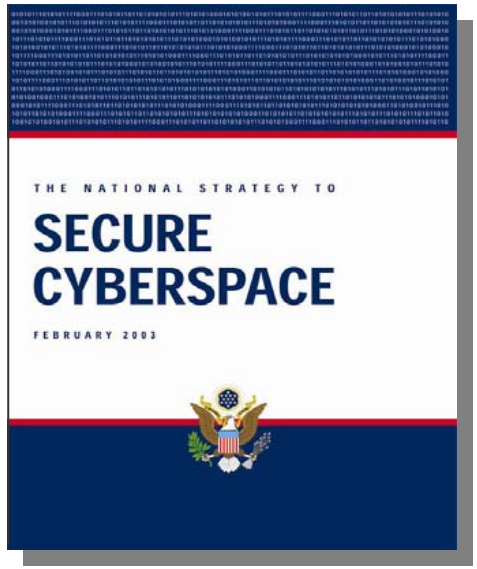
Software Assurance:

A Strategic Initiative of the U.S. Department of Homeland Security to Promote Integrity, Security, and Reliability in Software



Security in the Software Lifecycle

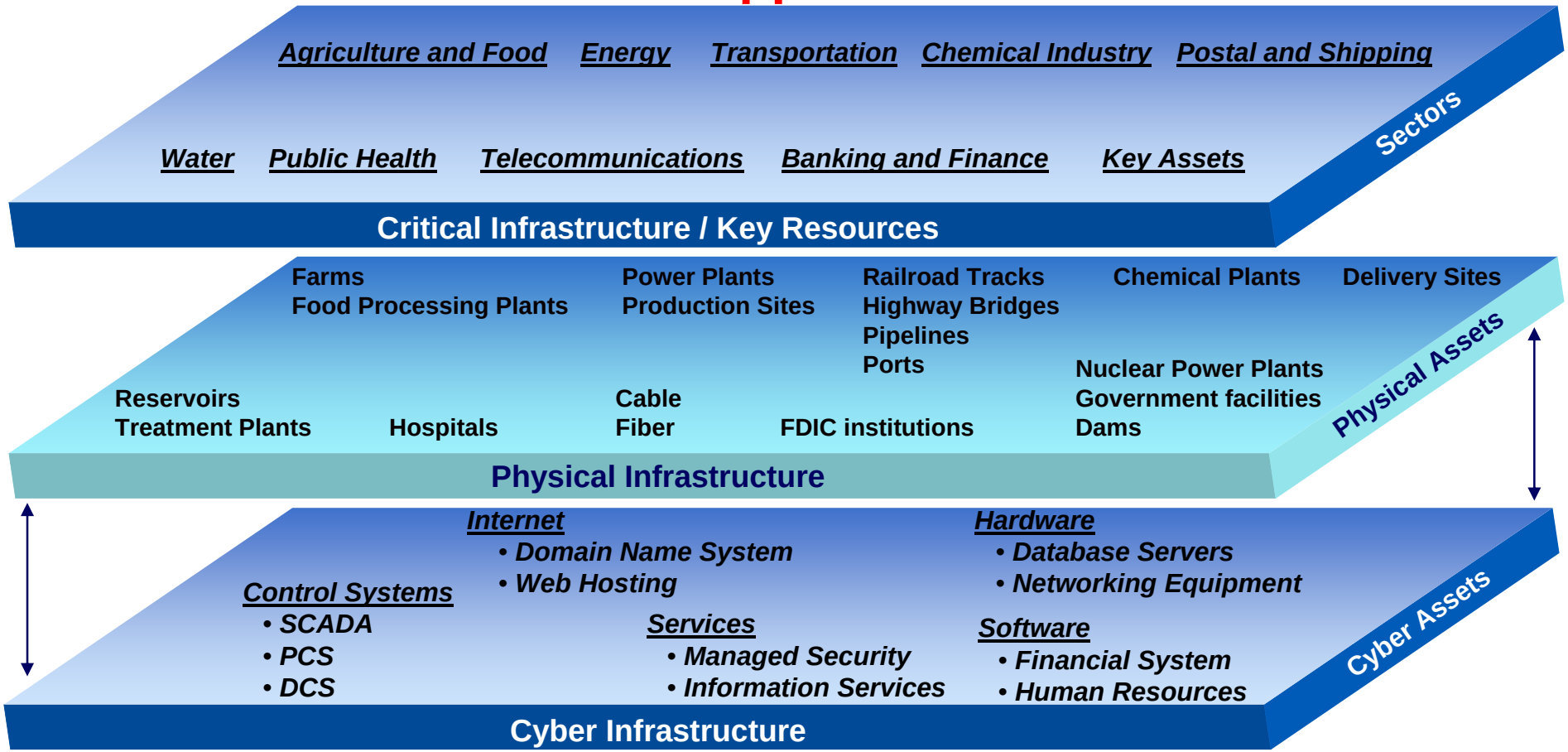
May 9, 2007

Homeland Security

Joe Jarzombek, PMP
Director for Software Assurance
National Cyber Security Division
US Department of Homeland Security

Cyberspace & physical space are increasingly intertwined and software controlled/enabled

Need for secure software applications



Homeland Security

"In an era riddled with asymmetric cyber attacks, claims about system reliability, integrity and safety must also include provisions for built-in security of the enabling software."

Cyber-related Disruptions and the Economy

- Network disruptions lead to loss of:
  - Money and Time
  - Products and Sensitive information
  - Reputation
  - Life (through cascading effects on critical systems and infrastructure)
- Meta-trends:
  - Worms & viruses increasingly sophisticated
  - More variants of older, successful worms
  - New vulnerabilities have black market value; increasing “zero-day” exploits

- **$67.2 Billion a year is lost to cyber crime in the USA** (FBI 2005)
- **$50-200M in average shareholder losses** (CRS 2006)
- ***80%* of hack attacks emanate from outside of user enterprise** (2005 US-CERT-CSO E-crime Survey)
- **9 out of 10 businesses affected by cyber crime last year** (FBI 2005)

Business Losses and Damages

Love Bug: $15B in damages; 3.9M systems infected 2000	**Code Red:** $1.2B in damages; $740M for recovery efforts 2001	**Slammer:** $1B in damages 2002	**Blaster:** $50B in damages 2003	**My Doom:** $38B in damages 2004	**Zotob:** Damages TBD 2005

Homeland Security

Over $40 million in spyware damages – attacks now are "designed to silently steal data for profit or advantage without leaving behind the system damage that would be noticeable to the user.“
(Congressional Testimony, HE &Commerce Telecomm/Internet Subcommittee, Sep 12, 2006)

Why Software Assurance is Critical

- Software is the core constituent of modern products and services – it enables functionality and business operations
- Dramatic increase in mission risk due to increasing:
  - Software dependence and system interdependence (weakest link syndrome)
  - Software Size & Complexity (obscures intent and precludes exhaustive test)
  - Outsourcing and use of un-vetted software supply chain (COTS & custom)
  - Attack sophistication (easing exploitation)
  - Reuse (unintended consequences increasing number of vulnerable targets)
  - Number of vulnerabilities & incidents with threats targeting software
  - Risk of Asymmetric Attack and Threats
- Increasing awareness and concern

Software and the processes for acquiring and developing software represent a material weakness

Homeland Security

Software is exposed to threats *all* the time, *even while it's under development.*

How is software threatened or put at risk of exploitation *in development?*

By developers who, through ignorance, carelessness, or intention, sabotage or subvert the software by…

- Including weaknesses and vulnerabilities, due to…
  - failure to consider the threat environment and its implications (e.g., inadequate or inaccurate threat models, misuse/abuse cases, etc.)
  - poor design choices
  - use of non-secure technologies (e.g., unsecured HTTP and SOAP)
  - unsafe programming practices, languages, libraries, tools
  - coding errors
  - inadequate non-functional security assessments and tests, or intentionally "fudged" results
- Leaving in backdoors, trapdoors, debugging hooks
- Embedding malware (time and logic bombs, Trojan horses, etc.)
- Including undocumented features/functions ("rotten Easter eggs")

Non-secure configuration management practices make the above easier to accomplish.

Where weaknesses and vulnerabilities originate *during development*

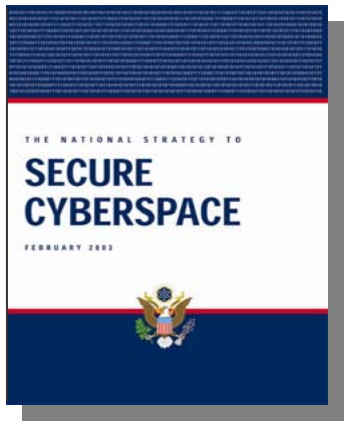
- Inadequate or spurious requirements
- Inadequate architecture, assembly option, detailed design
- Use of vulnerable processing models, software technologies
- Insecure use of development tools, languages, libraries
- Use of insecure development tools, languages, libraries
- Poor coding practices
- Coding errors
- Use of vulnerable, unpatched components
- Incorrect or mismatched security assumptions
- Inadequate reviews, testing, assessments
- Sabotaged test results
- Residual backdoors
- Sensitive info about software problems in user-viewable code, error messages
- Inadequate configuration documentation
- Insecure installation procedures, scripts, tools

Homeland Security

DHS Software Assurance Program Overview

- Program based upon the National Strategy to Secure Cyberspace - Action/Recommendation 2-14:

  "DHS will facilitate a national public-private effort to promulgate best practices and methodologies that promote integrity, security, and reliability in software code development, including processes and procedures that diminish the possibilities of erroneous code, malicious code, or trap doors that could be introduced during development."

