Certifying Open Source – The Linux Experience

K.S.(Doc) Shankar, IBM
Helmut Kurth, atsec Information Security

Abstract

As far as we know, no Open Source program has received a Common Criteria security certification—until now. Although some people believed that it was not possible for an Open Source program to receive a Common Criteria security certification, IBM and atsec have proven otherwise by obtaining a Common Criteria security certification for Linux.

It is also generally believed that security certifications are time consuming and can take years to accomplish. We were able to obtain the Common Criteria certification of Linux in four months. This paper discusses our experience with the certification process and our future plans.

Introduction

There has been considerable pressure from Linux® customers for Linux to obtain some form of security certification. All of Linux’s commercial competitors have already obtained certification. As Linux continues to penetrate the marketplace, customers, especially the government sector, have raised concerns regarding Linux security. Therefore, in order for Linux to continue to aggressively penetrate enterprises, some form of security certification is absolutely necessary. For these reasons, and after a careful analysis, IBM decided to sponsor a Common Criteria-based security certification for Linux [1].

The Common Criteria are composed of ISO standard (ISO 15408) for the security analysis of IT products. The governments of 18 nations officially adopt the Common Criteria, including the United States, Canada, Germany, France, and the UK. Since July 2002, the United States has required a Common Criteria or FIPS 140 certification for all IT-products used for the processing of security-critical data within US government systems.

This paper examines the following aspects of Linux Open Source and the Common Criteria certification process:

- Identifying security implications of Open Source code
- Determining which Evaluation Assurance Level (EAL) to achieve
Identifying security implications of Open Source code

An Open Source program is a program whose license gives users the ability to run the program for any purpose, to study and modify the program, and to redistribute copies of either the original program or the modified program (without having to pay royalties to previous developers). Two of the most visible Open Source projects are the Linux operating system and the Apache web server. Although this paper focuses on Linux, most of the strategies we described are applicable to all Open Source software.

Linux is a member of the large family of UNIX®-like operating systems including those from AT&T, Digital, IBM, HP, Sun, and others. Linux, however, is not a commercial operating system. Any one vendor does not control it. Linux was initially developed by Linus Torvalds [2] in 1991 for the IBM® PC based on the Intel® microcomputer. Today, however, Linux runs on just about any hardware platform. In addition, Linux has inspired the development of countless free software programs, including Apache, MySQL, Python, Perl, Samba, GNOME, Open SSL, Open SSH, Open LDAP, Open Source implementations of Kerberos, and Free S/WAN.

Since the beginning of Open Source development, debate has raged over whether Open Source software is more secure than other software, and there have been no definite conclusions. The fact remains that Open Source software gives attackers and defenders the same advantage [3]. If a defender does nothing about security, Open Source software essentially gives that advantage away to the attacker. However, Open Source software also offers great advantages to the defender by providing access to security techniques and knowledge that are normally not available with closed-source software. For example, most programmers who contribute to Open Source software take extra precautions to ensure their code is secure, as their reputations are at stake. Open Source code is subject to a community-wide review and audit (on the other hand, simply publishing code does not guarantee that people will examine the code for security flaws). Another advantage of Open Source software is the speed with which patches become available when bugs are found. Many Open Source developers consider it a personal challenge to develop fixes and release them to the Open Source community, sometimes within a few hours. This speed is unheard of in closed-source development. In summary, while there are many advantages to Open Source software and it is generally considered a good thing, Open Source does not guarantee security. Linux, however, has been scrutinized by many very good security engineers with regard to its trust and confidence.
Determining which Evaluation Assurance Level (EAL) to achieve

The Common Criteria [4] allow for a number of assurance levels from which to choose. The assurance levels range from EAL1 (the lowest) to EAL7 (the highest). Many of the commercial operating systems with which Linux competes have been certified at EAL4.

