Software Assurance Throughout the System Life Cycle

Mr. Thomas Hurt
Strategic Technology Protection and Exploitation
Office of the Under Secretary of Defense for Research and Engineering

22nd Annual NDIA Systems and Mission Engineering Conference
Tampa, FL | October 23, 2019
Modern military systems consist of hundreds of components with thousands of interconnections executing millions of lines of software.
Is the Future Sustainable?

New Features/Components added continuously

Everything is interconnected or networked (Internet of Things (IoT))

Technology continues to advance (methods of attack)

The addition of new components, changes to the network, and advancement of adversary technology creates an unsustainable continuous cycle of redesign and patching to protect against adversarial access.
Are We Really Protecting?

**Scope:**
Modern military systems consist of hundreds of components with thousands of interconnections executing millions of lines of software.

**Cybersecurity:**
The Risk Management Framework (RMF) is used to determine if a program receives an Authority to Operate (ATO) for a single version and configuration at the time of deployment.

---

Cybersecurity compliance is insufficient to address the scope of the threat.

Flaws in network architecture, mission software system design, and operations expose existing software weaknesses and known vulnerabilities to the adversary. Any of these weaknesses, when used by the attacker, can lead to degraded confidence, loss of lethality, or complete mission failure.
What If We Fixed The Vulnerabilities First?

New Features/Components added continuously

- Integrating new software, fixing known vulnerabilities first, is less challenging due to a reduction in exposure of vulnerabilities even when interoperating with vulnerable systems.

Everything is interconnected or networked (Internet of Things (IoT))

- In the event of unknown or unauthorized connection to secure systems, adversary remains unable to exploit vulnerabilities in connected devices.

Technology continues to advance (methods of attack)

- Removal of vulnerabilities prevents exploitation regardless of the adversary or attack method.

When DoD programs fix vulnerabilities, successful future cyberattacks are prevented. Instead of attempting to mitigate risk at the entry point, the vulnerabilities adversaries typically exploit have actually been removed.
Updated System Tactical Use Threads
How will a component be used? Its tactical criticality?

Engine Control SW (ECS) provides needed metrics
Input: Engine performance data; Output: Needed alerts/response
Read/write capabilities to data bus do needed functions

Mission Threads
What will my system do and how will it interact?

Engine functionality will be controlled by ECS
Engine Monitoring System will monitor engine performance
Performance issues will be transmitted by data bus to control panel

System Requirements
What is required to get from concept to product?

ECS has no known vulnerabilities
Monitoring SW cannot be exploited to access ESC or data bus
Secure Design/Architecture considerations for Data bus communication

Countermeasure Selection
Tactical criticality focuses on review of NVD and vendor SwA activities for COTS
Binary Analysis
Static & Origin Analysis
Intel & Attack Modeling
Coding Standards
Penetration Testing
Architecture/Design Inspections
Monitored Execution

MIL-STD-1533b

22nd Annual NDIA Systems and Mission Engineering Conference
Oct 2019
84% of breaches exploit the vulnerabilities in the application, yet funding for IT defense vs. software assurance is 23 to 1. 

Who Fixes the Most Vulnerabilities?

What is the percentage of known vulnerabilities remediated by each industry vertical, in order to reduce application-layer risk?

- **Manufacturing**: 81%
- **Financial Services**: 65%
- **Retail + Hospitality**: 60%
- **Technology**: 50%
- **Healthcare**: 43%
- **Government**: 27%

The data represents 38,670 application assessments submitted for analysis during the 18-month period from October 1, 2013 through March 31, 2015 by large and small companies, commercial software suppliers, open source projects and software outsourcers.

Contest: Need for Engineering-in Software Assurance Activities over the Software Development Life Cycle (SDLC)

- **Where Software Flaws Are Introduced**
  - Requirements Engineering: 70%
  - System Design: 20%
  - Software Architectural Design: 10%

- **Where Software Flaws Are Found**
  - Code Development: 3.5%
  - Unit Test: 16%
  - Integration Test: 50.5%
  - System Test: 9%
  - Acceptance Test: 21%
  - Operation: 21%

**Improved focus on engineering-in software assurance activities needed on the front end of the SDLC**

**Source:** Carnegie Mellon University, Software Engineering Institute (Critical Code; NIST, NASA, INCOSE, and Aircraft Industry Studies), used with permission.
Software Assurance Supplied to Traditional Systems Engineering Process

Operational Need

Delivered Capability

Disposal

Sustainment and Continuous Engineering
- Monitor for 3rd party vulnerabilities
- Continued Assessment & Timely Patching

Operation
- Implement operational monitoring and response
- Risk Management Framework (RMF)

Validation
- Conduct Third Party SwA Testing
- Validate security requirements/assumptions

Transition
- Transition data rights
- Ensure acquirer can rebuild & retest

Requirements

Validation Solution

Continuous Application Across the Acquisition Lifecycle

Design & Product

Integration
- Full system regression testing
- Automated reproducible build

Implementation
- Warning flags & Coding standards
- Hardening measures
- Code reviews

