SOFTWARE ASSURANCE TRACK

10:45 – 11:30 am
• **Mitigating the Risk of Zero-Day Attacks with Software Security Automation**
  • Joe Jarzombek (DHS), Tom Millar (DHS), and John Banghart (NIST)

11:45 – 12:30 pm
• **Measure Software Security**
  • Sean Barnum (MITRE)

1:30 – 2:15 pm
• **Cyber Observables eXpression (CybOX) - Use Cases**
  • Richard Struse (DHS) and Sean Barnum (MITRE)

2:30 – 3:15 pm
• **Workshop: Risk Analysis and Measurement with CWRAF**
  • Richard Struse (DHS) and Steve Christey (MITRE)

3:45 – 4:30 pm
• **Malware Attribute Enumeration and Characterization (MAEC)**
  • Penny Chase (MITRE) and Ivan Kirillov (MITRE)

4:45 – 5:30 pm
• **Toward CWE Compatibility Effectiveness and CWE Coverage Claims Representation (CCR)**
  • Paul E. Black (NIST) and Richard Struse (DHS)
Software Assurance: Mitigating Risk of Zero-Day Attacks with Software Security Automation

October 31, 2011
Software Assurance

- Tom Millar: addressing the operational needs; what’s the problem that has seen an exponential growth in vulnerabilities as a result exploitable software weaknesses being placed into operations, and what security automation is needed.

- John Banghart: addressing the NIST SP-enabled standards, such SCAP, Continuous Monitoring, and FISMA focused on securing what has been deployed.

- Joe Jarzombek: addressing the use of security automation enumerations and languages; how they can be used today and how they are maturing to better enable software security automation to prevent exploitable software from being deployed.
Today Everything’s Connected

Your System is attackable…

When this Other System gets subverted through an un-patched vulnerability, a mis-configuration, or an application weakness…
Security Feature

Buffer Overflow (CWE-120) Exploit (CAPEC-123)

SQL Injection (CWE-89) Exploit (CAPEC-66)

Exploitable Software Weaknesses are sources for future Zero-Day Attacks
Software Assurance

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

Languages, tools, enumerations and repositories

Automation

Including design, coding, testing, deployment, configuration and operation
Automation is *one piece* of the SwA puzzle.
“Enabling Distributed Security in Cyberspace: Building a Healthy and Resilient Cyber Ecosystem with Automated Collective Action” DHS Paper describes evolving environment

**BOLD** = Enabled/Supported by NCSD SwA
Ecosystem Areas Directly Enabled/Supported by Enumerations/Languages/Standards

First Wave
- OVAL, CVE, CVSS, CPE, XCCDF, SCAP, CWE, CVRF, SWID
- CCE, OVAL, CPE, CCSS, XCCDF, OCIL, SCAP, SWID
- ERAP, CRE, ERI, RCL, RPL, CWE
- CEE, CLS, CLT, CELR, CybOX, EMAP

Second Wave
- MAEC, OVAL, CVE
- MAEC, OVAL, CAPEC, CybOX
- MAEC, CAPEC, OVAL, CybOX
- MAEC, CAPEC, OVAL, CybOX
- MAEC, OVAL, AI, SWID

Third Wave and Beyond
- Collaborative threat intelligence
- Sensing and Warning
- Response
- Forensics and Damage Assessment
- Recovery
- Reconstitution

Ecosystem Areas:
- Design
- Engineering
- Architecture
- Supply Chain Assurance
- Modeling and Simulation
- Reconstitution

Languages/Standards:
- Test, attestation, and assurance
- CVE, OVAL, CWE, CAPEC, MAEC, CybOX, CWRAF, CWSS, CCR, SAFES, SACM, ISO 15026-2, ISO TR 20004, SWID, SwAAP
- CCE, OVAL, TNC
- CVE, OVAL, TNC
- XCCDF, SCAP, SWID
- ERAP, CRE, ERI, RCL, RPL, CWE
- CEE, CLS, CLT, CELR, CybOX, EMAP
- AI, ARF, ASR, PLARR
- CYBEX, IODEF, RID, TLP, MAEC, CybOX, CVE
- CPE, OVAL, AI, SWID