IBM and atsec first looked at obtaining an EAL4 certification. Although the EAL4 security level was technically attainable, it would have taken a long time to achieve. Given the marketing pressure, the objective was to demonstrate that Linux could be certified at a meaningful level fairly quickly. EAL3 certification was considered next. Because life cycle assurance requirements start at EAL3, meeting those requirements would be difficult for Open Source code in the first step of certification. Another consideration was that we did not want to develop any significant software for security functionality that would further slow down the certification effort. In other words, IBM and atsec wanted to certify Linux “as-is.” Therefore, as a start, we chose a current Linux release as the “Target of Evaluation” (TOE) and selected EAL2 as our first step. EAL2 can be augmented further to obtain higher assurance levels and meet the requirements for more elaborate Protection Profiles.

Identifying EAL2 and Security Target (ST) requirements

EAL2 provides assurance through an analysis of the operating system’s security functions. The EAL2 certification process involves analyzing the operating system’s functional and interface specifications, guidance documentation, and high-level design to understand the operating system’s security behavior. In addition, the following supports the analysis:

- Independent testing of the security functions
- Evidence of developer testing based on functional specification
- Selective independent confirmation of the developer test results
- Strength of function analysis
- Evidence of developer search for obvious vulnerabilities
- Verification that all the documents on the functional specification, high level design, and guidance are consistent with actual operation of the system
- A configuration list for the Linux distribution chosen (SuSE Linux Enterprise Server 8)
- Evidence of secure delivery procedures

The EAL2 assurance requirements fall into the following six categories:

1. **Configuration Management** - This class specifies how to establish that the Linux implementation meets the functional requirements. These objectives are met by requiring

---

1 Since the time this paper was written, SUSE Linux has been evaluated at the EAL3 assurance level with full compliance to the Controlled Access Protection Profile [5]. The additional experience with this evaluation is shortly discussed in the last section of this paper.
discipline and control in the process of refinement and modification of the Linux distribution and the related documentation. Basically, the Configuration Management system needs to provide methods for tracking changes and ensuring that all of the changes are authorized.

2. **Delivery & Operations** - This class provides requirements for the correct delivery, installation, generation, and startup of the evaluated Linux distribution.

3. **Development** - This class encompasses requirements for representing the TSF (TOE Security Function) at various levels of abstraction from the functional interface to the implementation representation. For EAL2, functional specification and the high-level design are needed. In addition, a correspondence demonstration between the functional specification and high level design needed to be provided.

4. **Guidance Documents** - This class provides the requirements for user and administrator guidance documentation. For the secure administration and use of Linux, it was necessary to describe all relevant aspects for the secure application.

5. **Security Testing** - The emphasis in this class is on confirmation that the security functions operate according to the operating system’s specification. Security testing provides assurance that the Linux distribution that was tested satisfies the security requirements. In this class, the aspects of coverage (completeness) and depth (level of detail) are separated for reasons of flexibility.

6. **Vulnerability Assessment** - This class defines requirements for the identification of exploitable vulnerabilities. Specifically, this class addresses any vulnerabilities introduced by the construction, operation, misuse, or incorrect configuration of the evaluated Linux distribution.

**Security Target**

The Common Criteria let you choose the security functions that are subject to the evaluation. The security functions need to be defined in detail in a document called the **Security Target**. The **Security Target** must justify that the set of security functions is useful and consistent. In addition, the threats that are meant to be countered by those functions, as well as the security requirements for the operational environment (like physical protection and trust of those users who are allowed to administer the system), need to be defined and explained.

As stated previously, the intention was to evaluate Linux “as-is” and to not develop additional security functions. The target of the evaluation was a server system, not a client, in order to emulate the way Linux is used for the handling of critical data by commercial enterprises and government agencies. The intention was not to have a wide set of security-related functions included in the evaluation, but instead to evaluate the basic security functions used almost everywhere. Additional security functions will be included in later evaluations.
The security functions included in this first evaluation of Linux were:

- User identification and authentication using passwords
- Discretionary access control for files and inter-process communication (IPC) objects
- Object reuse to ensure that no process can access data left by another process in an object (such as main memory or disk space)
- Security management that defines the functions to be used by root users to administer the Linux server
- Protection mechanisms that protect the kernel and applications from interference and tampering by untrusted processes

The evaluated configuration includes a few basic network services running as trusted processes (processes with root privileges) and also allows for other network protocols served by untrusted processes on unprivileged ports. For example, the evaluated configuration allows a web server to be run as an untrusted process on a port above 1024 (for example, 8080).

