

Linux Achieves CAPP/EAL4+

Can it achieve EAL5?

Doc Shankar, IBM

Helmut Kurth, atsec

Agenda

- What has Linux achieved so far?
- Open Source and EAL5
- Challenges for EAL5
- Roadmap to EAL5
- Remaining problems

CC evaluations of Linux so far

Product	Hardware	Kernel	PP	Assurance Level	Evaluator	Certifying Body	Application Date	Certification Date
SLES 8	xSeries® 335	2.4	ST	EAL 2+	atsec	BSI	02/03	08/03
SLES 8 SP3	xSeries 335, pSeries® 630, iSeries® 825, zSeries® 900, eServer® 325	2.4	CAPP	EAL3+	atsec	BSI	07/03	01/04
RHEL3 UP2	xSeries 335 AS/WS, pSeries 630 AS iSeries 825 AS, zSeries 990 AS, eServer 325 AS	2.4	CAPP	EAL3+	atsec	BSI	03/04	07/04
SLES 9	xSeries model x335 machine type 8676 pSeries model 520 machine type 9111 (LPAR SF220_049) iSeries model 520 machine type (9406) (OS/400® V5R3 LPAR) zSeries 990, eServer 325	2.6	CAPP	EAL4+	atsec	BSI	03/04	03/05
RHEL4 UP1	xSeries model x336 machine type 8837 (AS/WS) pSeries model 550 machine type 9124 with pSeries LPAR (AS only) iSeries model 550 machine type 9406 with OS/400 v5R3 LPAR (AS only) zSeries z/VM 5.1 Logical Partition (AS only) eServer model 326 based on the AMD 64 (Opteron) processor machine type 8848 (AS only)	2.6	CAPP	EAL4+	atsec	NIAP	02/05	10/05
SLES 8 SP3	Range of HP Pentium, Xeon and Itanium based systems	2.4	CAPP	EAL3+	atsec	BSI	04/04	09/04
RHEL 3 UP3	Range of HP Pentium, Xeon and Itanium based systems	2.4	CAPP	EAL3+	atsec	BSI	04/04	09/04

Parties involved in the evaluations (Sponsored by IBM)

- IBM:
 - Sponsor the project, project management, and coordination
 - Codevelop the audit subsystem
 - Develop design documentation (FS, HLD, LLD)
 - Develop test cases and test plan
 - Conduct developer testing
 - Document development/security procedures (i.e. Configuration Management for test suites, document control, and test results)
 - Produce Vulnerability Assessment Report
- Distributors – SUSE & Red Hat:
 - Codevelop the audit subsystem
 - Update development and security procedures documentation
- atsec:
 - Codevelop the evaluation strategy
 - Provide guidance documents and a configuration script
 - Perform the evaluation
- Certifying Bodies - BSI & NIAP:
 - Supervise the evaluation and issue the certificate

Linux and Common Criteria

- Until 2003, many people believed that Linux would not be able to get CC certified
- Now, two years later, no other operating system has got more Common Criteria certificates than Linux
 - Two distributions (Novell SUSE and Red Hat)
 - Two different kernel versions (2.4 and 2.6)
 - Many different hardware platforms
 - IBM Pentium, XEON, and Opteron systems
 - IBM pSeries, iSeries, and zSeries systems
 - HP Pentium, XEON, and Itanium systems
 - SGI Itanium systems
 - Assurance levels up to EAL4 augmented by ALC_FLR.3

Challenges for CAPP/EAL4

- New functionality
 - Now kernel version 2.6 (was 2.4 in the previous evaluations)
 - New design of the kernel audit functions
- Low-level design
 - Required for the kernel and all trusted processes
 - Large documents focusing on the security functions
 - Describing the details of the 2.6 kernel and trusted processes
- Additional vulnerability analysis
 - More in-depth analysis
 - Penetration testing
 - Crypto/Keygen/RNG/Primality tests
- Impacts on the SUSE development processes
 - Enhancements in flaw remediation
 - Acceptance procedures

Evaluation evidence open sourced

- Functional Specification*
 - Man pages existed, but not for all system calls and configuration files.
 - Additional man pages have been developed.
- High Level Design*
 - Very good general material and books exists, but partly not up-to-date and not focused on security
 - a new security focused High Level Design has been developed
- User Documentation*
 - Some very good security related documents and books exist, but they are generic and not dedicated to a specific distribution.
 - An additional Security Guide has been developed.
- Test Documentation**
 - Test cases for security functions didn't exist, so a comprehensive set of tests were developed for each assurance level.

Linux® now has a good starting point for further evaluations, and for the evaluation of other distributions.

* http://www-124.ibm.com/linux/pubs/?topic_id=5

** <http://ltp.sourceforge.net/EAL3.html>

EAL5 overview (1)

- Configuration Management:
 - ACM_SCP.2 versus ACM_SCP.3
 - Development tools under configuration management
- Delivery & Operations:
 - No difference
- Development:
 - ADV_FSP.2 versus ADV_FSP.3
 - Semiformal functional specification
 - ADV_HLD.2 versus ADV_HLD.3
 - Semiformal high level design
 - ADV_IMP.1 versus ADV_IMP.2
 - Full implementation representation (source code)
 - ADV_INT.1
 - Modular design with minimized interaction between the modules (largely independent modules)
 - ADV_RCR.1 versus ADV_RCR.2
 - Semiformal correspondence between Functional Specification and High Level Design
 - ADV_SPM.1 versus ADV_SPM.3
 - Formal security policy model

EAL5 overview (2)

- Guidance
 - No change
- Life Cycle
 - ALC_LCD.1 versus ALC_LCD.2
 - Use of a “standardized life cycle model”
 - ALC_TAT.1 versus ALC_TAT.2
 - Description and use of “implementation standards”
- Tests
 - ATE_DPT.1 versus ATE_DPT.2
 - Testing at interfaces defined in the low-level design
- Vulnerability Assessment
 - AVA_CCA.1
 - Informal covert channel analysis
 - AVA_VLA.2 versus AVA_VLA.3
 - Systematic vulnerability analysis

What needs to be done for EAL5

- Work to be done beyond EAL4 is identified in the next slides
- Color codes used:
 - Green: No or almost no work required.
 - Blue: Little work required. Not a major effort.
 - Orange: Significant work required, but doable
 - Red: Potential showstopper. Problem either can not be overcome or requires too much work to solve.