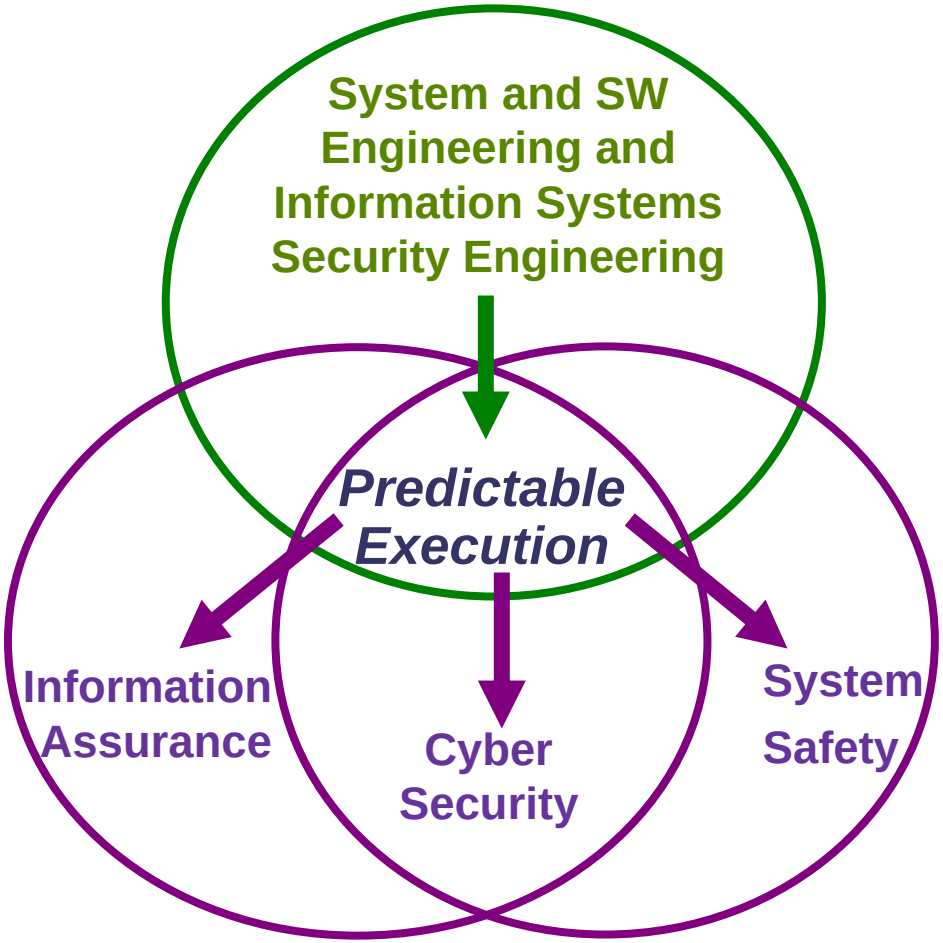


- DHS Program goals promote the security of software across the development, acquisition and implementation life cycle
- Software Assurance (SwA) program is scoped to address:
  - **Trustworthiness** - No exploitable vulnerabilities exist, either maliciously or unintentionally inserted
  - **Predictable Execution** - Justifiable confidence that software, when executed, functions as intended
  - **Conformance** - Planned and systematic set of multi-disciplinary activities that ensure software processes and products conform to requirements, standards/ procedures

Homeland Security

CNSS Instruction No. 4009, "National Information Assurance Glossary," Revised 2006, defines Software Assurance as: "the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle, and that the software functions in the intended manner".

SW Assurance related to Engineering Disciplines



For a safety/security analysis to be valid …

The execution of the system must be ***predictable.***

This requires …

- Correct implementation of requirements, expectations and regulations. — ***Traditional concern***
- Exclusion of unwanted function even in the face of attempted exploitation. — ***Growing concern***

Predictable Execution = requisite enabling characteristic

Homeland Security

Security in the Software Life Cycle:
Informed development and supply chain management

- Existing processes, methods, and techniques that can help them to specify, design, implement, configure, update, and sustain software that is able to accomplish the following:
  - Resist or withstand many anticipated attacks.
  - Recover rapidly and mitigate damage from attacks that cannot be resisted or withstood.
- The key to secure software is:
  - A security-enhanced software development life cycle process -- includes practices that not only help developers root out and remove exploitable defects (e.g., vulnerabilities) in the short term, but also, over time, increase the likelihood that such defects will not be introduced in the first place.
  - A security-enhanced acquisition / out-sourcing life cycle process -- includes practices that address risks associated with the software supply chain
- ***Functional Correctness must be exhibited even when software is subjected to hostile conditions; therefore, claims about system reliability, integrity and safety must include provisions for built-in security of enabling software***

Enhance "Assurance" Considerations: Leveraging CMM-based Process Improvement

Determine how "assurance" is factored into suppliers' process capabilities

- **An infrastructure for safety & security is established and maintained.**
  1. Ensures Safety and Security Competency within the Workforce;
  2. Establishes a Qualified Work Environment (including the use of qualified tools);
  3. Ensures Integrity of Safety and Security Information;
  4. Monitors Operations and Report Incidents (relative to environment in which software will be used);
  5. Ensures Business Continuity.
- **Safety & security risks are identified and managed.**
  6. Identifies Safety and Security Risks;
  7. Analyzes and Prioritizes Risks relative to Safety and Security;
  8. Determines, Implements, and Monitors the associated Risk Mitigation Plan.
- **Safety & security requirements are satisfied.**
  9. Determines Regulatory Requirements, Laws, and Standards;
  10. Develops and Deploys Safe and Secure Products and Services;
  11. Objectively Evaluates Products (using safety and security criteria);
  12. Establish Safety and Security Assurance Arguments (with supporting evidence).
- **Activities/products are managed to achieve safety & security requirements.**
  13. Establishes Independent Safety and Security Reporting;
  14. Establishes a Safety and Security Plan;
  15. Selects and Manages Suppliers, Products, and Services using safety and security criteria;
  16. Monitors and Controls Activities and Products relative to safety and security requirements.

Many suppliers use CMMs to guide process improvement & assess capabilities; yet many CMMs do not explicitly address safety and security.

Source for "Assurance" enhanced processes: *U.S. DoD and FAA joint project on Safety and Security Extensions for Integrated Capability Maturity Models, September 2004,* at http://www.faa.gov/about/office_org/headquarters_offices/aio/documents/media/SafetyandSecurityExt-FINAL-web.pdf

DHS Software Assurance Program Structure *

- As part of the DHS risk mitigation effort, the SwA Program seeks to reduce software vulnerabilities, minimize exploitation, and address ways to improve the routine development of trustworthy software products and tools to analyze systems for hidden vulnerabilities.
- The SwA framework encourages the production, evaluation and acquisition of better quality and more secure software; leverages resources to target the following four areas:
  - **People** – education and training for developers and users
  - **Processes** – sound practices, standards, and practical guidelines for the development of secure software
  - **Technology** – diagnostic tools, cyber security R&D and measurement
  - **Acquisition** – due-diligence questionnaires, contract templates and guidelines for acquisition management and outsourcing

Homeland Security

* July 28, 2006 statement of George Foresman, DHS UnderSecretary for Preparedness, before the U.S. Senate Committee on Homeland Security and Governmental Affairs, Subcommittee on Federal Financial Management, Government Information, and International Security

DHS Software Assurance (SwA) Program …

… encourages the production, evaluation and acquisition of better quality and more secure software through targeting

People	Processes	Technology	Acquisition
Developers and users education & training	Sound practices, standards, and practical guidelines for the development of secure software	Diagnostic tools, cyber security R&D, and measurement	Software Security Improvements through due-diligence questions, specs and guidelines for acquisitions/ outsourcing

Products

Build Security In - https://buildsecurityin.us-cert.gov	Practical Measurement Guidance for SwA and Information Security
SwA Common Body of Knowledge (CBK)	SwA Metrics & Tool Evaluation with Common Weakness Enumeration dictionary for tools
SwA Developers' Guide on Security-Enhancing SDLC	SwA in Acquisition: Mitigating Risks to Enterprise
SwA-related standards -- IEEE, ISO/IEC, OMG, NIST	

The May 15-17 DHS Software Assurance Working Group will include a day on the "assurance" case/argument (https://buildsecurityin.us-cert.gov)

Homeland Security

DHS SwA – People Focus

- Provide Guide to Software Assurance (SwA) Common Body of Knowledge (CBK)
  - Serves as a framework to identify workforce needs for competencies and leverage standards and “best practices” to guide software-related curriculum development
  - Addresses three domains: “acquisition & supply,” “development,” and “post-release assurance” (sustainment)
  - Draft v1.1 distributed on 25 Sep 2006 for review and comment
  - Draft content now being used by early adopters in graduate level courses in secure coding/programming and NDU Information Resource Management College (IRMC) CISO Certificate Program course on SwA
- Plans:
  - Next SwA CBK draft with “guiding principles” to be released May 2007
  - Develop pilot training/education curriculum consistent with CBK in conjunction with early adopters for distribution by September 2007
  - Provide input to IT Security Essential Body of Knowledge (EBK)

Homeland Security

DHS SwA – Process Focus

- Provide Software Assurance (SwA) Developers' Guidance
  - Provided practical guidance via "Build Security In" on US-CERT web site with regular updates based on feedback from stakeholders
  - Provided draft developers guide, "Securing the Software Lifecycle: Making Application Development Processes – and Software Produced by Them – More Secure" for public review and comment (draft v1.1 released July 2006)
- Plans:
  - Continue to provide periodic updates to https://buildsecurityin.us-cert.gov
  - Released developers' guide, draft v1.2 in March 2007 reflecting review comments
  - In collaboration with federal agencies, standards bodies, industry and academia:
    - provide draft guidance for specifying 'assurance arguments' from which to base claims about the safety, security and dependability of software – draft v0.5 to be released September 2007 for review and comment
    - Provide recommended changes to national and international standards on software testing and software assurance – via ongoing work and liaison with IEEE CS S2ESC, ISO/IEC JTC1 SC7/SC27/SC22, OMG, CNSS, and NIST

Homeland Security

DHS SwA – Technology Focus

- Provide SwA Technology Lifecycle Support Guidance
  - Sponsor work with NIST to inventory and measure effectiveness of SwA tools
  - Sponsor public-private work to provide a common dictionary of software weaknesses (CWE) - primarily those that can be discovered by tools
  - Provide common attack pattern enumeration (CAPEC) from which developers and users can understand the resilience of software relative to use, abuse and misuse.
  - Provide SwA Measures to support decision making throughout the software lifecycle
  - Provided draft SwA Landscape document, including organizing mechanisms for SwA ecosystem infrastructure, from which to clarify and specify interfaces and interoperability among various SwA initiatives
- Plans
  - NIST Special Pub 500-268, “Source Code Security Analysis Tool Functional Spec”
  - NIST Special Pub 500-269, “SwA Tools: Web Application Scanner Functional Spec”
  - NIST Special Pub 500-270, “Source Code Security Analysis Tool Test Plan”
  - In collaboration with NIST, provide a Test Case Generator from which to evaluate SwA tool compatibility and effectiveness – demonstrated in March 2007
  - Provided in March 2007 for review draft v1.0 SwA Measurement Guide, “Practical Guidance for Software Assurance and Information Security Measurement”