Regarding user authentication, the Pluggable Authentication Module (PAM) framework has been included in the evaluated configuration with a number of modules that enforce a strong password policy for password-based authentication. Other user authentication methods have been excluded and left for future evaluation. By including the PAM framework, we have opened the path for additional authentication methods in future evaluations.

Regarding discretionary access control, the evaluated configuration includes the ext3 file system, which supports POSIX-compatible access control lists. Support for ext3 allows for a more flexible access-control policy than is available with the standard UNIX® permission bits. In addition, the access-control mechanisms for IPC objects have been subject to the evaluation.

Missing or incomplete preparation of objects for reuse by other subjects have been common security problems in many operating systems. This initial evaluation ensured that all storage objects in memory or on disk did not contain information from their previous use when reallocated to another subject.

Regarding security management, the Security Target defines two roles: the “system administrator” and the “normal user.” A system administrator is any user who is allowed to su to root (which, in the evaluated configuration, is restricted to users belonging to the “trusted” group). Direct login to root is prohibited. All other users are regarded as “normal users.”

Another focus of the evaluation was the analysis of the mechanisms Linux uses to protect the kernel from interference and tampering by non-root application processes, and the separation the kernel enforces between processes running with different user IDs. These mechanisms strongly rely on the capabilities of the underlying hardware. Therefore, the hardware architecture and the way the hardware features are used for memory management and process separation needed to be looked at. In the case of our evaluation, the hardware platforms analyzed were the Intel Pentium® 4 and the XEON-based xSeries® systems of IBM. Those systems include single-
processor systems and multi-processor systems. More hardware platforms will be included in subsequent evaluations.

All of the security functions are described in more detail in the Security Target, which is publicly available on the website of the German Common Criteria certification body [6]. The Security Target as well as other documents used for the evaluation have been published on the IBM website [11].

The strategy for the development of the Security Target was to use the Controlled Access Protection Profile (CAPP) [5] as far as possible to describe the threats to be countered, the policies to follow, the assumptions and requirements on the environment as well as the requirements for the security functions themselves. The main reason for this strategy was to identify the missing functions required for full compliance with this Protection Profile and close those gaps with the development of the missing security functions. CAPP has been the basis for the evaluation of other operating systems like Solaris, AIX, HP-UX and Windows 2000.

Documenting the evaluation
The Common Criteria require the existence of a set of design and guidance documentation to be used as a basis for the analysis. The first document is the “functional specification,” which defines the external interfaces of the security functions.

In the case of Linux, the functional specification consists of three parts:
- The kernel system calls
- The processes or programs running with root privileges (these are either programs started as a daemon with root privileges or programs that are owned by root, have the s-bit set, and are executable by users other than root)
- The configuration files that influence the behavior of the security functions

In the UNIX world, the external interfaces of system calls and programs, as well as the configuration files, are described in man pages. When we compared the list of system calls in the SuSE Linux Enterprise Server 8 (which uses the standard 2.4.19 kernel version) with the existing man pages, a significant number (59) of those system calls were found to not have corresponding man pages, and the man pages for some of the configuration files did not describe all of the available parameters. In order for the evaluation to complete successfully, those deficiencies needed to be fixed. To close this documentation gap, man pages were developed for all system calls that were previously lacking them. Some of the existing man pages were also updated and enhanced for security-critical configuration files.

The next hurdle faced with respect to documentation was the requirement of the Common Criteria for a high-level design document that describes how the security functions are implemented. Although a large number of documents and books exist that describe how the kernel works, most of the literature is not detailed enough with respect to the implementation of security-related aspects. For example, the implementation of the access control lists in the ext3 file system and the implementation of access control for IPC objects are not described in enough
detail in the existing documentation. Also, documentation of some of the PAM modules used in the evaluation did not satisfy the requirements of the Common Criteria for a high-level design.

As the sponsor of the evaluation, IBM decided to develop a new high-level design document for SuSE Linux Enterprise Server 8, with a focus on the security functions defined in the Security Target. The new high-level design document has been extensively reviewed throughout its development by people from IBM, SuSE, and atsec to ensure that it correctly describes the security functions and contains the required level of detail.

