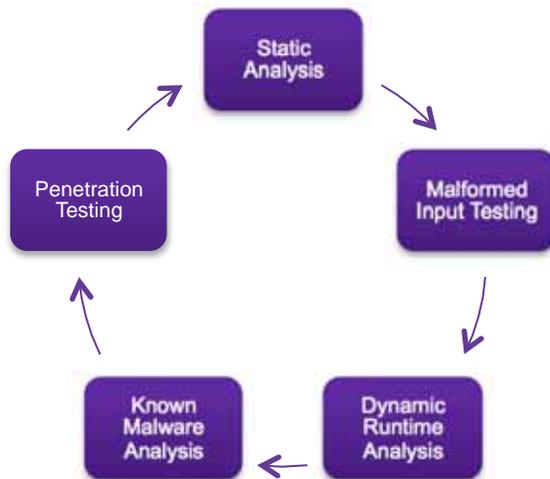
Technical Impact	Automated Analysis	Automated Dynamic Analysis	Automated Static Analysis	Black Box	Fuzzing	Manual Analysis	Manual Dynamic Analysis	Manual Static Analysis	White Box
Execute unauthorized code or commands		78, 120, 129, 131, 476, 805	78, 79, 98, 120, 129, 131, 134, 190, 426, 798, 805	79, 129, 134, 190, 426, 494, 698, 798		98, 120, 131, 190, 426, 494, 805	476, 798	78, 798	
Gain privileges / assume identity		601	306, 352, 426, 601, 798	259, 426, 798		259, 306, 352, 426	798	601, 798, 807	
Read data	209, 311, 327	78, 89, 129, 131, 209, 404, 665	78, 79, 89, 129, 131, 134, 352, 426, 798	14, 79, 129, 134, 319, 426, 798		89, 131, 209, 311, 327, 352, 426	209, 404, 665, 798	78, 798	14
Modify data	311, 327	78, 89, 129, 131	78, 89, 129, 131, 190, 352	129, 190, 319		89, 131, 190, 311		78	

- Section Contents**
- Software Assurance**
- Engineering for Attacks
- Software Quality
- Prioritizing Weaknesses
- Detection Methods
- Manageable Steps
- Pocket Guides
- Staying Informed
- Finding More Information
- Other Items of Interest**
- Discussion List
- CWE Newsletter
- Terms of Use



If I Had A Wish

- Automated toolsets that figure this out through a well defined workflow.
 - I know it is a lofty goal, but thinking big is what drives progress.
 - Today what we can do is capture good data and apply some wisdom in a manual manner.
- If I had one more wish, I would probably wish for time travel, because it just seems cool.



Questions?

Mike Ahmadi

mike@codenomicon.com

SYNOPSYS[®]

MITRE

Bob Martin

ramartin@mitre.org