Certifying Open Source—The Linux Experience

The Common Criteria is an international standard for evaluating the security functions of IT products. The authors describe how they obtained this security certification for Linux, the first open-source product to receive such certification.

K.S. (Doc) Shankar
IBM

Helmut Kurth
atsec Information Security

As Linux has continued to penetrate the marketplace, customers, especially in the government sector, have exerted considerable pressure on Linux developers to obtain some form of security certification. All of Linux's commercial competitors have already obtained certification, and for Linux to continue to aggressively compete with them, it must obtain certification as well. For these reasons, and after careful analysis, IBM decided to sponsor a Common Criteria-based security certification for Linux.

An international group of experts developed the Common Criteria to assess the security functions of computer products. In 1999, the International Organization for Standards (ISO) adopted the Common Criteria as ISO standard 15408.[1] The governments of 20 nations, including the US, Canada, Germany, France, Japan, and the United Kingdom, have officially adopted the Common Criteria. Since July 2002, the United States has required a Common Criteria or Federal Information Processing Standard (FIPS) 140 certification for cryptographic functions (see http://csrc.nist.gov/cryptval) for all IT products used for processing security-critical data within US government systems.

As far as we know, no open-source program has received a Common Criteria security certification—until now. Although some people believed that an open-source program couldn't receive a Common Criteria security certification, IBM and atsec have proved otherwise by obtaining this certification for Linux. It's also generally believed that security certifications are time consuming, often taking years to accomplish. We obtained the Common Criteria certification of Linux in four months. This article describes our experience with the certification process and examines several aspects of Linux open source.

Security implications of open-source code

An open-source program's license lets users run the program for any purpose, study and modify the program, and 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 article focuses on Linux, most of the strategies we describe are applicable to all open-source software.

Linux is a member of the large family of Unix-like operating systems from companies such as AT&T, Digital, IBM, Hewlett-Packard, and Sun. Linux, however, isn't a commercial operating system, and no one vendor controls it. Linus Torvalds, who developed Linux in 1991, initially designed it 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 MySQL, Python, Open LDAP, and open-source implementations of Kerberos.

Since the beginning of open-source development, debate has raged over whether open-source software is more secure than other software, but with no definite conclusion. Open-source software gives attackers and defenders the same advantage.[2] If a defender does nothing about security, open-source software essentially gives the advantage to the attacker. However, open-source software also offers the defender great advantages by provid-

ing access to security techniques and knowledge that are rarely 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. In addition, open-source code is subject to community-wide review and audit (although simply publishing code doesn't guarantee that people will examine it for security flaws).

Another advantage of open-source software is the speed with which developers make patches available when bugs are found. Many open-source developers consider it a personal challenge to develop and release fixes to the open-source community, sometimes within a few hours—a speed unheard of in closed-source development.

Certification process

The Common Criteria allow for several assurance levels, ranging from Evaluation Assurance Level 1 (the lowest level requiring only a minimal set of documentation and testing of the security functions) to EAL7 (the highest level requiring formally specified security functions, extensive testing of all details of those functions, and a sophisticated vulnerability analysis). Many of Linux's commercial competitors have been certified at EAL4.

IBM and atsec first considered obtaining an EAL4 certification for Linux. Although this security level was technically attainable, achieving it would have taken about three years. Given the market pressure, our objective became to demonstrate that Linux could be certified at a meaningful level fairly quickly (that is, in about six months). We considered EAL3 certification next. Because lifecycle assurance requirements start at EAL3, open-source code would have difficulty meeting those requirements in the first certification step. In addition, we didn't want to develop any significant software for security functionality that would further slow the certification effort. In other words, IBM and atsec wanted to certify Linux as-is. Therefore, as a start, we chose the SUSE Linux Enterprise Server 8, a widely used Linux distribution, as the target of evaluation (TOE) and selected EAL2 as our first step. (Since this article's writing, we've evaluated both SUSE Linux and Red Hat Linux at the EAL3 assurance level with full compliance to the controlled access protection profile.[3]) We can further augment EAL2 to obtain higher assurance levels and meet the requirements for more elaborate protection profiles.