Homeland Security

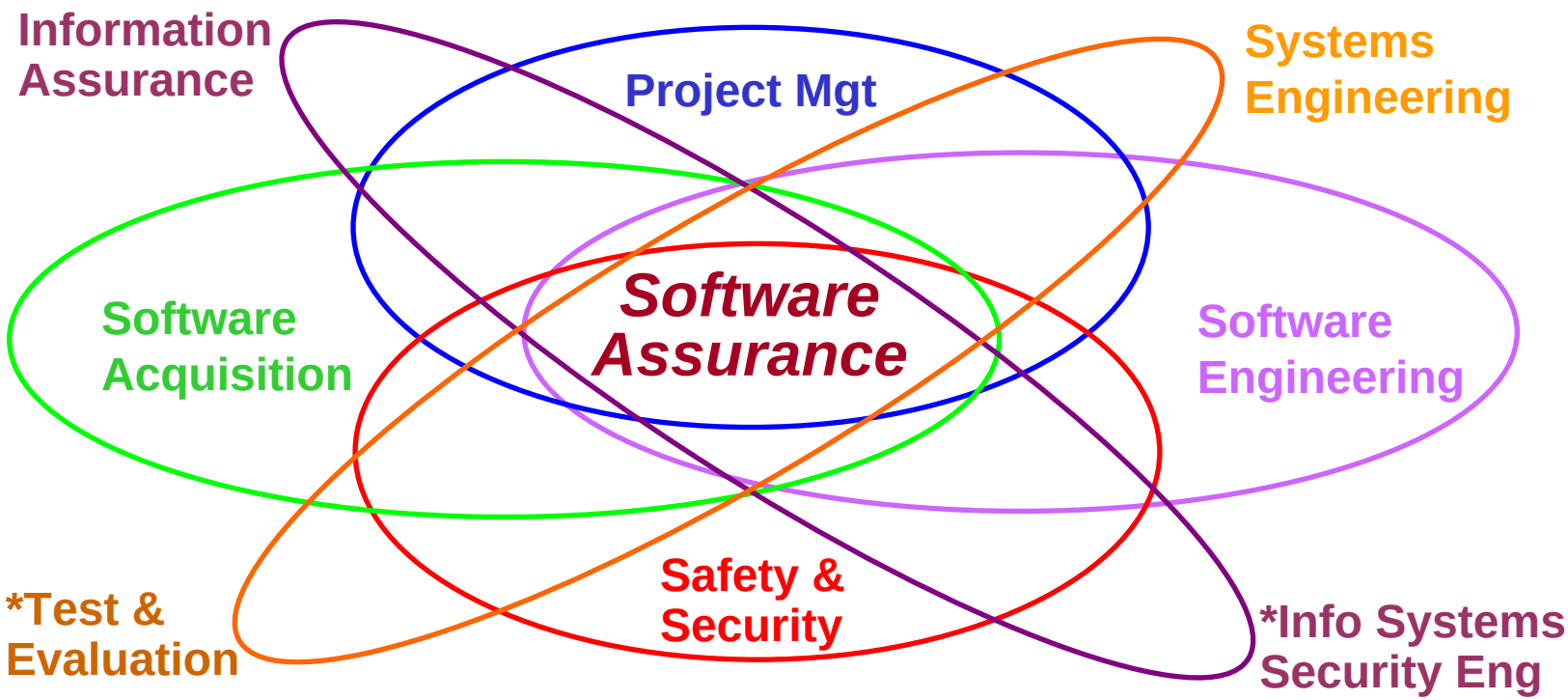
A SwA Ecosystem Demonstration will be held the evening of March 7th during the OMG SwA Workshop Reception that is open to those attending either the OMG SwA Workshop (March 5-7) or the SwA Forum (March 8-9). It will include a demo of the Test Case Generator being co-sponsored by DHS and NIST.

DHS SwA – Acquisition Focus

- Provide Software Assurance (SwA) Acquisition Guidance
  - Provided draft Acquisition Management guidance focused on enhancing supply chain management through improved risk mitigation and contracting for secure software
  - Collaborated on “due diligence” questionnaires for RFI/RFP and source selection decision making
  - Drafted templates and sample statements of work / procurement language for acquisition and evaluation based on successful models
  - Collaborated with agencies implementing changes responsive to the Federal Acquisition Regulation (FAR) IT security provisions of FISMA when buying goods and services and new core competency of “Software Acquisition Management” identified by Federal CIO Council’s IT Workforce Committee
- Plans:
  - Publicly release acquisition guide, draft v1.0, “Software Assurance (SwA) in Acquisition: Mitigating Risks to the Enterprise” in March 2007

Homeland Security

Disciplines Contributing to Software Assurance*



In Education and Training, Software Assurance could be addressed as:

- A “knowledge area” extension within each of the contributing disciplines;
- A stand-alone CBK drawing upon contributing disciplines;
- A set of functional roles, drawing upon a common body of knowledge; allowing more in-depth coverage dependent upon the specific roles.

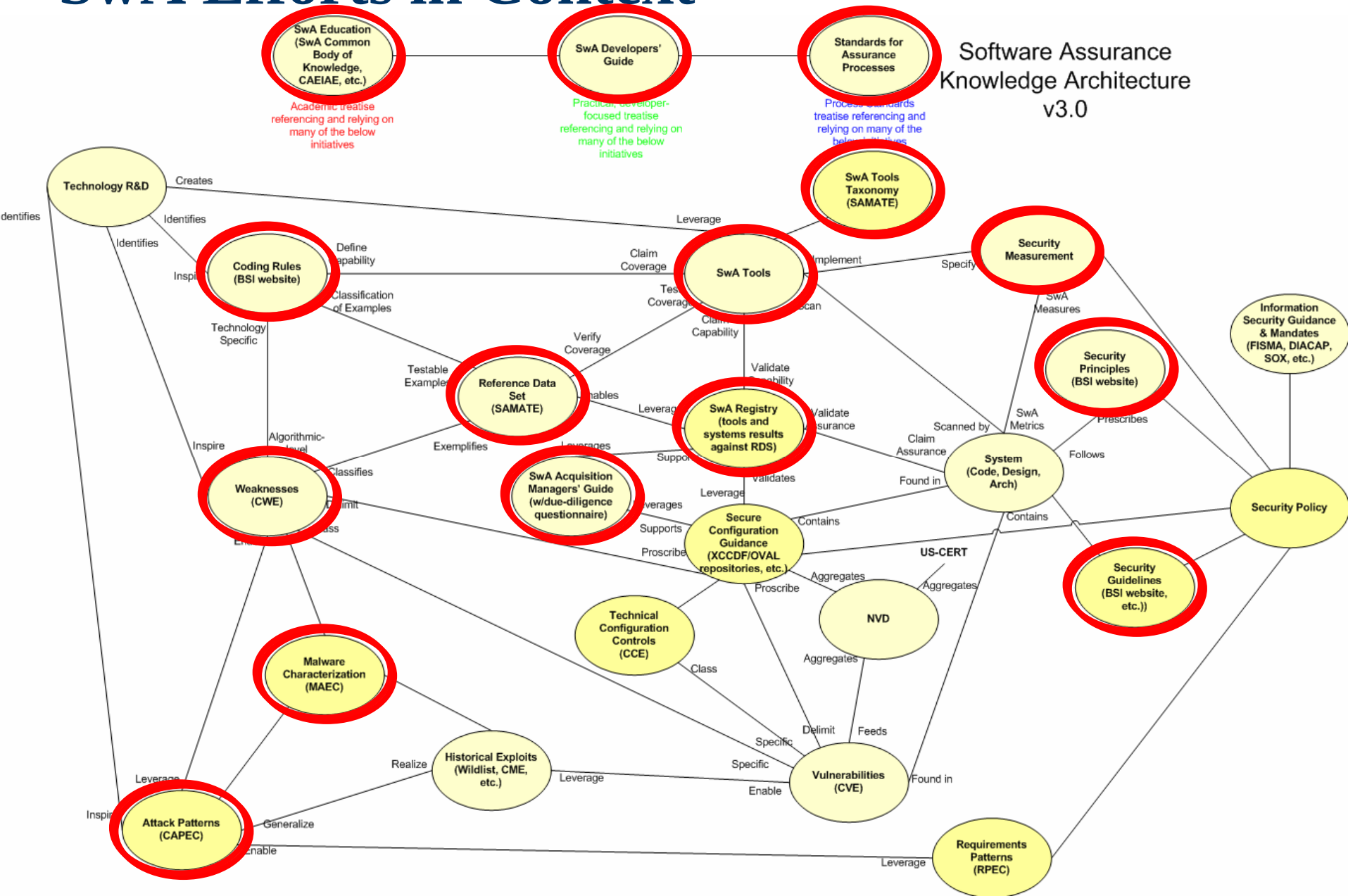
Intent is to provide framework for curriculum development and evolution of contributing BOKs

Homeland Security

* All require Measurement; see ‘Notes Page’ view for contributing BOK URLs and relevant links

The intent is not to create a new profession of Software Assurance; rather, to provide a common body of knowledge: (1) from which to provide input for developing curriculum in related fields of study and (2) for evolving the contributing disciplines to better address the needs of software security, safety, dependability, reliability and integrity.

SwA Efforts in Context



DHS Software Assurance Outreach Services

- Co-sponsor bi-monthly SwA WG sessions and semi-annual Software Assurance Forum for government, academia, and industry to facilitate the ongoing collaboration -- next Oct 2007
- Sponsor SwA issues of CROSSTALK (Oct 05, Sep 06, Mar 07); provide SwA articles in other journals to “spread the word” to relevant stakeholders
  - March 2007 issue on “Software Security”
  - Future issues on Software Acquisition, SOA, etc
- Provide free SwA resources via “BuildSecurityIn” portal to promote relevant methodologies
- Provide DHS Speakers Bureau speakers
- Support efforts of consortiums and professional societies in promoting SwA

Homeland Security

INPUT TargetVIEW

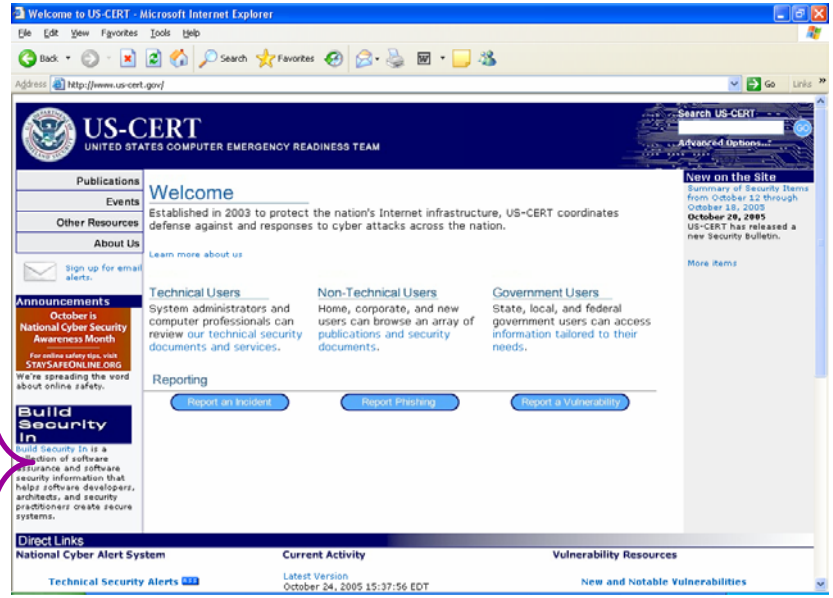
What if…

- **Government, in collaboration with industry / academia, raised expectations for product assurance with requisite levels of integrity and security:**
  - Helped advance more comprehensive software assurance diagnostic capabilities to mitigate risks stemming from exploitable vulnerabilities and weaknesses;
  - Promoted use of methodologies and tools that enabled security to be part of normal business.
- **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' products and services to the acquisition project and to the operations enabled by the software.
  - Information about evaluated 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 & make claims;
  - Qualified tools used in software lifecycle enabled developers/testers to mitigate security risks;
  - Standards and qualified tools would be used to certify software by independent third parties;
  - IT/software workforce had requisite knowledge/skills for developing secure, quality products.

