Software Assurance Landscape

1. Intro & Purpose of Landscape

Try to stay broad and focus on knowledge rather than get too specific (e.g. measurement or standardization). All of this is about developing and sharing knowledge. We still need to decide the appropriate scoping boundary for this paper between software assurance and systems or information assurance.

The security and integrity of information systems has become a critical issue within most types of organizations and finding better ways to address the topic has become the objective of many in industry, academia, and government. The development, sharing and leveraging of software assurance knowledge is becoming a key enabler to making the types of changes and improvements that are needed to address these issues.

There are a large number of software assurance knowledge development and sharing activities and initiatives being perused by a variety of groups including public standards groups, industry associations, commercial organizations, academia, and government.

This paper will attempt to:

- draw a somewhat broad picture of the organizations and efforts of the software assurance landscape
- identify and describe various knowledge resources being developed and made available by these efforts
- describe and explore how many of these efforts and knowledge resources are actually mutually supportive, well aligned, and complimentary
- Identify gaps and opportunities in the current landscape

Together these efforts and knowledge resources compose major segments of a comprehensive approach to economically addressing the software development, systems operation, accreditation/reporting, research and development and education needs of today's corporations, governments, and everyone else. The information technology security capabilities of our critical infrastructure and commerce capabilities are totally dependent on software, networks, and information and thus bringing improvements and manageability to these will benefit all.

By one count, there are over a hundred efforts, initiatives, and standards being developed, propagated, or discussed in the information security arena yet most people are probably only aware of a few.

2. Brief overview and scoping of "Software Assurance"

Define what we mean by "software assurance" in this context. What is it? Why is it important? Who cares about it? Define the stakeholders and actors.

Software Assurance Business Case
Software Assurance State of the Art Report (SOAR).
Software Assurance Knowledge Architecture whitepaper (GAP)

3. Software Assurance State of the Art/Practice Summary

CURRENT STATE:

- **Software and IT vulnerabilities jeopardize infrastructure operations, business operations & services, intellectual property, and consumer trust;**
- **Adversaries have capabilities to subvert the IT/software supply chain:**
  - Software & IT lifecycle processes offer opportunities to insert malicious code and to poorly design and build software which enables future exploitation
  - Government and businesses rely on COTS products and commercial developers using foreign and non-vetted domestic suppliers to meet majority of IT requirements.
  - Off-shoring magnifies risks and creates new threats to security, business property and processes, and individuals' privacy – requires more comprehensive domestic strategies to mitigate those risks.
  - Government lacks information on suppliers' process capabilities (business practices); cannot adequately determine security risks posed by the suppliers' products and services to the acquisition project and to the operations enabled by the software.
  - Government lacks mechanism for sharing information about evaluated products (multiple labs testing same products) and the responsive provisions for discovering exploitable vulnerabilities throughout the lifecycle.
- **Growing concern about inadequacies of suppliers' capabilities to build/deliver secure IT/software – too few practitioners with requisite knowledge and skills:**
  - Concern about suppliers and practitioners not exercising "minimum level of responsible practice" – no standards in place to benchmark or assess practices.
  - Current education & training provides too few practitioners with requisite competencies in secure software engineering – enrollment down in critical IT and software-related degree programs.

TARGETED END STATE (WHERE WE WANT TO BE):

- **Government, in collaboration with industry/academia, raised expectations for product assurance with requisite levels of integrity and security:**
  - Promoted use of methodologies and tools that enabled security to be part of normal business;
  - Structured and funded to advance more comprehensive software assurance diagnostic capabilities to mitigate risks stemming from exploitable vulnerabilities;
  - Identified assets requiring high assurance software; factoring those needs in procurements.
- **Acquisition managers & users factored risks posed by the supply chain as part of the trade-space in risk mitigation efforts:**
  - Information on suppliers' process capabilities (business practices) would be used to determine security risks posed by the suppliers' software products and services to the acquisition project and to the operations enabled by the software.
  - Information about evaluated software products would be available, along with responsive provisions for discovering exploitable vulnerabilities, and products would be securely configured in use.
- **Suppliers delivered quality products with requisite integrity and made assurance claims about the IT/software safety, security and dependability:**
  - Relevant standards would be used from which to base business practices and make claims about product/system safety, security and dependability;
  - Qualified tools used in software lifecycle would enable developers/testers to mitigate security risks;

- Standards and qualified tools would be used to certify software by independent third parties;
- IT/software workforce would have requisite knowledge/skills for developing secure, quality products.

CURRENT PROGRAM GAPS:

- **Processes and technologies are required to build trust into IT and software:**
  - Improvements are needed in the state-of-the-practice and state-of-the-art for IT & software development capabilities.
  - Government, through the Software Assurance Program, is working in collaboration with academia/industry/standards organizations to identify requisite practices for producing products with requisite levels of security and quality; however, it is primarily based on collaborative volunteer efforts -- funding and resources are needed to advance the evolution of requisite technologies and standards.
  - SwA program identifying/evolving software security methodologies and tools; collaboratively working with other federal programs and industry groups; ready for the next phase advance projects with industry, NIST and standards groups.
  - Relevant national and international standards are needed from which to base and evaluate business practices & make claims about software assurance.
  - More comprehensive diagnostic tools and supporting knowledge are needed to be qualified and used in software lifecycle to enable developers/testers to mitigate security risks.
  - Mechanisms (standards-based processes and tools) are needed to enable independent third-party labs to certify software relative to assurance claims.

- **National-level focus is needed to stay competitive in a global IT environment:**
  - Computing education & training needs to evolve with changing nature of IT/software business to produce software workforce with requisite knowledge and skills for developing secure, quality products.
  - Educational policy and investment is needed to foster innovation and increase IT-related enrollments.

4. Software Assurance Landscape Index

Link all entries in the index to their detailed descriptions in the following sections.

Communities & Leadership

- Software Assurance leadership organizations
  - DHS NCSD Software Assurance Program
  - NSA CAS
  - Object Management Group (OMG) Software Assurance Special Interest Group (SwA SIG)
  - NASA Software Assurance Research Program
  - NASA Reducing Software Security Risk (RSSR) Program
  - DoD Software Assurance Tiger Team
  - DISA Application Security Project
  - FAA and DoD Safety and Security Extension Project Team
  - National Cyber Security Taskforce (NCST)
  - High Confidence Software Systems (HCSS)
    - NITRD Supplement to the President's FY 2007 Budget
    - NSA, Brad Martin
    - OSD/DDR&E, Rob Gold
    - NSF, Helen Gill
  - National Infrastructure Security Co-ordination Centre (NISCC)
  - Open Web Application Security Project (OWASP)
  - Web Application Security Consortium (WASC)
  - Secure Software Forum
  - Application Security Industry Consortium (APPSIC)
- Standards Organizations
  - International Standards Organization (ISO)
  - Institute of Electrical and Electronics Engineers (IEEE)
  - International Electrotechnical Association (IEC)
  - Object Management Group (OMG)
  - National Institute for Standards and Technology (NIST)
  - American Society for Quality (ASQ)
  - Telcordia/Bellcore
  - Malcom Baldridge
  - International Systems Security Engineering Association (ISSEA)
  - Knowledge Software
- Professional Societies
  - Institute of Electrical and Electronics Engineers (IEEE)
  - Association for Computing Machinery (ACM)
  - Cyber Security Industry Alliance (CSIA)
  - Computing Technology Industry Association (CompTIA)
- Knowledge Coordinating organizations
  - Security Description and Exchange Format (SecDEF) Initiative
  - OMG MOF (MetaObject Facility)
  - Lots & lots more
- Knowledge portals
  - Building Security In (BSI)
    - Needs redesign with more collaboration (GAP)
    - Needs more promotion and fresh content (e.g., blogs) (GAP)
  - Information Assurance Technology Analysis Center (IATAC)
  - Data & Analysis Center for Software (DACS)
  - Open Web Application Security Project (OWASP)
  - Web Application Security Consortium (WASC)

- National Vulnerability Database (NVD)
- Commercial
  - Security Focus
  - Darkreading
  - Etc.

Developing and Maintaining Software-based Systems

Intro description of scope and purpose

- Software Security Principles
  - BSI catalog
- Processes & Practices
  - Software Assurance Developer's Guide
  - Policy
    - BSI articles
  - Security requirements definition
    - BSI articles
    - Requirements Patterns catalog (GAP)
    - SecDEF Security Requirements Description and Exchange Formats (SRDEF)
  - Architectural Risk Analysis
    - BSI articles
    - Threat Modeling
      - BSI articles
  - Secure design
    - Security patterns
      - SecurityPatterns.org
    - Secure design guidelines
      - BSI catalog
        - Need more, better and broader content (GAP)
  - Building Security In
    - Weakness & Vulnerability
    - Attacker's perspective
      - Attack Patterns
        - Attack Patterns whitepaper (BSI)
        - Common Attack Pattern Enumeration & Classification (CAPEC)
      - Threat Modeling
        - BSI articles
    - Security requirements
      - Misuse & Abuse Cases
        - Misuse & Abuse Cases whitepaper (GAP)
    - Etc.
  - Secure code review
    - BSI articles
  - Security testing
    - BSI articles
    - Risk-based testing
    - Penetration Testing
      - BSI articles
      - Red Teaming
  - Assembly, Integration & Evolution
    - BSI Articles
  - Enterprise Software Assurance (GAP)
    - Needs broad set of content covering the complex issue and guidance surrounding software assurance at the enterprise level (GAP)
  - Project Management
    - BSI articles
  - Risk Management
    - BSI articles
  - Process Capability (Best Practices)

        - Capability Maturity Model (CMM/CMMI) (SEI)
        - Systems Security Engineering Capability Maturity Model (SSE-CMM)
        - Integrated Capability Maturity Model (iCMM)
        - ASQ (American Society for Quality)
        - SPICE (Software Process Improvement and Capability dEtermination)
        - Praxis CbyC
        - Fraunhofer Institute for Experimental Software Engineering (IESE)
        - CSE (Center for Software Engineering), University of Southern California
        - CAESER (Centre for Advanced Empirical Software Research), U. of New South Wales
        - LASER (Laboratory for Advanced Software Engineering Research), University of Massachusetts Amherst
        - TickIT (U.K. and Sweden)
        - Information Technology Infrastructure Library (ITIL)
        - Information Services Procurement Library (ISPL)
        - Application Services Library (ASL)
        - Dynamic Systems Development Method (DSDM)
        - Control Objectives for Information and related Technology (CobiT)
        - Experimental Software Engineering Group (ESEG), University of Maryland
        - Informatics Process Group, University of Manchester, U.K.
        - Distributed Software Engineering Group, Imperial College, London, U.K.
        - EPOS (Expert System for Program and System Development)
        - Software Process Improvement Lab, Texas A&M University
        - Information Technology Services Qualification Center (ITsqc) – Carnegie Mellon
- Standards
    - ISO/IEC
        - ISO/IEC JTC 1/SC 27
        - ISO/IEC 21827
        - DCID 6/3
        - ISO SC22 Committee on Secure Language Usage
        - 15026
    - IEEE
    - NIST
        - NIST FISMA Standards Efforts
    - OMG
        - Knowledge Discovery Metamodel (KDM)
- Measurement
    - BSI articles
    - Software Measurement Laboratory (SML), University of Magdeburg, Germany
    - Software Quality Research Laboratory, U. of Tennessee
    - M Squared Technologies RSM (Resource Standard Metrics)
    - Semantic Designs
- Tools
    - NIST Software Assurance Metrics and Tool Evaluation (SAMATE) Project
        - Software Assurance Tool Taxonomy
        - Source Code Analysis Tool Functional Specification
    - CAMP
    - Weakness Identification Tools
        - COTS
            - Fortify Software Source Code Analysis Suite
            - Klocwork K7 Development Suite
            - Ounce Labs Prexis
            - Secure Software CodeAssure

- Coverity Prevent
- Grammatech CodeSonar
- Parasoft Jtest
- SPI Dynamics DevInspect & SecureObjects,
- Compuware DevPartner SecurityChecker
- Microsoft Prefix, Prefast, FXCop
- SofCheck Inspector
- LogicLibrary Logiscan

▪ Open Source
- Cigital ITS4
- Flawfinder
- Secure Software RATS (Rough Auditing Tool for Security)
- Splint 3.1.1

▪ GOTS

o Analysis Tools
▪ IDA Pro
▪ Grammatech Code Surfer
▪ Carnegie Mellon (Cylab) FX
▪ Praxis SPARKAda
▪ Understand C++
▪ HBGary Inspector
▪ Microsoft SLAM
▪ Reflective Prism