EAL2 analysis and requirements

The EAL2 certification process involves analyzing the operating system's functional and interface specifications, guidance documentation, and high-level design to understand its security behavior. EAL2 analysis also includes

- Independent security function testing
- Evidence of developer testing based on functional specification
- Selective independent confirmation of developer test results
- Strong function analysis
- Evidence of developer search for obvious vulnerabilities
- Verification that all functional specification, high-level design, and guidance documentation is consistent with the system's actual operation
- Configuration list for the Linux distribution chosen (in this case, SUSE Linux Enterprise Server 8)
- Evidence of secure delivery procedures

The EAL2 assurance requirements fall into six categories:

- *Configuration management* specifies methods for establishing that the Linux implementation meets the functional requirements. We met these objectives by requiring discipline and control in the refinement and modification processes of the Linux distribution and the related documentation. Basically, the configuration management system must let us track changes and ensure that all of the changes are authorized.
- *Delivery and operations* provide requirements for the evaluated Linux distribution's correct delivery, installation, generation, and startup.
- *Development* encompasses requirements for representing the TOE security function (TSF) at various abstraction levels, from the functional interface to the implementation representation. EAL2 analysis requires correspondence between the functional specification and high-level design.
- *Guidance documents* describe all relevant aspects for the secure administration and use of Linux.
- *Security testing* confirms that the security functions operate according to the operating system's specification, and thus that the tested Linux distribution satisfies the security requirements. In this class, the aspects of coverage (completeness) and depth (level of detail) are separated for flexibility.
- *Vulnerability assessment* defines requirements for identifying exploitable vulnerabilities. Specifically, this class addresses any vulnerability introduced by the construction, operation, misuse, or incorrect configuration of the evaluated Linux distribution.

We felt confident that we could meet those requirements with Linux and complete the evaluation within six months.

Security target

The Common Criteria lets you choose the security functions to be evaluated. The *security target* is a document that details the security functions; it must show that this set of functions is useful and consistent. It must also describe the threats these function are designed to counter as well as

the security requirements for the operational environment (such as physical protection and trust of the users administering the system).

As stated previously, our intention was to evaluate Linux as-is, without developing additional security functions. The evaluation targeted a server system, not a client, to emulate how commercial enterprises and government agencies use Linux for handling critical data. Rather than evaluate a wide set of security-related functions, we wanted to evaluate the basic security functions used in almost all Linux implementations. We'll include additional security functions in later evaluations.

Our intention was to evaluate Linux as-is, without developing additional security functions.

In the first evaluation, the security target included the following security functions:

- *Password-based user identification and authentication.* We included the pluggable authentication module (PAM) framework in the evaluated configuration with several modules that enforce a strong password policy for password-based authentication. We excluded other user authentication methods, leaving them for future evaluation.
- *Discretionary access control for files and interprocess communication (IPC) objects.* 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. We've also evaluated access-control mechanisms for 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).* Missing or incomplete preparation of objects for reuse is a common security problem in many operating systems. Our initial evaluation ensured that all storage objects in memory or on disk contained no information from their previous use when reallocated to another subject.
- *Security management that defines the functions root users will use to administer the Linux server.* The security target defines two roles: *system administrator* and *normal user*. A system administrator is any user who is allowed to become root (in the evaluated configuration, this is restricted to trusted users). Direct log in to root is prohibited. All other users are normal users.
- *Mechanisms that protect the kernel and applications from interference and tampering by untrusted processes.* We also evaluated the kernel-enforced separation between processes running with different user IDs. Because these mechanisms rely on the underlying hardware capabilities, we examined the hardware architecture and how it performs memory management and process separation.

Our evaluation analyzed the Intel Pentium 4 and XEON-based IBM xSeries systems, including both single- and multiprocessor systems. Subsequent evaluations will include more hardware platforms.

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 lets users run a Web server as an untrusted process on ports above 1024.

The security target, which is publicly available on the German Common Criteria certification body Web site (www.bsi.de/zertifiz/zert/reporte/0216b.pdf), describes the security functions in more detail. The IBM Web site also includes the security target as well as other documents used for the evaluation (http://oss.software.ibm.com/linux/pubs/?topic_id=5).