Standards:
- CVE, OVAL, TNC
- OVAL, CVE, CVSS, CPE, XCCDF, SCAP, CWE, CVRF, SWID
- CCE, OVAL, CPE, CCSS, XCCDF, OCIL, SCAP, SWID
- ERAP, CRE, ERI, RCL, RPL, CWE
- CEE, CLS, CLT, CELR, CybOX, EMAP
- AI, ARF, ASR, PLARR
- CYBEX, IODEF, RID, TLP, MAEC, CybOX, CVE
- CPE, OVAL, AI, SWID
- CVE, OVAL, CAPEC, CybOX, CVE
- MAEC, OVAL, CYBEX, IODEF, RID, TLP, MAEC, CybOX, CVE, CWE, CAPEC
Cyber Threats Emerged Over Time

- **1980’s**
  - Password cracking
  - Exploiting known vulnerabilities
  - Back doors
  - Disabling audits
  - Internet social engineering attacks
  - Password guessing

- **1990’s**
  - GUI intruder tools
  - Executable code attacks (against browsers)
  - Automated widespread attacks
  - Automated probes/scans
  - Network mgmt. diagnostics
  - Sniffers
  - Hijacking sessions

- **2000’s**
  - Email propagation of malicious code
  - Automated widespread attacks using NNTP to distribute attack
  - Arnoldic attacks
  - Sophisticated command & control
  - Anti-forensic techniques
  - Home users targeted
  - Distributed attack tools
  - Increase in wide-scale Trojan horse distribution
  - Windows-based remote controllable Trojans (Back Orifice)
  - Increase in tailored worms
  - Sophisticated denial-of-service attacks
  - Widespread denial-of-service attacks
  - Techniques to analyze code for vulnerabilities without source code

- **2010’s**
  - Binary encryption
  - Diffuse spyware
  - Attack sophistication
  - Increase in wide-scale Trojan horse distribution
  - Sophisticated command & control
Solutions Also Emerged Over Time

1980's
- Password cracking
- Packet spoofing
- Exploiting known vulnerabilities
- Burglaries

1990's
- Password guessing
- Exploiting known vulnerabilities
- Burglaries
- Automated probes/scans
- Sniffers
- Automated widespread probes/scans
- Executable code attacks (against browsers)
- Disabling audits
- Internet social engineering attacks
- GUI intruder tools
- Penetration testing

2000's
- Email propagation of malicious code
- DDoS attacks
- Binary encryption
- Increase in tailored worms
- Sophisticated command & control
- Anti-forensic techniques
- Home users targeted
- Distributed attack tools
- Widespread distribution of wide-scale Trojan horses
- Windows-based remote controllable Trojans (Back Orifice)
- Techniques to analyze code for vulnerabilities without source code

2010's
- Widespread denial-of-service attacks
- Techniques to analyze code for vulnerabilities without source code
- Distributed attack tools
- Widespread distribution of wide-scale Trojan horses
- Windows-based remote controllable Trojans (Back Orifice)
- Anti-forensic techniques
- Home users targeted
What Do The Informational Building Blocks for “Architecting Security” Look Like?

- Standard ways for **enumerating** “things we care about”
- **Languages/Formats** for encoding/carrying high fidelity content about the “things we care about”
- **Repositories** of this content for use in communities or individual organizations
- **Adoption/branding and vetting** programs to encourage adoption by tools and services
The Building Blocks Are:

- **Enumerations**
  - Catalog the fundamental entities in IA, Cyber Security, and Software Assurance
    - Vulnerabilities (CVE), configuration issues (CCE), software packages (CPE), attack patterns (CAPEC), weaknesses in code/design/architecture (CWE), observables (CYBOX)

- **Languages/Formats**
  - Support the creation of machine-readable state assertions, assessment results, and messages
    - Configuration/vulnerability/patch/asset patterns (XCCDF & OVAL), results from standards-based assessments (ARF), event patterns (CEE), malware patterns (MAEC), risk of a vulnerability (CVSS), config risk (CCSS), weakness risk (CWSS), assessment findings (SAFES/SACM), information messages (CYBEX/IODEF)

