Advances in Software Security Assessment and Verification Research

David Gilliam and John D. Powell

CSMIISS
JPL IT Symposium
Pasadena CA
November 4, 2002
Acknowledgments

• This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

• The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program lead at the NASA Software IV&V Facility.

• This activity is managed locally at JPL through the Assurance and Technology Program Office.
Research Goal

- Reduce security risk to the computing environment by mitigating vulnerabilities in the software development and maintenance life cycles

- Provide an instrument and tools to help avoid vulnerabilities and exposures in software

- To aid in complying with NASA NPG 2210 and NPG 2810
Security Instrument Components

- Software Security Assessment Instrument
  - Software Security Checklist (SSC)
  - Model-based security specification and validation approach (MBV)
- Flexible Modeling Framework Instrument (FMF)
Security Instrument Components (cont.)

- Vulnerability Matrix
  - Categorize vulnerabilities and group them
  - Vulnerability properties
- Software property-based testing tool (PBT)
  - Developed for testing JAVA code
  - Developing testing instrument for C code
- Other Security Assessment Tools (SATs)
  - List of security tools for assessing software and systems, purpose, use, and alternatives
Software Security Checklist (SSC)

- NASA NPG 2210 and NPG 2810
  - External Release of Software
  - Software Security Checklist

- Two phases
  - Phase 1: Provide instrument to verify external release of NASA software does not present a security risk to NASA and its partners
  - Phase 2: Provide instrument to integrate security as a formal approach to the software life cycle
Software Security Checklist (cont.)

• Phase 1
  – Checklist for use prior to external release of software
    • Concerns include the following:
      – HR information
      – Operational processes
      – Internal processes
      – Sensitive IP addresses, Host names, etc.
      – Crypto Certificates and Keys
      – Embedded userids and passwords
    • Document describing purpose and use of the checklist
Software Security Checklist (cont.)

– Proposed procedure for using the checklist
– Examples for automating software checking to the greatest extent possible
  • Perl script for checking strings with IP addresses
  • Perl script for checking dangerous subroutines
    – gets, fgets, strcpy, strcat, sprintf, printf, and fprintf, because of format string errors
    – strncpy and strncat because of the oft-forgotten NUL byte problem, and possibly others
  • The specific set of functions is system dependent; for example, some Windows functions require parameter checks that UNIX versions would not, i.e., "system" is not a function that works on Windows
Software Security Checklist (cont.)

• Phase 2
  – Integrating Security in the Software Lifecycle
    • Analyze and prioritize what items are of concern
      – Cost of life cycle (Risk cost grows exponentially throughout software life cycle)
      – Cost of where problems occur
      – Security risk analysis
    • Determine and prioritize most effective solutions for addressing the concerns
      – Need for software security for architect?
Software Security Checklist (cont.)

- Develop Checklists for use through the software life cycle process
- Develop Document describing purpose and use of the checklists
- Develop or Provide list of Tools/Instruments that can assist in automation of checks
  - these tools help a good analyst, but they cannot replace him or her (Halting problem, etc.)
- Provide Recommendations and lessons learned for software security architecture
Instrument Integration

- SCC Guides the process of identifying security needs of software before release
- Remaining parts of the security instrument provide analysts one means of support in addressing security needs.
  - VMatrix known vulnerabilities
  - PBT code level discovery of known/new vulnerabilities with feedback to Vmatrix and MBV.
  - MBV provides early lifecycle discovery of known/new vulnerability scenarios with feedback to Vmatrix and PBT
- Improved traceability through an integrated approach

November 4, 2002
Conclusion

- Assessment Instrument composed of four tools and reports:
  - Vulnerability Matrix (VMatrix)
  - Tester’s Assistant (PBT)
  - Flexible Modeling Framework (MBV, FMF)
  - Software Security Checklist (SSC)
- Tools can be used collectively or individually
- There is a potential for wider application of the instrument beyond assessment of security of software

November 4, 2002

FOR MORE INFO...

David Gilliam
JPL
400 Oak Grove Dr., MS 144-210
Pasadena, CA 91109
Phone: (818) 354-0900       FAX: (818) 393-1377
Email: david.p.gilliam@jpl.nasa.gov
Website: http://security.jpl.nasa.gov/rssr/

John Powell
MS 125-233
Phone: (818) 393-1377
Email: john.d.powell@jpl.nasa.gov