Rule Set Based Access Control (RSBAC)

Amon Ott

Email: ao@rsbac.org
WWW: http://www.rsbac.org

Abstract

The Rule Set Based Access Control (RSBAC) system is an open source extension to current Linux kernels, which has been continuously developed for several years.

It was designed according to the Generalized Framework for Access Control (GFAC) to overcome the deficiencies of access control in standard *nix systems, and to make a flexible combination of security models as well as proper access logging possible.

Access control is devided into enforcement, decision and data structures, and all access modes are grouped into abstract request types. This makes the framework and the existing model implementations easily portable to other operation systems.

Among the nine included access control models are well known ones, like MAC/Bell–LaPadula, as well as new models, which have been specially designed for *nix needs.

Installation requires a kernel patch, RSBAC configuration and a recompile. The complete set of administration tools contains a range of menues for most tasks.

Practical experience shows the system to be fast and stable for production use, what is one reason for its growing acceptance. There are already two Linux distributions with RSBAC included.

1 Introduction

1.1 History

The RSBAC project has been started as my master thesis in November 1996 at Hamburg University, which lasted a full year until November 1997. The first public version 0.9 for Linux kernel 2.0.30 was published on January, 9, 1998.

As of 27th of July, the current stable version is 1.1.1, which supports kernels 2.2.18/19 and 2.4.2-7. Pre-version 1.1.2-pre9 has been released, and 1.1.2-final will certainly be available when this paper is presented.

The next major release 1.2.0 will contain a lot of changes, e.g the first extension of the set of request types.

1.2 Motivation

It is well known that the classic *nix style access control is insecure. For me, there are three major reasons:

1. Small granularity: All you have are the access modes read, write and execute, set for the file or dir owner, the file's assigned group and all others. In very many cases, this is barely enough for secure administration.

2. Discrete control: You have to put trust into all users, who handle sensitive or critical data, that they administrate access control accordingly. Due to their lack of personal group management, they can hardly do proper access control setups.

   Also, all discrete access control is like an invitation to trojans and viruses, who can do anything the respective user is allowed to do.

3. Superuser root - the worst of these three problems: root has full access to everything, even the kernel memory, and is too often needed. Too much software has to start or even run under root account, e.g. many network daemons.

   Naturally, there are loads of exploits through this dangerous account.

1.3 Design Goals

The RSBAC system has been designed to meet the following items:

- Provide better access control models for different administration goals.
- Allow flexible selection and combination of several independent models within a generic framework.
- Be easily portable.
- Encapsulate as many parts as possible.
- Be extensible.
- Be filesystem independent
- Become a real, stable, usable and fast kernel extension.
- Target at many Orange Book / B1 requirements.

2 Overview of RSBAC

2.1 GFAC based

The RSBAC system is based on the Generalized Framework for Access Control (GFAC) by Marshall Abrams and Leonard La Padula, which describes a general framework approach to separate access control between enforcement, decision, access control data and authorities, who are allowed to modify that data.

The original design followed the suggestions in [LaPadula95] about how to implement GFAC in a *nix system and implemented some access control models briefly described there, namely MAC, FC and SIM. The MAC model implementation has been changed a lot since then.

Several publications already cover certain aspects of the whole RSBAC system, which you can download from the RSBAC documentation page at [RSBAC].

All code is published under GNU Publishing License and can also be downloaded from there.

2.2 Key Features

RSBAC has a flexible structure due to its separation between enforcement (Access Control Enforcement Facility, AEF), decision (Access Control Decision Facility, ADF) and data (Access Control Data, ACI). Because of its request abstraction, only AEF and parts of ACI are operation system dependent. The ADF, which contains all model implementations, should mostly need a recompile to work on other *nix style operation systems.

The framework supports almost any type of access control model. The model combination in ADF requires a metapolicy, which restrictively decides in cases of contradiction between model decisions.

Through the Runtime Module Registration facility (REG), decision modules as well as system calls or persistent generic lists can be added or removed at runtime, e.g. from a Loadable Kernel Module (LKM).

As a very important part, there is also a powerful logging system. Whether a decision is to be logged depends upon the request type and the decision, the user ID, the program running and the object that shall be accessed. Logging can be done with pseudonyms, thus providing some user privacy.