Homeland Security

Software Assurance Working Group Sessions every two months -- Next SwA Forum 2-3 Oct 2007 at Hilton, McLean, VA

www.us-cert.gov →



http://buildsecurityin.us-cert.gov

Joe Jarzombek, PMP
Director for Software Assurance
National Cyber Security Division
Department of Homeland Security
Joe.Jarzombek@dhs.gov
(703) 235-5126

Homeland Security

Homeland Security

Questions?

- - - - - - - -

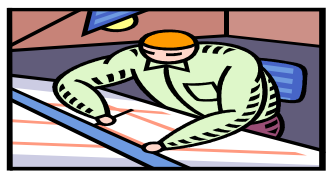
Back-up Slides

Reaching Relevant Stakeholders

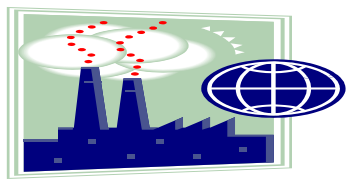
Leverage Evolving Efforts in Universities, Standards Organizations & Industry

Education	Professional Development	Training and Practices
• Curriculum • Accreditation Criteria	• Continuing Education • Certification	• Standards of Practice • Training programs
CNSS IA Courseware Eval	***Certified SW Development Professional (CSDP), IEEE***	***IEEE CS SW & Systems Engineering Standards Committee (S2ESC)***
IEEE/ACM SW Eng 2004 curriculum	***IEEE CSDP Prep Course***	***ISO/IEC JTC1/SC7/ SC27/ SC22 and other committees***
AACSB & ABET	***IEEE CS SWE Book Series***	
AIS IS & MSIS curriculum		

University acceptance



Individual acceptance



Industry acceptance

Homeland Security

Adopted from "Integrating Software Engineering Standards" by IEEE Computer Society Liaison to ISO/IEC JTC 1/SC 7, James.W.Moore@ieee.org, 23 February 2005

Bi-Monthly Software Assurance (SwA) Working Groups:

Next WG sessions held May 15-17, 2007 – Next SwA Forum 2-3 Oct 2007

	Tuesday, May 15th	Wed, May 16th	Thursday, May 17th
Morning 9:00am - 11:30am	**Session 1:** *Technology, Tools & Product Evaluation Working Group*	**Plenary Session**	**Session 6:** *Processes & Practices Working Group on "Argument/Case"*
	Session 2: *Business Case Working Group*		**Session 7:** *Acquisition and Measurement WGs*
Afternoon 1pm - 5pm	**Session 1:** *Technology, Tools & Product Evaluation Working Group*	**Session 4:** *Malware Working Group*	**Session 6:** *Processes & Practices Working Group on "Argument/Case"*
	Session 3: *Workforce Education & Training Working Group*	**Session 5:** *Acquisition Working Group*	**Session 7:** *Measurement Working Group*

Presentations from previous SwA WGs and Forums are on US-CERT Portal (https://us-cert.esportals.net/) under the appropriate Working Group in the Library folder. Access to WG folder is restricted to those who have participated in the WG. Contact DHS NCSD if you do not yet have access to the appropriate folders.

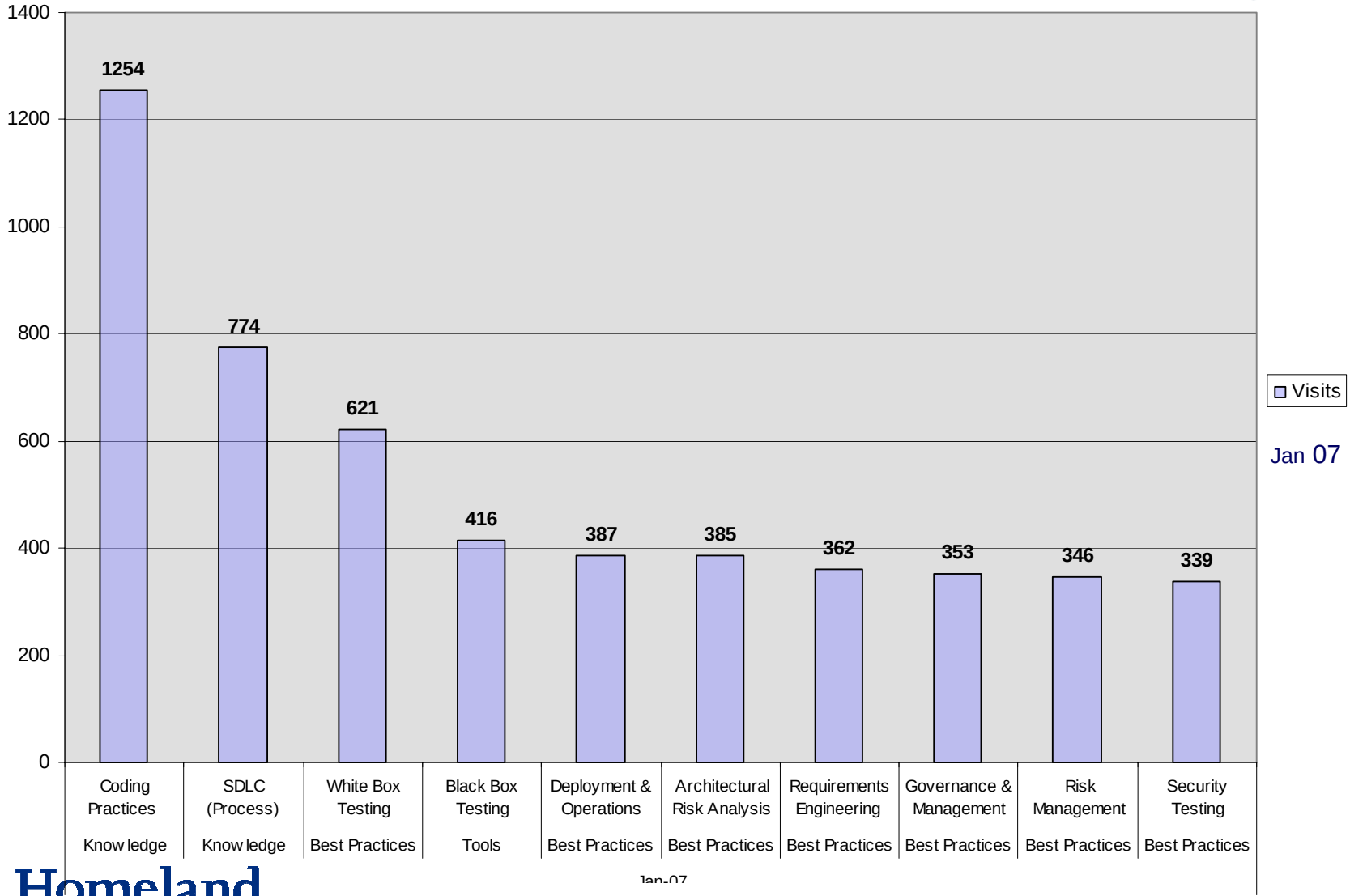
Build Security In (BSI) on US-CERT Summary - Trend

- **Increased Use:**
  - In Jan 07 BSI had 14,848 Unique Visitors, an increase of 3,859
  - Page Views count during the month of December increased by 48,268 hits.

- **Top Documents for Jan 07**
  - Most viewed content area for January was again ‘Coding Practices’ (‘Knowledge’ Category) with 1,254 views -- shows an increase of 349 hits compared to the December data.
  - For ‘Best Practices’ Category – White Box Testing, Deployment & Operations and Architectural Risk Analysis ranked as the most viewed content areas.
  - For ‘Tools’ Category - Black-Box Testing was again the most viewed content area.

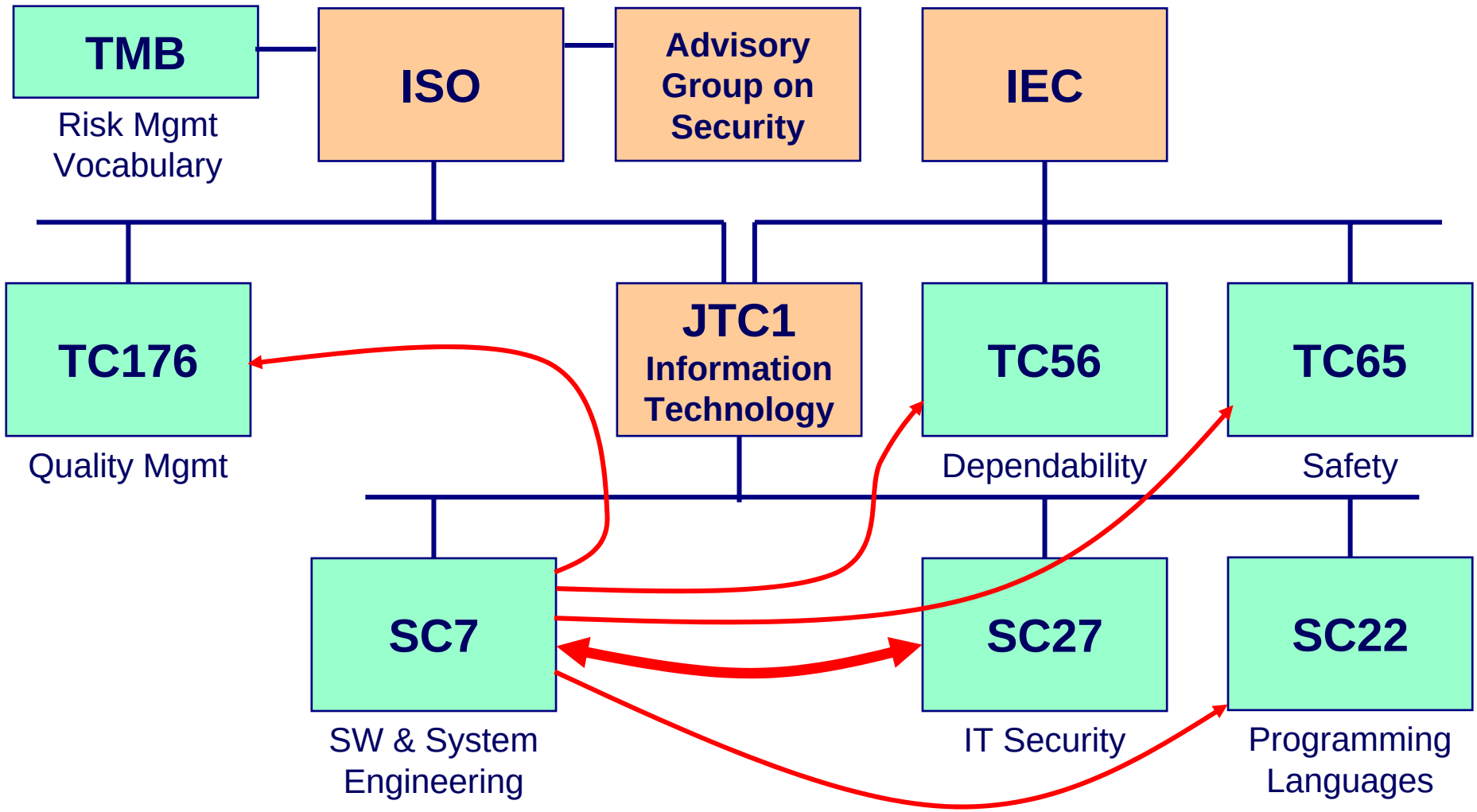
Homeland Security

"Build Security In" SwA on US-CERT Top Documents – What are BSI Visitors finding?