In developing the security target, we used the controlled-access protection profile (CAPP)[3] as far as possible to describe the threats to be countered, the policies to follow, the assumptions and requirements on the environment, and 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 the gaps by developing the missing security functions. CAPP is a US government-defined standard for commercial operating system security requirements, and has been the basis for evaluating other operating systems.

Required documentation and analyses

The Common Criteria require a set of design and guidance documents on which to base the analysis. The first document, the *functional specification*, defines the security functions' external interfaces. The functional specification for Linux has three parts:

- kernel system calls,
- 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 *setuid*-bit set, and are executable by users other than root), and
- configuration files that influence security function behavior.

In the Unix world, *man pages* describe the external interfaces of system calls and programs, as well as the configuration files. 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, we found that a significant number (59) of the calls had no corresponding man pages, and the man pages for

some of the configuration files didn't describe all the available parameters. To close the documentation gap, we developed man pages for the system calls that lacked them. We also updated and enhanced some of the existing man pages for security-critical configuration files.

The Common Criteria also requires a high-level design document describing the security functions' implementation. Although many documents and books describe how the kernel works, few provide enough detail about the implementation of security-related aspects, such as the access-control lists in the ext3 file system or access control for IPC objects. In addition, documentation of some of the PAM modules used in the evaluation failed to satisfy the Common Criteria's requirements for a high-level design.

As the sponsor of the evaluation, IBM decided to develop a new high-level design document for the SUSE Linux Enterprise Server 8, focusing on the security functions defined in the security target. People from IBM, SUSE, and atsec have extensively reviewed the new document throughout its development 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, we developed a security guide. The security guide describes how to get to the evaluated configuration in the installation process and gives guidance on managing the security functions so the system maintains its security level.

The security functions must be extensively tested, with tests covering all security functions and most of their parameters and details. To satisfy this requirement, we significantly extended the Linux Test Project (LTP) test suite with security-function-related tests. We integrated the tests into the LTP framework so that most of the tests are now fully automated.

The Common Criteria also requires a vulnerability analysis. At EAL2, it requires only a high-level vulnerability analysis. IBM developed such a vulnerability analysis that shows not only the vulnerabilities detected in the last several years that have been addressed, but also the residual vulnerabilities that hostile users could potentially exploit.

Successful completion of a Common Criteria security evaluation doesn't guarantee absolute security. The Common Criteria security evaluation states that the product has been analyzed with a specific level of rigor as defined by the EAL. The evaluation also states that the product, when operated in an environment compliant with the restrictions in the security target, can be trusted to enforce the security policy defined by the security functions in the security target. Violating the assumptions on the operational environment results in residual vulnerabilities that can be exploited. Therefore, any user of an evaluated product should cross-check the operational environment against the defined restrictions to verify that the environment enforces those restrictions.

Lessons learned

Over the years, many people have asked whether performing a Common Criteria security evaluation of an open-source system is possible. We've demonstrated the feasibility of obtaining an EAL2 certification.

Open source vs. commercial

At EAL2, the evaluation of open-source products is similar to that of commercial products. The development processes of open-source and commercial products have minimal differences at this level.

Nevertheless, the Common Criteria EAL2 requirements made it necessary to choose a defined distribution as the evaluation's target. The evaluation also assessed development, configuration management, and flaw-remediation procedures of the distributor (SUSE). With the experience gathered in this initial evaluation, we're confident that open-source software managed and maintained by a distributor can satisfy the development-process-related aspects of higher assurance levels.

We've also demonstrated that it's possible to quickly evaluate complex software such as Linux. The evaluation took four months from the start of the evaluation until the Common Criteria certification body accepted all of the required technical reports. Of course, one reason the process went quickly was that we used EAL2, a fairly low level of assurance. Our strategy was to obtain a certificate quickly and then move step-by-step to higher evaluation levels. That the evaluators had many years of experience in certification and had previously performed evaluations of Unix-type operating systems also helped.

Lack of previous evaluations

Security evaluations are common in operating system-type products. In fact, Linux was the only server operating system with a growing market share that hadn't undergone a security evaluation, simply because evaluations cost money, which neither the open-source community nor the existing Linux distributors have been willing or were able to spend. A Linux evaluation also requires that Linux adapt its security functionality to commercial enterprise and government agency security requirements. The open-source community hasn't always regarded such adaptation as a high priority; the community is often more interested in technologically challenging functions than in satisfying standard business requirements.

