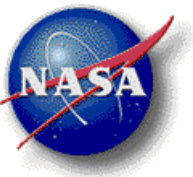


JPL

SAS '05

Reducing Software Security Risk through an Integrated Approach

David P. Gilliam, John D. Powell

Jet Propulsion Laboratory,
California Institute of Technology

Matt Bishop
University of California, Davis

Acknowledgement

- **NOTE:**
  - **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 by the NASA Software IV&V Facility**
  - **This activity is managed locally at JPL through the Assurance and Technology Program Office**

Current Collaborators

- David Gilliam – Principle Investigator, JPL
- John Powell – JPL Software Engineer
- Josef Sherif – JPL Software Security Engineer
- Matt Bishop –Professor of Computer Science, University of California at Davis

- http://rssr.jpl.nasa.gov

Agenda

- Goal
- Problem
- Approach
- Verification of PatchLink & Results
- Importance/Benefits
- Future

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 security requirements and appropriate best practices

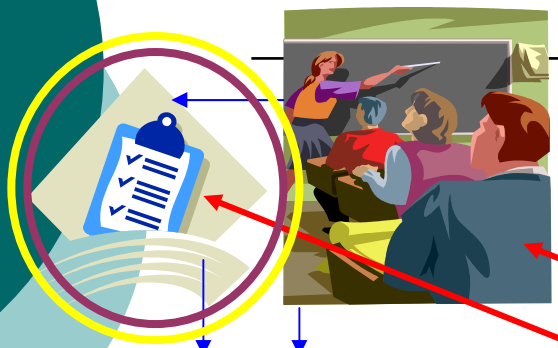
Problem

- Cost of Fixing Security Weaknesses in Software and Systems Is Expensive
- Security Weaknesses Can Lead to Loss / Corruption / Disclosure / Availability of DATA and Systems Impacting Missions
- Poor Security Requirements
- Poor System Engineering
  - Leads to poor design, coding, and testing
- Cycle of Penetrate and Patch
- Piecemeal Approach to Security Assurance

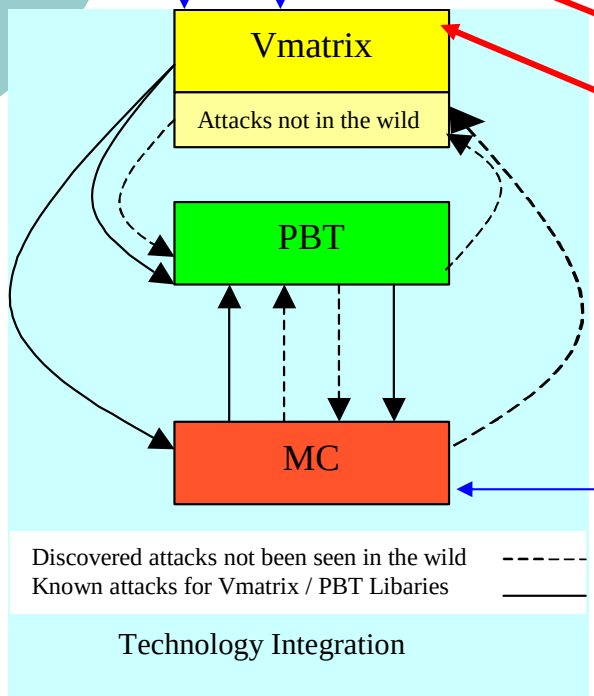
Approach

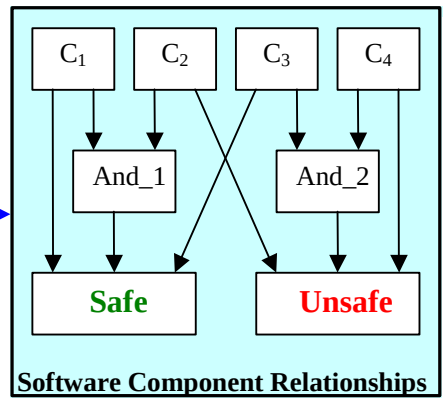
- Develop a Software Security Assessment Instrument for the Life Cycle
  - Several Foci
    - Training/Education
    - Security Checklist for the Life Cycle
    - Application of Lightweight Formal Verification Techniques for Security Weaknesses in Code and Systems

Reducing Software Security Risk Through an Integrated Approach



- **Software Vulnerabilities Expose IT Systems and Infrastructure to Security Risks**
- **Goal: Reduce Security Risk in Software and Protect IT Systems, Data, and Infrastructure**
  - Security Training for System Engineers and Developers
  - Software Security Checklist for end-to-end life cycle
  - Software Security Assessment Instrument (SSAI)