- **Knowledge Repositories**
  - Packages of assertions supporting a specific application
    - Vulnerability advisories & alerts, (US-CERT Advisories/IAVAs), configuration assessment (NIST Checklists, CIS Benchmarks, NSA Configuration Guides, DISA STIGS), asset inventory (NIST/DHS NVD), code assessment & certification (NIST SAMATE, DoD DIACAP & eMASS)

**Tools**
- Interpret IA, Cyber Security, and SwA content in context of enterprise network
- Methods for assessing compliance to languages, formats, and enumerations
# Cyber Ecosystem Standardization Efforts

<table>
<thead>
<tr>
<th>Question</th>
<th>Relevant Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>What IT systems do I have in my enterprise?</td>
<td>CPE (Platforms)</td>
</tr>
<tr>
<td>What known vulnerabilities do I need to worry about?</td>
<td>CVE (Vulnerabilities)</td>
</tr>
<tr>
<td>What vulnerabilities do I need to worry about right now?</td>
<td>CVSS (Scoring System)</td>
</tr>
<tr>
<td>How can I configure my systems more securely?</td>
<td>CCE (Configurations)</td>
</tr>
<tr>
<td>How do I define a policy of secure configurations?</td>
<td>XCCDF (Configuration Checklists)</td>
</tr>
<tr>
<td>How can I be sure my systems conform to policy?</td>
<td>OVAL (Assessment Language)</td>
</tr>
<tr>
<td>How can I be sure the operation of my systems conforms to policy?</td>
<td>OCIL (Interactive Language)</td>
</tr>
<tr>
<td>What weaknesses in my software could be exploited?</td>
<td>CWE (Weaknesses)</td>
</tr>
<tr>
<td>What attacks can exploit which weaknesses?</td>
<td>CAPEC (Attack Patterns)</td>
</tr>
<tr>
<td>How can we recognize malware &amp; share that info?</td>
<td>MAEC (Malware Attributes)</td>
</tr>
<tr>
<td>What observable behavior might put my enterprise at risk?</td>
<td>CybOX (Cyber Observables)</td>
</tr>
<tr>
<td>What events should be logged, and how?</td>
<td>CEE (Events)</td>
</tr>
<tr>
<td>How can I aggregate assessment results?</td>
<td>ARF (Assessment Results)</td>
</tr>
<tr>
<td>Question</td>
<td>SCAP Components</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>What IT systems do I have in my enterprise?</td>
<td>CPE (Platforms)</td>
</tr>
<tr>
<td>What known vulnerabilities do I need to worry about?</td>
<td>CVE (Vulnerabilities)</td>
</tr>
<tr>
<td>What vulnerabilities do I need to worry about right now?</td>
<td>CVSS (Scoring System)</td>
</tr>
<tr>
<td>How can I configure my systems more securely?</td>
<td>CCE (Configurations)</td>
</tr>
<tr>
<td>How do I define a policy of secure configurations?</td>
<td>XCCDF (Configuration Checklists)</td>
</tr>
<tr>
<td>How can I be sure my systems conform to policy?</td>
<td>OVAL (Assessment Language)</td>
</tr>
<tr>
<td>How can I be sure the operation of my systems conforms to policy?</td>
<td>OCIL (Interactive Language)</td>
</tr>
<tr>
<td>What weaknesses in my software could be exploited?</td>
<td>CWE (Weaknesses)</td>
</tr>
<tr>
<td>What attacks can exploit which weaknesses?</td>
<td>CAPEC (Attack Patterns)</td>
</tr>
<tr>
<td>How can we recognize malware &amp; share that info?</td>
<td>MAEC (Malware Attributes)</td>
</tr>
<tr>
<td>What observable behavior might put my enterprise at risk?</td>
<td>CybOX (Cyber Observables)</td>
</tr>
<tr>
<td>What events should be logged, and how?</td>
<td>CEE (Events)</td>
</tr>
<tr>
<td>How can I aggregate assessment results?</td>
<td>ARF (Assessment Results)</td>
</tr>
</tbody>
</table>