Auditing, for example, is an important requirement for many business areas in which accountability is essential. Although Sun Solaris, IBM AIX, HP-UX, and Microsoft Windows 2000 Advanced Server all have extensive audit capabilities that are compliant with CAPP auditing requirements, Linux has only a few auditing add-ons, most of which are prototypes in character, and none that satisfies the CAPP requirements completely.

Main difficulties

The main difficulty encountered in the Linux evaluation was the existing documentation's status. This might seem paradoxical considering the many documents on Linux available online and in books, the man pages that are part of most packages, and the body of Linux how-tos. What was lacking was the design documentation required to perform a security analysis quickly and efficiently. The Linux Documentation Project (www.tldp.org) contains Linux documentation, but the documents published there focus primarily on guidance and only partly on design. Some useful documentation exists on the kernel;[4,5] however, although these sources are detailed and up-to-date, their focus isn't security, and they lack some of the implementation details. Moreover, little documentation exists on nonkernel functions, such as the PAM framework or the implementation of other security functions running with root privileges. In addition, we identified severe deficiencies in the man pages describing the system calls and security-critical configuration files.

In Linux, the code and design documentation development processes aren't strictly coupled. Often, the people who develop the design documentation aren't part of the development team. Those who update the software and those who update the design documentation aren't necessarily in sync. Except for the man pages, no rules or guidelines for structuring and producing design documentation exist. Consequently, design documentation differs significantly in the level of detail provided. In some cases, information overlaps between sources; in other cases, functions aren't described at all.

Unlike developers of open-source operating systems, most commercial developers have a defined process for updating the design documentation when modifications are made and maintain the design and user documentation under strict management control. To overcome the documentation problem in the Linux evaluation, IBM developed the missing man pages and created a new security-focused high-level design document as input for the evaluation. This approach will let us develop the design document even further than necessary for EAL2 to ultimately achieve higher evaluation levels.

The Linux Test Project (http://ltp.sourceforge.net), which defines a suite of tests for Linux, made the testing situation better than the documentation situation, and in fact made the situation for Linux with respect to testing comparable to the situation an evaluator finds when evaluating a commercial operating system for the first time.

Certain aspects of open-source software make evaluating open-source operating systems easier than evaluating commercial operating systems. For example, the open-source community responds to reported security problems quickly. Although today's security-aware vendors also react quickly, open source still has a timing advantage over commercial products.

Another important positive aspect of open-source software, especially at the lower Common Criteria assurance levels (up to and including EAL4), is source code availability. Only at EAL5 do the Common Criteria require the developer to provide the full source code to the evaluation facility. At EAL4, only a subset of the source code must be provided, and at EAL1 through EAL3, no source code need be provided. Because open-source products provide access to the full source code, an evaluation facility can use the code as additional input when assessing the product. How useful this access is without corresponding up-to-date low-level design documentation depends on the evaluators' experience. In the Linux evaluation, all evaluators had in-depth knowledge of Unix-type operating systems in general and Linux in particular. The evaluators used the source code extensively in identifying security-critical implementation errors as well as assessing vulnerability. This let them identify many security problems that would have been difficult to identify without access to the source code.

Of the security problems identified and solved within the evaluation, evaluators found that no man pages for more than 50 system calls of the 2.4.19 kernel existed. IBM developed the missing man pages and released them to the open-source community. In addition, in the PAM framework alone, evaluators detected a total of six security-critical implementation flaws. SUSE corrected all of these flaws and included the corrections in the PAM update that was part of the evaluated configuration. These flaws included

- a heap corruption problem that could cause a system crash;
- a false return code passed back to the caller, which could cause a user's password to be set to zero; and
- a flaw in the implementation of stacked modules causing those modules to omit some required checks, which could lets a user access the system without the correct password.

The main lesson learned in this project is that a Common Criteria evaluation is possible for open-source products, even those as complex as a full Linux distribution. Evaluation of open-source products can take place relatively quickly. The evaluation officially began 6 March 2003 and ended 10 July 2003. Within this time frame, the design documentation and test documentation produced by IBM and SUSE, and the evaluation documentation produced by atsec amounted to more than 800 pages (see the "Certification Process Resources" sidebar for more on this documentation). In addition, we now have a better understanding of how to achieve compliance with higher Common Criteria assurance levels.