- Weakness
  - o Common Weakness Enumeration (CWE™)
  - o Secure Coding Rules
    - ▪ BSI catalog
    - ▪ Tool vendors
- Patterns
  - o Design patterns
  - o Security patterns
    - ▪ SecurityPatterns.org
  - o Requirements patterns catalog (GAP)
  - o Attack patterns
    - ▪ Common Attack Pattern Enumeration & Classification (CAPEC)
- Modeling
  - o Knowledge Discovery Metamodel (KDM)
  - o Microsoft's System Definition Model (SDM)
  - o BSI articles
- Software Components
  - o COTS (too numerous to mention – category alone likely sufficient for our purposes)
    - ▪ Secure operating systems
      - Green Hills Software
      - OpenBSD
      - Coyotos
  - o GOTS (too numerous to mention – category likely alone sufficient for our purposes)
  - o Open Source
    - ▪ SourceForge
    - ▪ Free Software Foundation
    - ▪ Freshmeat
    - ▪ OpenSource Directory (OSDIR)
    - ▪ BerliOS

Operation and Maintenance of Systems and Networks

Intro description of scope and purpose

- Announcements
  - US-CERT
  - Common Announcement Interchange Format (CAIF)
  - National Infrastructure Security Co-ordination Centre (NISCC)
- Threat Assessment
  - National Infrastructure Security Co-ordination Centre (NISCC)
  - WASC Threat Classification
- Incidents
  - US-CERT
  - National Infrastructure Security Co-ordination Centre (NISCC)
  - SecDEF Incident Object Description and Exchange Formats (IODEF)
  - SecDEF Intrusion Detection Description and Exchange Formats (IDDEF)
  - TERENA Computer Security Incident Response Teams Task Force (TF-CSIRT)
  - BSI articles
- Vulnerability
  - Common Vulnerabilities and Exposures (CVE®)
  - Open Vulnerability and Assessment Language (OVAL™)
  - National Vulnerability Database (NVD)
  - US-CERT
  - Secunia
  - Open Source Vulnerability Database (OSVDB)
  - OWASP Top Ten
  - SANS Top Twenty
  - National Infrastructure Security Co-ordination Centre (NISCC)
  - SecDEF Susceptibility and Flaw Description and Exchange Formats (SFDEF)
  - SecDEF Vulnerability & Exploit Definition and Exchange Format (VEDEF)
  - Email Lists
    - Full Disclosure
    - BugTraq
    - WebAppSec
  - Tools & Standards
    - Common Vulnerability Scoring System (CVSS)
    - CERIAS/Perdue University's Cassandra
    - CERIAS/Perdue University's CVE Change Logs
- Configuration
  - Common Configuration Enumeration (CCE™)
  - Extensible Configuration Checklist Description Format (XCCDF)
  - Center for Internet Security (CIS) Benchmarks
  - DISA Security Technical Implementation Guides (STIGS)
  - NIST Security Configuration Checklists
  - SecDEF Secure Configuration Description and Exchange Formats (SCDEF)
  - Interoperability
    - OMG Common Secure Interoperability, Version 2
    - SecDEF Secure Interoperability Description and Exchange Formats (SIDEF)
- Management
  - Distributed Management Task Force (DMTF)
    - DMTF's Common Information Model (CIM)
    - DMTF's Web-Based Enterprise Management (WBEM)
    - DMTF's Web Services for Management (WS-Management)
    - DMTF's Systems Management Architecture for Server Hardware (SMASH)

- Microsoft's Dynamic Systems Initiative (DSI)

- Attacks, Malware, Spyware & Grayware
  - Common Attack Pattern Enumeration & Classification (CAPEC)
  - Malware
    - Common Malware Enumeration (CME™)
    - ICSA Labs Wild List
    - Offensive Computing
    - NSA Malicious Code Tiger Team
    - Malware Attribute Enumeration and Categorization (MAEC) (GAP)
    - Antivirus
      - Symantec
      - Norton
      - Etc.
  - Spyware
    - Antispyware Coalition
  - Grayware (droppers, keyloggers, etc.)
  - Exploits
    - SecDEF Vulnerability & Exploit Definition and Exchange Format (VEDEF)
- Forensics
  - SecDEF Forensic Investigation Description and Exchange Formats (FIDEF)
- Vulnerability Detection Tools
  - Citadel Security Software
  - BearingPoint
- Many, many more

Evaluating, Certifying, Reviewing, and Monitoring Compliance of Software-base Systems

Intro description of scope and purpose

- Standards
  - International
    - ISO/IEC 15443-3 FRITSA
    - ISO/IEC 15408 (Common Criteria)
  - National
    - NIST FISMA Standards Efforts
    - NSTISSP 11
    - Common Criteria Evaluation & Validation Schema (CCEVS)
    - NIST SAMATE RDS
  - DOD
    - DOD 8500
    - DCID 6/3
    - DIACAP
    - IATF
    - NIAP
  - Public law
    - SOX
    - BASEL II
    - HIPAA
    - FIRPA
  - Commercial
    - Payment Card Industry (PCI) Data Security Standard
  - Open Source
    - SourceForge
    - Free Software Foundation
- C&A
  - Software Assurance Registry (systems results against SAMATE RDS) (GAP)
  - Etc.
- Tool evaluations
  - NIST Software Assurance Metrics and Tool Evaluation (SAMATE) Project
    - Reference Data Set
      - Software Assurance Registry (tools results against SAMATE RDS) (GAP)
  - CAMP
  - Universities
    - Georgia Tech Research Institute (GTRI) – Freeland Abbott
- People Certifications
  - SANS Institute
- Labs
  - Commercial
    - West Coast Labs (UK)
    - ICSA Labs
    - Wyle Labs
    - Mier Communications
    - Tolly
    - AppLabs (Key Labs)
  - NIAP
    - URS EG&G
    - General Dynamics – Network Systems (GD-NS)

- NAVAIR
- Lockheed Martin
- BAESystems
- SRI International
- Tresys
- proServices
- Innerwall

- National Voluntary Laboratory Accreditation Program (NVLAP)

Formalization and Enabling Technologies for Implementing Security Guidelines and Specifications

Intro description of scope and purpose
Blue = Work in progress
Red = Work not started yet

- Interoperability Tooling Standards designed to connect individual tools into full solution required to support security guidelines and specifications
  - OMG Standards adopted by ISO:
    - ISO/IEC 19502 (MOF) / OMG MOF;
    - ISO/IEC 19501 (UML) / OMG UML2;
    - ISO/IEC 19503 (XMI) / OMG XMI;
  - OMG standards proceeding with ISO adoption:
    - OMG ADM Knowledge Discovery Metamodel (KDM)
    - OMG Semantics of Business Vocabulary and Rules (SBVR)
  - OMG standards:
    - OMG Common Warehouse Metamodel (CWM)
    - OMG Business Process Metamodel (BPM)
    - OMG Information Management Metamodel (IMM)
    - OMG ADM Metrics Metamodel
    - OMG Software Assurance Metamodel
- Formalization of security standards and guidelines as enabled by interoperability tooling standards
  - Common Weakness Enumeration (CWE)
  - ISO/IEC 15408 (CC) - Protection Profile
- SwA tools complying with interoperability tooling standards
  - List of tool types and tool vendors that comply with interoperability standards:
    - Formal specification tool:
      - IBM/Rational
      - Unisys
      - Telelogic
    - Code Generators:
      - IBM
      - Borland
      - CA
      - SUN
    - Architecture Analysis:
      - IBM/Rational
      - Borland
    - Repository:
      - ASG
      - Adaptive
    - Reusibale Assets Repository:
      - Flashline/BAE
    - Source code extractors :
      - ASG
      - Relativity
    - Binary Extractors:
      - KDM Analytics
    - Rule Mining (technology that extracts polices from existing code):
      - KDM Analytics
      - Relativity

  - SwA Ecosystem Infrastructure Development Kit built on OMG open standards to enable security-related tools to interoperate
  - Infrustructure RunTime System with APIs for tool plug & play
  - Document all tool types and tools candidates for SwA Ecosystem Infrastructure plug & play environment creating full SwA solution
- SwA tool types and tools currently available but not compliant with any interoperability standards (point tools)
  - Source code Analyzers:
    - refer to SAMATE project http://samate.nist.gov/index.php/Source_Code_Analyzers
  - Binary code Analyzers:
    - refer to CAMP project
- Methodologies
  - Tooling methodologies – how tools will adopt to infrastructure
  - SwA methodology – how to apply tools to perform full SwA evaluations
  - Task–driven methodology – how to apply tools to perform particular SwA task
- Tools Certification & Evaluation
  - Certify tools for SwA Ecosystem infrastructure compliance
  - Evaluation of source and binary code analysis tools
    - NIST SAMATE Project
    - CAMP Project
- Reference implementation for Task-oriented SwA activities/projects
  - CWE-enabled Test case generator for NIST's SAMATE project: produces tests that are used to evaluate tools compliance for CWE
  - CWE-enabled analysis of software used in Lab evaluation
  - Protection Profile specification-driven analysis used in Lab evaluations
- Technologies in support of Vulnerability description formats and data exchange formats
  - SecDEF Susceptibility and Flaw Description and Exchange Format
  - SecDEF Vulnerability & Exploit Definition and Exchange Format
  - SecDEF Incident Object Description and Exchange Formats (IODEF)
  - SecDEF Incident Detection Description and Exchange Formats (IDDEF)
  - SecDEF Secure Configuration Description and Exchange Formats (SCDEF)
  - SecDEF Secure Interoperability Description and Exchange Formats (SIDEF)
  - Open Vulnerability and Assessment Language (OVAL)

Research & Development (R&D)

Intro description of scope and purpose

- Academic
  - CMU Cylab
  - CERIAS/Purdue
  - Software Engineering Research Laboratory, University of Colorado, Boulder
  - Carnegie Mellon
- Government
  - National Infrastructure Security Co-ordination Centre (NISCC)
  - NSA
  - Defense R&D Canada (DRDC)
  - Lawrence Livermore National Lab
- Non-Profit
  - Kestrel Institute
- Industry
  - IBM Research
- Lots missing here (most companies listed throughout the other sections have some sort of research activities.)

Education

Intro description of scope and purpose

- CBK
- National Centers of Academic Excellence in Information Assurance Education (CAEIAE)
    - Auburn University
    - University of Alaska Fairbanks
    - Naval Postgraduate School
    - University of California at Davis
    - California State Polytechnic University, Pomona
    - United States Air Force Academy
    - Florida State University
    - Nova Southeastern University
    - Clark Atlanta University
    - Georgia Institute of Technology
    - Kennesaw State University
    - Idaho State University
    - University of Idaho
    - DePaul University
    - Illinois Institute of Technology
    - University of Illinois at Urbana-Champaign
        - Information Trust Institute (ITI)
            - The Trustworthy Cyber Infrastructure for Power (TCIP) Center
            - The NSA IA Education Center
            - Trusted ILLIAC
            - The Boeing Trusted Software Center
            - NCASSR
    - Purdue University
    - Iowa State University
    - University of Louisville
    - University of New Orleans
    - Capitol College
    - Johns Hopkins University
    - Towson University
    - University of Maryland, Baltimore County
    - University of Maryland University College
    - Boston University
    - Northeastern University
    - University of Massachusetts, Amherst
    - University of Detroit, Mercy
    - Walsh College
    - Eastern Michigan University
    - University of Minnesota
    - Mississippi State University
    - University of Nebraska at Omaha
    - New Jersey Institute of Technology
    - Stevens Institute of Technology
    - New Mexico Tech
    - Pace University
    - Polytechnic University
    - Rochester Institute of Technology
    - State University of New York, Buffalo
    - State University of New York, Stony Brook

  - Syracuse University
  - U.S. Military Academy, West Point
  - North Carolina State University
  - University of North Carolina, Charlotte
  - East Carolina University
  - Air Force Institute of Technology
  - Ohio State University
  - University of Tulsa
  - Oklahoma State University
  - Portland State University
  - Carnegie Mellon University
  - Drexel University
  - East Stroudsburg University of Pennsylvania
  - Indiana University of Pennsylvania
  - Pennsylvania State University
  - University of Pittsburgh
  - West Chester University of Pennsylvania
  - Dakota State University
  - Fountainhead College of Technology
  - University of Memphis
  - Southern Methodist University
  - Texas A&M University
  - University of Dallas
  - University of North Texas
  - University of Texas, Dallas
  - University of Texas, San Antonio
  - Norwich University
  - George Mason University
  - James Madison University
  - Virginia Polytechnic and State University
  - University of Washington
  - George Washington University
  - Information Resources Management College
- Other Educational Institutions/Training
  - Stanford
  - Defense Acquisition University (DAU)
  - National Defense University (NDU)
  - Computing Technology Industry Association (CompTIA)
  - Educause
  - York University (york.ac.uk)
  - SANS Institute
  - Global Knowledge
  - DevelopMentor
  - National Information Assurance Training and Education Center (NIATEC)

Acquisition & Marketing

Intro description of scope and purpose

- Assurance cases
  - Software Assurance Case whitepaper (GAP)
  - KDM ecosystem
- DHS Acquisition manager's guide
- NSTISSP 11
- NIST SP800-23
- US Air Force Enterprise Agreement with Microsoft

Forums, Conferences, Colloquia, Working Groups, etc.

