Software Assurance: Trusting the Untrustable

Dr. Larry Wagoner
NSA IAD/JHU APL

Software as a Core Concern

- Then
  - Domestic dominated market and small number of suppliers
  - Stand alone systems
  - Software small and simple
  - Software small part of functionality
  - Custom and closed development processes (vetted personnel)
  - Adversaries known, few, and technologically less sophisticated
- Now
  - Who, what, where of software origin is mostly unanswerable
  - Globally network environment
  - Software large and complex
  - High reliance on software and software as core of systems
  - Business Decision to use commercial software and software reuse
  - Adversaries numerous and sophisticated

Desired Attributes of Software

- Dependability
  - Reliability
  - Security
    - Confidentiality
    - Integrity
    - Availability
  - Safety
  - Maintainability
  - Other -ilities

Summary from CIO Survey

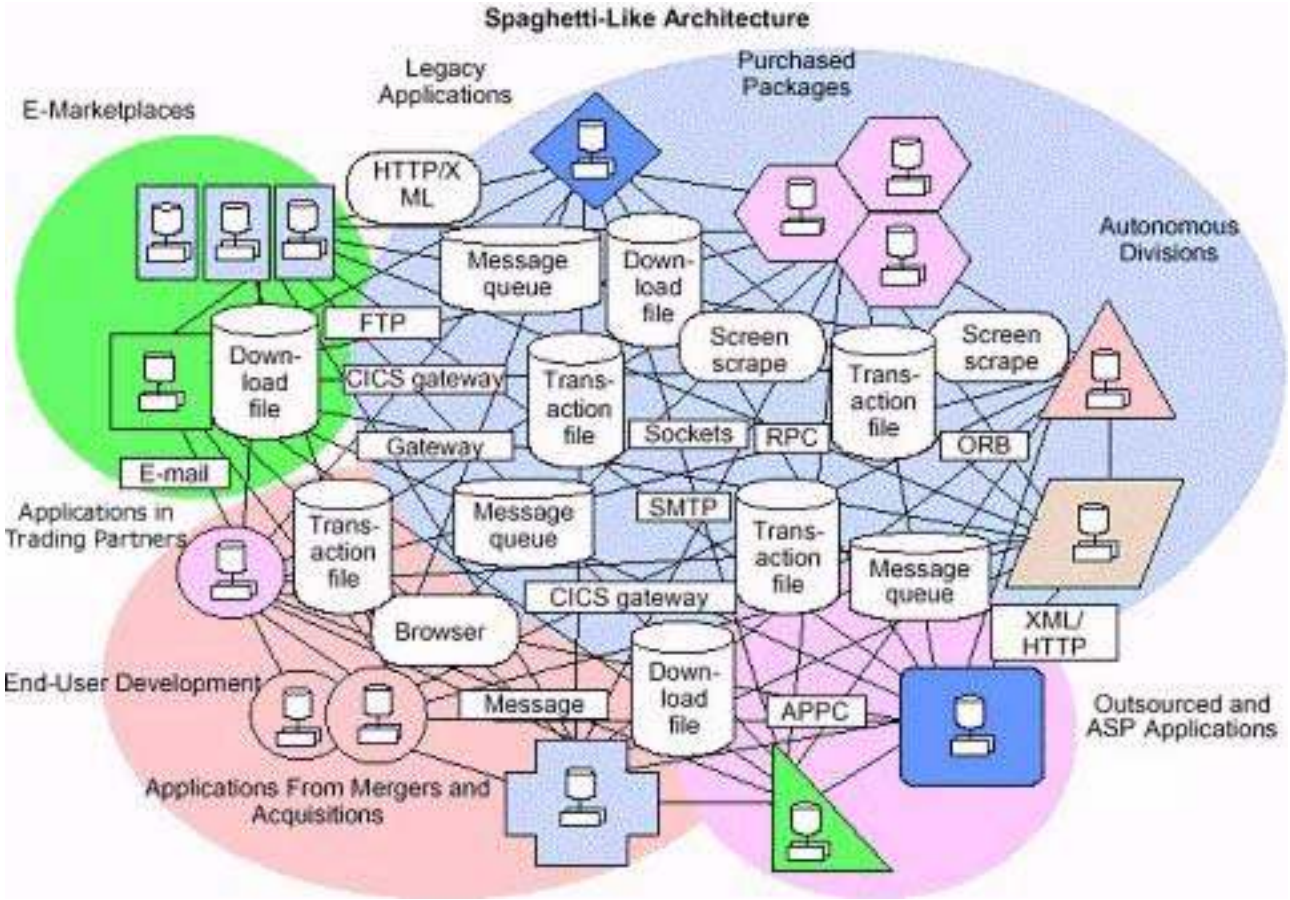
- Reliable software and vulnerability-free software are the top priorities
- CIOs have low to medium confidence in software's ability to be free of flaws, security vulnerabilities and malicious code flaws
- 86% of CIOs rate the fundamental security of software as vulnerable or extremely vulnerable
- The majority have had to redeploy staff, incur increased IT costs and suffer reduced productivity due to software flaws
- Internal testing, contracts/SLAs and reputation among peers are the most preferable means for CIOs to determine if software is free of flaws
- The majority would like vendors to certify software meets a designated security target and to scan for flaws and security vulnerabilities using qualified tools