Standardization Efforts leveraged by the Security Content Automation Protocol (SCAP)
Efforts focused on mitigating risks and enabling more robust continuous monitoring and faster incident response

| What IT systems do I have in my enterprise? | • CPE (Platforms) |
| What known vulnerabilities do I need to worry about? | • CVE (Vulnerabilities) |
| What vulnerabilities do I need to worry about right now? | • CVSS (Scoring System) |
| How can I configure my systems more securely? | • CCE (Configurations) |
| How do I define a policy of secure configurations? | • XCCDF (Configuration Checklists) |
| How can I be sure my systems conform to policy? | • OVAL (Assessment Language) |
| How can I be sure the operation of my systems conforms to policy? | • OCIL (Interactive Language) |
| What weaknesses in my software could be exploited? | • CWE (Weaknesses) |
| What attacks can exploit which weaknesses? | • CAPEC (Attack Patterns) |
| How can we recognize malware & share that info? | • MAEC (Malware Attributes) |
| What observable behavior might put my enterprise at risk? | • CybOX (Cyber Observables) |
| What events should be logged, and how? | • CEE (Events) |
| How can I aggregate assessment results? | • ARF (Assessment Results) |
Software Assurance: Mitigating Risk of Zero-Day Attacks with Software Security Automation

October 31, 2011
Leverage Common Weakness Enumeration (CWE) to mitigate risks to mission/business domains

CWE is a formal list of software weakness types created to:
• Serve as a common language for describing software security weaknesses in architecture, design, or code.
• Serve as a standard measuring stick for software security tools targeting these weaknesses.
• Provide a common baseline standard for weakness identification, mitigation, and prevention efforts.

Some Common Types of Software Weaknesses:

- Buffer Overflows, Format Strings, Etc.
- Structure and Validity Problems
- Common Special Element Manipulations
- Channel and Path Errors
- Handler Errors
- User Interface Errors
- Pathname Traversal and Equivalence
- Errors
- Authentication Errors
- Resource Management Errors
- Insufficient Verification of Data
- Code Evaluation and Injection
- Randomness and Predictability
International in scope and free for public use, CWE™ provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.
“Making Security Measureable”: measurablesecurity.mitre.org

Sponsored by DHS with MITRE as technical lead

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

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

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”)
$W_d$: The set of all *discovered* software weaknesses in $W$.
$W$: The set of all vulnerabilities in $W$
$V_d$: The set of all *discovered* vulnerabilities in $V$
What does the future hold?
We know it’s *not* this, at least not in the near-term
Maybe the problem grows unbounded?
One reasonable near-term goal
Is this really better? Yes
For the software we’re responsible for:

where should we start?

Vulnerabilities identified with a CVE are a good starting point.
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)
National Vulnerability Database (NVD)

CVE Entry + NVD

- CVSS Scores
- Affected Platforms
- Root-cause Weaknesses (CWE’s)
- References to Advisories
- References to Mitigations
- References to Tools
- OVAL-based Checks

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

website: nvd.nist.gov
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

How do we identify these?

which weaknesses are most important?
Prioritizing weaknesses to be mitigated

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
NSTB Assessments
Summary Report:
Common Industrial Control System Cyber Security Weaknesses

