Rule Set Based Access Control (RSBAC)

Linux Kernel Security Extension

Linux Kongress 2004 - One Day Workshop

Amon Ott <ao@rsbac.org>

Contents:

1 Introduction to RSBAC
1.1 History
1.2 Motivation
1.3 Design Goals

2 Overview of RSBAC

3 Architecture and Implementation of the Framework
3.1 Subjects and Objects
3.2 List of Requests with Targets
3.3 Architectural Diagram
3.4 Module Registration (REG)
3.5 Network Templates

Contents II:

4 Selection of Implemented Models
4.1 Authentication Enforcement (AUTH)
4.2 Role Compatibility (RC)
4.3 Access Control Lists (ACL)
4.4 File Flags (FF)
4.5 Linux Capabilities (CAP)
4.6 Process Jails (JAIL)
4.7 Resource Control (RES)
4.8 Pageexec Support (PAX)

Contents III:

5 Installation
5.1 Linux Kernel
5.2 Administration tools
5.3 First Boot

6 Administration
6.1 Command Line Tools
6.2 Menues

7 Usage Areas
7.1 Workstations
7.2 Servers

Contents IV:

8 Practical Experience
8.1 Running Systems
8.2 Stability
8.3 Performance

9 Online Resources

10 How to Identify Security Requirements on a Server
10.1 System Base
10.2 Services
10.3 Users, User IDs and Paths
10.4 Logging

Contents V:

11 Selecting a Security Model Combination
11.1 General Criteria
11.2 Model Specifics
11.3 Experiences

Contents VI:

12 Breaking the Requirements into Model Specific Designs
12.1 Base Protection and Service Encapsulation
12.2 AUTH
12.3 FF
12.4 JAIL
12.5 RC
12.6 ACL
12.7 Logging
12.8 Special RSBAC Goodies

Contents VII:

13 Sample Configuration
13.1 Select Simple Server Type
13.2 Specify Requirements
13.3 Select Models
13.4 Design a Configuration
13.5 Implement It

14 Improvement Discussion
14.1 Outlook on v1.2.4
14.2 ???

Contents VIII:

15 Ending It Up

15.1 Conclusion: What we learned
15.2 How to Go On
15.3 Final Questions.

1 Introduction

1.1 History

1.2 Motivation
1.3 Design Goals

1.1 Introduction: History

- RSBAC Project started as Master Thesis in November 1996
- First public RSBAC version 0.9 for Linux kernel 2.0.30 on January, 9, 1998
- Current stable release 1.2.3 for kernels 2.4.26-27 and 2.6.6-8
- 1.2.4 with many changes (see Outlook)

1.2 Introduction: Motivation

- Classic Linux/Unix Access Control is insecure
  - Small Granularity
  - Discrete Control
    - Trusted user?
    - Malware: Invitation to Trojans and Viruses
  - Superuser root
    - Full Access
    - Too often needed
    - Too many exploits (root kits, kernel module attacks etc.)
- Better models for other administration goals
- Flexible Model selection and combination
- Good portability.

2 Overview of RSBAC

- Free Open Source (GPL) Linux kernel security extension

- Independent of governments and big companies
- Several well-known and new security models, e.g. MAC, ACL and RC
- Control over individual user and program network accesses
- Any combination of models possible
- Easily extensible: write your own model for runtime registration.

2 Overview of RSBAC II

- Support for current 2.4 and 2.6 kernels
- Stable for production use since March 2000
- Several publications (see Homepage)
- Linux distributions with RSBAC: Adamantix and Gentoo Hardened
- Debian kernel patch package, Sniffix Live CD System, Simple Live-CD
- Outdated Linux distributions with RSBAC: ALTLinux Castle and Kaladix.

2 Overview of RSBAC III

- Access Control Framework for current Linux Kernels

- Based on Generalized Framework for Access Control (GFAC) by Abrams and LaPadula
- Flexible structure
  - Separation between enforcement (AEF), decision (ADF) and access control information (ACI)
  - Only AEF and part of ACI system dependent
  - Almost any type of model supportable
  - Model independent -> meta policy
  - Runtime Module Registration (REG)

2 Overview of RSBAC IV

- Powerful logging system
  - Request and decision based
  - User based
  - Program based
  - Object based.