2.3 RSBAC in the wild

The RSBAC system has been in stable production use since March 2000. It supports all current Linux kernels from the 2.2 and 2.4 series.

Downloads and feedback are constantly increasing, and there are even two Linux distributions running with RSBAC kernels, ALTLinux Castle and Kaladix.

3 Architecture and Implementation of the Framework

3.1 Subjects, Objects and Requests

In RSBAC, subjects are defined as processes acting on behalf of user IDs. The following object types (here named as target types) are defined:

- FILE
- DIR
- FIFO (also known as named pipe)
- SYMLINK
- DEV (devices by block/char and major:minor)
- IPC (Inter Process Communication)
- SCD (System Control Data)
- USER
- PROCESS
- NONE (no object associated with this request)

Access modes are grouped into abstract request types. Whenever a subject wants to access an object, the respective request call with parameters request type, subject, object and attribute data is issued. One system call can lead to several request calls, e.g. sys_open can lead to SEARCH, CREATE, TRUNCATE and all OPEN request types.

3.2 List of Requests with Targets

The following request types with respective target types are defined:

- R_ADD_TO_KERNEL: NONE
- R_ALTER: IPC
- R_APPEND_OPEN: FILE, FIFO, DEV, IPC
- R_CHANGE_GROUP: FILE, DIR, FIFO, IPC, USER, PROCESS, NONE
- R_CHANGE_OWNER: FILE, DIR, FIFO, IPC, PROCESS, NONE
- R_CHDIR: DIR
- R_CLONE: PROCESS
- R_CLOSE: FILE, DIR, FIFO, DEV, IPC
- R_CREATE: DIR (where), IPC
- R_DELETE: FILE, DIR, FIFO, IPC
- R_EXECUTE: FILE
- R_GET_PERMISSIONS_DATA: FILE, DIR, FIFO, IPC, SCD
- R_GET_STATUS_DATA: FILE, DIR, FIFO, SYMLINK, IPC, SCD
- R_LINK_HARD: FILE, FIFO
- R_MODIFY_ACCESS_DATA: FILE, DIR, FIFO
- R_MODIFY_ATTRIBUTE: All target types
- R_MODIFY_PERMISSIONS_DATA: FILE, DIR, FIFO, IPC, SCD, NONE
- R_MODIFY_SYSTEM_DATA: SCD
- R_MOUNT: DIR, DEV
- R_READ: DIR, SYMLINK, IPC (optional: FILE, FIFO, DEV, IPC-sock)
- R_READ_ATTRIBUTE: All target types
- R_READ_OPEN: FILE, FIFO, DEV, IPC
- R_READ_WRITE_OPEN: FILE, FIFO, DEV, IPC
- R_REMOVE_FROM_KERNEL: NONE
- R_RENAME: FILE, DIR, FIFO
- R_SEARCH: DIR, FIFO
- R_SEND_SIGNAL: PROCESS
- R_SHUTDOWN: NONE
- R_SWITCH_LOG: NONE
- R_SWITCH_MODULE: NONE
- R_TERMINATE: PROCESS
- R_TRACE: PROCESS
- R_TRUNCATE: FILE
- R_UMOUNT: DIR, DEV, NONE
- R_WRITE: DIR, SCD (optional: FILE, FIFO, DEV, IPC-sock)
- R_WRITE_OPEN: FILE, FIFO, DEV, IPC

Please note that target type NONE is internally converted into SCD target 'other' by RC and ACL model.

3.3 Architectural Diagram

Figure 1 on page 4 shows the RSBAC implementation in the Linux kernel.

A typical system call interception (AEF component) places two calls to ADF: a request for decision and, if access has been granted and the system call functionality has been successfully performed, a notification. ACI data is only updated on the notification call, because the system call might fail from other reasons.

3.4 Module Registration (REG)

Additional decision functions and system calls can be added at runtime through the Module Registration facility (REG). This allows for new models to be implemented as a kernel module.

Decision and notification functions, system calls and generic persistent lists can be added or removed whenever necessary. All such registrations are controlled with handles private to the registrant to protect against unwanted modification. Also, every Linux kernel module can avoid unloading with a module use count.

To show most REG features, three sample modules are included in the installation tarball.