- **Security Instrument Includes:**
  - Model-Based Verification
  - Property-Based Testing
  - Security Checklist
  - Vulnerability Matrix
  - Collection of security tools

Inception-to-Retirement Process

- Coincides with Organizational Polices and Requirements
- Security Risk Mitigation Process in the Software Lifecycle
- Software Lifecycle Integration
  - Training
  - Software Security Checklists
    - Phase 1: Security Checklist for Life Cycle
    - Phase 2: Security Checklist and Process for External Release of Software
  - Security Assurance Instruments:
    - Early Development – Model Checking / FMF
    - Implementation – Property Based Testing
    - Prototyped on PatchLink Unix Agent Software

Importance/Benefits

- Enhances a Secure Trusted Network Environment
- Reduces Cost of Maintenance
- Reduces Loss or Destruction of DATA and Systems
- Improves NASA's Overall Security Posture
  - Fewer Intrusions and Audit Findings Leads to a Better Image (OMB & Public)

Current Work

- Prototype of SSAI Techniques on PatchLink Unix Agent
  - In Use at NASA-Wide for Reporting System Patch Status to NASA HQ
    - Part of NASA's Report to OMB
    - PatchLink Used to Report Patch Compliance
    - PatchLink Used to Report Compliance with IT System Security Benchmarks
  - Report Requested by NASA CIO
  - Report Submitted to IV&V Center
- PatchLink Vendor is Modifying Code to Address Findings in Submitted Report
  - PatchLink Provided Responses to Findings
  - A Positive Example of Vendor and Customer Working Together to Increase Security Using SSAI Techniques

Current Work (Cont.)

- Model-Based Verification (MBV)
  - Verification of critical security properties for the design of the Patchlink agent
    - Traditional Model Checking (MC)
      - Design was verified to uphold the properties for the system design as a whole.
    - (MC) with Flexible Modeling Framework (FMF)
      - Warning: potential denial of service (DOS) attacks
        - Component designs susceptible to DOS
        - Complex interaction with between multiple components (“subsystems”) mitigates risk
      - Result: Risk mitigation recommendation to the development team* for implementation
        - Rigorous functional verification, against the design, of “weak and mitigating” components
        - Additional verification of security properties for the implementation of interfaces involved in the “complex interactions” that mitigate DOS attacks

*** Patchlink was a case study where an implementation already existed. Therefore a formal risk mitigation recommendation for development was not delivered but the operational team at Patchlink team was made aware of the results for future use should the current implementation fall prey to a DOS attack.**

Current Work (Cont.)

- Validation results of the FMF approach
  - FMF provided risk mitigation recommendations identifying areas that will likely need attention in later phases. Areas where:
    - Additional testing will be needed because minor deviations from design details pose serious security risks. (Patchlink: “weak” components)
    - Testing strategies that addresses the critical function derived from some specific complex component interactions. (Patchlink: Interaction Mitigating DOS)
      - Often found to be undocumented derived interface requirements
        - Documentation will be critical during maintenance
  - FMF produce a flexible/adaptable model of system component interaction issues and constraints exists for later use. (Patchlink: Use of PBT in conjunction with FMF) Uses such as:
    - Identification of system interaction anomalies well in advance of system integration testing. (during design)
    - Reducing the risk of introducing security vulnerabilities during software changes (i.e. enhancement, repair, reconfiguration, maintenance) via component dependencies captured in the model
    - Easily reusing security properties in the implementation and testing phase with PBT
  - Over the full system FMF results are consistent with traditional MC

Example Analysis

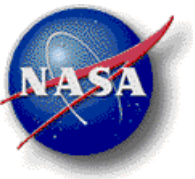
- Property-Based Testing (PBT)
- Goal: Verify updates
  - Translates to “check that the CRC checksum is validated before copying”
  - Copying uses routine “copyFile”
  - Checking done in two places
    - checkCRC
    - decompress_file

Result

- Confirmed that the invariant holds for all cases that the data exercised
  - Numerous test cases run
  - No formal path analysis done, but tests appeared complete

Other Properties Tested

- Run client at lower priority
  - Did this by hand, as it occurs in a shell script
- Listen and respond only to client-initiated connections
  - Found out there was one case in which this was not true
  - Turned out to be a known situation that was not a security problem



JPL

Results

- Some properties tested
- Some properties written but not tested
- Some problems with instrumenter that did not appear in UCD tests
  - All being fixed; none affected testing
- No security problems identified
  - But one property had to be restructured to take into account an expected interaction not explained before the property was written