3 Architecture and Implementation of the Framework

3.1 Subjects and Objects
3.2 List of Requests with Targets
3.3 Architectural Diagram
3.4 Module Registration (REG)
3.5 Network Templates

3.1 Architecture: Subjects and Objects

- Subjects:
  - Processes acting on behalf of users,
  - executing one program file with a set of dynamic libraries

- Object Types (Target Types):
  - FILE
  - DIR
  - FIFO
  - SYMLINK
  - DEV (devices by block/char and major:minor)
  - IPC (Inter Process Communication)
  - SCD (System Control Data)
  - USER
  - PROCESS
  - NETDEV
  - NETTEMP
  - NETOBJ

3.2 Architecture: List of Requests

- Requests:
  - Abstraction of what a subject wants to do with an object

- 46 Request Types:

R_ADD_TO_KERNEL: NONE
R_ALTER: IPC
R_APPEND_OPEN: FILE, FIFO, DEV, IPC
R_CHANGE_GROUP: FILE, DIR, FIFO, SYMLINK, IPC, PROCESS, NONE
R_CHANGE_OWNER: FILE, DIR, FIFO, SYMLINK, IPC, PROCESS, NONE
R_CHANGE_DAC_EFF_OWNER: PROCESS
R_CHANGE_DAC_FS_OWNER: PROCESS
R_CHDIR: DIR
R_CLONE: PROCESS
R_CLOSE: FILE, DIR, FIFO, DEV, IPC, NETOBJ

3.2 Architecture: List of Requests II

R_CREATE: DIR (where), IPC, NETTEMP, NETOBJ
R_DELETE: FILE, DIR, FIFO, SYMLINK, IPC, NETTEMP, NETOBJ
R_EXECUTE: FILE
R_GET_PERMISSIONS_DATA: FILE, DIR, FIFO, SYMLINK, IPC, SCD
R_GET_STATUS_DATA: FILE, DIR, FIFO, SYMLINK, IPC, SCD, PROCESS, NETDEV
R_LINK_HARD: FILE, FIFO, SYMLINK
R_MODIFY_ACCESS_DATA: FILE, DIR, FIFO, SYMLINK
R_MODIFY_ATTRIBUTE: All target types
R_MODIFY_PERMISSIONS_DATA: FILE, DIR, FIFO, SYMLINK, IPC, SCD, NONE
R_MODIFY_SYSTEM_DATA: SCD, PROCESS, NETDEV
R_MOUNT: FILE, DIR, DEV
R_READ: FILE, DIR, FIFO, DEV, IPC, NETTEMP, NETOBJ
R_READ_ATTRIBUTE: All target types
R_READ_OPEN: FILE, FIFO, DEV, IPC