4 Implemented Models

A range of models have already been implemented within the RSBAC framework. They can be selected at kernel configuration and then be combined for an individual protection profile.

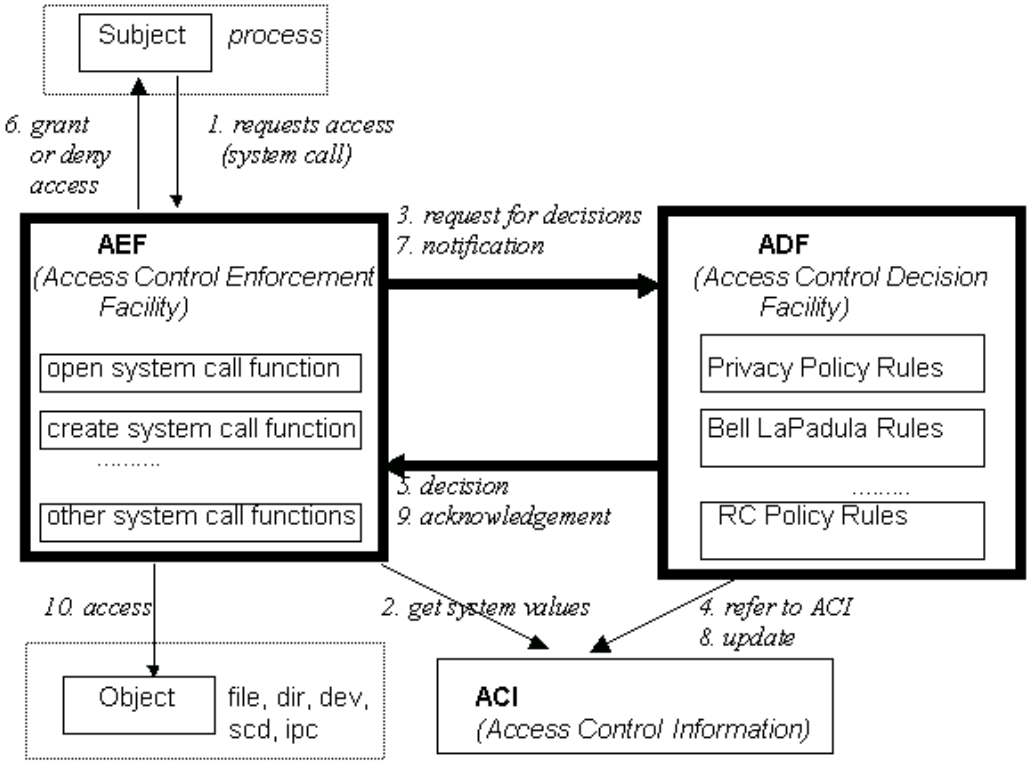

Figure 1: RSBAC Architecture

4.1 MAC

Mandatory Access Control (MAC) stands for the Bell – La Padula model implementation. There are 253 classification levels and 64 categories available.

For those programs that are not MAC aware – like most *nix programs –, current security levels can be automatically adjusted as necessary, but within read and write level boundaries.

4.2 FC

The simple Functional Control (FC) role model defines three roles, 'Normal User', 'Security Officer' and 'System Administrator', and three types, 'General', 'Security', 'System'.

Normal users may access general data, security officers general and security data and system administrators general and system data.

4.3 SIM

The Security Information Modification (SIM) model only allows write access to objects marked as 'Security Data' for users marked as 'Security Officers'.

4.4 PM

Simone Fischer-Hübner's complex Privacy Model (PM) can control the processing of personal data in accordance to the EU privacy laws.

This model defines object classes with sets of purposes, tasks, necessary accesses and much more. Please see e.g. [FiHue97] or [FiHueOtt98] for more information.

4.5 MS

Malware Scanning (MS) is not an access control model – it is an on-access malware scanner prototype in the kernel, which monitors file and socket read accesses. For performance, file scanning results are stored persistently (with version) and reset on all write accesses. Due to its prototype status, it only detects a few DOS and Linux viruses.

The MS design and module are described in [OttFiSwi98]

4.6 FF

The File Flags module provides inheritable flags for FILE, DIR, FIFO and SYMLINK objects.