Intro description of scope and purpose

- DHS/DOD Software Assurance Forum
- DHS/DOD Software Assurance Working Groups
- SAMATE Workshops and Working Groups
- OMG SwASIG Workshops and Working Groups
- RSA
- USENIX
- Software Security Summit
- A slew of gov't ones are missing here
- Etc.

5. Software Assurance Domain Summaries

This area will give textual summaries of the context and relationships between knowledge, activities and initiatives in each area.

Communities & Leadership

Need a prose summary of this area

Developing and Maintaining Software-based Systems

When talking about the security and integrity of software systems during their development and maintenance there are a variety of existing and newly formed efforts, as well as on-going research, that are introducing standardization concepts and methods. Traditional software development has focused on developing functionality within predictable cost and schedules. With the wider use and dependence on software in all types of industries, government, academia, and individual's lives it has become clear that the security features and operational secureness of the software must be considered more aggressively. Gathering and sharing good practices in this area has been going on for several years by efforts like Purdue's CERIAS project. In recent years the Open Web Application Security Consortium (OWASP) and Web Application Security Consortium (WASC) have helped codify and promulgate best practice guidelines for securing web-based applications to the point that the Federal Trade Commission has used the OWASP Top Ten list of secure web application guidance as part of the test for whether companies being investigated by the FTC have done their due diligence with respect to a company's web sites and web applications.

The Department of Defense (DoD) and the Department of Homeland Security (DHS) have both been working to better address the operational secureness of the software used in their respective areas of responsibility through a variety of efforts. The joint DoD/DHS Software Assurance Working Groups and Forums are a collection of bi-monthly invitation only groups working to define, organize, and improve many different aspects of secureness of software. Current working groups, which meet every other month in the Washington D.C. area, include working groups for: Software Assurance Business Case; Software Assurance Acquisition; Software Assurance Processes and Practices; Software Assurance Measurement; Software Assurance Tools and Technology; and Software Assurance Workforce Training. Every third gathering includes a Software Assurance Forum that provides a larger context for keynote presentations and public discussions. The DoD/DHS Software Assurance Working Groups and Forums have many "products" underway as vehicles for capturing the work of the groups and making their work generally available.

The first three "products" of the current working groups are: the DHS "Build Security In" web site, a collection of best practices and discussions about many aspects of building secure software-based systems that is presented in a development methodology neutral manner; the DHS Software Assurance Common Body of Knowledge document, a collection of knowledge and practices about developing and fielding secure software systems but targeted as additions to the practices of software engineering, systems engineering, project management, testing, acquisition, maintenance, deployment, and usage of software-based systems rather than a new area of practice in and of itself; and the DHS Security in the Software Lifecycle document, a collection of ideas and approaches for making software development processes and software produced by them more secure. Enhancements continue on the web site to reflect the ongoing work of the working groups and the various related activities in the community and similar enhancements are planned for the two documents as well.

Several national and international organizations, including NIST, DoD, NSA, OMG, and ISO/IEC/IEEE, as well as commercial and academic organizations are working with and within the DoD/DHS Software Assurance Working Groups and Forums framework on complimentary activities. Some, like the Software Assurance Metrics and Tool Evaluation (SAMATE) at NIST are being funded partially by DHS. The NIST SAMATE effort is focused on developing formal taxonomies to describe the different types of tools that can be applied to the software and systems assurance challenge. Through as series of workshops, conferences, and the activities of the Tools and Technology Working Group, the SAMATE project is developing standard ways of describing the features and capabilities of tools that can be brought to bare on this arena.

In addition to its work within the DoD/DHS Software Assurance Working Groups and Forums, DoD has: a Tiger Team that has been studying their challenges with regard to securing software-based systems; a Defense Science Board of the topic; the development of a Center for Assured Software for DoD, and they are reaching out to industry for ideas and solution approaches through efforts with the National Defense Industrial Association (NDIA), the Association for Enterprise Integration (AFEI), and the Object Management Group (OMG).

To support the software and systems assurance efforts, the Object Management Group (OMG) has created the Software Assurance (SwA) Special Interest Group (SIG) which is pulling together the knowledge, technology, and methods to develop some appropriate OMG technology standards to support assessing the secureness of software-based systems. The NDIA, after convening a workshop and summit on software assurance to explore the range of opportunities for a long-term solution to the issue of software assurance, has established the NDIA System Assurance Committee, which is chartered to assure the effective functionality of DoD's command, control, communications and related weapon systems with high confidence that the systems are not vulnerable to intrusion and cannot be compromised. The primary focus of this effort is the development of an Acquisition Handbook for System Assurance. This effort is in parallel, and leveraged with, the efforts of the DoD/DHS Software Assurance Acquisition Working Group.

The DoD's Center for Assured Software (CAS) is in the NSA. The CAS, which subsumed the current NIAP program office, which manages the application of the Common Criteria evaluations for the U.S. government, is actively pursuing the application of automation to the task of assuring that software-based systems are free of known types of software weaknesses and do not have malicious functionality hidden within them.

The Common Weakness Enumeration (CWE) effort is a community effort, tied closely to the NIST SAMATE tool evaluation work, the NSA assurance efforts, and the OMG's Software Assurance SIG. Aimed at developing a common dictionary of the software, design, and architecture weaknesses that lead to vulnerable software, the CWE initiative is bringing together the knowledgeable commercial, academic, and government parties to hammer out a reference source that can be used as the basis of teaching, training, assessment, and certification of systems for known types of weaknesses that can be exploited. Closely tied to the CVE initiative, which catalogues specific cases where these weaknesses have been discovered in software products that are in operational use, the CWE initiative is helping create the foundation for tools and assessment methods to vet software for these issues before they end up in the operational versions of the software applications and operating systems.

A major aspect of the SAMATE project is the development and open sharing of test applications that have been salted with known weaknesses so that those that wish to see how effective a particular tool or technique is in finding that type of weakness will have readily available test materials to use. The SAMATE Test Reference Datasets (TRDs) are organized by CWE weakness types and will also include varying levels of complexity, as appropriate to each type of weakness, so that tools that are more or less effective in finding complex examples of weaknesses can be identified. Correct constructs, that are closely aligned to the CWEs but are correct implementations, will also be included in the TRDs to help identify the false positive effectiveness of the tools.

It is important that the SAMATE TRDs and the CWE dictionary definitions of the weaknesses clearly articulate what a weakness's complexity means so that the selection of tools are not based on their ability to handle the most complex situations when the types of applications that an organization will use the tools to examine are much simpler and thus a less "capable" tool could be chosen – or in most cases – several tools that would otherwise be passed by would be available for selection based on their effectiveness against the simpler examples of a weakness's complex situations. So an important aspect of the SAMATE TRDs will be the concise explanation of what the various complexities of the weaknesses are and how a potential user of test results can look at their own software, determine its complexity for a particular weakness, and identify what level of the weakness complexity TRDs they need a tool to be effective against.

The public SAMATE TRDs would be accompanied by the ability for NIST to create new custom TRDs on an adhoc basis to support tools being assessed for their ability to find CWEs in sample applications that they have not previously seen, that include test applications indicative of the various complexities those CWEs can manifest, and are customized to a subset of CWEs that the tool claims to be effective against. Claims of compatibility with a set of CWEs (called CWE Compatibility) will be accepted along with the tools results against a custom set of TRDs. The results for tools will not be evaluated, but rather they will be posted to record each particular tool's effectiveness for the set of CWEs it claims to cover and then each organization will need to review the results to decide whether a particular tool was effective enough for the level of complexity that is representative of the software that the organization wishes to use the tool on.

The OMG's Software Assurance SIG will be using CWEs as one type of software issue that tools will need to be able to locate within the eventual OMG Software Assurance technology approach. One part of the OMG's approach will include the use of the Semantics of Business Vocabulary and Business Rules (SBVR) language to articulate modelable expressions of the different CWEs. SBVR expressions of each CWE will be included in the definitions of each CWE as the official version of the SBVR expression of that CWE.

ISO/IEC/IEEE standards efforts
ISO SC22 Committee on Secure Language Usage
Common Attack Patterns Enumeration and Categorization (CAPEC)
Malware Attribute Enumeration and Categorization (MAEC)

The two major segments of this topic area are traditional applications and web-based applications.

[lots more going in here]

Operation and Maintenance of Systems and Networks

Once a system's components and networks are operational there are several activities needed to ensure that they work in a secure manner and in conformance to the objectives and policies of the organization. One of the oldest of the efforts helping with this is the Common Vulnerabilities and Exposures (CVE) Initiative, which provides unique identifiers for each publicly known software vulnerability, along with a list of the other names for the vulnerability. While CVE is focused on assigning the unique identifier, the National Vulnerability Database (NVD) goes further by taking each CVE and adding additional information about the vulnerability, its impact, remediation, and severity. The severity rating is provided through the Common Vulnerability Scoring System (CVSS) base score, which provides a standardized way of establishing the fundamental risk posed by a vulnerability based on its basic type and nature. Organization can then use the rest of the CVSS measurement approach to add in a measure based on the current risk from the activity related to exploiting the vulnerability and a measure based on the status and placement of the vulnerable software in the organization making the risk calculation. Another effort, the Open Vulnerability and Assessment Language (OVAL) provides xml-based definitions on how to evaluate whether a specific vulnerability is on a system or it can be used to define how to check for whether a system is configured in a particular manner. OVAL works with the XML Configuration Checklist Data Format (XCCDF) standard to describe low-level policy about how a system should be configured. The Common Configuration Enumeration (CCE) is used to identify common configuration concepts so that they can be correlated and managed more consistently across different tools and platforms. The Cassandra effort, part of Perdue's Center for Education and Research in Information and Security (CERIAS) project, allows organizations to sign up for email notifications about new CVE's being issued for applications or operating systems that they describe in a profile they register in Cassandra. The UPPN (Platform) effort is working to identify a standard method for describing concepts like a particular operating system version or application version so that efforts like OVAL and XCCDF can use them as building blocks within their definitions as well as integrate with inventory management capabilities through common concepts and identifiers. The Center for Internet Security (CISecurity) develops and maintains concensus guides on how to securely configure many common operating systems and applications using XCCDF and OVAL. Similarly, the National Institute of Standards and Technology (NIST) provides machine consumable guidance on recommend configuration settings for compliance to FISMA using XCCDF and OVAL. The Department of Defense (DoD) Information Systems Agency (DISA) uses CVE and OVAL to monitor the Information Assurance status of the DoD systems on the Global Information Grid (GIG) by having its own repository of OVAL definitions to check for the vulnerabilities they care about as well as OVAL definitions to check systems for compliance to the DISA System Technical Implementation Guides (STIGS). The DoD also communicates to its constituents about which vulnerabilities need to be addressed by enumerating them through a list of CVEs within each of the Information Assurance Vulnerability Alerts, Bulletins, and Technical Advisories (IAVA, IAVB, and IAVTA). Focusing attention at critical issues by listing the CVEs is also used by the SANS Institute in their semi-annual Top Twenty List.

The Open Management Consortium's SML (Service Modeling Language), Microsoft's System Definition Model (SDM) and Dynamic Systems Initiative (DSI) are efforts within a set of software users and suppliers for describing status of systems and managing them through system models and model transition statements.

In addition to managing the vulnerabilities and configurations of systems, organizations must manage the malicious threats facing theirs systems and networks. Initiatives contributing to organizing and improving this area include: the Common Malware Enumeration (CME) effort, which is helping bring clarity to which of the numerous virii circulating are in fact the same threat and can be managed with the same methods; the Anti Spyware Consortium (ASC), which is working to organize and define the variety of types of spyware; and the ICSA Lab's Wild List effort which keeps track of any virii that has been reported more than two times; the new “Offense Software” effort, which is also aimed to help manage the threats posed by malware; and the National Security Agency (NSA) guidance paper on dealing with malcode.

