The Open Source Security Myth — And How to Make it A Reality

Michael Davis

Dynamic Security Concepts, Incorporated

Disclaimer

- DSCI's Sponsorship of this speaker does not necessarily express or imply approval or endorsement of the speaker's views.

- DSCI makes no promises or warranties of any kind, express or implied, including those of merchantability and fitness for a particular purpose, as to the content of this presentation. In no event shall DSCI be liable for any damages resulting from use of this presentation even if DSCI has been informed of the possibility of such liability

Agenda

- Introduction
- Benefits – Up Front
- Functional versus Assurance
- Assurance Model (ISO 15408)
- Development in Depth
- The Path Ahead
- Summary

Introduction

- Presentation Title – confessions
- Information Availability ≠ Security
- Not just an Open Source problem
- Benefits of Open Source Community
- Software Lifecycle
- Mistakes – also known as
- Status Quo
- Good Examples

Benefits – Up Front

- Improve Open Source Software (OSS) development
  - Lower the bar for project participation
  - Communicate design decisions
  - Communicate alternatives considered, but not implemented
- Assess Assurance
  - Gain confidence that the OSS does what it claims
  - Gain confidence that the OSS does not do what it isn't supposed "to do"

Functional versus Assurance

- Functional Requirements and Statements
- Assurance Requirements and Statements

- The Common Criteria Model (ISO 15408)
  - CC Functional Requirements and Statements
  - CC Assurance Requirements and Statements
  - CC Evaluation Methodology (CEM)
    - Roles
    - Requirements

Assurance Model (ISO 15408)

- Configuration Management
- Delivery and Operation
- **Development**
- Guidance Documents
- Life Cycle Support
- Tests
- Vulnerability Assessment

Configuration Management

- Defined: Configuration management (CM) helps to ensure that the integrity of the TOE is preserved, by requiring discipline and control in the processes of refinement and modification of the TOE and other related information. CM prevents unauthorized modifications, additions, or deletions to the TOE, thus providing assurance that the TOE and documentation used for evaluation are the ones prepared for distribution.

- Open Source: *de facto* Standard is CVS

https://www.cvshome.org/

Delivery and Operation

- Defined: Delivery and operation defines requirements for the measures, procedures, and standards concerned with secure delivery, installation, and operational use of the TOE, ensuring that the security protection offered by the TOE is not compromised during transfer, installation, start-up, and operation.

- Open Source: Some use of PGP

Development

- **Defined:** Development defines requirements for the stepwise refinement of the TSF from the TOE summary specification in the ST down to the actual implementation. Each of the resulting TSF representations provide information to help the evaluator determine whether the functional requirements of the TOE have been met.

- **Open Source:** Status Quo is high level functional description and comments in the source code.

Guidance Documents

- Defined: Guidance documents defines requirements directed at the understandability, coverage and completeness of the operational documentation provided by the developer. This documentation, which provides two categories of information, for users and for administrators, is an important factor in the secure operation of the TOE.

- Open Source: Varies significantly by project with architecture, system administrator, technical user, and “typical” user

Life Cycle Support

- **Defined:** Life cycle support defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment.

- **Open Source:** Ad Hoc

Tests

- Defined: Tests states testing requirements that demonstrate that the TSF satisfies the TOE security functional requirements.

- Open Source: Caveat Emptor – but what should expect for nothing?

Vulnerability Assessment

- **Defined:** Vulnerability assessment defines requirements directed at the identification of exploitable vulnerabilities. Specifically, it addresses those vulnerabilities introduced in the construction, operation, misuse, or incorrect configuration of the TOE.

- **Open Source:** ?

Subject for DEFCON 13?