Current flags are: read_only, execute_only, search_only, write_only, secure_delete, no_execute, no_delete_or_rename (not inherited), append_only and add_inherited (not inherited itself). The add_inherited flag denotes, whether the flag values of an object's parent object are added to its own flags.

4.7 AUTH

The authentication model (AUTH) restricts the capability of a process to CHANGE_OWNER. Only user IDs a process has an AUTH capability for can be reached, all other setuid requests are denied. AUTH thus controls, under which user IDs programs can be executed, and easily restricts login paths to the system.

AUTH capabilities can be set at the program file and are then inherited at execution, or they can be set directly on the process by other processes, which have a special flag auth_may_set_cap set. Additionally, there is a shortcut flag auth_may_setuid, which turns the capability check for this program off.

The capability setting scheme makes a daemon based authentication enforcable: an authentication daemon can set only those capabilities for a process, which the process has successfully authenticated for.

As an important base model, AUTH is the only model which is not independent of all others: all changes to AUTH settings are controlled by requests to ADF.

4.8 RC

The Role Compatibility model (RC) defines 64 roles and 64 types per target type. For ease of use, filesystem targets (FILE, DIR, FIFO, SYMLINK) share the same RC type set. For each pair of role and type, a compatibility vector of request types is defined. A subject may access an object with a request type, if the subject-object compatibility vector has the bit for this request set.

To allow control of requests with target NONE, those requests are checked against the SCD target 'other'.

Every user gets one default role assigned. Additional to types, roles can also be compatible with other roles, which means, a process running with a role can change to all compatible roles. Role compatibility thus defines a chain, possibly a circle, of roles that can be reached from a certain role.

Roles can not only be assigned to users, but also to program files. This can be done temporarily from execution start to the first setuid, via the initial_role attribute, or permanently with the force_role attribute. The latter also controls some special cases for role assignments. Initial roles are typically used for login programs, forced roles for administration tools or daemons.

For administration, there is a powerful separation of administration duty scheme, which e.g. allows to create closed or overlapping workgroups. For this, the role vectors admin_roles and assign_roles are defined for roles, and the additional access rights admin, assign, access_control and supervisor are defined for all types. The scheme is, like all other model details, described in the online model description at [RSBAC].

With its level of abstraction and design for *nix needs, the RC model gives a fast and flexible access control setup. It is thus recommended for most purposes.

4.9 ACL

Access Control Lists (ACL) define, what subject may access which object with which request types. They are always attached to objects. Subjects can be RC roles, thus extending the RC model, individual users or ACL groups.

Every user is allowed to define individual global or private groups of users. Global groups can also be used for administration by other users, private groups are unusable for those. In a simple scenario, one user could administrate a set of global groups for all others.

However, it is also possible to e.g. setup workgroups, where the group leader defines all group memberships, but a system wide security officer assigns all necessary access rights for this group.

If there is no ACL entry for a subject at an object, the object parent's ACL entries are used, but filtered through the object's inheritance mask. On top of all object trees, there is a default ACL for each target type. The whole inheritance scheme is similar to that of a well known traditional PC network system.

For administration, there are three special access rights: Access Control allows to grant or revoke all standard rights, Forward allows to forward the standard rights you have to others, and Supervisor allows everything. In default kernel config, the Supervisor right can never be masked out.

Like in RC model, to allow control of requests with target NONE, those requests are checked against the SCD target 'other'.

The ACL model is recommended in those cases, where the RC model role or type abstraction is not sufficient to cover all necessary access control settings. However, it is much more difficult to keep

the overview of a complex ACL setup than of an RC setup.

5 Installation under Linux

All newer RSBAC versions for Linux come in three parts: a kernel files tar archive, a kernel patch and an administration tools package. In most cases, all three parts are needed and must be of the same version.

5.1 Linux Kernel

The kernel files tar archive is simply unpacked in the kernel source tree. Then the patch for the target kernel version is applied using the patch utility. After kernel configuration (RSBAC default settings are fine for beginners), call 'touch Makefile', compile and install the new kernel and modules, if any.

Specially for beginners, it can be useful to also compile and install a second kernel with Maintenance Mode enabled, or to enable Soft Mode in the first one. These options can give you emergency access after an administration error.