Homeland Security

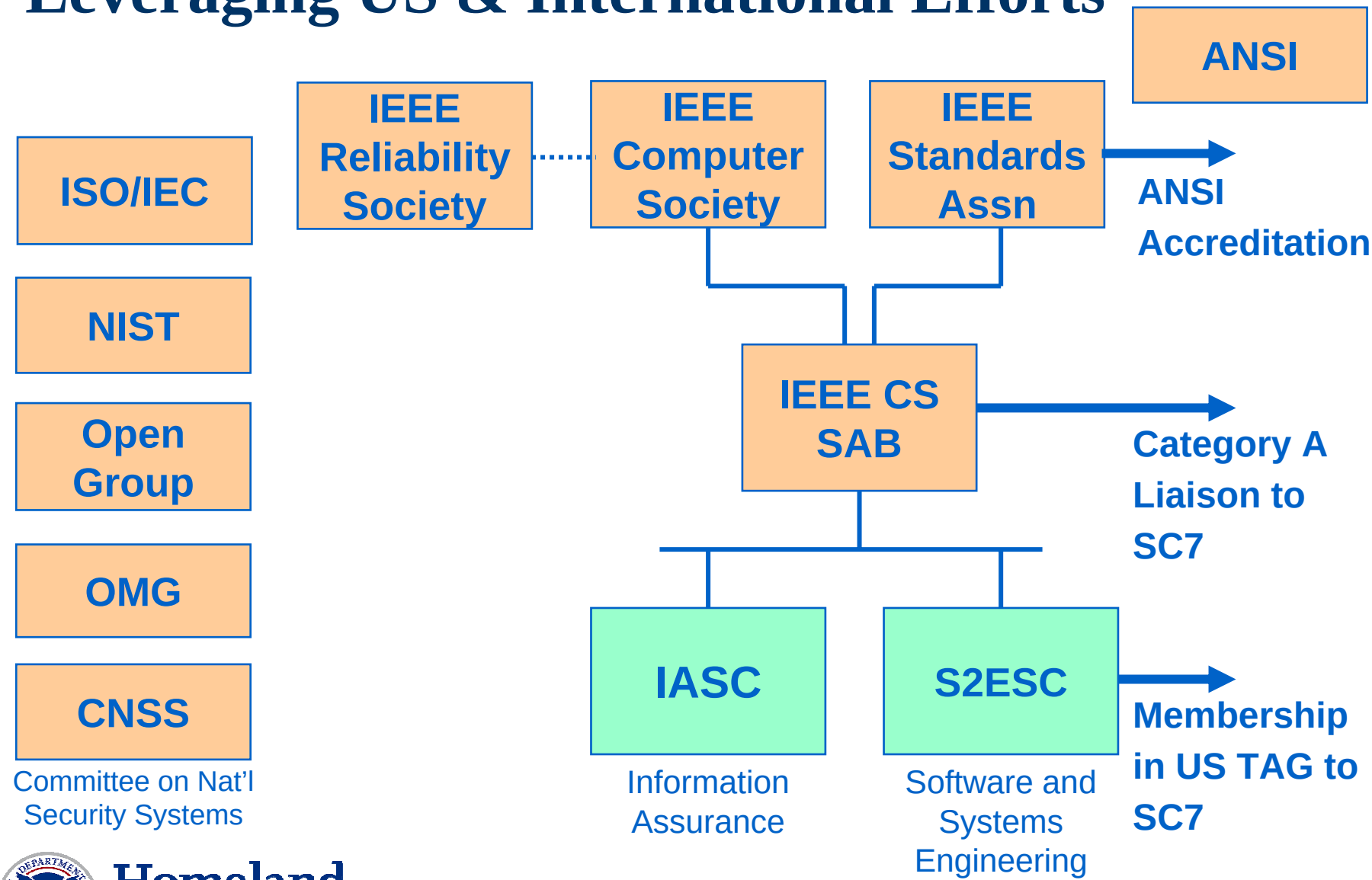
SwA Concerns of Standards Organizations



Homeland Security

* DHS NCSD has membership on SC7, SC27 & IEEE S2ESC leveraging Liaisons in place or requested with other committees

Leveraging US & International Efforts



Homeland Security

Some Current Efforts of NCSD SwA Working with Standards Organizations

- IEEE S2ESC
  - Develop criteria for assurance case / argument
  - Use as an “integrator” of standards for packaging / transition to industry.
- ISO SC7
  - Incorporate “raise the floor” assurance practices into life cycle standards.
  - Incorporate “raise the ceiling” practices into separate standards strongly related to the life cycle standards.
  - Use Safety & Security practices as a benchmark for measuring success.
- ISO SC22
  - Develop coding guidelines for common programming languages.
  - WG established to identify secure constructs in languages
- ISO SC27
  - Expand context to include assurance concerns.

Homeland Security

Key Standards for Software & System Processes

- ISO/IEC 15288, System Life Cycle Processes
  - 25 processes spanning the life cycle of a system.
  - The standard is primarily descriptive.
- ISO/IEC 12207:1995, Software Life Cycle Processes
  - 17 processes spanning the life cycle of a software product or service.
  - The standard is somewhat prescriptive in defining a minimum level of responsible practice.
  - Describes processes meeting the needs of organizational process definition.
- ISO/IEC 12207:Amd 1
  - Describes processes to meet the needs of process assessment and improvement.
- ISO/IEC 15026, Integrity Levels ➔ Assurance
  - Describes additional techniques needed for high-integrity systems.
  - Currently, not process-oriented, but is being repositioned.
- ISO/IEC 16085, Risk Management Process
- ISO/IEC 15939, Measurement Process
- Other standards treating specific processes in greater detail

Homeland Security

Scope of ISO/IEC JTC1 SC7 "System and Software Assurance"

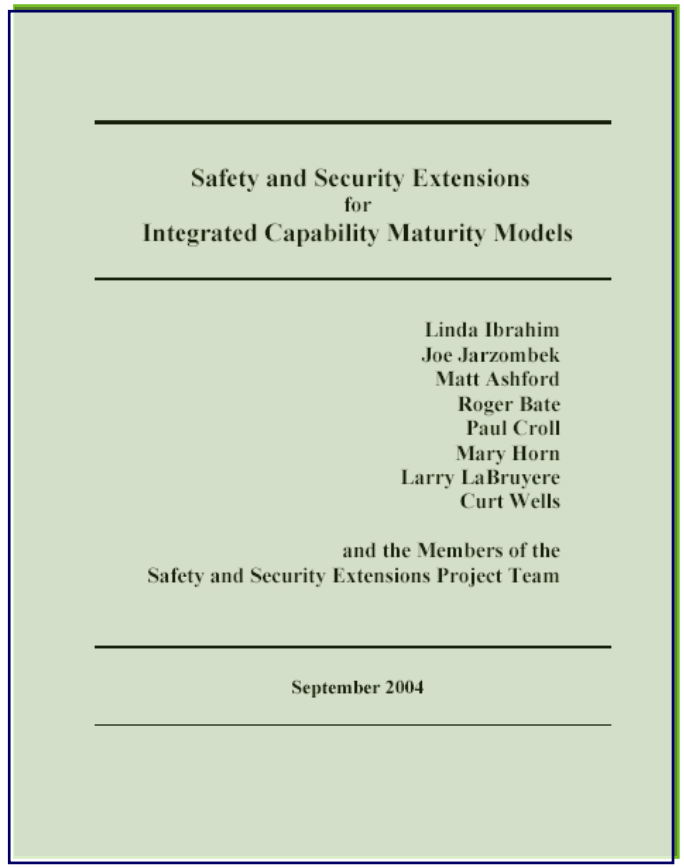
"System and software assurance focuses on the management of risk and assurance of safety, security, and dependability within the context of system and software life cycles."

Terms of Reference changed: ISO/IEC JTC1/SC7 WG9, previously "System and Software Integrity"

Adopted from Paul Croll's SSTC May 2005 presentation, "Best Practices for Delivering Safe, Secure, and Dependable Mission Capabilities"

“Safety & Security Extensions for Integrated Capability Maturity Models” – Input to SC7

1. Ensure Safety and Security Competency
2. Establish Qualified Work Environment
3. Ensure Integrity of Safety and Security Information
4. Monitor Operations and Report Incidents
5. Ensure Business Continuity
6. Identify Safety and Security Risks
7. Analyze and Prioritize Risks
8. Determine, Implement, and Monitor Risk Mitigation Plan
9. Determine Regulatory Requirements, Laws, and Standards
10. Develop and Deploy Safe and Secure Products and Services
11. Objectively Evaluate Products
12. Establish Safety and Security Assurance Arguments
13. Establish Independent Safety and Security Reporting
14. Establish a Safety and Security Plan
15. Select and Manage Suppliers, Products, and Services
16. Monitor and Control Activities and Products



Safety and Security Extensions
for
Integrated Capability Maturity Models

Linda Ibrahim
Joe Jarzombek
Matt Ashford
Roger Bate
Paul Croll
Mary Horn
Larry LaBruyere
Curt Wells