CIO Executive Council
The Professional Organization for CIOs

Realities of Relying on Software

- Software has defects – many defects have security implications
- Current software patching solutions are always struggling to stay ahead of attacks
- Newly invented attacks may make good secure software vulnerable
- Software may be used in a different environment than for which it was written
- Hackers are continiously trying to break into your systems at every level

Reality of Software



Consists of complex, multiple technologies with multiple suppliers

- **Based on average defect rate, deployed software package of one million Lines Of Code has 6000 defects**
- **Assume only 1% of those defects are security vulnerabilities, there are 60 different opportunities for someone to attack the system**

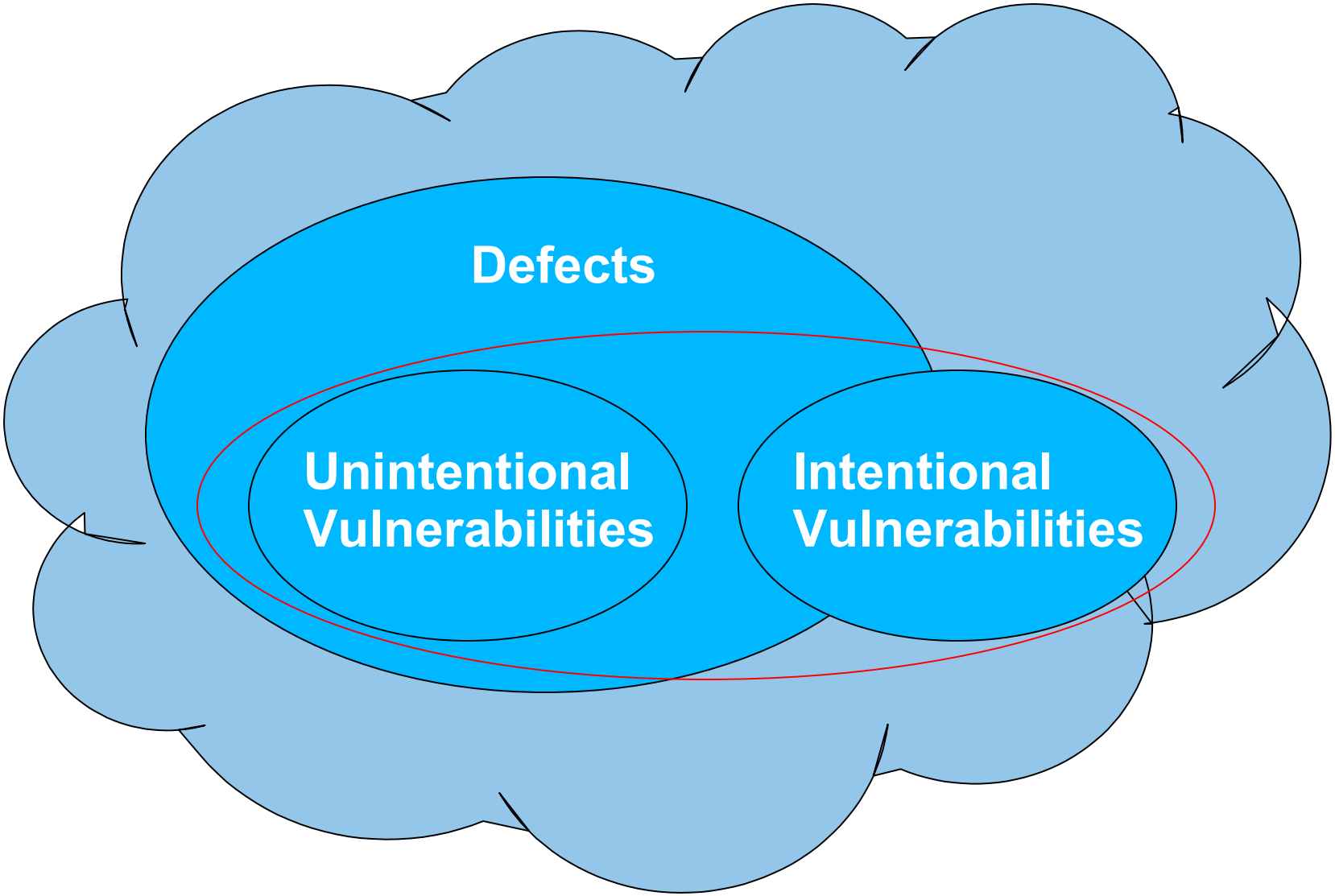
Software Assurance (SwA) is Needed

- Software Assurance
  - 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 software functions in the intended manner – Source: Committee on National Security Systems (CNSS) Instruction No. 4009, "National Information Assurance Glossary." Revised 2006 http://www.cnss.gov/instructions.html

DoD Software Assurance Problem

- Software is fundamental to the Global Information Grid (GIG) and critical to weapons, business and support systems
- Threat Agents: Nation-state, terrorist, criminal, rogue developer who:
  - Exploit vulnerabilities remotely
  - Gain control through supply chain opportunities
- Vulnerabilities
  - Unintentional vulnerabilities maliciously exploited (e.g. poor quality or fragile code)
  - Intentionally implanted logic (e.g. back doors, logic bombs, spyware)
- Adversary may steal or alter mission critical data, or corrupt or deny the function of mission critical platforms

Software Program Space



Unintentional Vulnerabilities

- Classic case: buffer overflow

```
foo (char *s)
{
    char buf[10];
    strcpy (buf,s);
    printf (“buf is %s\n”,s);
}
foo (“This will overflow you really fast”)
```

Intentional Vulnerabilities

- Classic case: time/logic bomb

```
int main(void) {
  time_t curtime;
  struct tm *loctime;
  // Get time of day
  curtime = time(NULL);
  // Converts date/time to a structure
  loctime = localtime (&curtime);
  // Output ASCII data/time
  printf ("Local time is: %s", asctime(tblock));

 // if April 1st set off time bomb
  if (loctime->tm_mon == 4) && (loctime-> tm_day == 1))
      // do some bad things
}
```

Unintentional Vulnerability

- Classic case: buffer overflow

```
foo (char *s)
{
char buf[10];
strcpy (buf,s);
printf (“buf is %s\n”,s);
}
```