To address the required administrator guidance for the installation and management of the evaluated configuration, a specific Security Guide was developed. The Security Guide describes how to get to the evaluated configuration in the installation process and provides guidance on how to manage the security functions so that the system maintains its level of security.

Another area where enhancements were required is testing. To pass an evaluation successfully, it must be shown that the security functions were extensively tested and that all security functions with most of their parameters and details were covered by the tests. To satisfy this requirement, the test suite developed within the Linux Test Project (LTP) was significantly extended with tests related to security functions. The tests were integrated into the framework defined by the Linux Test Project and most of the tests are now be executable in a fully automated way.

Another document required by the Common Criteria is a vulnerability analysis. At EAL2, only a high-level vulnerability analysis is required. IBM developed such a vulnerability analysis, which shows not only the vulnerabilities that were detected in the last several years that have been addressed, but also the residual vulnerabilities that could potentially be exploited by hostile users.

The successful completion of a Common Criteria security evaluation is not a guarantee of absolute security. The Common Criteria security evaluation states that the product has been analyzed with a specific level of rigor as defined by the Evaluation Assurance Level. The evaluation also states that the product, when operated in an environment compliant with the restrictions given in the Security Target, can be trusted to enforce the security policy defined by the security functions of the Security Target. Residual vulnerabilities can still exist that can potentially be exploited when the assumptions on the operational environment are violated. Therefore, any user of an evaluated product should perform a crosscheck of the operational environment with the defined restrictions to verify that the environment enforces those restrictions.

**Learning from the certification process**

A question that has been asked for quite some time is: Is it possible to perform a Common Criteria security evaluation of an Open Source system? Obtaining an EAL2 certification has demonstrated feasibility.
How is the evaluation of an Open Source product different from the evaluation of a commercial product?

At the evaluation level we chose, EAL2, we did not encounter many differences from a commercial product evaluation. At EAL2, the development process of an Open Source product and a commercial product has minimal differences.

Nevertheless, the Common Criteria requirements at EAL2 made it necessary to choose a defined distribution as the target for the evaluation. The development, configuration management, and flaw remediation procedures of the distributor (SuSE) were also assessed in this evaluation. With the experience gathered in this initial evaluation, there is now confidence that the development process-related aspects of higher assurance levels can also be satisfied by Open Source software that is managed and maintained by a distributor.

What has also been demonstrated is the fact that an evaluation of complex software such as Linux can be done quickly. The evaluation took just four months from the start of the evaluation until all of the required technical reports were produced and accepted by the Common Criteria certification body. Of course, one of the reasons the process went quickly was that EAL2 was chosen as the evaluation level. The strategy was to obtain a certificate quickly and then move step-by-step to higher levels of evaluation. The fact that evaluators involved had many years of experience in the certification business and who had previously performed evaluations of UNIX-type operating systems also helped.

Why hasn’t Linux been evaluated before?

Other UNIX-type operating systems have been successfully evaluated in the past. These operating systems include Sun Solaris (several versions), HP-UX (several versions) and IBM AIX (several versions). Microsoft® Windows® 2000 Advanced Server has also successfully passed an evaluation.

These successful evaluations show that performing a security evaluation is common for operating system-type products. Linux was the only server operating system with a growing market share that had not undergone a security evaluation for the simple reason that an evaluation costs money, which neither the Open Source community nor the existing Linux distributors have been willing or able to spend.

A Linux evaluation also requires that Linux adapt its security functionality to the security requirements of commercial enterprises and government agencies. The Open Source community has not always regarded such adaptation as a high priority. The community is often more interested in technologically challenging functions than it is in satisfying standard business requirements. An example is auditing, which is an important requirement for many business areas where accountability is essential. Sun Solaris, IBM AIX, HP-UX, and Microsoft Windows
2000 Advanced Server all have extensive audit capabilities. Linux, on the other hand, has only a few auditing add-ons, most of which are prototypes in character.

The ‘Controlled Access Protection Profile’ (CAPP) [5], a standard defined by the US government on security requirements for commercial operating systems, includes specific requirements for auditing. None of the current approaches for an audit subsystem for Linux satisfies these requirements completely. All of the competing operating systems mentioned previously provide auditing capabilities that are compliant with the CAPP requirements.