5.2 Administration tools

Just untar the tar archive, ./configure, make and make install. If your kernel source tree is not at /usr/src/linux, use the –with-kerneldir configure option.

5.3 First Boot

If the recommended AUTH model has been included, you will have to use the kernel parameter rsbac_auth_enable_login when for the first time booting an RSBAC kernel. This parameter makes the init code set the auth_may_setuid flag for /bin/login to allow this program to CHANGE_OWNER.

Also with AUTH, several daemons will fail to run, because they are not allowed to setuid to their designated user IDs. The necessary AUTH capabilities will have to be set later.

After boot, you should login as root and create a user account with ID 400, which is the Security Officer etc. account in the default settings. You will need it for administration.

As a first task, the Security Officer (ID 400) should now set the AUTH capabilities for the failing daemons with 'rsbac_fd_menu filename'. You will find the necessary values from the denied CHANGE_OWNER requests in the system log.

6 Administration

6.1 Attributes

Model administration is mostly setting various attributes for objects, e.g. RC types, and model specific items, e.g. RC roles.

6.2 Command Line Tools

For all settings, there are command line tools available. Most of the tools have a set of options and parameters, which are displayed when they are called without any parameter. Figure 2 on page 7 shows such a help screen.

6.3 Menues

Most RSBAC settings can also be done with administration menues, which are strongly recommended for interactive use. Figure 3 on page 7 shows the RSBAC main menu screen.

7 Usage Areas

The RSBAC system can be useful in many environments. Some examples for workstations and servers are given here.

7.1 Workstations

On a workstation, the main goal would be to protect against unwanted configuration changes, as well as against any type of malware infection.

A given system setup can be easily maintained in its original state, reducing administration work.

7.2 Servers

Usually, the first step to secure a server system is to protect its executables, libraries and configuration files against unauthorized modifications. After that, all services can be encapsulated into individual sandboxes.

Examples of servers that need service encapsulation or compartmentation are:

Firewalls: DNS and mail forwarders, Web and FTP proxies

(Virtual) Webservers: Apache, Zope etc., CGIs, separation of virtual domains

(Virtual) Mail Servers: Sendmail, QMail, Postfix, POP3, IMAP, mailing lists, separation of mail areas

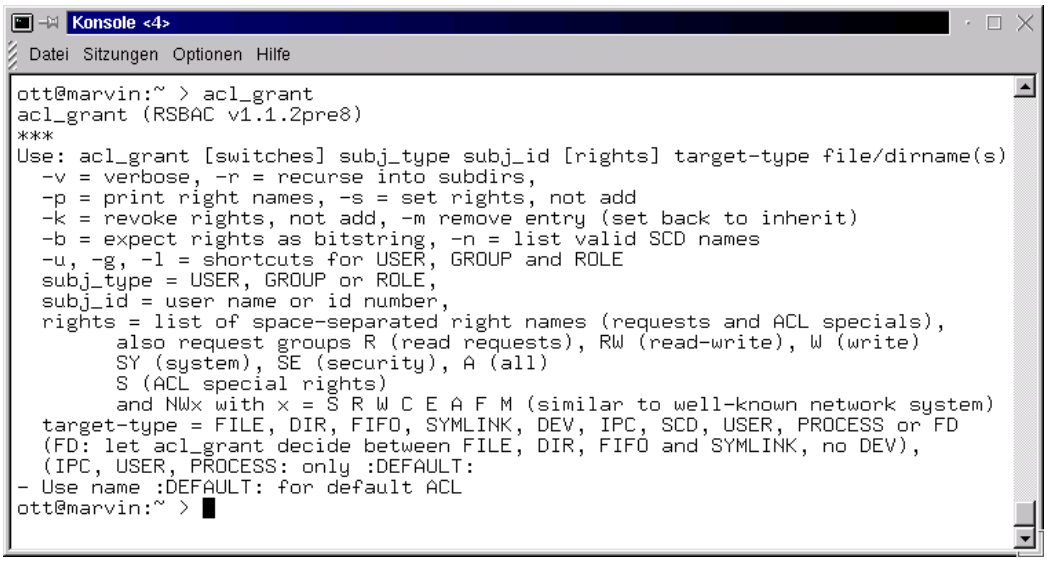

Figure 2: acl_grant commandline tool help screen.