Intentional Vulnerability

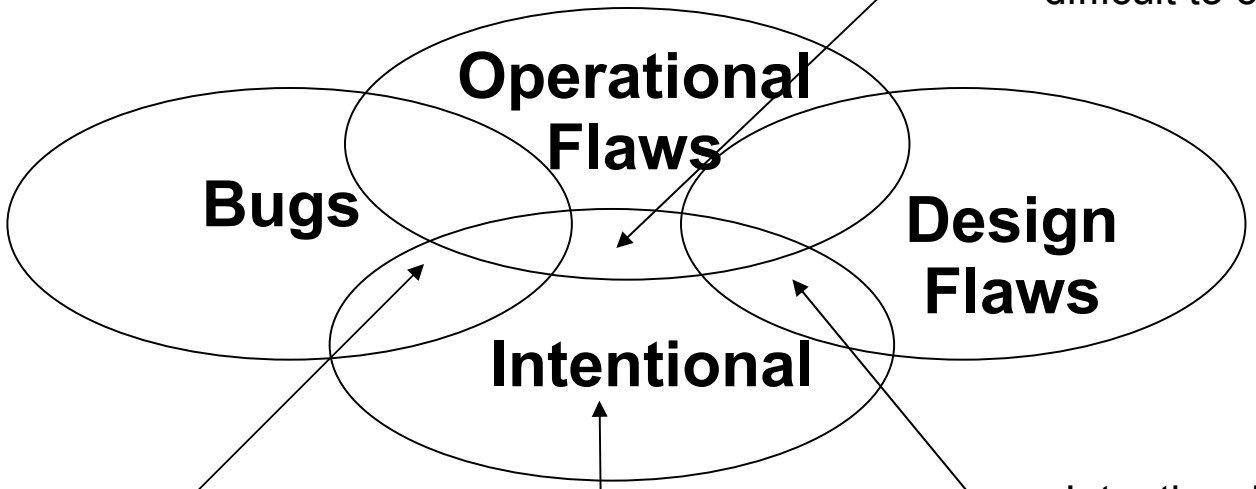
- Classic case: buffer overflow

```
foo (char *s)
{
char buf[10];
strcpy (buf,s);
printf (“buf is %s\n”,s);
}
```

Challenges

- Software size and complexity
- Pace of change
- Time and cost of evaluation
- Unintentional vs. Intentional vulnerabilities