**What were the main difficulties encountered in the Linux evaluation?**

The main difficulty encountered in the Linux evaluation was the status of the existing documentation. This may seem paradoxical considering the large number of documents on Linux available on the Internet and in published books, the man pages that are part of most packages, and the body of Linux HOW-TOs. What was lacking was the design documentation that is required to perform a security analysis quickly and efficiently. The Linux Documentation Project [7] contains Linux documentation, but the documents published there focus primarily on guidance documentation and only partly on design documentation. Some useful documentation exists on the kernel (see *Linux Kernel 2.4 Internals* by Tigran Aivazian [8] or the second edition of *Understanding the Linux Kernel* [9]). Although these sources are quite detailed and up-to-date, they are not focused on security and lack some of the details required for the formal evaluation with respect to the implementation of the security functions. Also, very little documentation exists on non-kernel functions, such as the PAM framework or the implementation of other security functions running with root privileges.

Additionally, severe deficiencies in the man pages that describe the system calls and security-critical configuration files were identified.

The problem with Linux is that development of the code and development of the design documentation are not strictly coupled. In many cases, the people who develop the design documentation are not part of the development team. Those who update the software and those who update the design documentation are not necessarily in sync. With the exception of the man pages, there are no rules or guidelines for structuring and producing design documentation. This lack of guidelines leads to design documentation that differs significantly in the level of detail provided. In some cases information overlaps between sources and in other cases functions are not described at all.

Commercial operating systems are designed differently than Open Source operating systems because most commercial developers have a defined way to update the design documentation when modifications are made. Also, the maintenance of the design and user documentation is usually under strict management control.

To overcome the documentation problem in the Linux evaluation, IBM decided to develop the missing man pages and create a new security-focused high-level design document as input for the
evaluation. This approach will enable the development of the design document even further than necessary for EAL2 in order to ultimately achieve higher evaluation levels.

Another important aspect of an evaluation is testing. The situation was better in comparison due to the Linux Test Project [10]. The Linux Test Project maintains a defined test suite for Linux, which was used as a starting point for the testing required for the evaluation. With the test suite, the situation for Linux with respect to testing was quite comparable to the situation an evaluator finds when a commercial operating system is evaluated for the first time.

Certain aspects of Open Source software make an evaluation of Open Source easier than that of a commercial operating system. One such aspect is the fast response time in the Open Source community when it comes to reported security problems. Although it is acknowledged that security-aware vendors also react very quickly nowadays, Open Source still has a timing advantage over commercial products.

Another very important positive aspect of Open Source software, especially at the lower Common Criteria assurance levels (up to and including EAL4), is the full availability of the source code. Only at EAL5 do the Common Criteria require that the developer provide the full source code to the evaluation facility. At EAL4, only a subset of the source code needs to be provided to the evaluation facility, and at EAL1 through EAL3 no source code needs to be provided.

Because Open Source products by their nature provide access to the full source code, an evaluation facility can use this availability as additional input when performing the assessment. How useful this access is without corresponding up-to-date low-level design documentation depends very much on the experience of the evaluators. In the case of the Linux evaluation that was performed, all evaluators had in-depth knowledge of UNIX-type operating systems in general and Linux in particular. The evaluators made extensive use of the source code in the identification of security-critical implementation errors as well as in the vulnerability assessment. This led to a number of security problems that were identified in the evaluation (and corrected by the developers), which would not have been possible to identify as easily without access to the source code.

Examples of security problems identified and solved within the evaluation are:

- more than 20 system calls of the 2.4.19 kernel were not described by man pages. The missing man pages were developed by IBM and released to the Open Source community.
- just in the pluggable authentication module (PAM) framework a total of six security critical implementation flaws were detected throughout the evaluation. As far as the PAM configuration used for the evaluation was affected, all of them had been corrected and those corrections are included in the update to the PAM package that is part of the evaluated configuration. The flaws included
  - a heap corruption problem potentially leading to a system crash
a false return code passed back to the caller potentially resulting in a user’s password being set to zero
a flaw in the implementation of stacked modules resulting in some required checks not being performed. Depending on the PAM configuration this could have resulted in a user getting access to the system without knowing the correct password

The main lesson learned in this project was the demonstration that Open Source products, even those as complex of a full Linux distribution, can be evaluated according to the Common Criteria. Open Source products can also be evaluated relatively quickly. The evaluation officially started on March 6, 2003 and was finished on July 10, 2003. Within this timeframe, the design documentation and test documentation produced by IBM and SuSE, and the evaluation documentation produced by atsec as the evaluation facility, amounted to more than 800 pages!