May 2010

Idaho National Labs SCADA Report
<table>
<thead>
<tr>
<th>Weakness Classification</th>
<th>Vulnerability Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE-19: Data Handling</td>
<td>CWE-228: Improper Handling of Syntactically Invalid Structure</td>
</tr>
<tr>
<td></td>
<td>CWE-229: Improper Handling of Values</td>
</tr>
<tr>
<td></td>
<td>CWE-230: Improper Handling of Missing Values</td>
</tr>
<tr>
<td></td>
<td>CWE-20: Improper Input Validation</td>
</tr>
<tr>
<td></td>
<td>CWE-116: Improper Encoding or Escaping of Output</td>
</tr>
<tr>
<td></td>
<td>CWE-195: Signed to Unsigned Conversion Error</td>
</tr>
<tr>
<td></td>
<td>CWE-198: Use of Incorrect Byte Ordering</td>
</tr>
<tr>
<td>CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer</td>
<td>CWE-120: Buffer Copy without Checking Size of Input (&quot;Classic Buffer Overflow&quot;)</td>
</tr>
<tr>
<td></td>
<td>CWE-121: Stack-based Buffer Overflow</td>
</tr>
<tr>
<td></td>
<td>CWE-122: Heap-based Buffer Overflow</td>
</tr>
<tr>
<td></td>
<td>CWE-125: Out-of-bounds Read</td>
</tr>
<tr>
<td></td>
<td>CWE-129: Improper Validation of Array Index</td>
</tr>
<tr>
<td></td>
<td>CWE-131: Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td></td>
<td>CWE-170: Improper Null Termination</td>
</tr>
<tr>
<td></td>
<td>CWE-190: Integer Overflow or Wraparound</td>
</tr>
<tr>
<td></td>
<td>CWE-680: Integer Overflow to Buffer Overflow</td>
</tr>
<tr>
<td>CWE-398: Indicator of Poor Code Quality</td>
<td>CWE-454: External Initialization of Trusted Variables or Data Stores</td>
</tr>
<tr>
<td></td>
<td>CWE-456: Missing Initialization</td>
</tr>
<tr>
<td></td>
<td>CWE-457: Use of Uninitialized Variable</td>
</tr>
<tr>
<td></td>
<td>CWE-476: NULL Pointer Dereference</td>
</tr>
<tr>
<td></td>
<td>CWE-400: Uncontrolled Resource Consumption (&quot;Resource Exhaustion&quot;)</td>
</tr>
<tr>
<td></td>
<td>CWE-252: Unchecked Return Value</td>
</tr>
<tr>
<td></td>
<td>CWE-690: Unchecked Return Value to NULL Pointer Dereference</td>
</tr>
<tr>
<td></td>
<td>CWE-772: Missing Release of Resource after Effective Lifetime</td>
</tr>
<tr>
<td>CWE-442: Web Problems</td>
<td>CWE-22: Improper Limitation of a Pathname to a Restricted Directory (&quot;Path Traversal&quot;)</td>
</tr>
<tr>
<td></td>
<td>CWE-79: Failure to Preserve Web Page Structure (&quot;Cross-site Scripting&quot;)</td>
</tr>
<tr>
<td></td>
<td>CWE-89: Failure to Preserve SQL Query Structure (&quot;SQL Injection&quot;)</td>
</tr>
<tr>
<td>CWE-703: Failure to Handle Exceptional Conditions</td>
<td>CWE-431: Missing Handler</td>
</tr>
<tr>
<td></td>
<td>CWE-248: Uncaught Exception</td>
</tr>
<tr>
<td></td>
<td>CWE-755: Improper Handling of Exceptional Conditions</td>
</tr>
<tr>
<td></td>
<td>CWE-390: Detection of Error Condition Without Action</td>
</tr>
</tbody>
</table>
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?
Leveraging Vignettes in Cyber Security Standardization for Key ICT Applications in various Domains

<table>
<thead>
<tr>
<th>Technology Groups</th>
<th>Business/Mission Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Applications</td>
<td>e-Commerce</td>
</tr>
<tr>
<td></td>
<td>Banking &amp; Finance</td>
</tr>
<tr>
<td></td>
<td>Energy (i.e., SmartGrid)</td>
</tr>
<tr>
<td></td>
<td>Chemical</td>
</tr>
<tr>
<td></td>
<td>Manufacturing</td>
</tr>
<tr>
<td></td>
<td>Shipping</td>
</tr>
<tr>
<td></td>
<td>Freight, ships</td>
</tr>
<tr>
<td></td>
<td>Transportation (i.e., rail)</td>
</tr>
<tr>
<td></td>
<td>National Defense</td>
</tr>
<tr>
<td></td>
<td>Intel networks</td>
</tr>
<tr>
<td></td>
<td>Homeland Security</td>
</tr>
<tr>
<td></td>
<td>Secret Service, TSA, etc.</td>
</tr>
<tr>
<td></td>
<td>Government (other than Nat'1 Def &amp; HS)</td>
</tr>
<tr>
<td></td>
<td>Emergency Services, law enforcement services, etc.</td>
</tr>
<tr>
<td></td>
<td>Public Health</td>
</tr>
<tr>
<td></td>
<td>Food &amp; Water</td>
</tr>
<tr>
<td></td>
<td>Telecommunications</td>
</tr>
<tr>
<td></td>
<td>Teleworking</td>
</tr>
<tr>
<td></td>
<td>e-Voting</td>
</tr>
</tbody>
</table>

Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups.
CWRAF/CWSS in a Nutshell

<table>
<thead>
<tr>
<th>CWSS Score</th>
<th>CWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>CWE-79</td>
</tr>
<tr>
<td>95</td>
<td>CWE-78</td>
</tr>
<tr>
<td>94</td>
<td>CWE-22</td>
</tr>
<tr>
<td>94</td>
<td>CWE-434</td>
</tr>
<tr>
<td>94</td>
<td>CWE-798</td>
</tr>
<tr>
<td>93</td>
<td>CWE-120</td>
</tr>
<tr>
<td>93</td>
<td>CWE-250</td>
</tr>
<tr>
<td>92</td>
<td>CWE-770</td>
</tr>
<tr>
<td>91</td>
<td>CWE-829</td>
</tr>
<tr>
<td>91</td>
<td>CWE-190</td>
</tr>
<tr>
<td>91</td>
<td>CWE-494</td>
</tr>
<tr>
<td>90</td>
<td>CWE-134</td>
</tr>
<tr>
<td>90</td>
<td>CWE-772</td>
</tr>
<tr>
<td>90</td>
<td>CWE-476</td>
</tr>
<tr>
<td>90</td>
<td>CWE-131</td>
</tr>
</tbody>
</table>

User-defined cutoff

“Vignette”

CWSS Scoring Engine

Most Important Weaknesses

W

W_d

Most Important Weaknesses

CWRAF/CWSS in a Nutshell
Common Weakness Risk Analysis Framework (CWRAF) and Common Weakness Scoring System (CWSS)

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.
Common Weakness Risk Analysis Framework (CWRAF) and Common Weakness Scoring System (CWSS)

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.
Which static analysis tools find the CWE’s I care about?
CWRAF/CWSS Provides Risk Prioritization for CWE throughout Software Life Cycle

- Enables education and training to provide specific practices for eliminating on software fault patterns;
- Enables developers to mitigate top risks attributable to exploitable software;
- Enables testing organizations to use suite of test tools & methods (with CWE Coverage Claims Representation) that cover applicable concerns;
- Enables users and operation organizations to deploy and use software that is more resilient and secure;
- Enables procurement organizations to specify software security expectations through acquisition of software, hosted applications and services.
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?

<table>
<thead>
<tr>
<th>CWSS Score</th>
<th>CWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>CWE-79</td>
</tr>
<tr>
<td>95</td>
<td>CWE-78</td>
</tr>
<tr>
<td>94</td>
<td>CWE-22</td>
</tr>
<tr>
<td>94</td>
<td>CWE-434</td>
</tr>
<tr>
<td>94</td>
<td>CWE-798</td>
</tr>
<tr>
<td>93</td>
<td>CWE-120</td>
</tr>
<tr>
<td>93</td>
<td>CWE-250</td>
</tr>
<tr>
<td>92</td>
<td>CWE-770</td>
</tr>
<tr>
<td>91</td>
<td>CWE-829</td>
</tr>
<tr>
<td>91</td>
<td>CWE-190</td>
</tr>
<tr>
<td>91</td>
<td>CWE-494</td>
</tr>
<tr>
<td>90</td>
<td>CWE-134</td>
</tr>
<tr>
<td>90</td>
<td>CWE-772</td>
</tr>
<tr>
<td>90</td>
<td>CWE-476</td>
</tr>
<tr>
<td>90</td>
<td>CWE-131</td>
</tr>
</tbody>
</table>

CWE Scoring Engine

Most Important Weaknesses

Wd

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
Common Vulnerabilities and Exposures (CVE)
Open Vulnerability Assessment Language (OVAL)
Malware Attribute Enumeration and Characterization (MAEC)
Cyber Observables eXpression (CybOX)