Development in Depth

- What is the Requirement?
  - Capture the Design philosophy
  - Traceability
    - Functional Description → HLD
    - HLD → LLD (Implementation)

- "Tools"

Don’t Fake It

Don’t:

- Paste pseudo code before real code
- Bury comments that address configuration management issues (e.g. This “fixes” that)
- Include comments that address the obvious
- ***LEAVE*** Design and Code out of sync
- Assume that well commented, modular code is enough

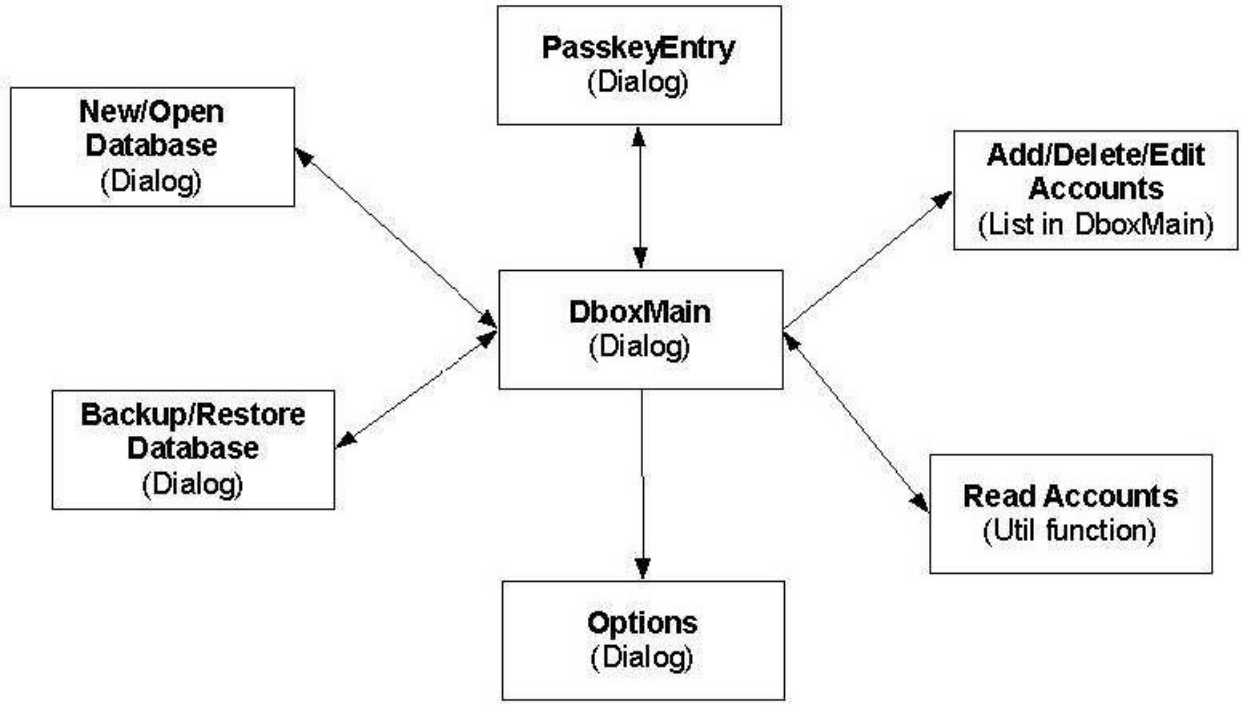
Documentation Requirements

- The requirement is to understand flow and purpose of “what is there”.
- A security review needs traceability between functional description and HLD and between HLD and LLD (Implementation)
- Capture the Design philosophy! (Overall Architecture and subsystem design documents)
- Document alternatives considered
- Vulnerabilities are (traceable) to the boundaries. Therefore: (1) Identify the boundaries, and (2) Describe protection (bounds checking, error cases, etc.)

Documentation Tools

- Text (e.g. Mozilla)
- Source Code comments
- Parsing Source Code comments
- Flow Charts
- UML – Static Structure (Class Diagrams)
- UML – Interaction Diagram
- UML – Sequence Diagram
- FSM and State Charts

Password Safe



- Andrew Mullican (now a Developer)
- Software Patterns Analysis

Michael Davis Michael@WvDavis.us
609-625-3942
mDavis@DSCinc.net www.DSCinc.net

Devil's Advocate

Commercial Developers are more likely to develop more useful documentation from high-level down to meaningful source comments:

- Market driven to develop code correctly
- Customer demand for reliability
- Requirement to make code "approachable"

The Path Ahead

- The OpenSSL Project
- Improved comments in the source code must be the foundation for progress:
  - The current work is focus on the source code.
  - Tools are available – parsing source code comments (e.g. JavaDocs)
- Effort driven from the bottom up

Summary

- What I don't know:
  - How to motivate developers to write more documentation
  - How much is enough
- What I do know:
  - Tools are available
  - Open exchange of information is essential
  - Feedback priorities