- Immature science of finding vulnerabilities

Malicious Vulnerability Classes and Relationships



Intentional operational flaw - e.g. password too difficult to change

Intentional bug – e.g. intentionally planted buffer overflow

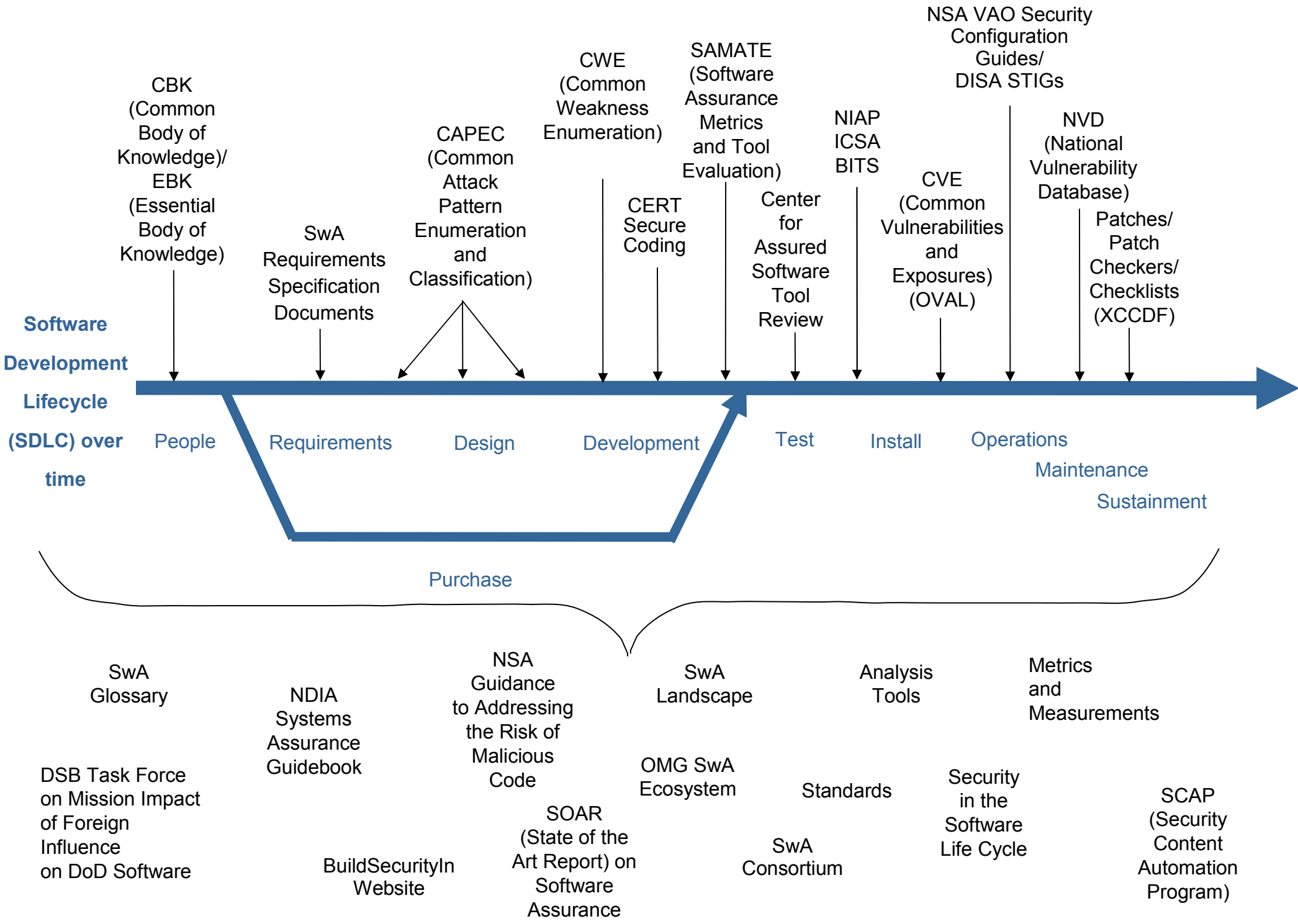
Intentionally malicious code – e.g. time/logic bomb, covert channel