Challenges for EAL5 (1)

- Configuration management issues
 - Development tools under configuration management
 - Easily covered. Development tools are also Open Source and managed in the same way as Linux.
- Development
 - Semiformal functional specification
 - Man pages can be argued to be mostly semiformal. Limited additional work required.
 - Semiformal high-level design
 - Current high level design is informal. Some major effort required to identify a useful semiformal notation and augment the current high-level design.
 - Implementation representation
 - Full source code is available.
 - Internal structure
 - Kernel has modular design but minimization of inter-module interactions was not a design criterion. Restructuring of the kernel is not possible (this is by far too much work and will not be accepted by the kernel maintainers).

Challenges for EAL5 (2)

- Development
 - Semiformal correspondence between functional specification and high-level design
 - Can be developed together with the semiformal high-level design.
 - Formal Security Policy Model
 - Can be done. Some work, but not too difficult.
- Life Cycle
 - “Standardized life-cycle model”
 - Is the Open Source development a “standardized life cycle model”?
 - Use of “implementation standards”
 - Does the Linux development have “implementation standards”?
- Testing
 - Testing at internal module interfaces
 - Possible but requires significant work. Additional debugging tools may be required.

Challenges for EAL5 (3)

- Vulnerability analysis
 - Informal Covert Channel Analysis
 - Required for leaking of passwords and cryptographic keys. Can be based on some existing assessments.
 - Systematic vulnerability analysis
 - Doable, but requires a significant effort. Part of it can be done using existing source code analysis tools and penetration testing tools. Additional tools may be required to develop.

Summary of Challenges

- The main challenges are:
 - Development of a semi-formal high-level design
 - Testing at the internal module interfaces
 - Systematic vulnerability analysis
- Open issues are
 - Is the design “good enough” with respect to modularity and minimization of inter-module dependencies?
 - Does the life-cycle model follow a “standardized life-cycle model”?
 - Does the implementation make use of “implementation standards”?

The Open Issues in more Detail

- CC requirement for “modularity and minimization of module interaction and inter-dependencies”
 - *This family addresses the internal structure of the TSF. Requirements are presented for modularity, layering (to separate levels of abstraction and minimise circular dependencies), minimization of the complexity of policy enforcement mechanisms, and the minimization of the amount of non-TSP-enforcing functionality within the TSF — thus resulting in a TSF that is simple enough to be analyzed. (CC, part 2, para 337)*
 - The Linux kernel is structured into modules. Still, the complexity is quite high and the interactions between modules are large and complex. Also, some trusted programs show a high complexity in the interaction of their modules.
 - The requirements for modularity and minimization of complexity have been also stated in the TCSEC starting from class B2. The Linux kernel as it is today would probably not be compliant with the TCSEC requirements for modularity at the B2 level.

The open issues in more detail

- CC requirement for “standardized life-cycle model”
 - *A standardized life-cycle model is a model that has been approved by some group of experts (e.g. academic experts, standards bodies). (CC, part 2, para 397)*
 - The Open Source development model has been extensively studied and is now accepted by academia as a useful and successful life-cycle model.

This problem seems to be solved!

The open issues in more detail

- CC requirement for “use of implementation standards”
 - *ALC_TAT.2.3D The developer shall describe the implementation standards to be applied.*
 - The CC never define the term “implementation standard”!
 - Is it “coding style”?
 - See the “Linux kernel coding style” document.
 - Other aspects?
 - See the “Linux kernel management style” document.
 - Those documents cover the kernel, they are not necessarily applied for trusted programs.
 - Kernel coverage should be sufficient to satisfy ALC_TAT.2.3 (full coverage of the TOE is required starting with EAL6).

Remaining Open Issue

Is the complexity of the Linux kernel
“small enough to be analyzed”?

(often neglected third requirement of the Reference Monitor principle)

There is no clear answer to this question!

Summary

- If the structure of the Linux kernel satisfies the requirements of ADV_INT.1, EAL5 is possible.
- Significant work is required with respect to:
 - Development of a semi-formal high-level design
 - Development of a test strategy, test plan and test cases for testing at module interfaces of the low-level design
 - Performing a systematic vulnerability analysis
- Other problems are minor and can be solved easily.

Other Issues

- EAL5 is not covered by the CCRA
 - Certificate will not be accepted world-wide
- EAL5 is not covered by the CEM
 - Evaluation methodology differs with different schemes
- No general-purpose operating system has been evaluated at EAL5 so far
 - Smart card products (chips and smart card operating systems)
 - Logical partitioning system (IBM PR/SM)
 - All of them are significantly less complex than the Linux kernel

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of IBM.

SUSE and its logo are registered trademarks of Novell.

IBM, IBM logo, AIX, AS/400, eServer, xSeries, iSeries, pSeries, zSeries, are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the US, other countries or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.