Finally, communicating with other organizations about incidents and ongoing issues has spawned a couple efforts of standardization. These include the VEDEF, SIDEF, SCDEF, SFDEF, IDMEF, IODEF, and FIDEF efforts being fostered by the NISCC and the CAIF effort out of Stutgardt Germany. Individual event information has had some work in the IETF...

CIEL, OSVDB, OWASP Top 10, FTC Guidance and Penalties, VISA and MasterCard PCI (Payment Card Industry) security standards, SEI’s Cylab, USCERT VulnNotes, Advisories, and Incident handling.

[I am sure there is more that needs to go in here]

Evaluating, Certifying, Reviewing, and Monitoring Compliance of Software-based Systems

NIAP
Common Criteria
DITSCAP
DICAP
SoX
HIPAA
FISMA
DoD/Certification and Accreditation Revitalization

Formalization and Enabling Technologies for Implementing Security Guidelines and Specifications

As software organizations recognized the need to evolve their systems beyond homogeneous and monolithic solutions to ones with architectures that support COTS, multi-operating environments and multi-languages, security became a major challenge. At the same time the tooling industry that provides enabling technologies for building secure software systems did not keep pace with software system evolution. This created a large gap where point tools were available but unable to be applied by a significant portion of the software industry. This caused an even greater security exposure. The gap became even larger when communities, in an effort to increase system security, focused on developing/imposing more standards and guidelines, but ignored evolution of technologies that would enable organizations to cost effectively implement them and certify against them.

Achieving a breakthrough in addressing this vicious circle of lower and lower levels of assurance requires the collaboration of all the key stakeholders, including tool vendors, software integrators and software suppliers. One effort targeted at addressing this problem is the SwA Initiative of the Object Management Group (OMG). Under the OMG umbrella, a set of stakeholders have developed a set of interoperability standards that would enable point tools to integrate into a complete solution which software suppliers would be able to use as enabling development technologies during SDLC. A number of these are already ISO standards and the others are currently undergoing that procedure.

Further collaboration between these stakeholders has led to an understanding that no one type of tool could fully address the security issue, but rather, it requires the integration and collaboration

of all of them. This collaboration has resulted in the formation of the Software Assurance Ecosystem, another potential piece of the puzzle targeted at addressing these issues.
The Software Assurance Ecosystem would provide a collaborative environment for technologies, tools and methodologies focused on software assurance. The Ecosystem is entirely based on OMG open standards for information exchange between all software assurance tools, creating a full and comprehensive solution that dramatically reduces the time and cost associated with software assurance activities.

The key enabler of this option is the Software Assurance Ecosystem Infrastructure, which is a tools and tooling standards integration and inter-operation environment. The infrastructure provides an OMG open standards based platform that supports the adaptation of all existing tool vendor products and reduces the barriers to entry for new tool vendors through its creation of a standard definition of software system artifacts and security/business rule claims. It brings together the three key domains of: static analysis, formal methods and extraction technology – and allows domain experts in each area to provide best in class solutions that integrate and inter-operate.

The SwA Ecosystem Infrastructure would provide a platform/enabling technology for secure programming, security testing and certification. It would bring together current and new players. SwA tool vendors that are not yet complying with any of the interoperability standards can bridge that gap through “bridge” technology that would be provided as part of the overall infrastructure.

The SwA Ecosystem Infrastructure is defined but not implemented yet. It is proposed to be implemented by providing a Software Development Kit, Run Time System and APIs that would serve as plug-ins for tools.

What matters is use, not existence of a technology. The technology transition goal for security technologies funded by the U.S. government should be pervasive use in applicable contexts, and not achieving a “checkmark” saying “released to public” or “intellectual property rights for government research have been sold/given to a commercial company.” Where possible, the technology transition approach should be selected to encourage multiple competing vendors, so that the government and industry users can pick the best value for the money. In some cases, simply funding research sufficient for the commercial world to build a tool may be sufficient. In other cases, widespread use may be encouraged by releasing a tool (which may be just a prototype) to the public domain or as open source software (OSS), then allowing multiple vendors to productize and commercialize it. That way, everyone can use the work developed and paid for by the government, and commercial ventures can improve it, sell support, etc. as they see fit. In some cases, requiring free use for the government may be sufficient, but such approaches have risks: if the technology transition approach does not support competition, the government may find its “free” use requires more money than a competed product due to poor sustainment and support. In yet other cases, widespread use may be best achieved by working with standards bodies to develop enhancements (e.g., to existing language standards), or by developing new standards, so that the commercial world will implement the approach in a consistent way. For example, this might be a useful approach for transitioning language subsets into practice. The government should be wary of any transition technique that enables any single vendor to monopolize an approach, since without competition, the government may end up paying far more than necessary or receive far less service than it needs.

Research & Development (R&D)

Need a prose summary of this area

Education

Need a prose summary of this area

Acquisition & Marketing

Need a prose summary of this area

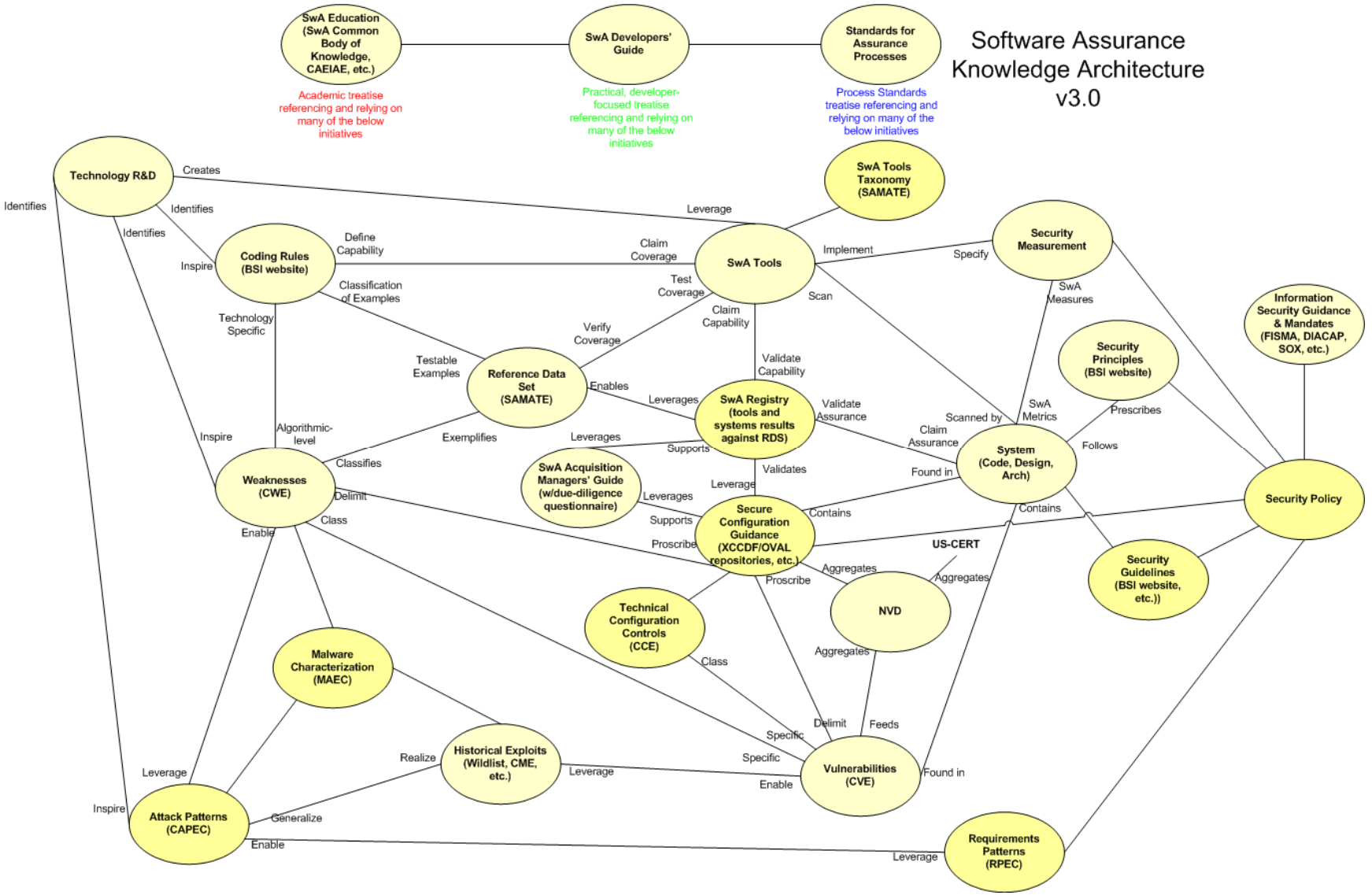
Forums, Conferences, Colloquia, Working Groups, etc.

Need a prose summary of this area

6. Graphical Representations of Landscape

Describe purpose, context and scope for graphical representations.
Varying levels of abstraction as appropriate

Software Assurance Knowledge Architecture



Software Assurance Knowledge Architecture Elements

Attack Patterns

Description:
An Attack Pattern is a blueprint for exploiting a software vulnerability. Attack Patterns are developed from analyzing existing software exploits (hacks, malware, spyware, etc.) and abstracting and identifying common patterns in these approaches. Attack Patterns are excellent mechanisms for identifying and qualifying the risk that a given exploit will occur within a software system. They are also useful as counterexamples for software designers and implementers to avoid or block relevant patterns of attack.
Status: Nascent
Related Organizations, Activities & Knowledge:

- BSI – Attack Pattern article series
- Common Attack Pattern Enumeration and Classification (CAPEC)

Coding Rules

Description:
Coding Rules are representations of how to check for the presence of specific weaknesses within a specific context such as a given programming language. There are often multiple coding rules for any given weakness. These coding rules can be used to guide manual code review or to feed automated scanning tools.
Status: Warrants Enhancement
Related Organizations, Activities & Knowledge:

- BSI – Coding Rules catalog

Historical Exploits

Description:
Status:
Related Organizations, Activities & Knowledge:

- Wildlist
- Common Malware Enumeration (CME)

Information Security Guidance & Mandates

Description:
Status:
Related Organizations, Activities & Knowledge:

- FISMA
- DIACAP
- SOX
- HIPAA
- Etc.

Malware Characterization

Description:
Status: Gap
Related Organizations, Activities & Knowledge:

- Malware Attribute Enumeration and Categorization (MAEC)

NVD

Description:
The U.S. National Institute of Standards and Technology's (NIST) National Vulnerability Database (NVD) “is a comprehensive cyber security vulnerability database that integrates all publicly available U.S. Government vulnerability resources and provides references to industry resources.
Status: Ongoing
Related Organizations, Activities & Knowledge:

- NVD
- CVE
- OVAL
- XCCDF
- NIST Checklist Program
- CVSS

Reference Data Set

Description:
A set of test applications that have been salted with known weaknesses so that those that wish to see how effective a particular tool or technique is in finding that type of weakness will have readily

available test materials to use. The reference datasets will also include varying levels of complexity, as appropriate to each type of weakness, so that tools that are more or less effective in finding complex examples of weaknesses can be identified. Correct constructs, that are closely aligned to the weaknesses but are correct implementations, will also be included in the reference data sets to help identify the false positive effectiveness of the tools.
Status: Nascent
Related Organizations, Activities & Knowledge:

- SAMATE
- CWE
- NSA CAS
- OMG SwA SIG

Requirements Patterns
Description:
Status: Gap
Related Organizations, Activities & Knowledge:
Secure Configuration Guidance
Description:
Most organizations have established criteria for how they want the operating systems and standard applications configured for secure operation. These criteria are usually different from the way the software suppliers ship the software from their distribution facility. Originally these configuration guides took the form of paper documents or electronic documents that describe how to configure the software. In the future these configuration guidelines will be conveyed as machine readable xml policy documents that use a combination of XCCDF and OVAL along with CCE control names and UPPN package and platform identifiers.
Status: Warrants Enhancements
Related Organizations, Activities & Knowledge:

- NSA Guides
- DISA STIGS
- CIS Benchmarks
- Vendor guidelines on locking down their product
- XCCDF
- OVAL
- CCE
- UPPN