and the Members of the
Safety and Security Extensions Project Team

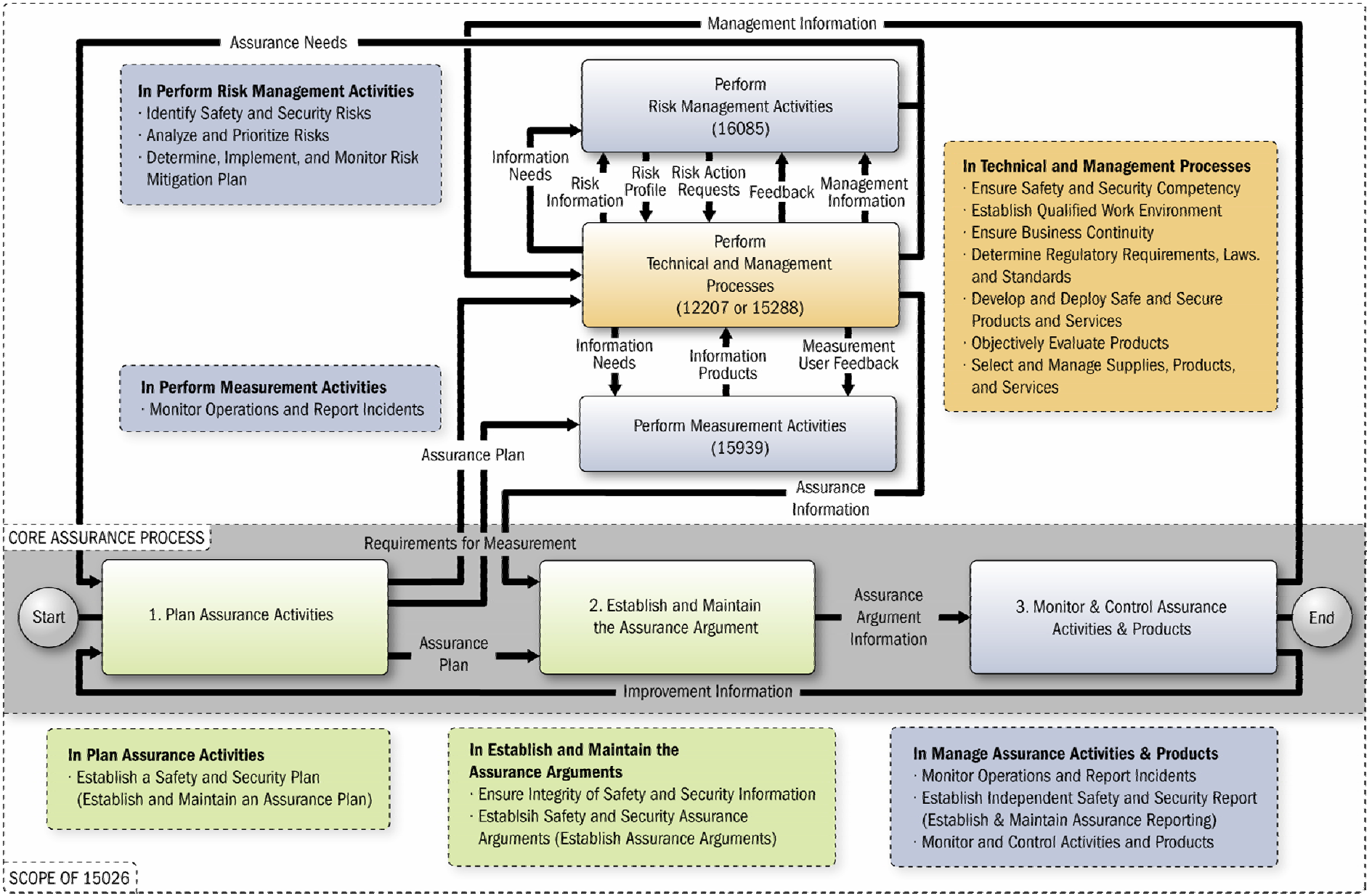
September 2004

www.faa.gov/ipg

Source: United States Department of Defense and Federal Aviation Administration joint project on, Safety and Security Extensions for Integrated Capability Maturity Models, September 2004

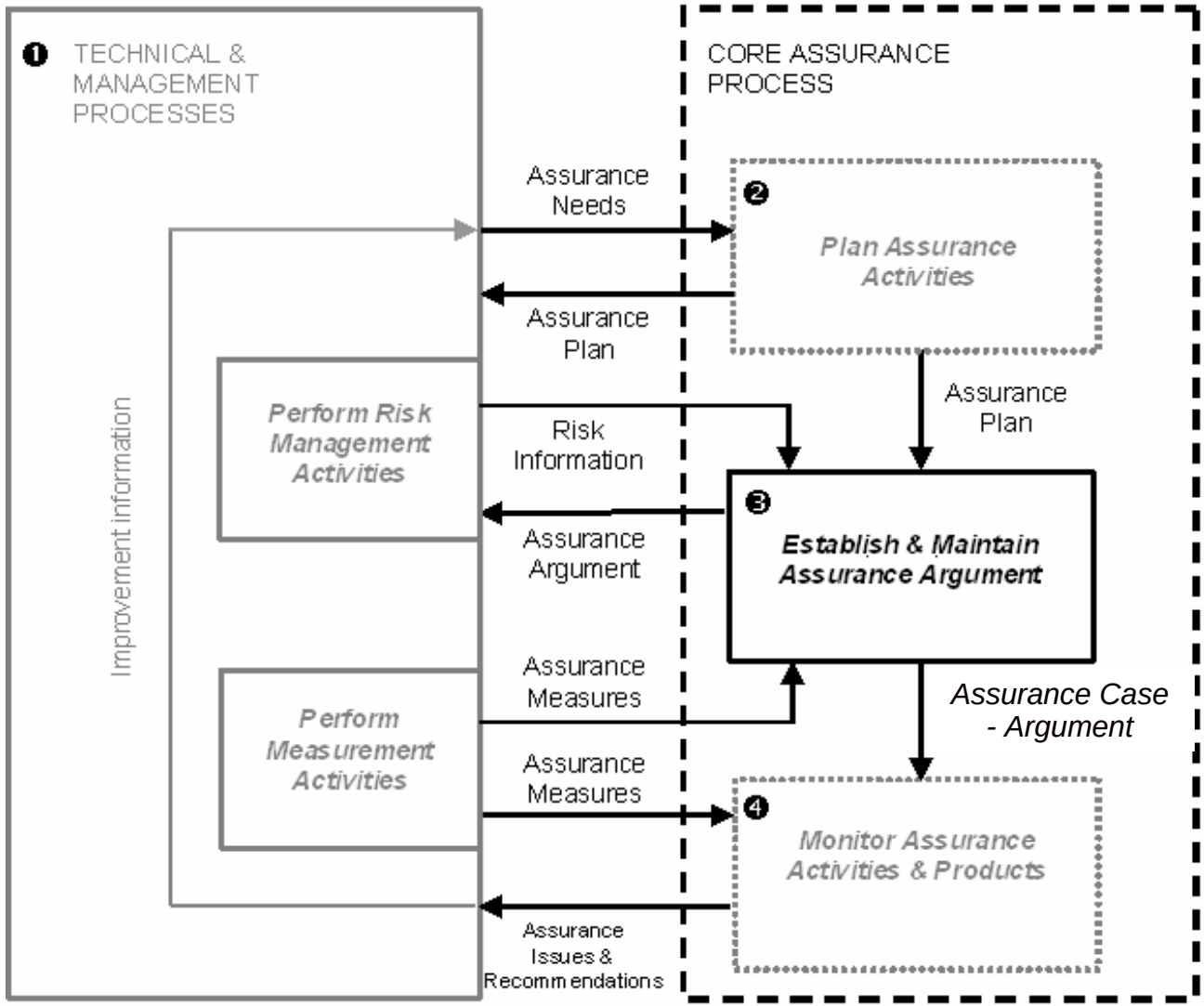
From synthesis and harmonization of practices from 8 standards (4 on security and 4 on safety)

ISO/IEC SC7 Framework for System & SW Assurance



ISO/IEC JTC1 SC7 – System and Software Assurance
Interface with ISO/IEC Standards – Assurance Case/Argument

- Describes interfaces/ amplifications to the Technical & Management processes of ISO/IEC 15288 System Lifecycle & 12207 Software Lifecycle
- Describes interfaces/ amplifications to ISO/IEC 16085 Risk Management Process and 15939 Measurement Process and ISO/IEC 27004 Security Metrics
- Establishes centrality of the Assurance Argument
- Leverages IT security concepts and terminology in ISO/IEC15443
- Leverages OMG's ADM Task Force – Knowledge Discovery Meta-model



Source: ISO/IEC 15026-D4, JTC1, SC7, WG9 (currently in the process of modifying the context interrelationships)

The Assurance Case/Argument – Requires Measurement

- Set of structured assurance claims, supported by evidence and reasoning, that demonstrates how assurance needs have been satisfied.
  - Shows compliance with assurance objectives
  - Provides an argument for the safety and security of the product or service.
  - Built, collected, and maintained throughout the life cycle
  - Derived from multiple sources
- Sub-parts
  - A high level summary
  - Justification that product or service is acceptably safe, secure, or dependable
  - Rationale for claiming a specified level of safety and security
  - Conformance with relevant standards and regulatory requirements
  - The configuration baseline
  - Identified hazards and threats and residual risk of each hazard and threat
  - Operational and support assumptions

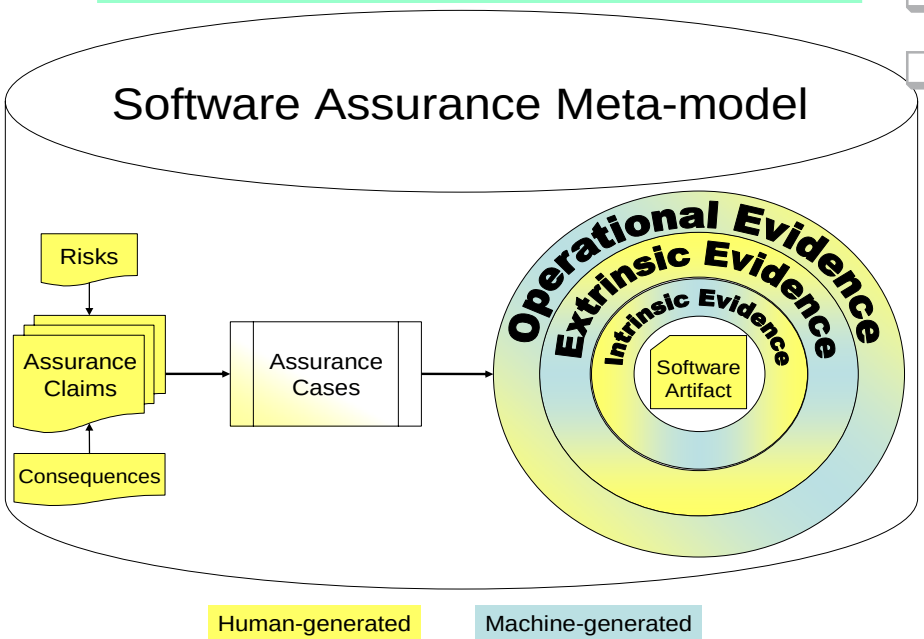
*Adopted from Paul Croll, ISO SC7 WG9 Editor for Systems and Software Assurance

The Assurance Case/Argument

Structure

Part 1 — A coherent argument for the safety and security of the product or service