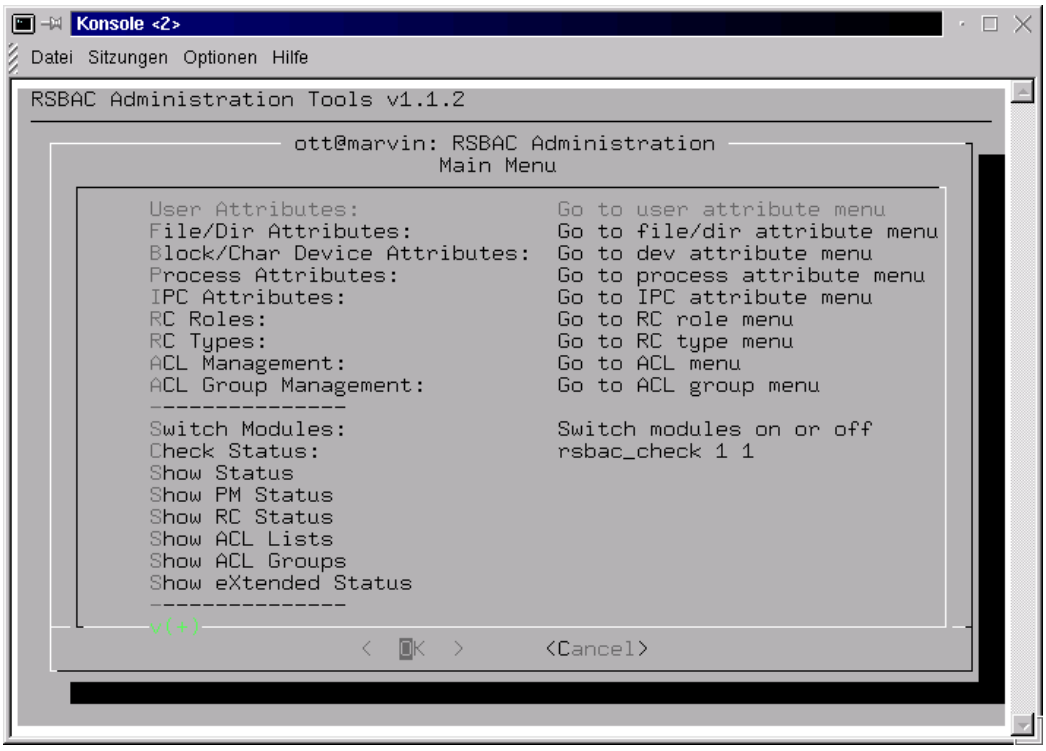

Figure 3: RSBAC Main Menu.

File Servers: Samba, Coda, separation of organizational areas like workgroups, etc.

Application Servers: separation of user accounts, protection against malware or user attacks

8 Practical Experience

During several years of RSBAC development, a lot of experience with the system has been gained.

8.1 Running Systems

Compuniverse Linux Firewalls have been delivered and have since then been running with RSBAC for over a year. They show that base protection and service encapsulation are possible without drawbacks in usability. The RSBAC models used are AUTH, FF and, most of all, RC. Some software has been specially selected for easier protection, e.g. a POP3 server with a separate authentication daemon.

Feedback by email or on the RSBAC mailing list shows many server systems running with RSBAC, some of them in areas with special security needs. Also, there are now two Linux distributions configured and delivered with RSBAC, ALTLinux Castle and Kaladix.

8.2 Stability

In uni-processor mode, RSBAC has shown to be very stable for over a year. In SMP mode, versions up to 1.1.0 had some stability problems on several sample systems, version 1.1.1 had only very few problems. The remaining problems should be solved in 1.1.2-pre9, but are still tested.

8.3 Performance

The main performance influences are the number and dynamic change of attribute objects, the number and types of decision modules and, of course, the amount of logging.

Kernel compile time benchmarks on standard 2.4.6 kernel sources in default settings have been run for RSBAC version 1.1.2-pre8 with kernel 2.4.6. The test system had one Celeron CPU with 333 MHz, 256MB RAM and different RSBAC configurations. Each single test consisted of three 'make bzImage' runs, measured by the 'time' utility, in single user mode. To eliminate caching issues, one extra test compile was done before the timed compilations. Before each run, a 'make clean' was called.