Intentional design flaw – e.g. intentionally weak crypto, admin accounts with default passwords

How to Assure Software

- Build it right
  - Easy to say...
- Use safer languages or language subsets
- Follow published guidance
- Use tools to identify vulnerabilities
  - Static analysis
  - Dynamic analysis
  - Concolic analysis

Overview of Current SwA Efforts


CBK (Common Body of Knowledge)/ EBK (Essential Body of Knowledge)
SwA Requirements Specification Documents
CAPEC (Common Attack Pattern Enumeration and Classification)
CWE (Common Weakness Enumeration)
CERT Secure Coding
SAMATE (Software Assurance Metrics and Tool Evaluation)
Center for Assured Software Tool Review
NIAP ICSA BITS
CVE (Common Vulnerabilities and Exposures) (OVAL)
NSA VAO Security Configuration Guides/ DISA STIGs
NVD (National Vulnerability Database)
Patches/ Patch Checkers/ Checklists (XCCDF)
Software Development Lifecycle (SDLC) over time
People
Requirements
Design
Development
Test
Install
Operations
Maintenance
Sustainment
Purchase
SwA Glossary
NDIA Systems Assurance Guidebook
NSA Guidance to Addressing the Risk of Malicious Code
SwA Landscape
Analysis Tools
Metrics and Measurements
DSB Task Force on Mission Impact of Foreign Influence on DoD Software
BuildSecurityIn Website
SOAR (State of the Art Report) on Software Assurance
OMG SwA Ecosystem
SwA Consortium
Standards
Security in the Software Life Cycle
SCAP (Security Content Automation Program)

Efforts for Specific Parts of the SDLC

- **Common Body of Knowledge (CBK) for SwA**
  - Compilation of SwA knowledge and best practices
  - Serve as input in curricula for university and college courses
- **Essential Body of Knowledge (EBK)**
  - Serve as the basis for non-academic workforce training programs and classes
- **SwA Requirements Specification Documents**
  - DoD effort to create a boilerplate set of requirements for software
  - SwA in Acquisition: Mitigating Risks to the Enterprise
    - Due Diligence questionnaires
    - Sample statements for contract language
- **CAPEC (Common Attack Pattern Enumeration and Classification)**
  - Defines a standard taxonomy of definitions, classifications and categorizations of software targeting attack patterns
- **CWE (Common Weakness Enumeration)**
  - Common language for describing software security weaknesses in architecture, design, and code
  - Standard metric for software security tools that target those weaknesses
  - Common baseline standard for identification, mitigation, and prevention of weaknesses
  - **SBVR (Semantics of Business Vocabulary and Rules)**
    - a metamodel specification for capturing expressions in a controlled natural language and representing them in formal logic structures

Efforts for Specific Parts of the SDLC (cont.)