Security Guidelines
Description:
Status: Warrants Enhancements
Related Organizations, Activities & Knowledge:
Security Measurement
Description:
Status:
Related Organizations, Activities & Knowledge:
Security Policy
Description:
Status:
Related Organizations, Activities & Knowledge:
Security Principles
Description:
Status: Ongoing
Related Organizations, Activities & Knowledge:

- BSI – Principles

Standards for Assurance Processes
Description:
Status:
Related Organizations, Activities & Knowledge:

- ISO ???
- CVE
- CWE
- OVAL
- XCCDF
- UPPN

- KDM
- SBVR

SwA Acquisition Manager's Guide
Description:
Status:
Related Organizations, Activities & Knowledge:

SwA Developers' Guidance
Description:
Status:
Related Organizations, Activities & Knowledge:

SwA Education
Description:
Status:
Related Organizations, Activities & Knowledge:

SwA Registry
Description:
Status:
Related Organizations, Activities & Knowledge:

SwA Tools
Description:
Status:
Related Organizations, Activities & Knowledge:

SwA Tools Taxonomy
Description:
Status: Nascent
Related Organizations, Activities & Knowledge:

- SAMATE SwA Tools Taxonomy

System
Description:
Status:
Related Organizations, Activities & Knowledge:

Technical Configuration Controls
Description:
Security Policies normally include guidance on how to secure the configuration of software-based systems and their applications as well as guidance on the process and physical protections that are to be used. The portions of guidance that relate to actual settings and configurations of the software components are referred to as the Technical Configuration Controls but there is no consistency in the names or terminology used to describe the controls.
Status: Nascent
Related Organizations, Activities & Knowledge:

- Common Configuration Enumeration (CCE)
- XCCDF
- OVAL
- NIST Checklist Program

Technology R&D
Description:
Status:
Related Organizations, Activities & Knowledge:

Vulnerabilities
Description:
Software in systems and networks can be compromised through vulnerabilities. Vulnerabilities can be from either weaknesses in the software that need to be addressed or they can come from mis-configured software that allows activities and access that should not be allowed.
Status: Ongoing
Related Organizations, Activities & Knowledge:

- NVD
- CVE
- CVSS
- OVAL
- OSVDB
- CWE

- CCE

Weaknesses

Description:

Exploitable vulnerabilities in software are caused by weaknesses in the architecture, design, or implementation of the software or a combination of several weaknesses. There has been no central collection or source of definitions and terminology for discussing and describing weaknesses in software.

Status: Warrants Enhancement

Related Organizations, Activities & Knowledge:

- CVE
- CVSS
- OWASP
- WASC
- Microsoft's Secure Software Development Lifecycle

Software Assurance Organizational Architecture

Software Assurance Organizational Architecture Elements

Software Assurance Activity Architecture

Software Assurance Activity Architecture Elements

7. Software Assurance Knowledge, Activities and Initiatives

Anti-Spyware Coalition – http://www.antispywarecoalition.org
ASC is "dedicated to building a consensus about definitions and best practices in the debate surrounding spyware and other potentially unwanted technologies. Composed of anti-spyware software companies, academics, and consumer groups, ASC seeks to bring together a diverse array of perspective on the problem of controlling spyware and other potentially unwanted technologies."
Related with: (Related knowledge, activity or initiative) (Nature of Relationship)

The Application Security Industry Consortium (AppSIC) – http://www.appsic.org/
The Application Security Industry Consortium is a small community of security and technology experts united to establish and define the international cross-industry application security guidelines and measures, seeking to: bridge the gulf between application security issues & business needs; develop a yardstick for secure software development processes; generate application security assessment criteria; proliferate knowledge on application security; and provide insight on security return on investment.

Build-Security-In – http://buildsecurityin.us-cert.gov
Build Security In (BSI) is a project of the Strategic Initiatives Branch of the National Cyber Security Division (NCSD) of the U.S. Department of Homeland Security (DHS). "BSI content is based on the principle that software security is fundamentally a software engineering problem and must be addressed in a systematic way throughout the software development life cycle. BSI contains and links to a broad range of information about best practices, tools, guidelines, rules, principles, and other knowledge to help organizations build secure and reliable software."

Center for Internet Security (CIS) Benchmarks – http://www.cisecurity.org
The CIS Benchmarks are "consensus best practice standards for security configuration and are widely accepted by U.S. government agencies for FISMA compliance, and by auditors for compliance with the ISO standard as well as GLB, SOx, HIPAA, FIRPA and other the regulatory requirements for information security. For the first time ever, a large group of user organizations, information security professionals, auditors and software vendors have defined consensus technical control specifications that represent a prudent level of due care and best-practice security configurations for computers connected to the Internet."

CERIAS/Perdue University's Cassandra – https://cassandra.cerias.purdue.edu/main/index.html
CERIAS/Purdue University's free Cassandra tool monitors changes and updates to the U.S National Vulnerabilities Database (formerly ICAT) and the Secunia vulnerability databases. Cassandra saves lists of products, vendors, and keywords from these sources into "profiles" and emails any updates to subscribers. Users can create as many profiles as they want for networks, typical installs, important hosts, or any other areas of interest. CVE Change Logs, another free CERIAS tool, monitors changes to the CVE List.

Common Announcement Interchange Format (CAIF) – http://www.caif.info
CAIF is an XML-based format created by RUS-CERT at the University of Stuttgart, Germany, to store and exchange security announcements in a normalized way. It provides a basic but comprehensive set of elements designed to describe the main aspects of an issue related to security. The set of elements can easily be extended to reflect temporary, exotic, or new requirements in a per-document manner. CAIF documents are able to incorporate OVAL Definitions.

Common Configuration Enumeration (CCE™) – http://cce.mitre.org
CCE is the part of CVE that focuses on security configuration issues and exposures. CCE provides unique identifiers to system configurations in order to facilitate fast and accurate correlation of configuration data across multiple information sources and tools.

Common Malware Enumeration (CME™) – http://cme.mitre.org
CME provides single, common identifiers to new virus threats to reduce public confusions during malware outbreaks. CME is not an attempt to replace the vendor names currently used for viruses and other forms of malware, but instead aims to facilitate the adoption of a shared, neutral indexing capability for malware.

Common Vulnerabilities and Exposures (CVE®) – http://cve.mitre.org
International in scope and free for public use, CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVE's common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services. CVE is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

Common Weakness Enumeration (CWE™) – http://cwe.mitre.org
Targeted to developers and security practitioners, CWE is a formal or dictionary of common software weaknesses 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, and to provide a common baseline standard for weakness identification, mitigation, and prevention efforts. CWE is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

Common Vulnerability Scoring System (CVSS) – http://www.first.org/cvss/
Commissioned by the U.S. National Infrastructure Advisory Council (NIAC) in support of the global Vulnerability Disclosure Framework and currently maintained by the Forum of Incident Response and Security Teams (FIRST), CVSS is a "vendor agnostic, industry open standard designed to convey vulnerability severity and help determine urgency and priority of response. It solves the problem of multiple, incompatible scoring systems and is usable and understandable by anyone."

DISA Security Technical Implementation Guides (STIGS) – http://iase.disa.mil/stigs/index.html
The U.S. Department of Defense's (DOD) Defense Information Systems Agency's (DISA) Security Technical Implementation Guides (STIGS) are configuration standards for DOD information assurance and information assurance-enabled devices and systems.

Distributed Management Task Force (DMTF) – http://www.dmtf.org
The Distributed Management Task Force, Inc. (DMTF) is an international, industry "organization dedicated to the development of management standards and the promotion of interoperability for enterprise and Internet environments." DMTF standards "provide common management infrastructure components for instrumentation, control and communication in a platform-independent and technology neutral way."

DMTF's Common Information Model (CIM) – http://www.dmtf.org/standards/cim/
DMTF's Common Information Model (CIM) "is a common data model of an implementation-neutral schema for describing overall management information in a network/enterprise environment."

DMTF's Web-Based Enterprise Management (WBEM) – http://www.dmtf.org/standards/wbem/
DMTF's Web-Based Enterprise Management (WBEM) "is a set of management and Internet standard technologies developed to unify the management of enterprise computing environments."

DMTF's Web Services for Management (WS-Management) –
http://www.dmtf.org/standards/wsman/
DMTF's Web Services for Management (WS-Management) specification "promotes interoperability between management applications and managed resources by identifying a core set of Web service specifications and usage requirements to expose a common set of operations that are central to all systems management."

DMTF's Systems Management Architecture for Server Hardware (SMASH) –
http://www.dmtf.org/standards/smash
"DMTF's Systems Management Architecture for Server Hardware (SMASH) initiative is a suite of specifications that deliver architectural semantics, industry standard protocols and profiles to unify the management of the data center."

Extensible Configuration Checklist Description Format (XCCDF) –
http://checklists.nist.gov/xccdf.html
XCCDF was created by the U.S. National Security Agency (NSA) and National Institute of Standards and Technology (NIST) to be a specification language for providing a "uniform foundation for expression of security checklists, benchmarks, and other configuration guidance [to] foster more widespread application of good security practices." The default configuration checking technology for XCCDF is OVAL.

Microsoft's Dynamic Systems Initiative (DSI) -
http://www.microsoft.com/windowsserversystem/dsi/default.mspx
"The Dynamic Systems Initiative (DSI) is a commitment from Microsoft and its partners to deliver "self-managing dynamic systems" to help IT teams capture and use knowledge to design more manageable systems and automate ongoing operations, resulting in reduced costs and more time to proactively focus on what is most important to the organization."

Microsoft's System Definition Model (SDM) –
http://www.microsoft.com/windowsserversystem/dsi/sdm.mspx
System Definition Model (SDM) is a unifying thread enabling integrated innovation from Microsoft and its partners across application development tools, operating systems, applications, hardware, and management tools. SDM is a model that is used to create definitions of distributed systems. The SDM "blueprint can be created and manipulated with various software tools and is used to define system elements and capture data pertinent to development, deployment, and operations so that the data becomes relevant across the entire IT life cycle."

Microsoft's Secure Development Life Cycle (Secure SDLC) – http://msdn.microsoft.com/security/

National Cyber Security Partnership (was Cyber Security Industry Alliance) –
http://www.cyberpartnership.org/
The National Cyber Security Partnership (NCSP) is led by the Business Software Alliance (BSA), the Information Technology Association of America (ITAA), TechNet and the U.S. Chamber of Commerce in voluntary partnership with academicians, CEOs, federal government agencies and industry experts. Following the release of the 2003 White House National Strategy to Secure Cyberspace and the National Cyber Security Summit, this public-private partnership was established to develop shared strategies and programs to better secure and enhance America's critical information infrastructure. The partnership established five task forces comprised of cyber security experts from industry, academia and government. Each task force is led by two or more co-chairs. The NCSP-sponsoring trade associations act as secretariats in managing task force work flow and logistics. The task forces include: Awareness for Home Users and Small Businesses; Cyber Security Early Warning; Corporate Governance; Security Across the Software Development Life Cycle; and Technical Standards and Common Criteria. The task forces will be releasing separate work products beginning in March 2004 and ending in April 2004.

National Vulnerability Database (NVD) – http://nvd.nist.gov
The U.S. National Institute of Standards and Technology's (NIST) National Vulnerability Database (NVD) "is a comprehensive cyber security vulnerability database that integrates all publicly available U.S. Government vulnerability resources and provides references to industry resources. It is based on and synchronized with the CVE vulnerability naming standard." NVD also includes OVAL-IDs as references and is searchable by CVE-ID and OVAL-ID. NVD is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

NIST Security Configuration Checklists – http://checklists.nist.gov
The U.S. National Institute of Standards and Technology (NIST) Security Configuration Checklists Program for IT Products are checklists of settings and options selections that minimize the security risks associated with computer hardware and software systems used within the federal government. "Such checklists, when combined with well-developed guidance, leveraged with high-quality security expertise, vendor product knowledge, operational experience, and accompanied with tools, can markedly reduce the vulnerability exposure of an organization." The NIST Checklist Effort is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

NIST FISMA Standards Efforts – http://checklists.nist.gov/NSCA.html
The Federal Information Security Management Act (FISMA) Implementation Project was established in January 2003 to produce several key security standards and guidelines required by Congressional legislation. These publications include FIPS 199, FIPS 200, and NIST Special Publications 800-53, 800-59, and 800-60. Additional security guidance documents are being developed in support of the project while not called out directly in the FISMA legislation. These publications include NIST Special Publications 800-37, 800-53, and 800-53A. The U.S. National Institute of Standards and Technology (NIST) Computer Security Division continues to produce other security standards and guidelines in support of FISMA available at http://csrc.nist.gov/publications/nistpubs.