In addition we now have a better understanding of what needs to be done to achieve compliance with higher Common Criteria assurance levels. The experience gathered in this initial evaluation allowed us to apply for a re-evaluation at a higher assurance level, as explained in more detail in the last section of this paper.

Resources from the Certification Process
To improve the security of Linux, most of the documentation developed within the evaluation of Linux has been made generally available as part of the Open Source repository [11]. This documentation includes:

- Security Target
- New man pages
- High-level design
- Security Guide
- Test cases

By making this documentation available, other distributions will be able to pick up this material, adapt it to their systems, and undergo a Common Criteria evaluation themselves without the need to invest a significant amount of time and money in preparing these documents.

A question that remains is what to do with the vulnerability analysis. Making the vulnerability analysis public might give potential hackers ideas because not all vulnerabilities can be effectively eliminated. On the other hand, the vulnerability analysis can provide additional information for those responsible for setting up and managing an environment with Linux systems. As is the case with Open Source software in general, there are pros and cons to publishing the vulnerability analysis. The discussion is not so critical for the current fairly high-level vulnerability analysis that was developed, but the relevance of this issue will change with evaluation levels that require an in-depth vulnerability assessment.
Next steps
Performing a security evaluation should never be a one-time accomplishment. To maintain the security level achieved, the security certificate must be maintained. In the case of Linux, the intent is to go a step further: to increase, step-by-step, the assurance level and the security functionality until Linux achieves the highest assurance level of any commercial operating system product, while offering the richest set of security functions!

The next step in this direction has already been taken. Linux, like its commercial competitors IBM AIX 5.2, Sun Solaris 8, and Microsoft Windows 2000 Advanced Server, has been successfully evaluated for compliance with the requirements of the US government defined CAPP. CAPP includes extended functionality, such as an audit subsystem. Additionally, the evaluation assurance level was increased to EAL3, requiring an even more strict security analysis of Linux. The security functionality included in the EAL3 evaluation went beyond those required by CAPP including a newly developed subsystem for auditing security relevant events.

The main difference between EAL2 and EAL3 are the additional requirements for configuration management, “developer security”, more detailed high-level design and more rigorous testing. The configuration management and developer security requirement have been addressed by the procedures of the distributor (SuSE). The high-level design document produced for the EAL2 evaluation has been enhanced with more details and the additional requirements for testing have been addressed by additional test cases as well as the use of the gcov tool to demonstrate which kernel internal interfaces are called by individual test cases. The documents for this evaluation have also been published [11]. The test plan and test cases used for the evaluation has been published on the web site of the Linux test project [12].

The result is a more useful system where customers can place an even higher level of trust in the correctness and effectiveness of the security functions provided by Linux. EAL3 certification was another significant step in establishing Linux as a trusted base for critical applications.

As a further step Linux is currently in evaluation for compliance with the requirements of the EAL4 level. This includes the development of a low-level design of the Linux kernel (the evaluation will be based on the 2.6 version of the kernel) as well as a more sophisticated vulnerability analysis being performed. The experience gathered in the EAL2 and the EAL3 evaluation have given the confidence that compliance with EAL4 can be achieved by the end of 2004. Keeping in mind that the initial evaluation of other operating systems for compliance with this level took more than 3 years, the step-by-step approach taken in the evaluation of Linux seems to quite efficient. With compliance with EAL2 achieved after 4 month, compliance with EAL3 achieved after another 6 month and compliance with EAL4 estimated to take another 12 month the overall time frame to take Linux to EAL4 would be less than 2 years.

References
7. www.tldp.org
8. www.tldp.org/LDP/lki

Legal Statement

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

IBM, AIX, and xSeries are registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Intel and Pentium are registered trademarks of Intel Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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