- **CERT Secure Coding Project website**
  - **Recommendations for avoiding vulnerabilities in C/C++ and Java**
  - **http://www.cert.org/secure-coding/**
- **SAMATE (Software Assurance Metrics and Tool Evaluation)**
  - Develop metrics to gauge the effectiveness of existing software assurance tools
  - Assess current software assurance methodologies and tools to identify deficiencies that may introduce software vulnerabilities or contribute to software failures
  - Products and activities
    - Taxonomy of classes of software assurance tool functions
    - Workshops for software assurance tool developers and researchers and users to prioritize particular SA tool functions
    - Specifications of SA tool functions
    - Detailed testing methodologies
    - Workshops to define and study metrics for the effectiveness of SA functions
    - A set of reference applications with known vulnerabilities
    - Papers in support of SAMATE metrics, including a methodology for defining functional specifications, test suites, and Software assurance tool evaluation metrics
- **CAS (Center for Assured Software) Tool Review**
  - Methodical evaluation of five of the leading source code analysis tools

Efforts for Specific Parts of the SDLC (cont.)

- **NIAP (National Information Assurance Partnership)/ICSA Labs/BITS (Banking Industry Technology Secretariat) Testing**
  - **Independent testing of software products**
- **CVE (Common Vulnerabilities and Exposures)**
  - Repository of information about common vulnerabilities tracked and reported by others
  - **OVAL (Open Vulnerability and Assessment Language)**
    - A standard schema and language for expressing information about the publicly-known vulnerabilities and exposures defined and categorized in the CVE
- **NSA VAO Security Configuration Guides/DISA STIGs**
- **NVD (National Vulnerability Database)**
  - A comprehensive cyber security vulnerability database that integrates all publicly available U.S. Government vulnerability resources and provides references to industry resources
- **Patches/Patch Checkers/Guidance**
  - **XCCDF (Extensible Configuration Checklist Description Format)**
    - Specification language for writing security checklists, benchmarks, and related kinds of documents

Across the SDLC Efforts

- **SwA Glossary**
  - Glossary of SwA terms created by the NSA Center for Assured Software
- **NDIA Systems Assurance Guidebook**
  - Organized using ISO/IEC 15288, System Life Cycle Processes
- **NSA Guidance to Addressing the Risk of Malicious Code**
  - **Designed to provide consistent IA guidance to the system development programs supported by the ISSEs**
  - **Organized using ISO/IEC 12207, Software Life Cycle Processes**
- **BuildSecurityIn Website**
  - DHS sponsored, CERT hosted
  - Core website location for information and links to best practices, tools, guidelines, rules, principles and other resources for building security into software at every phase of its development
  - https://buildsecurityin.us-cert.gov

Across the SDLC Efforts (cont.)

- **DSB Task Force on Mission Impact of Foreign Influence on DoD Software**
  - Goal is to characterize the causes for and level of DoD dependence on foreign-sourced software and assess the risks presented by that dependence
  - Identify policies or technological research that can be undertaken to improve the ability to determine and sustain the trustworthiness and assurability of software
  - Identify requirements for intelligence gathering to better characterize and monitor the threat from foreign suppliers
  - Prioritize DoD software components according to their need for high levels of trustworthiness
  - Identify the organizations within DoD and the Intelligence Community that are considered key stakeholders for software assurance, and determine their current capabilities, as well as any research, technology, policy, or legal challenges they need to address
  - Available at http://www.acq.osd.mil/dsb/reports/2007-09-Mission_Impact_of_Foreign_Influence_on_DoD_Software.pdf

Across the SDLC Efforts (cont.)