Table 1 on page 9 shows the average times in seconds that were produced. The significant kernel time increase with all options is mostly due to the MS module with read check enabled, which marks all files ever read as scanned and thus produces a huge amount of attribute objects in large lists. Lookups in large lists are slow.

9 Online Ressources

All sources and a lot of documentation are available online at the RSBAC homepage at http://www.rsbac.org.

There is also an RSBAC mailing list for discussion, bug reports and update notes. Postings to the mailing list go to rsbac@rsbac.org, requests go to majordomo@rsbac.org. A Web interface with archive is available at http://www.compuniverse.de/lwgate/rsbac.

10 Outlook

Important changes are planned for the next major release 1.2.0. Some of them are:

- Real network access control: socket templates and socket targets, new requests BIND, CONNECT, etc.
- Better user authentication: kernel space user management(?), RSBAC standard AUTH daemon(?), biometric authentication(?)
- PM overhaul with menues
- Filesystem redirection support(?)
- Tool for automatic rule creation from access logs
- Samba integration of ACL model

References

[Abrams+90] Abrams, M. D., Eggers, K. W., La Padula, L. J., Olson, I. M., A Generalized Framework for Access Control: An Informal Description, Proceedings of the 13th National Computer Security Conference, Oktober 1990

[CU] Brauch, K., Ott, A., Compuniverse Homepage, http://www.compuniverse.de

RSBAC options	Total time	Kernel+User	Kernel	User/Process
none / clean kernel	711.75	711.74	34.83	676.91
Maint kernel (no mods, no debug)	719.09 (+1.03%)	719.09 (+1.03%)	41.02 (+17.77%)	678.07 (+0.17%)
Maint kernel (no modules)	719.20 (+1.05%)	719.19 (+1.05%)	39.04 (+12.09%)	680.15 (+0.48%)
RC + AUTH, no other options	719.36 (+1.07%)	719.35 (+1.07%)	45.41 (+30.38%)	673.94 (-0.44%)
AUTH + ACL, no other options	721.18 (+1.32%)	721.19 (+1.33%)	44.56 (+27.94%)	676.63 (-0.04%)
REG+FF+RC+AUTH+ACL, Net support, ind. Log (def. config)	729.33 (+2.47%)	729.33 (+2.47%)	52.76 (+51.48%)	676.57 (-0.05%)
All models + options, except MS	763.35 (+7.25%)	763.07 (+7.21%)	81.63 (+134.37%)	681.44 (+0.67%)
All models and options	854.69 (+20.08%)	854.21 (+20.02%)	169.65 (+387.08%)	684.56 (+1.13%)

Table 1: Standard 2.4.6 kernel source compile benchmarks with different RSBAC options. Times are given in seconds.

[EU95] Directive 95/46/EC of the European Parliament and of the Council, On the protection of individuals with regard to the processing of personal data and on the free movement of such data, Brussels, 1995

[FiHue97] Fischer-Hübner, S., A Formal Task-based Privacy Model and its Implementation: An updated Report, Second Nordic Workshop on Secure Computer Systems (NORDSEC'97), Helsinki, 1997

[FiHueOtt98] Fischer-Hübner, S., Ott, A., From a Formal Privacy Model to its Implementation, Proceedings of the 21st National Information Systems Security Conference (NISSC '98), Arlington, VA, 1998, http://www.rsbac.org/niss98.htm

[LaPadula95] La Padula, L. J., Rule Set Modeling of a Trusted Computer System, Essay, in: Information Security: An Integrated Collection of Essays, Hrsg.: Abrams, M. D., Jajodia, S., Podell, H. J., IEEE Computer Society Press, 1995

[OttFiSwi98] Ott, A., Fischer-Hübner, S., Swimmer, M., Approaches to Integrated Malware Detection and Avoidance", Proceedings of the 3rdNordic Workshop on Secure IT Systems, Trondheim, November 5-6, 1998, http://www.rsbac.org/nordse98.htm

[RSBAC] Ott, A., RSBAC Homepage, http://www.rsbac.org