NIST Software Assurance Metrics and Tool Evaluation (SAMATE) Project –
http://samate.nist.gov/index.php/Main_Page
The U.S. National Institute of Standards and Technology's (NIST) SAMATE project supports the Department of Homeland Security's Software Assurance Tools and R&D Requirements Identification Program. The objective is the identification, enhancement, and development of software assurance tools that ensure that software processes and products conform to requirements, standards, and procedures. "NIST is leading in (A) testing software evaluation tools, (B) measuring the effectiveness of tools, and (C) identifying gaps in tools and methods." SAMATE is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

National Infrastructure Security Co-ordination Centre (NISCC) –
http://www.niscc.gov.uk/niscc/index-en.html
The role of the UK National Infrastructure Security Co-ordination Centre (NISCC) is to "minimize the risk to the [UK] Critical National Infrastructure (CNI) from electronic attack." As CNI issues transcend geographical borders and problems can strike anywhere in the world, NISCC operates in a global context through four broad work streams: (1) Threat Assessment using a wide range of resources to investigate, assess, and disrupt threats; (2) Outreach by promoting protection and assurance by encouraging information sharing, offering advice, and fostering best practices; (3) Response by warning of new threats, advising on mitigation, managing disclosure of vulnerabilities, and helping the CNI investigate and recover from attack; and (4) Research and Development by devising the most advanced techniques and methods to support efforts across all work streams.

Open Vulnerability and Assessment Language (OVAL™) – http://oval.mitre.org
OVAL is an international, information security community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. OVAL includes a language used to encode system details, and an assortment of content repositories held throughout the community. The language standardizes the three main steps of the assessment process: representing configuration information of systems for testing; analyzing the system for the presence of the specified machine state (vulnerability, configuration, patch state, etc.); and reporting the results of this assessment. The repositories are collections of publicly available and open content that utilize the language. OVAL is sponsored by the U.S. Department of Homeland Security National Cyber Security Division.

Open Web Application Security Project (OWASP) – http://www.owasp.org
Open Web Application Security Project (OWASP) is an open community effort dedicated to enabling organizations to develop, purchase, and maintain applications that can be trusted. OWASP "advocates approaching application security as a people, process, and technology problem because the most effective approaches to application security includes improvements in all of these areas." Similar to other open-source software projects, OWASP produces many types of materials in a collaborative, open way and all of its tools, documents, forums, and chapters are free and open to anyone interested in improving application security.

SANS Top Twenty – http://www.sans.org/top20/
The SANS Top Twenty is a SANS/FBI consensus list of the Twenty Most Critical Internet Security Vulnerabilities, which "enables cyber security professionals to tune their defensive systems to reflect the most important new vulnerabilities that attackers are exploiting to take over computers and steal sensitive or valuable information." The list includes CVE Identifiers to uniquely identify the vulnerabilities it describes.

Security Description and Exchange Format (SecDEF) Initiative –
http://www.vedef.org/index.html
The UK Security Description and Exchange Format (SecDEF) initiative is "a federated effort to encourage the crystallisation of various XML-based description and exchange formats to support information exchange requirements related to security information where there is a need to cross management domains. SecDEF coordination is provided by the UK Government's Central Sponsor for Information Assurance (CSIA) organisation, a unit of the UK Cabinet Office, which is charged with working with partners in the public and private sectors, as well as its international counterparts, to help safeguard the nation's IT and telecommunications services."

SecDEF Forensic Investigation Description and Exchange Formats (FIDEF) –

http://www.vedef.org/index.html
The need for an extension to Incident Object Description and Exchange Format (IODEF) to “accommodate the specific information exchange needs of forensic investigation specialists has been identified, but as yet no lead organization” for the Forensic Investigation Description and Exchange Formats (FIDEF) initiative has been agreed upon.

SecDEF Intrusion Detection Description and Exchange Formats (IDDEF) –
http://www.vedef.org/iddef/index.html
“A requirements analysis exercise identified information exchange requirements for the structured XML-based exchange of data relating to intrusion detection events. A gap analysis exercise identified that the Intrusion Detection Message Exchange Format (IDMEF) being produced by the Internet Engineering Task Force (IETF) Intrusion Detection Working Group (IDWG) meets the needs of this requirement.

SecDEF Incident Object Description and Exchange Formats (IODEF) –
http://www.vedef.org/iodef/index.html
A requirements analysis exercise identified information exchange requirements for structured XML-based exchange of data relating to incident objects. A gap analysis exercise identified that the Incident Object Description and Exchange Format (IODEF)—as initiated by TF-CSIRT, the European Task for on Computer Security Incident Response Teams under the auspices of the IETF Extended Incident Handling Working Group—meets the needs of this requirement.

SecDEF Secure Configuration Description and Exchange Formats (SCDEF) –
http://www.vedef.org/index.html
“The need for a way to specify configure information in a transportable manner, be it for new installations, ad hoc or planned changes, or remediation, has been identified, but as yet no lead organization” for the Secure Configuration Description and Exchange Formats (SCDEF) initiative has been agreed upon.

SecDEF Susceptibility and Flaw Description and Exchange Formats (SFDEF) –
http://www.vedef.org/index.html
“Formerly referred to as the Penetration Testing DEF (PTDEF), the need for an information exchange mechanism to take the results of discovery of installation specific Susceptibilities and Flaws and interface into Trouble Ticket systems has been identified, but as yet no lead organization” for the Susceptibility and Flaw Description and Exchange Formats (SCDEF) initiative has been agreed upon.

SecDEF Secure Interoperability Description and Exchange Formats (SIDEF) –
http://www.vedef.org/sidef/index.html
“A requirements analysis exercise identified a need for a standardised representation of structured XML-based data object labels over and above the basic "DC.rights" representation” of DCMI.org. A Secure Interoperability Description and Exchange Formats (SIDEF) initiative to address this issue is currently under initial development in both the UK Government and NATO.

SecDEF Security Requirements Description and Exchange Formats (SRDEF) –
http://www.vedef.org/index.html
“The need for a way to specify requirements for security, either in support or acquisition activities, or as parameters for connections between management domains has been identified, but as yet no lead organization” for the Security Requirements Description and Exchange Formats (SRDEF) initiative has been agreed upon.

SecDEF Vulnerability & Exploit Definition and Exchange Format (VEDEF) –
http://www.niscc.gov.uk/niscc/vedef-en.html
The free exchange of information on new vulnerability and exploit amongst responsible vendors, Computer Security Incident Response Teams (CSIRTs), and their user communities is crucial to incident prevention. The objective of VEDEF is to create a single standard for transferring vulnerability and exploit information as structured data amongst interested parties; at present the problem is not the lack of data formats, but rather the proliferation of competing and generally incompatible proposals for such formats.

TERENA Computer Security Incident Response Teams Task Force (TF-CSIRT) –
http://www.terena.nl/activities/tf-csirt/

The Trans-European Research and Education Networking Association (TERENA) is an association of organisations are involved with “the provision and use of computer network infrastructure and services for research and education in Europe. TERENA's principal members are the National Research and Education Networking organisations (NRENs) of a large number of countries in and around Europe.” TERENA’s TF-CSIRT Task Force “promotes the collaboration between Computer Security Incident Response Teams (CSIRTs) in Europe. The main goals of the Task Force are to provide a forum for exchanging experiences and knowledge, establish pilot services for the European CSIRTs community, promote common standards and procedures for responding to security incidents, and assist the establishment of new CSIRTs and the training of CSIRTs staff.”

US Air Force Enterprise Agreement with Microsoft –
http://www.gcn.com/print/24_1/31468-1.html
The U.S. Air Force in January 2005 entered into two service-wide contracts with Microsoft Corporation in which all software on Air Force desktop computers are configured to one of three security setting configurations that meet Air Force requirements. Microsoft is responsible for identifying vulnerabilities and implementing fixes across the enterprise.

US-CERT Vulnerability Notes, Technical Alerts, and Security Bulletins –
http://www.us-cert.gov
US-CERT publishes information on a wide variety of vulnerabilities, descriptions of which are available from the US-CERT Web site in a searchable database format, and are published as "US-CERT Vulnerability Notes" at http://www.kb.cert.org/vuls. US-CERT also publishes "Technical Cyber Security Alerts" at http://www.us-cert.gov/cas/techalerts/ that provide timely information about current security issues, vulnerabilities, and exploits, and "Cyber Security Bulletins" at http://www.us-cert.gov/cas/bulletins/that provide weekly summaries of new vulnerabilities along with patch information when available.

Web Application Security Consortium (WASC) – http://www.webappsec.org
The Web Application Security Consortium (WASC) is an international group of experts, industry practitioners, and organizational representatives who produce open source and widely agreed upon best-practice security standards for the World Wide Web. WASC facilitates the exchange of ideas, organizes industry projects, and consistently releases technical information, contributed articles, security guidelines, and other useful documentation.

8. Targeted Capabilities

The following capability target descriptions came primarily from an appendix of the ??? document created by ??? on ??? (Joe?)
Each of these capabilities needs mapped to the various organizations, activities and knowledge of the SwA landscape.

For each topic, an estimate of its difficulty and the timeframe to begin receiving results is included. Short-term results are expected to yield benefits within 2 years; medium-term results are expected to yield benefits in 3-5 years, and long-term results are expected to yield benefits in longer time frames. In a few cases, the short-term areas are those where much is known, and what is needed (at least in part) is the development or refinement of tools that can be applied at the scales of ordinary software development. Generally, a more difficult area will not yield results until later (if they were easy, we could do them and receive their results more rapidly). Regardless of the time frame for expected outcomes, R&D investment is needed in the near-term to obtain these medium and long term outcomes. Of course, these are merely estimates; researchers and developers may identify breakthroughs that transform a difficult problem into a simpler one, and we encourage such insights. Some topics are necessarily dependent on other topics; some of these dependencies are noted, but again, insight may break or change anticipated dependencies.

The R&D topics in this appendix are grouped into the following general categories: development processes, scanning/ detection of security vulnerabilities, countermeasures, development tools, application environment, requirement/design/validation, and education. This grouping is simply an aid to understanding the various topics; the topics could certainly be grouped in other ways. The topics were collected from a series of brainstorming meetings with a variety of experts, so they represent a variety of concepts and directions. This appendix ends with a few general notes about such R&D.

Development Processes

- *Identify and validate minimum subsets of organizational practices & process requirements for secure software development*. Medium difficulty; Mid-term results. Note that this can build on substantial previous work such as the Safety and Security Extensions to Integrated Capability Maturity Models, SSE-CMM, CMMI, iCMM, Trusted CMM (TCMM), Trusted Software Methodology (TSM), and others to identify the minimum needs for various levels of assurance, and validate that these are needed and sufficient for secure development. Note that this builds on the knowledge of individuals (identified separately).

- *Provide incentives for COTS vendors to seriously incorporate security concerns during development & sustainment*. Hard difficulty; Long-term results. Perhaps an office can be established, whose job is to devise such incentives, help establish policies to support them, and work with industry. In some cases this may be possible via standards processes, open publication of research results, and various methods of releasing research prototypes (e.g., making them freely available to lower deployment costs). This will require gaining a greater understanding of the economics of information security technology. [HPL Influencing vendors]

- *Develop more cost-effective methods for high assurance software development*. Hard difficulty; long-term results. There is a great deal of research available in formal methods. However, the costs, time, and advanced expertise required to use them, along with their difficulty in scaling and sometimes poor interoperation with existing infrastructures, have made them impractical on many projects. Build on existing research to find cost-effective methods, most likely building on previous formal methods research. The goal is to have

development tools that, when used, produce software that is "correct by construction." This also requires that such approaches be able to efficiently handle the inevitable changes in software without a significant amount of rework. Involves a great deal of fundamental research. [HPL High assurance development]

- *Improved validation methods*. Medium difficulty; mid-term results. Improve model-based mathematical methods that evaluate and quantify system security properties, and/or incorporate techniques such as finite state automata and process algebra models. [EIP Assurance/Validation].