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)

Common Vulnerabilities and Exposures (CVE)
Open Vulnerability Assessment Language (OVAL)
Malware Attribute Enumeration and Characterization (MAEC)
Cyber Observables eXpression (CybOX)
It can be used to begin the translation of SwA Activities across organizational leadership.
**SwA and Operational Resilience**

- **Compliance**
- **Monitoring**
- **Incident Management and Control**
- **Enterprise Focus**
- **Vulnerability Analysis and Resolution**
- **Asset Management**
- **Controls**
- **Measurement and Analysis**

**Questions:***
- How do we prevent this next time?
- Are we being attacked?
- Who is attacking and what do they want?
- Are we at risk?

**Abbreviations:**
- OVAL SCAP
- CVSS CAPEC MAEC KDM
- MAEC
- BPMN

**Adapted from** September 2010 SwA Forum, CERT RMM for Assurance, Lisa Young, SEI

**Courtesy of** Michele Moss
Software Assurance Ecosystem: The Formal Framework

The value of formalization extends beyond software systems to include related software system process, people and documentation.

- **Process, People & Documentation**
  - Evaluation Environment
    - Some point tools to assist evaluators but mainly manual work
    - Claims in Formal SBVR vocabulary
    - Evidence in Formal SBVR vocabulary
    - Large scope requires large effort

- **Software System / Architecture Evaluation**
  - Many integrated & highly automated tools to assist evaluators
  - Claims and Evidence in Formal vocabulary
  - Combination of tools and ISO/OMG standards
  - Standardized SW System Representation In KDM
  - Large scope capable (system of systems)
  - Iterative extraction and analysis for rules

- **Claims, Arguments and Evidence Repository**
  - Formalized in SBVR vocabulary
  - Automated verification of claims against evidence
  - Highly automated and sophisticated risk assessments using transitive inter-evidence point relationships

- **Reports Risk Analysis, etc)**

- **Process Docs & Artifacts**

- **Hardware Environment**

- **Software System Artifacts**

- **IA Controls**

- **CWE**

- **Protection Profiles**

- **Formalized Specifications**

- **Executable Specifications**

- **Technical Evidence**

- **Claims, Arguments and Evidence Repository**
SwA Working Groups – Next meeting: Week of Nov 28, 2011 @ MITRE in McLean, VA

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

See Language for sharing correlation of incident information -- Cyber Observables eXpression (CybOX) at http://cybox.mitre.org
IT/Software Supply Chain Management is a National Security & Economic Issue

- Adversaries can gain “intimate access” to target systems, especially in a global supply chain that offers limited transparency.

- Advances in science and technology will always outpace the ability of government and industry to react with new policies and standards:
  - National security policies must conform with international laws and agreements while preserving a nation’s rights and freedoms, and protecting a nation’s self interests and economic goals.
  - Forward-looking policies can adapt to the new world of global supply chains.
  - Information standards, process standards, and product standards must mature to better address supply chain risk management, security, & systems/software assurance.
  - Assurance Rating Schemes for software products and organizations are needed.

- IT/software suppliers and buyers can take more deliberate actions to security-enhance their processes and practices to mitigate risks:
  - Government & Industry have significant leadership roles in solving this.
  - Individuals can influence the way their organizations adopt security practices.

Globalization will not be reversed; this is how we conduct business – To remain relevant, standards and capability benchmarking measures must address “assurance” mechanisms needed to manage IT/Software Supply Chain risks.
Next SwA Working Group sessions 28 Nov – 2 Dec 2011 at MITRE, McLean, VA

SOFTWARE ASSURANCE FORUM

“Building Security In”

https://buildsecurityin.us-cert.gov/swa
Public/Private Collaboration Efforts for Software Supply Chain Risk Management

Next SwA Forum meets 28 Nov – 2 Dec 2011 at MITRE, McLean, VA
Public/Private Collaboration Efforts for Software Supply Chain Risk Management

Next SwA Forum meets 12-16 Sep 2011 at SEI, Arlington, VA