Performing a security evaluation should never be a one-time accomplishment. Maintaining the security level achieved requires maintaining the security certifi-

Certification process resources

To improve Linux security, we've made most of the documentation developed within the evaluation generally available as part of IBM's open-source repository (http://oss.software.ibm.com/linux/pubs/?topic_id=5). This documentation includes security target, new man pages, high-level design, security guide, and test cases.

Other distributions can pick up this material, adapt it to their systems, and undergo a Common Criteria evaluation without having to invest a significant amount of time and money preparing these documents.

We've not yet decided what to do with the vulnerability analysis. Making the analysis public might benefit potential hackers 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 a Linux-based environment. 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 issue's relevance will change with evaluation levels that require in-depth vulnerability assessments.

cate. In the case of Linux, we must go a step further: 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.

We've already taken the next step in this direction. Linux, like its commercial competitors, has been successfully evaluated for compliance with CAPP requirements. We've also increased the evaluation assurance level to EAL3, requiring an even stricter security analysis of Linux. The security functionality included in the EAL3 evaluation went beyond that required by CAPP, including a new subsystem for auditing security-relevant events.

The main differences between EAL2 and EAL3 are additional requirements for configuration management, developer security, more detailed high-level design, and more rigorous testing. SUSE procedures have addressed the configuration management and developer security requirements. IBM added detail to the high-level design document produced for the EAL2 evaluation, test cases, as well as demonstrations using the `gcov` tool to determine which kernel internal interfaces are called by individual test cases, have addressed the additional testing requirements. The documents for this evaluation are available at the IBM Web site (see the sidebar). The test plan and test cases used in the evaluation are on the Linux Test Project Web site (http://ltp.sourceforge.net/EAL3.html).

The result is a more useful system that allows customers to place a higher level of trust in the correctness and effectiveness of Linux's security functions. EAL3 certification was another significant step in establishing Linux as a trusted base for critical applications.

As a further step, atsec is currently evaluating Linux for compliance with EAL4, which 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) and requires a more sophisticated vulnerability analysis. The experience gathered in the EAL2 and EAL3 evaluations have given us the confidence that we can achieve compliance with EAL4 by the end of 2004. The project is well on schedule. Given that the initial evaluation of other operating systems for compliance with this level took more than three years, our step-by-step approach seems quite efficient. Having achieved compliance with EAL2 in four months, compliance with EAL3 in another six months, and compliance with EAL4 in an (estimated) additional 12 months, the overall time frame to take Linux to EAL4 will be less than two years. □

References

1. *ISO/IEC Standard 15408, Evaluation Criteria for IT Security*, parts 1 to 3, Int'l Organization for Standards, 1999.
2. C. Cowen, "Software Security for Open-Source Systems," *IEEE Security & Privacy*, vol. 1, no. 1, Jan./Feb. 2003, pp. 38–45.
3. *Controlled Access Protection Profile* (CAPP), version 1.d, Nat'l Inst. of Standards and Technology, validated protection profile, 1999; http://niap.nist.gov/cc-scheme/pp/PP_CAPP_V1.d.html.
4. T. Aivazian, "Linux Kernal 2.4 Internals," Linux Documentation Project, 2002; www.tldp.org/LDP/lki.
5. D.P. Bovet and M. Cesati, *Understanding the Linux Kernel*, O'Reilly & Assoc., 2003.

Acknowledgments

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

***K.S. (Doc) Shankar** is a security architect at IBM. His research interests include public-key infrastructure; cryptographic frameworks and systems; and operating system, workstation, and network security. He received MS and PhD degrees in electrical engineering and computer sciences from the University of California at Berkeley. Contact him at dshankar@us.ibm.com.*

***Helmut Kurth** is the cofounder, chief scientist, and head of the Common Criteria Evaluation Facility at atsec information security. His research interests include secure operating systems, cryptographic protocols and smart card security. He has a master's in applied mathematics from the University of Bonn. He has been a member of the steering committee of the European Symposium on Research in Computer Security (ESORICS) since it was founded. Contact him at helmut@atsec.com.*