- *Improved capture of evidence and metrics*. Medium difficulty; mid-term results. Devise new approaches to capture evidence in a more automated and analyzable fashion, including evidence from updates, and to provide evidence based on evolving assurance foundations/technologies. [EIP evidence and metrics, process]

- *Develop interoperable methods for assurance*. Hard difficulty; long-term results. There are many different processes for increasing assurance, but less information on how to combine the information on those processes that rigorously aggregates them into defensible final results. Research is needed into how to better combine assurance methods that analyze and establish system security properties, including strategies for combining approximate and detailed methods (both automatic and interactive). [EIP Foundations/interoperable methods]

- *Compose secure systems from independent secure components.* Hard difficulty; long-term results. This is often referred to as the "composability problem." In practice, components will be from a variety of vendors. At best, some of the components will have undergone separate evaluations, but not as a system as a whole. In addition, systems are increasingly composed dynamically (at run time). Assuring system security properties from the composition of various parts is an extremely difficult problem, yet whether or not we can provide assurance, such composed systems will continue to be fielded. Thus, approaches that at least provide some assurance in these cases is needed. Although the general composibility problem is a hard problem, there should be ways to scope the problem to special (yet useful) cases where it can be solved more easily. Develop a scientific basis for extracting and combining properties and environmental assumptions across systems, develop support from managing hierarchically designed or layered systems, and develop techniques for constructing composite views, analyzing interactions, and performing automatic checking of properties. Identify methods for managing the consequences of interfering properties and mitigating interference. Possible approaches include a Composability tool, i.e. look at configuration in system/enclave, support "what if" questions, and support threat modeling/ vulnerability analysis. Note that there are various levels of composition (module, system, enclave, enterprise). The method of determining/defining such properties needs to be sufficiently simple that it can be applied to today's complex components. [HPL Secure system composition, EIP Security Foundations/develop and assurance security properties, EIP Foundations/Composition and decomposition]

- *Compose secure systems from potentially insecure independent components.* Hard difficulty; long-term results. In practice, some of the components in use will be less secure or insecure, as well as being from a variety of vendors and not evaluated as a whole. Thus, the issue of composing secure systems from secure components needs to be extended to also handle potentially insecure components. This is in addition to previous topic noted above, namely, of the components will have undergone separate evaluations, but not as a system as a whole. This is presumably dependent on solving

the problem of composing secure systems from secure independent components. [HPL Secure system composition, EIP Security Foundations/develop and assurance security properties, EIP Foundations/Composition and decomposition]

- *Simplify identification of validly secure configurations*. Easy difficulty; near-term results. Current practice often involves identifying a single specific configuration for a specific environment (a "hardening" process), but actual environments often vary and users are uncertain how they can modify the configuration while maintaining the security they need. Some existing tools can analyze configurations, but they often can only detect specific configuration errors, instead of being able to report that there are no errors and the configuration is actually secure. Improved methods to automate creation and analysis of configurations is needed.

- *Develop and analyze statistics of common vulnerabilities*. Easy difficulty; near-term results. Collect specific information on the vulnerabilities (unintentional and malicious) discovered in a variety of products, both proprietary and open source software, including the specific lines of code that caused the vulnerability, the lines used to fix them, categories of the vulnerability, and the general reasoning that was used both by the developer who inserted the vulnerability and the approach used to identify and fix the vulnerability. Perform in-depth analysis to determine precisely what the most common vulnerabilities are (both unintentional and malicious) in various environments (e.g., by operating system and programming language), and provide unattributed results publicly. The results must be sufficiently detailed so that various processes and tools can be designed to specifically counter the most common vulnerabilities, so that new/improved languages can be developed to counter the most common vulnerabilities, and so that education can be focused on the specifics of the most common vulnerabilities. There are some rudimentary high-level statistics available (e.g., high-level statistics on buffer overflows or race conditions). However, currently available results are insufficiently detailed to show exactly what the most common programming structures are that produce those vulnerabilities, and that a given process or tool actually counters the vulnerability or at least the percentage of current vulnerabilities that the process/tool will prevent. Indeed, it is often extremely difficult to identify the specific lines of code at fault in a program given today's vulnerability reporting systems, even when the program code is publicly accessible.

- *Develop and analyze improved statistics of common attacks*. Easy difficulty; near-term results. Although many organizations do collect some attack statistics, improved information on current attacks would be valuable. Improved visualization approaches would be helpful. Note that although this is useful in the broader context of information assurance, for the purposes of this appendix the issue is to aid in understanding of attacks to ensure that developed software is resistant to them.

- *Standard test cases/problem set*. Easy difficulty; near-term results. Publicizing a set of specific examples of real or representative cases, including malicious code attacks, would enable researchers to demonstrate whether or not their approaches would be effective at countering them. For example, publishing a set of code with known unintentional and intentional vulnerabilities, and determining how well processes or tools either detected or prevented them. Such test cases can demonstrate the effectiveness of certain processes or tools, and may also be helpful as a demonstration and motivation for improving software assurance.

- *Experiments to demonstrate how malicious code can/cannot be inserted into products*. Easy difficulty; near-term results. Perform a series of experiments to determine how

insertion of additional binary and/or source code during a normal commercial development process could survive to the shipped product, using a variety of insertion points. The purpose would be to learn where to focus thought and tool/analysis to detect the intentionally malicious insertion problem.

- *Metrics and practical processes to measure them for software dependability and defensibility (safety, security, and survivability)*. Hard difficulty; Long-term results. There are a number of metrics now, but they have a large number of known weaknesses. Some metrics simply measure properties of the code (e.g., McCabe complexity) but do not strongly predict the assurance of the resulting software. Some standards (e.g., the Common Criteria) define an evaluation process, but these only measure a product long after the fact, and their results are primarily a measure of the effort expended in the process, not the security of the actual product (e.g., a product evaluated at only EAL2 may actually be far more secure than a product evaluated at EAL 4). Penetration testing (a common process, included at the Common Criteria at EAL2 and higher) is extremely dependent on the cleverness of the particular people doing it, and on the amount of artificial constraints placed on them; while often used, it is not as repeatable as desired. Many existing measures are not predictive, a serious weakness. None give high-confidence answers to questions such as the likelihood of a system being vulnerable to a given type of attacker, and the length of time/effort it will take a given attacker to penetrate the system and/or that the software already contains a malicious flaw inserted by the attacker. [HPL Metrics for security]

Scanning/detection of security vulnerabilities

- *Improve source & object code scanners and automated security testing tools for non-malicious code*. Medium difficulty; mid-term results. Develop improved source and object code scanning tools ("static analysis tools") and automated execution testing tools ("dynamic analysis tools") that search for unintentional security vulnerabilities and encourage widespread use. There are many existing tools in this space (e.g., static tools include RATS, flawfinder, Ounce Labs' Prexis, and many others). However, there is certainly room for improving their false positive and false negative rates. Where available, tools should take advantage of source code, even if they are fundamentally object code scanners. However, there is also a need for improved tools for detecting vulnerabilities in object code where the original source code is not available. Improved decompilers could server as a useful supporting tool. [HPL Security of foreign and mobile code]

- *Develop detectors of malicious code by simulation*. Hard difficulty; mid-term results. Develop systems to detect common intentional security attacks, by executing them in "safe" environments, trying to trigger them, & seeing if their actions have dangerous consequences. This would include build changes, etc. This could be used by COTS vendors, or by the U.S. government (say in a "Software Defense" group) to try to trigger these and report back. [HPL Security of foreign and mobile code]

- *Improve source & object code scanners for malicious code*. Hard difficulty; long-term results. This would include build changes, etc. [HPL Security of foreign and mobile code]

- *Improve binary code coverage tools.* Easy difficulty; near-term results. In many cases source code is unavailable, but much information can be revealed from what portions of the binary code actually execute. Such tools exist, but are currently difficult to use. R&D is especially needed to help such tools resist being misled by embedded interpreters (since they will otherwise consider the interpreted information as simply data), even if the

data being interpreted as instructions are all briefly accessed. This is needed to identify code that is not being used in normal operation, and thus may be unnecessary code with a potential for unintentional vulnerabilities (such as an "Easter egg") or a logic bomb (malicious code that is only enabled under certain conditions).

- *Automated pedigree analysis of a program.* For source code: Easy difficulty; near-term results. For binary files: Medium difficulty; medium-term results. Automate methods to determine the pedigree of portions of software that may have originally come from other sources. Such tools already exist for comparing pairs of programs (typically for use in copyright infringement cases), but the goal here would be scale up such programs for use in determining if there is a match without a priori information. The goal here would to be identify pieces of code that have been reused but in fact have known vulnerabilities. In many cases, code may be reused, and vulnerabilities fixed in one location, but with various copies of the unfixed versions hidden in surprising locations. It should not be hard to extend existing tools with the databases necessary to support such analysis against source code; it may be more difficult to apply such technologies against binary executables.

- *Improved detection of similarities between portions of a program.* Medium difficulty; medium-term results. Develop tools to determine the portions of a program that are substantially "similar," sufficient to identify the portions that were probably developed by the same developer and/or at least the same group/process. If a particular developer/group were identified as a specific threat, such analysis could be used to identify the portions of the program most likely to need more detailed analysis. This may also involve comparing portions of different programs, to identify software portions developed by the same person/organization. Note that this has some similarities with the pedigree analysis, and there may be opportunities for synergy.

- *Improved detection of differences between portions of a program.* Medium difficulty; medium-term results. Develop tools to detect portions of a program that is significantly "different" from the rest, to identify portions developed by a different source and/or process that may need special attention.

- *Tools to identify susceptibility to denial-of-service attacks.* Medium difficulty; medium-term results. Existing approaches to perform real-time deadline analysis, static tools that identify loops, and various real-time load tools may aid in such detection.

- *Automated covert channel detection*. Hard difficulty; long-term results. Research techniques so tools can be developed to automate or dramatically reduce the labor-intensive task of identifying significant covert channels in hardware and software.

- *Improve analysis tool interoperability.* Medium difficulty; mid-term results. There are many tools available that can aid analysis (such as decompilers, debuggers, slicers, etc.), but the individual tools often do not work well together. Identify how analysts use and want to use tools together (including identifying information flows and common processes), and identify how to improve tools to improve interoperability (including defining standard interchange formats to support such flows).

Countermeasures

Some view countermeasures as outside the scope of software assurance. After all, the goal is better assurance, not measures to attempt to "patch over" inadequate assurance.

Nevertheless, we must deal with inadequately assured software (at least in the short term!), and since no approach is perfect, a defense-in-depth approach is warranted.
Note that it may be possible to improve many countermeasures by performing some kind of static analysis. For example, it may be possible to automatically generate a wrapper, or tune a run-time detection countermeasure, by analyzing the program first (in source or binary form).

- *Improve patch management*. Easy difficulty; near-term results. Patch management tools are already available, but for a variety of reasons patches are still unevenly applied or applied too late. These reasons include fear that the patch will cause subtle failures, difficulty in applying patches, and difficulty in confirming that patches have been applied. R&D is needed to investigate the causes for uneven or late application of patches, and to identify ways to improve it. This would include investigating techniques to support rapid testing of patches, safe rollback of bad patches, automating workarounds instead of only the patches (to give administrators alternatives if they perceive the patch as too dangerous), and rolling back and replaying of data processed (possibly across a network) should a patch fail. In short, R&D needs to identify ways of countering problems such as bad patches so that administrators will be more willing to apply patches.

- *Improved mechanisms to simplify upgrading programs without restarting them, using special hardware, or complex software development.* Medium difficulty; mid-term results. Installing patches often involves restarting that component (and on Windows, it may involve restarting the entire operating system, possibly several times). It many cases it should be possible to switch to a new component without restarting it (from the user's point of view), and/or with an extremely rapid "restart" where the components actually run in parallel for a period of time. There are techniques for doing this today, primarily to implement fault-tolerance (e.g., for finance, space, and long-running supercomputer tasks). However, they often require special hardware or complex software development effort. General approaches to rapidly upgrade programs without significant downtime would simplify patch application.

- *Devise mechanisms to detect/counter run-time vulnerability exploits*. Easy difficulty; mid-term results. Devise and promulgate changes to application infrastructure (such as compilers and operating systems) to detect attempted exploitation of common vulnerabilities, and counter them. Crispin Cowan's work on StackGuard, FormatGuard, and RaceGuard, IBM's ProPolice, Microsoft's /GS compiler option, and Red Hat's exec-shield are examples of this approach. Software Fault Isolation (SFI) could also be considered such an approach (though using hardware to implement it). When used to detect common vulnerabilities, such approaches can be useful in countering unknown vulnerabilities of a given known type. In some cases, there may be a reason to keep the details of these defenses classified, primarily if knowledge of the defense is sufficient to eliminate the defense, but classified defenses cannot be applied in many important locations where defenses are needed. Thus, defenses that continue to work even if an attacker knows exactly how they work have many advantages.