Part 2 — A set of supporting evidence

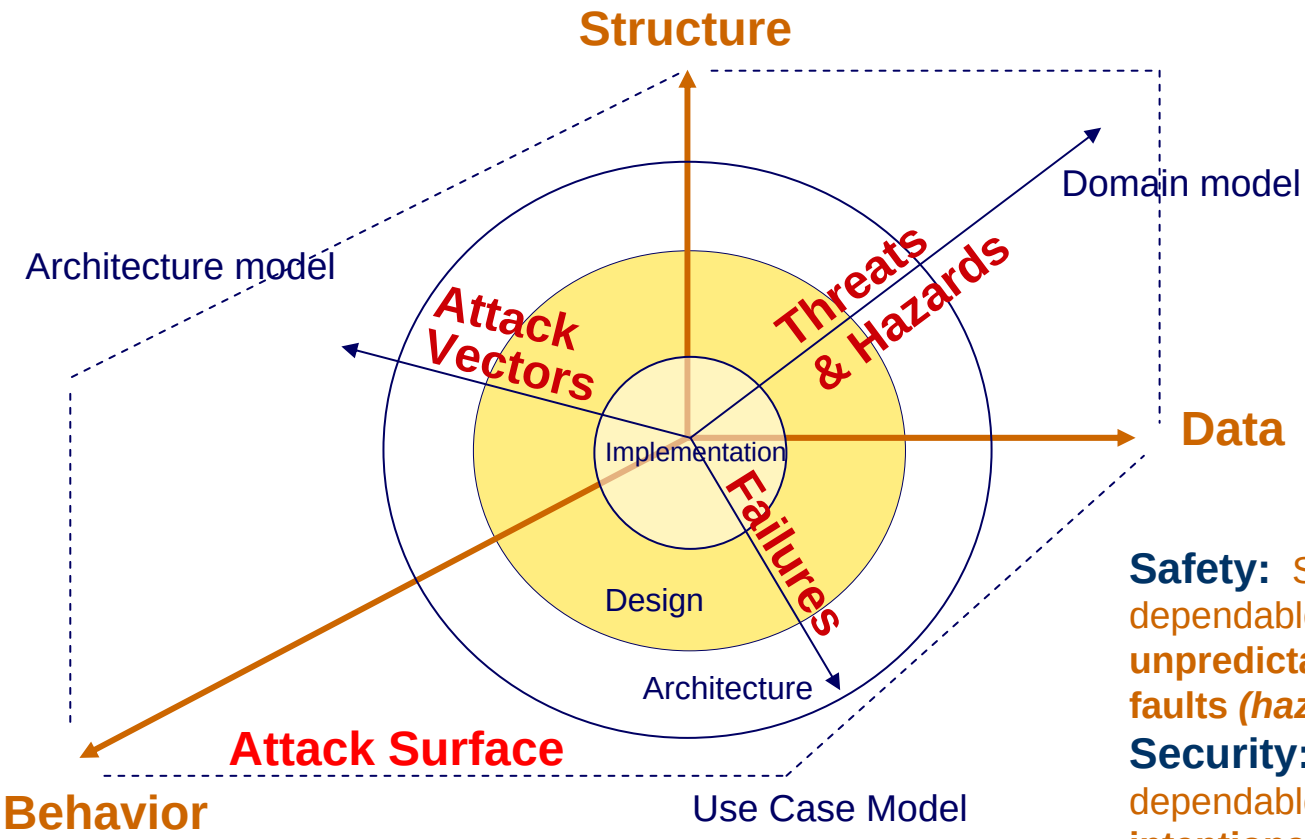


Attributes

- Clear
- Consistent
- Complete
- Comprehensible
- Defensible
- Bounded
- Addresses all life cycle stages

*Adopted from ISO/IEC JTC1 SC7 and OMG SwA

Partition of Concerns in Software-Intensive Systems



Safety: Sustaining predictable, dependable execution in the face of **unpredictable but unintentional faults *(hazards)***

Security: Sustaining predictable, dependable execution in the face of **intentional attacks *(threats)***

Considerations for Assurance Arguments:

-- What can be understood and controlled (failures & attack surface/vectors)?

-- What must be articulated in terms of "assurance" claims and how might the bounds of such claims be described?

From facilitated discussions in SwA WG on Practices and Processes, Aug & Nov 2005

Questions the following slides should answer:

- What do we mean by “secure software”?
- What are the threats to all software?
- What makes software vulnerable to those threats?
- How does the way software comes into existence affect its security?
- What techniques and tools can be used to produce (more) secure software?
- What resources are available to help developers produce secure software?

Homeland Security

Why care?

- Software is everywhere.
- It isn't just applications. It's also…
  - operating systems
  - frameworks and middleware
  - security systems
  - communications/networking systems
  - Embedded systems
  - firmware (like software, it's executable, readable, and writeable)
- SOA implemented in and dependent on software
- Software monitors and controls life-critical physical systems.
- Software manipulates, protects, and exposes extremely sensitive information.
- Software is itself protected by other software.
- The vast majority of software is *not* "built from scratch".

Homeland Security

What threatens software?

- External threats
  - Human attackers
  - Malware
- Insider threats
  - Rogue developers
  - Rogue administrators
  - Rogue users
- Embedded threats
  - Exploited backdoors
  - Malware

– **#1 TARGET FOR *ALL* THREATS**

– **Design defects and implementation flaws that manifest as exploitable weaknesses and vulnerabilities**

Homeland Security

How is software threatened?
in development

By developers who, through ignorance, carelessness, or intention, sabotage or subvert the software by…

- Including weaknesses and vulnerabilities, due to…
  - failure to consider the threat environment and its implications (e.g., inadequate or inaccurate threat models, misuse/abuse cases, etc.)
  - poor design choices
  - use of non-secure technologies (e.g., unsecured HTTP and SOAP)
  - unsafe programming practices, languages, libraries, tools
  - coding errors
  - inadequate non-functional security assessments and tests, or intentionally "fudged" results
- Leaving in backdoors, trapdoors, debugging hooks
- Embedding malware (time and logic bombs, Trojan horses, etc.)
- Including undocumented features/functions ("rotten Easter eggs")

Non-secure configuration management practices make the above easier to accomplish.

How is software threatened?
in distribution and deployment

- "Insiders" (developers, shippers, admins.) who, through ignorance, carelessness, or intention, sabotage or subvert the software by…
  - Not applying integrity mechanisms to executables (e.g., code signatures, digital watermarks)
  - Not using secure download channels (e.g., authenticated SSL)
  - Failing to ship on tamperproof media
  - Tampering with executables before shipping or installing
  - Misconfiguring the software and its environment upon installation
  - Planting rootkits/malware in install. package and/or host platform
  - Failing to apply security patches to COTS and OSS components
- External attackers who intercept and tamper with software downloads/distributions

Homeland Security

How is software threatened?
in operation

- Direct attack by human attackers or malicious processes
- Abuse of privileges by intended users, operators, administrators
- Non-secure administration, e.g., failure to patch
- Delivery of malware (Trojan horses, worms, viruses, etc.)

Homeland Security

But we're not connected to the Internet!

If software interacts with *anything* — human, other software — through *any* kind of interface, its weaknesses and vulnerabilities will be exposed and may be exploited.

Homeland Security

Attack patterns

- **Direct attacks:** Target known or suspected weaknesses and vulnerabilities in the software itself
- **Indirect attacks:** Target…
  - the software's interfaces to other software
  - faults at the boundary between the software and its execution environment
  - environment parameters provided to the software
  - execution environment resources on which the software depends (objective: denial of service)
  - defense in depth measures protecting the software

Homeland Security

Categories of attack patterns

- Reconnaissance
- Privilege escalation *(subversion)*
- Command injection *(subversion)*
- Malicious code *(subversion or sabotage)*
- Denial of service *(sabotage)*
- Integrity violation *(subversion)*
- Confidentiality violation *(sabotage)*

Homeland Security

What makes software vulnerable?

- It's big and complicated, and getting more so – humans can no longer fully comprehend it.
- Component-based development: COTS, OSS, and reuse means no-one really knows where most of it comes from, or how it was built.
- It contains lots of faults and weaknesses. Many of these are exploitable.
- It comes in binary executable form, which makes finding those faults and weaknesses a lot harder.
- It's exposed to threats *all* the time, *even while it's under development.*

Homeland Security

Where weaknesses and vulnerabilities originate *during development*

- Inadequate or spurious requirements
- Inadequate architecture, assembly option, detailed design
- Use of vulnerable processing models, software technologies
- Insecure use of development tools, languages, libraries
- Use of insecure development tools, languages, libraries
- Poor coding practices
- Coding errors
- Use of vulnerable, unpatched components
- Incorrect or mismatched security assumptions
- Inadequate reviews, testing, assessments
- Sabotaged test results
- Residual backdoors
- Sensitive info about software problems in user-viewable code, error messages
- Inadequate configuration documentation
- Insecure installation procedures, scripts, tools

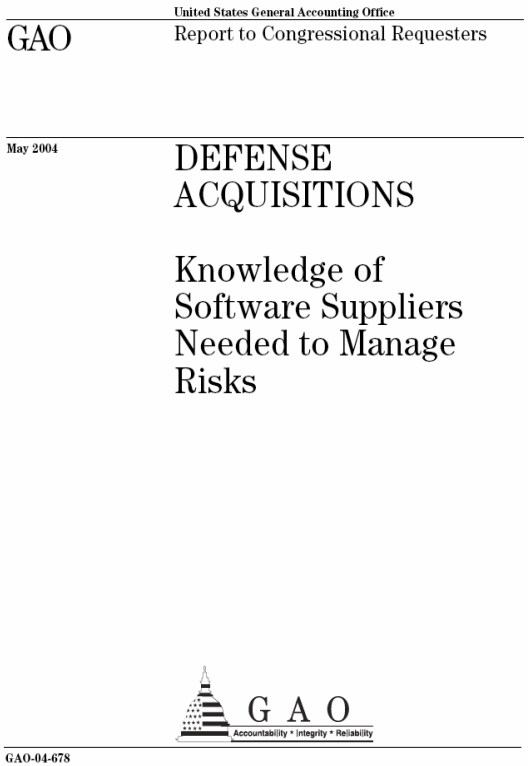
Homeland Security

Malicious Code
embedded during development

- Trojan horses
  - Seems to do one thing, but actually does another
- Time bombs
  - Execution is triggered at a predefined time (on computer clock)
- Logic bombs
  - Execution is triggered by a predefined event, input, or geospatial condition
- Malicious undocumented functions (“rotten Easter eggs”)

Homeland Security

Hard Problem:
Software of Unknown Pedigree (SOUP)


United States General Accounting Office

GAO

Report to Congressional Requesters

May 2004

DEFENSE ACQUISITIONS

Knowledge of Software Suppliers Needed to Manage Risks

G A O

Accountability * Integrity * Reliability

GAO-04-678

December 1999: Defense Science Board Task Force on Globalization and Security cites "Vulnerability of essential U.S. systems incorporating commercial software".

July 2003: The New York Times announces " Uneasiness about security as government buys software."

May 2004: GAO publishes *Defense Acquisitions: Knowledge of Software Suppliers Needed to Manage Risk*

June 2004: IDG News Service reports "Security vendor says offshore development needs check. Extra steps called for to ensure secure code."

2005/2006: GAO highlights risks of offshoring to firms in ambivalent and even hostile countries in *Offshoring of services: an overview of the issues* (Nov. 2005) and *Offshoring: U.S. semiconductor and software industries increasingly produce in China and India* (Sep. 2006)

2006: ACM publishes *Globalization and Offshoring of Software*, citing risks to national security from government use of offshored software. #1 Risk: difficulty understanding code pedigree could allow hostile nations, terrorists, criminals to subvert or sabotage software in government systems.

November 2006: Computerworld announces "DOD report to detail dangers of foreign software. Task force says U.S. adversaries may sabotage code developed overseas."

March 2007: CSIS publishes F*oreign Influence on Software: Risks and Recourse*.

While foreign development risks need domestic mitigation strategies, several studies indicate that domestic suppliers are likely to produce exploitable software unless they put security practices in place throughout the software development life cycle.

