Priority R&D topics for high assurance

- *Develop more cost-effective methods for high assurance software development*. This includes by reference, at the least, "improved source scanning" and "correct by construction" methods.
- *Compose secure systems from independent secure components*.
- *Improve binary scanning tools*.
  - This is a subset of the points below involving scanning tools (improve source & object code scanners and automated security testing tools for non-malicious code; develop detectors of malicious code by simulation; and improve source & object code scanners for malicious code).
  - This is included in high assurance for several reasons, including for checking if malicious code has been introduced into the binary (when the source is available) as well as for handling COTS binary-only releases.
- *Truly trustworthy computing base*.
- *Develop methods to minimize/control the functionality of product*.

For medium and low assurance, the following were identified as the highest priorities:

- *Develop cost effective methods for improving software development*
- *Compose secure systems from independent secure components.*
- *Metrics and practical processes to measure them for software dependability and defensibility (safety, security, and survivability). In particular, an improved metric of exploitability, to answer questions such as "has assurance improved? which approach is better and by how much? How assured am I?" A plausible final answer would be "time to exploit."*
- *Devise mechanisms to detect/counter run-time vulnerability exploits.*

Development Processes

- *Identify and validate minimum subsets of organizational practices & process requirements for secure software development*.
- *Provide incentives for COTS vendors to seriously incorporate security concerns during development & sustainment*.

Scanning/ detection of security vulnerabilities

- *Improve source & object code scanners and automated security testing tools for non-malicious code*.
- *Develop detectors of malicious code by simulation*.
- *Improve source & object code scanners for malicious code*.
- *Improve binary code coverage tools.*
- *Automated pedigree analysis of a program.*

Scanning/ detection of security vulnerabilities

- *Improved detection of similarities between portions of a program.*
- *Improved detection of differences between portions of a program.*
- *Tools to identify susceptibility to denial-of-service attacks.*
- *Automated covert channel detection*.
- *Improve analysis tool interoperability.*

Countermeasures

- *Improve patch management*.
- *Improved mechanisms to simplify upgrading programs without restarting them, using special hardware, or complex software development.*
- *Devise mechanisms to detect/counter run-time vulnerability exploits*. *Improved self-detection or protection from vulnerabilities.*
- *Improve privilege & damage minimization*.
- *Develop methods to minimize/control the functionality of products.*
- *Develop methods to identify "code that does things not specified.*

Countermeasures

- *Developing a truly trustworthy computer base*.
- *Improved visualization techniques of security properties in existing systems.*
- *Develop security escorts*.
- *Develop "plug-in" approaches to control sharing of sensitive information in networked environments*.
- *Develop Denial of Service (DoS) countermeasures*.

Development Tools

- *Devise mechanisms for non-repudiable SCM*.
- *Devise safe & secure language subsets/extensions with automated checking*.
- *Devise new secure high-level languages and code generators*.
- *Automatic generation of artifacts by development tools that can be provided to end-users or evaluators to support assurance.*
- *Assurance of development tools*.

Application Environment

- *Improve GUI security*.
- *Improved methods of developing secure GUI programs so they can be effectively analyzed.*
- *Devise improved infrastructure mechanisms for application security*.

Requirement/Design/Validation

- *Improved mechanisms for specifying security properties.*
- *Improved modeling*.
- *Improved approaches/patterns for secure architecture/design*.
- *Improve anti-reverse engineering mechanisms*.
- *Security with Privacy.*
- *Effective methods of tagging data in a distributed, heterogenous, cross-domain environment with high granularity.*
- Software self-attribution.

Education

- *Identify and validate individual knowledge requirements to develop secure software*.
- *Promulgate existing knowledge in how to develop secure software, including high assurance software*.
- *Develop, validate, and provide incentives for secure software development curricula in CS/SE programs*.

Secure Kernels