- *Improved self-detection or protection from vulnerabilities.* Medium difficulty; mid-term results. Identify/improve methods for programs to detect, protect, and/or correct themselves and/or their data against vulnerabilities. Many programming methods encourage the insertion of self-checking routines (such as "assertions") to detect errors. The effectiveness of such methods in preventing vulnerabilities is unclear, nor is it clear how to add such internal mechanisms to best counter such vulnerabilities. There has also been some work in including proofs of correctness ("proof checking code"). A subset of this is in [EIP Assurance/protect its own integrity]

- *Improve privilege & damage minimization*. Medium difficulty; mid-term results. Devise improved automated methods for limiting the operations permitted to applications ("sandboxes"), limiting damage (e.g., through fault tolerance), or allowing undoing of their damage. This should be without impeding normal use (including where possible real-time), and without requiring complex user configuration or mediation. Where programmatic control is required, devise ways to perform this in ways simple for developers to use correctly, across multiple platforms, even when the application is not highly privileged (otherwise, privilege minimization might require too many privileges). This includes better ways to ensure that the program is truly confined (a serious problem with today's sandboxes), and other ways to increase assurance in the damage minimization mechanism. Improve methods for characterizeing the properties of the resulting system. Address foreign and mobile code, e.g., strongly limiting or preventing execution of files sent via email (even indirectly). Previous research on "wrappers" can support this work. Improved fault tolerant architectures need to be devised to improve system resilience. More sophisticated systems should support correlation of data for detection and analysis, as well as toleration and adaptation. Encourage COTS hardware vendors to improve support for machine/system virtualization, so sandboxes are easier to implement and have a negligeable performance impact. Encourage COTS hardware vendor to include other mechanisms that can aid secure implementations. [HPL Security of foreign and mobile code; EIP Foundations/Access control and process separation; EIP Foundations/Control of information flow]

- *Develop methods to minimize/control the functionality of products.* Medium difficulty; mid-term results. For example, develop automated ways to configure COTS products to simply install only the functionality necessary (and no more), and to remove functionality. Develop methods to remove functionality preferably without requiring support from a COTS vendor (e.g., identify through program tracing statically and/or dynamically to determine what should not be used in a given circumstance, and modify the executable by "patching NOPs" or breakpoints so that the functionality is no longer available). This will require methods to identify what functions are actually required for a given domain and use. This is related to privilege minimization, but here the issue is to limit functionality (since with fewer functions, there is less to exploit). Existing products already support installing only some components; the goal of this research is to go far beyond those capabilities. One of the challenges is that COTS vendors typically only test a few configurations; minimized configurations are unlikely to have been tested rigorously, are rarely tested in the environment it will be used in, and modifying the executable will mean that the resulting product has not been tested at all (and risks significant difficulty if a problem later arises – was the problem in the software, or in the way it was modified?).

- *Develop methods to identify "code that does things not specified.*" Hard difficulty; long-term results. In particular, develop techniques to identify "Easter Eggs." This may be able to build on the code coverage tools noted earlier, but the goal is to be able to identify all such code, and not depend exclusively on dynamic execution (since dynamic execution will often not cover code that is nevertheless necessary and works as specified).

- *Developing a truly trustworthy computer base*. Medium difficulty; mid-term results. Create a complete trustworthy computing base that can be proven and verified in high-assurance settings (e.g., supervisory control of critical systems and processing at multiple security classification levels). Pieces of this goal exist at least in part. For example, there are specialized kernels that already support many of these goals, and it is likely to be an easy difficulty and near-term result to further examine them to gain some assurance against malicious insertion of code. However, gaining stronger assurance of

the absence of malicious code in them, including the larger functionality required for many of today's functions, and including the larger suite necessary (hardware, middleware, etc.) will be significantly harder. [HPL]

- *Improved visualization techniques of security properties in existing systems.* Medium difficulty; mid-term results. Improved methods are needed for visualizing the security-relevant properties of code, execution paths, data, and behavior. This is not simply "generating a UML diagram"; the goal is to extract and visualize those properties specifically relevant to the assurance and/or security properties of the product. [EIP Assurance]

- *Develop security escorts*. Hard difficulty; mid-term results. Develop "security escorts" to follow actions of "not yet fully trusted" software (eg, possibly attached to packets) to report but not restrict full use of applications. Note that this will require tools to support automated analysis of those reports to find the relevant information. Previous research on "wrappers," audit reduction, and network monitoring may support this work. Note that the telecommunications community refers to "software auditing" components that have a somewhat similar concept, though they are usually used to support dependability not information assurance. [HPL Security of foreign and mobile code]

- *Develop "plug-in" approaches to control sharing of sensitive information in networked environments*. Hard difficulty; mid-term results. Mandatory access control implementations developed to support the "Orange Book" did not do well in the marketplace, and currently most sensitive information is protected procedurally. "Plug-in" approaches that can reliably label data, and control their access, are needed. [HPL Controlled sharing of sensitive information in networked environments, EIP control of information flow].

- *Develop Denial of Service (DoS) countermeasures*. Hard difficulty; long-term results. For many systems, preventing denial of service is key requirement. In many cases, there may be ways to make it more difficult for an attacker to overwhelm a system, and/or detect malicious code designed to permit or create a denial of service. Note that some components not normally considered to be security-critical may, through their connection with other components, be able to perform a denial of service attack, and a malicious developer might insert such attacks into components considered "less critical." [HPL Denial of Service]

Development Tools

- *Devise mechanisms for non-repudiable SCM*. Easy difficulty; near-term results. Software configuration management (SCM) tools are designed to prevent non-developers from seeing or changing code, but there is little discussion of security requirements necessary for dealing with malicious developers and administrators. R&D is needed to specifically identify the security requirements (in particular, for non-repudiable, immutable history of software change sources to identify the specific actions/results of malicious developers and malicious administrators). There is also a need to identify specific mechanisms to support these requirements, particularly for distributed SCM systems. May involve creating and vetting a PP.

- *Devise safe & secure language subsets/extensions with automated checking*. Medium difficulty; mid-term results. Includes working with standards bodies to incorporate them; a long tail needed to embed in industry. Devise safe & secure language subsets (including improved/subsetted libraries) which counter common mistakes or to prevent certain

malicious attacks. Devise automated tools to check that the code meets these subsets. This may involve working to develop standards or extensions to standards, so that checks for the same subsets will be implemented by competing tool/compiler vendors. There are a number of existing language subsets; work is needed to make them easier to use, include stronger defenses, and become standardized/more widely used. There may need to be multiple levels of subsets/extensions, e.g., one for high assurance (where backwards compatibility is less important) and another for medium assurance (where backwards compatibility with the "standard" language is more important). [HPL secure system composition]

- *Devise new secure high-level languages and code generators*. Medium difficulty; mid-term results. Where necessary, develop new secure languages, particularly for specialized application areas. Perform automated code generation, since reducing the amount of manual code generation will reduce the likelihood of implementation errors in that code (and if there are errors, they can be fixed just once in the code generation tool). This includes the development of new/improved visual programming languages and ways of analyzing their results. [HPL secure system composition]

- *Automatic generation of artifacts by development tools that can be provided to end-users or evaluators to support assurance.* Hard difficulty; long-term results. Programs can be accompanied with information that end-users or evaluators could use to verify and gain assurance. Related work includes proof-carrying code, model-checking code, and compilers that provide additional information that simplify correlation of source code to binary code, but significant improvements could improve assurance.

- *Assurance of development tools*. Hard difficulty; long-term results. High assurance software development requires assurance that the tools themselves will not cause problems, including the absence of malicious code and extremely high reliability.

Application Environment

- *Improve GUI security*. Medium difficulty; mid-term results. Ramp up GUI security (e.g., X windows, Microsoft Windows) so that GUIs provide more isolation yet are still practical to use. For example, see the relevant SELinux work.

- *Improved methods of developing secure GUI programs so they can be effectively analyzed.* Medium difficulty; mid-term results. It is considered a best practice to separate the GUI from a software "engine," but this typically involves recreating specialized protocols for each application or using standard protocols (such as HTML and HTTP) that require extra effort to use effectively in an application. The difficulty of developing and analyzing such programs often encourages high assurance systems to stick to character-based interfaces, even where users would prefer a different interface. Devise improved ways of incorporating GUIs in high assurance applications.

- *Devise improved infrastructure mechanisms for application security*. Hard difficulty; mid-term results. Often application software has more information available to it about the security requirements on its data, but infrastructure components such as operating systems do not provide a good match for describing and enforcing them. Improved mechanisms are needed, preferably ones that are portable across multiple infrastructures. Note: this requires the cooperation of COTS infrastructure component vendors [HPL Application Security.]

Requirements/Design/Validation

- *Improved mechanisms for specifying security properties.* Hard difficulty; long-term results. This includes domain specific specification languages, specification methods and languages (including ways to simplify formal specification languages), and non-functional (security, performance, etc.) aspect specification. [EIP Foundations/specification].

- *Improved modeling*. Hard difficulty; long-term results. Devise methods and instances for developing rigorous descriptions of application domains and of security dependencies between security relevant components. Support generation of common models for both simulation and verification, and automated abstraction of models. Support system level modeling and analysis. [EIP modeling & simulation, EIP robust system design] [HPL].

- *Improved approaches/patterns for secure architecture/design*. Easy difficulty; near-term-results. Identify improved ways to structure systems to minimize privileges at the beginning of their development, including the identification of useful higher-level "design patterns" that counter various vulnerabilities. Although identifying some approaches and patterns can be identified (and better documented) immediately, note that this will probably be a continuing effort to capture and refine such information.

- *Improve anti-reverse engineering mechanisms*. Medium difficulty; mid-term results. The Software Protection Initiative has developed several techniques to counter reverse engineering, but stronger mechanisms would raise the bar against more sophisticated adversaries. Note that anti-reverse engineering methods obscure the binaries in ways that make it more difficult to assure it; methods to assure anti-reverse-engineered software may be valuable.

- *Security with Privacy.* Hard difficulty; long-term results. Find methods so that security and privacy can occur together, instead of being traded between each other. [HPL]

- *Effective methods of tagging data in a distributed, heterogenous, cross-domain environment with high granularity.* Medium difficulty; mid-term results. In principle, tagging data is easy, and there are some existing standards for such tagging. In practice, such tagging is difficult. Improved methods of applying tagging in complex environments are needed.

- *Software self-attribution.* Medium difficulty; mid-term results. Identify techniques to embed attribution (tracking) techniques into fielded software (a "software lo-jack"), so that a running software program's location can be identified. For attribute of where a piece of software came from, see the Software Configuration Management item.

Education

- *Identify and validate individual knowledge requirements to develop secure software*. Medium difficulty; Near-term results. A serious problem today is that most developers do not know how to develop secure software. Very recently a few books have appeared on the topic, but there is no formal identification of the topics that developers must know, nor validation that those are the important topics. Such material should include security concepts in requirements, design, implementation, and test, including understanding of common types of vulnerabilities and how to avoid them. Such identification and validation is necessary to enable approaches such as individual developer certification programs (which could be imposed as a requirement on government contractors) and/or to develop minimum requirements for CS/SE university accreditation.

- *Promulgate existing knowledge in how to develop secure software, including high assurance software*. Easy difficulty; near-term results. For example, there are current techniques involving applying formal methods to prove security properties, but current syntheses of this work is difficult to find.

- *Develop, validate, and provide incentives for secure software development curricula in CS/SE programs*. Medium difficulty; Mid-term results. There is very little experience in how to educate developers in how to develop secure software, and too much resistance from universities who view secure development topics as "just training" or "not relevant to the curricula." Even departments with security material rarely discuss how to develop or sustain applications for security. Little has been done to develop and validate curricula, compare various ways of inserting this education into university programs, and which ones work.

9. Software Assurance Roadmap

(Opportunities for Enabling and Enhancing the state of the Software Assurance Community)

Call out all the gaps identified in the index above. Identify priority and likely appropriate sponsors and implementation agents.

This area should likely be broken out into a separate document as there are likely to be multiple perspectives on this (DHS, NSA, etc.).

10. Categorization Schemas

Potential dimensions to categorize SwA landscape elements

- Type of Element
  - Organizations :
    - Purpose
    - Context
    - Related Activities
    - Knowledge
      - Created
      - Contribute
      - Managed
      - Leveraged
  - Activities
    - Type:
      - Initiatives
      - Events
    - Sponsor
    - Participants
    - Knowledge created
    - Knowledge leveraged
  - Knowledge
    - Purpose
    - Context
    - Sponsor (creation)
    - Owner
    - Source activity
    - Related Knowledge

- Security Vulnerability Objective
  - Prevention
  - Detection
  - Mitigation
  - Eradication