Where vulnerabilities originate
during deployment and operation

- Insecure configuration of software and its environment
- Inadequate allocation of resources
- Failure to apply patches
- Software “ageing”

Homeland Security

Secure software...

- Preserves all of its required properties in the face of threats to those properties
  - Dependability is the #1 desirable property for all software
    - If it doesn't work correctly and predictably at all times, what good is it?
- Can resist and/or tolerate most threats that attempt to subvert or sabotage it
- Can terminate, limit the damage, and rapidly recover from the few that succeed

Homeland Security

Dependability properties

- Quality (correctness and predictability)
- Reliability
- Fault-tolerance
- Trustworthiness
- Safety (the above intensified: the software's failure would threaten human life, health, or sustainability of environment)
- ***Security*** (integrity, availability, confidentiality against reconnaissance)
- Assurability

Homeland Security

What makes software secure?

- **Attack-resistance**
  - Components and system recognize and resist attack patterns.
  - System recognizes suspicious component behaviour and
    - isolates/constrains that behavior, or
    - terminates execution of the component
- **Attack-tolerance**
  - Components keep operating in spite of errors caused by attack
  - System keeps operating in spite of attack-caused component errors/failures
- **Attack-resilience**
  - System constrains damage from attacks, isolates itself from attack source
  - System rapidly recovers (to acceptable performance level)

Homeland Security

Security throughout the life cycle

- Security-enhancing process improvement model
  - e.g., FAA iCMM/CMMI safety & security extensions, SSE-CMM
- Security-enhancing life cycle methodologies
  - e.g., CLASP, SDL, McGraw's 7 Touchpoints, TSP-Secure, AEGIS, RUPSec, SSDM, Oracle Secure SW Assurance, Waterfall-Based SW Security Engineering Process
- Establishing security entry and exit criteria for each life cycle phase
- Including appropriate and sufficient security reviews, analyses, tests at each phase
  - e.g., threat models, attack trees, vulnerability analyses, code reviews, black box tests, risk analyses, assurance cases
- Secure SCM
- Education, training, awareness, professional certification
- QA of security of software processes and practices

Homeland Security

Secure requirements engineering

- Risk-driven vs. functionality-driven:
  - non-functional requirements (what software must *be*, vs. what it must *do*)
  - constraint requirements
  - negative requirements
    - Need to allow time for translating these into requirements for functionality (what can be built/tested)
      - e.g., no buffer overflows = must do input validation; must be fault-tolerant = must have exception handling that...)

Homeland Security

Reducing SW security risk: acquisition

- Include security requirements and evaluation criteria in all RFPs
- Strict monitoring/control of “non-traditional” acquisitions (e.g., OSS, shareware, freeware downloads)
- Supplier and integrator background checks (COTS)
- Supplier and integrator SDLC process reviews/capability assessments with security criteria
- Procurement and contract language requiring COTS suppliers to warrant safe, secure product behaviour
- Code analysis and assessment of products for known and common vulnerabilities *before* purchase (COTS) or download (OSS, freeware)
- Pedigree analysis: in an ideal world, acquisition policy would favor software with *known* pedigree

Homeland Security

Technological pre-commitments

- Commitments to use specific technologies and products are increasingly made at the enterprise level, then backed up by policy.
- Requirements of individual systems are seldom considered.
- Software and system engineering become exercises in working around undesirable features and properties.
  - Requirements have to be written in ways that ensure they can be satisfied within the constraints imposed by technological pre-commitments.
  - Additional requirements must be added to mitigate known vulnerabilities and security mismatches that use of pre-committed technologies/products introduce.
- Thorough, iterative risk analyses throughout system lifecycle should capture unacceptably high cost of workarounds and countermeasures, make case for waiving pre-commitments to high-risk technologies/products.

Homeland Security

What does Common Criteria evaluation say about software's security?

- It doesn't look at the right products.
  - Products without significant security functionality are not eligible for CC evaluation.
- The questions it asks are not adequate to determine whether software is secure
  - Focus of CC evaluations is on *correctness* and *security policy conformance* of TOE's security functions and controls.
  - Little if any CC language addresses software security concerns.
    - Software assurance language was added to draft CC v.3.
      - ISO/IEC period for considering draft expired before v.3 adoption

Homeland Security

What does Common Criteria evaluation say about software security? cont'd

- It doesn't look at the product in helpful ways.
  - CC evaluation is based predominantly on documentation analysis.
  - Direct testing of TOE limited to correctness of security functions.
- It doesn't adequately address the product's development process.
  - No rigour in product security engineering required below EAL5.
  - No formal methods are required until EAL7.
    - Most products are evaluated at EAL4 or below.

Homeland Security

Reducing SW security risk: source selection

Analyze security of each item to be purchased/licensed to determine feasibility of its use in the intended system and environment:

- Code reviews, vulnerability assessments
  - Select item only if there are feasible, affordable countermeasures for its vulnerabilities
- Identification of security assumption mismatches: what the software actually expects (of its environment, etc.) and does, vs. what it will actually be provided and how it will be used
  - Select item only if there are feasible, affordable workarounds or countermeasures for its security assumption mismatches
- Evaluate security evidence: vulnerability/incident reports, patch history; C&A documents, CC evaluation reports; relevant standards conformance; supplier reputation, track record, development process, willingness to warrant security.
  - In future, software security assurance cases will make this type of evaluation easier

Homeland Security

Secure software architecture and design

- System processing model doesn't preclude secure behaviors, interactions
- Minimisation of vulnerabilities—quantity and exposure—through security measures and countermeasures (discussed later)
- Secure intercomponent and extrasystem interfaces (APIs, RPCs, UIs) Prevents excessive trust in high risk (including SOUP) components
- Absolutely minimises privileges granted to *all* processes/components at all times
- Isolates and constrains environment in which high-risk software operates
- Minimises untrusted software access to/interaction with trusted software

Homeland Security

Secure software architecture and design cont'd

- Addresses mismatches in components' assumptions about each other:
  - Component A may expect Component B to provide certain
    - functionality (e.g., signature validation)
    - properties (e.g., fault tolerance)
    - outputs (format, length, etc.)
    - interfaces (APIs, RPCs, protocols)
- Addresses inaccurate assumptions about the environment:
  - Component may expect the execution environment to provide
    - certain functionality (e.g., PKI)
    - certain protection (e.g., sandboxing)
    - certain inputs (i.e., environment parameters

Homeland Security

Security issues of component-based software

- Mismatches in component assumptions about each other and execution environment: Component may expect…
  - certain functionality in another component (e.g., signature validation)
  - certain functionality in the environment (e.g., PKI)
  - certain properties in other components (e.g., fault tolerance)

Homeland Security

Sources of inaccurate assumptions

- Incomplete, omitted, overly-general, or poorly-stated functionality-constraining and nonfunctional property requirements
- Failure to translate such requirements into actionable requirements
- Architecture and design that do not satisfy their actionable non-functional (property) and negative (constraint) requirements
- Ignoring the security implications of different languages, tools, and technologies, and how they are used in implementing the software
- Failure to evaluate security of non-developmental components, alone and in combination with other components, before selection
- Security reviews/tests not included in each SDLC phase

Homeland Security

Sources of inaccurate assumptions (cont'd)

- Test cases limited to normal operating conditions
- Lack of risk-driven security testing, i.e., abnormal conditions, test cases based on attack patterns
- Lack of stress testing, i.e., abnormal activity, inputs, etc. to validate design assumptions
- Inadequate preparation of the software for distribution/deployment
- No verification that security standards have been conformed to
- Software design does not match intended operational environment

Homeland Security

SOUP = inaccurate security assumptions

- Unable to infer component trustworthiness from knowledge of development process
- Unable to infer component trustworthiness from supplier reputation
- Disjoint product and patch release schedules
- Disjoint supplier priorities vs. system requirements
- Publishing of known vulnerabilities: attackers know at least as much as system developers
  - Attackers don't care about license Ts&Cs "preventing" reverse engineering, which means they probably know much more.
- Potential hostile foreign influence on offshore developers may result in products with embedded malicious code, rotten Easter eggs, intentional vulnerabilities

Homeland Security

Reduce SOUP risk: architecture

- Define different candidate system architectures in which to evaluate components, model component risks
  - include threat, attack, vulnerability modeling for each candidate architecture
  - evaluate both architecture and components together
    - architecture provides framework for revealing intercomponent behaviors, assumption (mis)matches
    - candidate components verify security of architecture-defined component combinations, configurations, process flows

Homeland Security

Secure implementation and testing

- Secure coding practices supported by tools
- Write, acquire, reuse only components proven dependable, free of exploitable faults and weaknesses
- Security testing
  - White box:
    - static and dynamic code analysis
    - fault injection/propagation analysis
  - Black box
    - fault injection
    - fuzzing
    - penetration testing
    - vulnerability scanning

Homeland Security

Reduce SOUP risk: testing, risk management

- Black box—and when source code is available, white box—security testing
  - individual components
  - pairs of components
  - whole system
- Ongoing risk analysis and reengineering
  - find known-pedigree components with req'd capabilities to replace SOUP
  - redesign system so SOUP components' capabilities are no longer needed
  - apply new countermeasures to further reduce SOUP component risk

Homeland Security

Secure distribution, deployment, sustainment

- Trusted distribution techniques
  - code obfuscation
  - digital watermarking
  - code signing
  - authenticated, encrypted download channels
- Installation configuration that ensures
  - secure interactions with execution environment
  - adequate allocation and safe management of environment resources
- Sustainment and support
  - impact analyses of new requirements, own and supplier updates, patches
  - ongoing risk assessment to identify new requirements
  - forensic analysis (post-incident) to identify new requirements

Homeland Security

SW security measures and countermeasures

- Programmatic
  - input and output validation wrappers
  - obfuscation (to deter reverse engineering)
  - secure exception handling (in custom software)
  - fault tolerance measures
    - redundancy
    - diversity (redundancy using different components with comparable functions)

Security measures and countermeasures cont'd

- Development tools and languages
  - type-safe languages
  - safe versions of libraries
  - secure compilers
  - secure compilation techniques
- Environment-level measures
  - virtual machines/sandboxes
  - chroot jails
  - trusted OS with mandatory integrity policy/compartments
  - secure microkernels
  - TPMs
  - program shepherding
  - altered memory maps
  - system call filters

Homeland Security

Security measures and countermeasures cont'd

- Add-ons
  - code signing with signature validation
  - obfuscation and digital watermarking (to deter reverse engineering)
  - malware/spyware scanners (host level)
  - application security gateways/firewalls
  - intrusion detection/prevention (network and host based)
- Development process (see *Security in the Software Life Cycle*)

Homeland Security

Resources

- US-CERT BuildSecurityIn portal

  `https://buildsecurityin.us-cert.gov/`

- K.M. Goertzel, *et al: Security in the Software Life Cycle* Draft Version 1.2 (DHS CS&C NCSD Software Assurance Program, September 2006) – new version planned for 2007

- NIST SAMATE portal

  `http://samate.nist.gov/`

Homeland Security