- **SOAR (State of the Art Report) on Software Assurance**
  - Identifies and describes the current "state of the art" in software security assurance, including trends in the following areas:
    - Techniques that are now being used, or are being published (e.g., as standards), to produce-or increase the likelihood of producing-secure software. Examples: process models, life cycle models, methodologies, best practices
    - Technologies that exist or are emerging to address some part of the software security challenge, such as virtualized execution environments, "safe" and secure versions of programming languages and libraries, and software security testing tools
    - Current activities and organizations in government, industry, and academia, in the U.S. and abroad, that are devoted to systematic improvement of the security of software
    - Research sector trends-both academic and non-academic, U.S. and non-U.S.-that are intended to further the current activities and state of the art for software security
  - http://iac.dtic.mil/iatac/download/security.pdf

Across the SDLC Efforts (cont.)

- **Standards**
  - IEEE Revision of ISO/IEC 15026:2006, System and Software Integrity Levels
  - IEEE Standard 1074-2006, is a revision of the IEEE Std. 1074-1997, Developing Software Project Life Cycle Processes
    - intended to add support for prioritization and integration of appropriate levels of security controls into software and systems
    - formed the basis for developing ISO/IEC 12207.1—Standard for Information Technology—Software life cycle processes and 12207.2—Software life cycle processes—Life cycle data
  - ISO/IEC JTC1/SC22 Other Working Group: Vulnerabilities (OWGV)
    - examining vulnerabilities from a safety and security approach
    - to produce a Technical Report (TR) entitled “Guidance to Avoiding Vulnerabilities in Programming Languages through Language Selection and Use”
    - http://www.aitcnet.org/isai/
- **Analysis Tools**
  - Many commercial and free tools for software security
  - One survey: http://samate.nist.gov/index.php/Tools

Across the SDLC Efforts (cont.)

- **Metrics and Measurements**
- **Security in the Software Life Cycle**
  - Developers guide that describes a multitude of
    - Secure software process models
    - Software methodologies (security enhanced or not)
    - Security testing techniques

Across the SDLC Efforts (cont.)

- **SCAP (Security Content Automation Program)**
  - Repository of security content to be used for automating technical control compliance activities (e.g. FISMA/800-53), vulnerability checking (both application misconfigurations and software flaws), and security measurements.
- **SwA Consortium**
  - Gather and coordinate consumer software assurance needs, requirements, concerns, and priorities
  - Define requirements for risk assessment and testing of software; do so using language that includes standard representations of software vulnerabilities (e.g., CVE, CWE);
  - Identify and provide information about end-user tools that can solve specific user/consumer software security problems (e.g., anti-malware, anti-spyware);
  - Establish a scheme for rating software products' security, quality, assurance;
  - Identify software best practices, guidance, etc., of benefit from a consumer perspective;
  - Fund research that will benefit consumers and fill perceived R&D gaps.

Across the SDLC Efforts (cont.)

- **OMG SwA EcoSystem**
  - Leverage related OMG specifications such as the Knowledge Discovery Metamodel (KDM), the Semantics of Business Vocabulary and Rules (SBVR), and the upcoming Software Assurance Meta-model
    - To provide an effective vehicle for communications of assurance information between developers and stakeholders on the one hand, and certifiers and evaluators on the other
    - To provide a repository for gathering assurance claims and arguments
    - To improve the objectivity and accuracy of evidence collection;
    - To enable the rapid evaluation of evidence and building of evidence correlation models
    - To automate validation of claims against evidence, based on arguments
    - To enable more accurate and highly automated risk assessments

Other Efforts

- **CVSS (Common Vulnerability Scoring System)**
  - A framework for assessing and quantifying the impact of software vulnerabilities
- **CPE (Common Platform Enumeration)**
  - a structured naming scheme for information technology systems, platforms, and packages
  - includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name.
- **CCE (Common Configuration Enumeration)**
  - provides unique identifiers to system configurations in order to facilitate fast and accurate correlation of configuration data across multiple information sources and tools

Thank you.

Questions?