

# Software Assurance Tools to Improve Evidence Strength

Elizabeth Fong

NIST

SCC Workshop, January 7 and 8 2013

Disclaimer – Any products/companies mentioned are for  
information only - no endorsement implied

# How Much Evidence is Enough to Certify systems?

- - Users need information to establish required metrics (acceptance criteria)
  - e.g. Reliability, Trustworthiness, etc.
- - Produce evidences to support argument
  - Test results
  - Artifacts

Measurement of Evidence that the operational and maintenance requirements and constraints are identified correctly and satisfied

## Outline

- Software Metrics and Measurement
- Software Assurance Tool Evaluation
- Software Code Label
- Research leading to Scoring

# Some Current SAMATE Activities

- Static Analysis tool Exposition (SATE)
- SAMATE Reference Dataset (SRD)
- Precisely Define Some Common Weakness Enumeration (CWE) Entries (CWE Formalization/Effectiveness)
- Statistics and Universe of Program Workshop Planning (SUPER Workshop)

# Software Metrics Classification

For Product Metrics:

- **Size Metrics** (No. of elements)
- **Structure Metrics** (component and structure levels)
- **Complexity Metrics** (computational, algorithmic, logical, functional, etc.)
- **Quality Metrics** (functional, non-functional, reliability, usability, efficiency, maintainability, portability)

# Measurement Scales

## Nominal:

placing it into a category of some kind

## Ordinal:

Ranking the various data values

## Interval

Can be ranked between values

## Ratio

possess an absolute zero

# Measurement Methods

Measurable with units:

i.e., LOC, No. of Source Files, No. of defects per thousand LOC

Measurable with scales:

i.e., Cyclomatic complexity, risk (H,M,L)

Assurance Case with claims, arguments, evidence

i.e., Safety case

Scoring and checklist

i.e., SCAP, CVSS

# Determine the Strength of the Evidence Data

- Application of Software Assurance tools based upon the tool types



# Software Assurance Tool Types

- Static Source Code Analysis Tool
- Dynamic Analysis Tool
- Special Purpose Tool
  - Security-orient tool
  - Compliance-orient tool
  - Pedigree analysis tool

# Static Analysis tools

- **Grep-like** (pattern matching, lots of False Positive, not smart)
- **Smart tool** (understand flow, discriminate)
- **General tool** (broad coverage of weaknesses)
- **Specialized tool** (cover only a few weakness but more depth)

# Dynamic Analysis tool

- Web application scanner
- Penetration tester
- Fuzzing tool

# Software Label

- Software Facts should be:
  - Voluntary
  - Absolutely simple to produce
  - In a standard format for other claims
- What could be easily supplied?
  - Source available? Yes/No/Escrowed
  - Default installation is secure?
  - Accessed: network, disk, ...
  - What configuration files? (registry, ...)
  - Certificates (e.g., "No Severe weaknesses found by CodeChecker ver. 3.2")
- Cautions
  - A label can give false confidence.
  - A label shut out better software.
  - Labeling diverts effort from real improvements.

Software Facts	
Name	InvadingAlienOS
Version	1996.7.04
Expected number of users	15
<hr/>	
Modules	5 483 Modules from libraries 4 102
<hr/>	
	% Vulnerability
<hr/>	
<b>Cross Site Scripting</b> 22	65%
<i>Reflected</i> 12	55%
<i>Stored</i> 10	55%
<hr/>	
<b>SQL Injection</b> 2	10%
<hr/>	
<b>Buffer overflow</b> 5	95%
<hr/>	
<b>Total Security Mechanisms</b> 284	100%
Authentication 15	5%
Access control 3	1%
Input validation 230	81%
Encryption 3	1%
AES 256 bits, Triple DES	
<hr/>	
Report security flaws to: ciwnmcyi@mothership.milkyway	
<hr/>	
<b>Total Code</b> 3.1415×10 <sup>9</sup> function points	100%
C 1.1×10 <sup>9</sup> function points	35%
Ratfor 2.0415×10 <sup>9</sup> function points	65%
<hr/>	
<b>Test Material</b> 2.718×10 <sup>6</sup> bytes	100%
Data 2.69×10 <sup>6</sup> bytes	99%
Executables 27.18×10 <sup>3</sup> bytes	1%
<hr/>	
<b>Documentation</b> 12 058 pages	100%
Tutorial 3 971 pages	33%
Reference 6 233 pages	52%
Design & Specification 1 854 pages	15%
<hr/>	
<b>Libraries:</b> Sun Java 1.5 runtime, Sun J2EE 1.2.2, Jakarta log4j 1.5, Jakarta Commons 2.1, Jakarta Struts 2.0, Harold XOM 1.1rc4, Hunter JDOMv1	
<hr/>	
Compiled with gcc (GCC) 3.3.1	
<hr/>	
Stripped of all symbols and relocation information.	

# Software Rating systems

- OWASP Application Security Verification Standard (ASVS)
  - 4 levels of security rating: L1 – verified by SwA tools
  - L2 – Verified manually
  - L3 - Verified at design phase
  - L4 – Verified internally
- Veracode Security Rating System
  - e.g. AAA (First A represents testing by static analysis.
    - Second A represents testing by dynamic analysis.
    - Third A represents human testing)
- Coverity Software Integrity Rating
  - Level 1, Level 2 (determined by Coverity static analysis)

# How Are Facts Verified and Certified

- Government versus Private
- Mandatory versus voluntary
- Self-claimed versus Third Party
- Open versus Closed

# Scoring Systems with CWRAF

- Business Value Context
- Technical Impact scorecard
- Example of Scoring
  - CVSS
  - CWSS

# How Much Evidence is Enough?

- Progress in tool capabilities
- Standardized dictionary of weaknesses (CWEs)
- Quality of analysis
- Independent V&V
  - Labeling
  - Scoring