Other Work

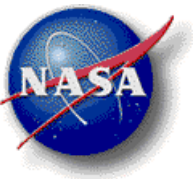
- Training Presentation Currently in Draft
- Provide to Software Quality Improvement (SQI) as a Course
  - Project Management Course
  - System Engineer / Developer Course

Relevance to NASA Accomplishments

- Increases NASA's Security Reliability of Systems and Software
- Helps to Prevent Negative Public Exposure Due to Security Breach
- Prototyped the SSAI Instrument on PatchLink Agents
  - Used large scale across NASA on its systems
  - Findings leading to improved vendor product

Next steps

- Integrate the Overall SSAI Process in the Project Life Cycle at NASA Centers
- Continue Using SSC in Life Cycle and External Release
- Begin Teaching Security as Part of Life Cycle Curriculum
  - Project Managers
  - System Engineers and Developers



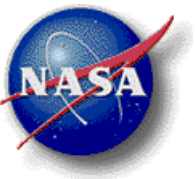
JPL

FOR MORE INFO... ***Web Site: http://rssr.jpl.nasa.gov/***

David Gilliam, JPL
400 Oak Grove Dr., MS 144-210
Pasadena, CA 91109
Phone: (818) 354-0900
Email: david.p.gilliam@jpl.nasa.gov

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

Matt Bishop, UC Davis
Department of Computer Science
Kemper Hall
phone: +1 (530) 752-8060
fax: +1 (530) 752-4767
email: bishop@cs.ucdavis.edu

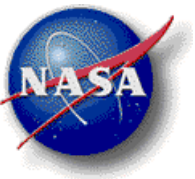


JPL

QUESTIONS?

? ? ?

Examples and Backup Slides

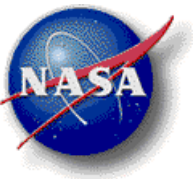


Properties

```
location funcall com.siusoft.lib.PLCRC::checkCRC(String fileName,
    String CRC) returns x if "x == 0" {
        assert fileok2(fileName, crc);
}

location funcall
    com.siusoft.lib.SiuCompress::cx_decompress_file(String dst,
        String src, PLCRC crc) returns x if "x != 'a'" {
            assert fileok(dst, src, crc);
}

location funcall com.siusoft.lib.SiuFile::copyFile(String source,
     String dest) {
        assert copyfile(source, dest);
}
```



Invariant

- True if updates verified before file copied

```
(fileok(x, y, z) and copyfile(a, y)) or
    (fileok2(x1, y1) and copyfile(a1, x1))
```

Model Checking & The Flexible Modeling Framework

- **MC of FMF combinations allows partial answers to otherwise intractable system state spaces**
- **MC with FMF Benefits Software in Early Lifecycle**
  - Earlier Discovery of Software Errors
  - Correction is easier / less expensive
  - Modeling in FMF components compatible with the software development process
  - Modular model design allows easy extension of existing models
    - Multiple client scenarios for the server login example were quickly modeled and verified
    - The various client scenarios allows extensive off-nominal verification with ease
  - Rapidly changing requirements and designs
    - Multiple design trade offs in login protocol were easily explored

Model Checking & The Flexible Modeling Framework – Server Login Model

- Varying levels of detail were defined for different system parts
  - Multiple login failure propagation scenarios
    - known but not formally defined for different systems using the login protocol
  - Model extensions are readily possible for many if not all of these scenarios
    - **Developed quickly**
    - **Adapted at will**
    - **Cross tested against**
      - **Client scenarios**
      - **Protocol design trades**

Property-Based Testing

- Property-based testing tool – Tester's Assistant (Matt Bishop, UC Davis)
  - Perform code slicing on applications for properties for a known set of vulnerabilities
  - Test for vulnerabilities in code on the system or whenever the computing environment changes
  - Initially, checks software developed in JAVA and C
    - The goal is to have the tool check other programming and scripting languages as well (C++, Perl, ActiveX, etc.)

Software Security Checklist (SSC)

- Two Phases
  - Phase 1:
    - Provide instrument to integrate security as a formal approach to the software life cycle
    - Requirements Driven
    - Pre-Requirements
      - Understand the Problem and Scope
    - Requirements Gathering and Elicitation
      - Be Aware of Applicable Requirements Documents
      - Provide Trace to External Requirements Docs

SSC (Cont.)

- Phase 2:
  - External Release
    - Release Process
    - Areas for Protection:
      - Protect People
      - Protect ITAR and EAR
      - Protect Trade Secrets – Patents
      - Protect Organizational Resources
    - Considerations
      - Insecure Subsystem Calls
      - Embedded IP Addresses or Phone Numbers
    - Web Site for Questions and Tools for Code Checking

SSC (Cont.)

- Phase 2 Checklist and Process in Use at JPL