Verification
- Static source code weakness analysis
- Binary Analysis
- Origin analysis
- Web app scanners & fuzzers
- Negative testing
- Automated test suite w/overage
- Penetration Testing

SUPPORT:
- Risk Management: Assurance case
- Configuration Management: Version control, Access control, Code signing
- Measures & Metrics

NOTE: Lifecycle processes typically occur simultaneously, not in sequence; see ISO/IEC 15288 & 12207

NOTE: Implementation, Integration & Verification are often performed continuously & simultaneously with the aid of IDEs & other tools.
Use SwA Tools Throughout the System Life Cycle

With the integration and automation of software assurance tools throughout the system life cycle, programs can make informed decisions on the identification and mitigation of risk.
MDA Software Assurance Approach

Phase 1 SwA Pilot Program in partnership with OSD
• Partnered with DoD Joint Federated Assurance Center (JFAC) Service Providers for SwA Risk Assessment Pilot Program
• Collaborated with Carnegie Mellon University Software Engineering Institute (CMU/SEI) to identify SwA Gaps in existing policy and guidelines

Phase 2
• Developed SwA Contract Language, MDA Policies and Guidelines based on identified gaps, BMDS Software Threat Models to support threat analysis and SwA Training for MDA personnel

Phase 3
• As of 22 March 2019, all MDA Programs under authority of the MDA Authorizing Official (AO) begin planning, budgeting, and incorporating SwA requirements into contracting efforts.
MDA Software Assurance Approach (cont.)

Ensure SwA requirements are on contract
- MDA Risk Management Framework enabled by SwA Overlay

Include SwA-specific CDRLs in ASSIST Database
- Software Assurance Evaluation Report
- Software Attack Surface Analysis Report
- Software Threat Analysis Report
- Vulnerability Assessment Report

Verify developer key SwA practices at milestone reviews
- SwA Entrance and Exit Criteria Added
  - Systems Requirements Review (SRR)
  - Preliminary Design Review (PDR)
  - Critical Design Review (CDR)
  - Test Readiness Review (TRR)

Note: A plus sign (“+”) indicates the baseline RMF control requirements specified in NIST SP 800-53 applies. The letter “E” indicates that there is a control extension for the applicable system type contained in the MDA SwA Overlay that applies. The blank cell indicates that the control is not required.
DoD systems continue evolution toward extreme reliance on software for execution of critical and tactical functionality.

The traditional application of “cyber protection” to DoD systems is sufficient to mitigate or prevent 10-to-16% of recorded cyberattacks.¹

Systems security engineering ensures implementation of SwA tools and practices from the start, transitions assurance requirements into sustainment, and is fundamental to ensure lethality of our weapons systems while under cyberattack.

Program’s, organization’s, and industry’s integration of more rigorous and robust software assurance engineering into their systems engineering process will reduce vulnerabilities and field more secure, reliable, and resilient weapons systems.

¹ Examples listed on following slide for reference.
Sample of Reporting on Cyber Attacks

- Is poor software development the biggest cyber threat?” CSO Online, September 2, 2015

- “Most Cyber Attacks Occur From This Common Vulnerability,” Forbes, March 10, 2015
  https://www.forbes.com/sites/sap/2015/03/10/most-cyber-attacks-occur-from-this-common-vulnerability/#af90ba57454d

  https://www.csiac.org/journal-article/engineering-software-assurance-into-weapons-systems-during-the-dod-acquisition-life-cycle/
For Additional Information

Mr. Thomas Hurt
Director, JFAC / Deputy Director, Software Assurance
Strategic Technology Protection and Exploitation
Office of the Under Secretary of Defense for Research and Engineering
571-372-6129
thomas.d.hurt.civ@mail.mil
**Technology: Binary Analysis Capabilities**

- **Execution of binary samples in an instrumented virtual environment using a combination of automated and manual testing to identify potentially malicious code.**
- **Compute complete control flow and remove dead code that can be used for exploitation.**

**Flowchart:**
- **Software Binary Sample**
  - **Unpacking**
  - **Static Analysis**
  - **Dynamic Analysis**
  - **Manual Analysis**

**Steps:**
1. **Unpacking of software executable to run static test and examine structural features of the binary sample.**
2. **Extract structuring and behavioral functions for analysis and include findings in risk report.**

*Source: AFLCMC TSN Lackland AFB*
Data Rights for SwA

Data Rights Requirements for Software Assurance
• Static analysis tools require source code
• Dynamic analysis tools require executable & environment
• Buildable source code & data rights required to have effective independent evaluation and competition of custom code.
• Need source code to be able to practically fix vulnerabilities (in most cases)

NDAA 2018 FY 2018 Sec. 871 Data Rights
As part of any negotiation for the acquisition of noncommercial computer software, the Secretary of Defense shall ensure that such negotiations consider, to the maximum extent practicable, acquisition, at the appropriate time in the life cycle of the noncommercial computer software, of all software and related materials necessary—
  (1) to reproduce, build, or recompile the software from original source code and required libraries;
  (2) to conduct required computer software testing; and
  (3) to deploy working computer software system binary files on relevant system hardware.