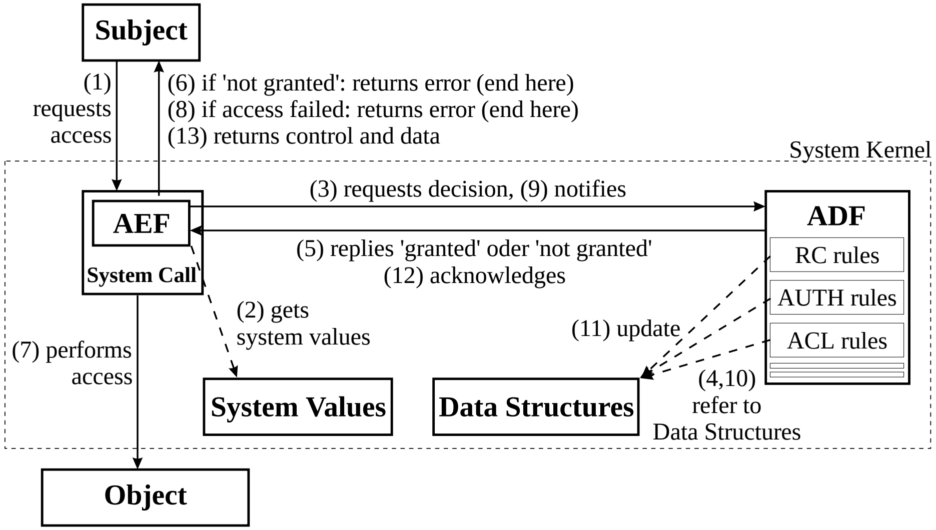
3.2 Architecture: List of Requests III

R_READ_WRITE_OPEN: FILE, FIFO, DEV, IPC
R_REMOVE_FROM_KERNEL: NONE
R_RENAME: FILE, DIR, FIFO, SYMLINK
R_SEARCH: DIR, SYMLINK
R_SEND_SIGNAL: PROCESS
R_SHUTDOWN: NONE
R_SWITCH_LOG: NONE
R_SWITCH_MODULE: NONE
R_TERMINATE: PROCESS (notify only)
R_TRACE: PROCESS
R_TRUNCATE: FILE
R_UMOUNT: FILE, DIR, DEV
R_WRITE: FILE, DIR, FIFO, DEV, SCD, NETTEMP, NETOBJ
R_WRITE_OPEN: FILE, FIFO, DEV, IPC
R_MAP_EXEC: FILE, NONE

3.2 Architecture: List of Requests IV

R_BIND: NETDEV, NETOBJ
R_CONNECT: NETOBJ
R_LISTEN: NETOBJ
R_ACCEPT: NETOBJ
R_SEND: NETOBJ
R_RECEIVE: NETOBJ

3.3 Architectural Diagram



Subject
(1) requests access
(6) if 'not granted': returns error (end here)
(8) if access failed: returns error (end here)
(13) returns control and data
System Kernel
AEF
System Call
(3) requests decision, (9) notifies
ADF
RC rules
AUTH rules
ACL rules
(5) replies 'granted' oder 'not granted'
(12) acknowledges
(2) gets system values
(11) update
(7) performs access
System Values
Data Structures
(4,10) refer to Data Structures
Object

3.4 Module Registration (REG)

- Runtime registration of decision functions (Rule Sets) and system calls
- Model implementation e.g. as kernel module
- Add or remove models, syscalls or generic (persistent) lists in a running system
- Easy control of module removal by the module itself
- Sample modules provided.

3.5 Network Templates

- Description of network endpoints
  - Ordering Number
  - Name (for human use only)
  - Address family (UNIX, INET, IPX, ...)
  - Address (E.g. 192.168.10.0 or "/dev/log")
  - Valid length (e.g. 24 Bits or 8 Byte)
  - Type (ANY, STREAM, DGRAM, ...)
  - Protocol (ICMP, TCP, UDP, ...)
  - Local network device (E.g. eth0)
  - Min and max port (E.g 1024-65535)

- Attribute values attached to templates
- Persistent default values for NETOBJ attributes

- Matched from lowest to highest template ordering number
- Used for local and remote endpoint, depending on request type.

3.5 Network Templates II: Examples

- Only apache may bind to port 80 at eth0

- Proxy may only connect to external addresses, not LAN
- Proxy may only accept connections from internal addresses

- Local users may only connect to mail and proxy server
- Local users (including root) may only use network families UNIX and INET.

4 Selection of Implemented Models

4.1 Authentication Enforcement (AUTH)
4.2 Role Compatibility (RC)
4.3 Access Control Lists (ACL)
4.4 File Flags (FF)
4.5 Linux Capabilities (CAP)
4.6 Process Jails (JAIL)
4.7 Resource Control (RES)
4.8 Pageexec Support (PAX)

4.1 Models: Authentication (AUTH)

- Restriction of CHANGE_OWNER with target PROCESS (setuid)

- CHANGE_OWNER capabilities (inherited from file to process): sets of reachable user IDs

- auth_may_setuid and auth_may_set_cap

- Daemon based authentication enforcable:
  - Process authenticates against daemon
  - Daemon sets capability for auth'd user at process
  - Process calls setuid.

4.1 Models: AUTH II

- Limited lifetime of all AUTH capabilities
- New in 1.2.2: Capabilities for effective and fs uids
- New in 1.2.3: AUTH learning mode.

4.2 Models: Role Compatibility (RC)

- Role and type based model:
  - User default role
  - Process current role
  - Object type
- Compatibility of roles
  - with object types (access rights in RSBAC framework granularity)
  - with other roles (change role actively)
- Forced and Initial Roles for program files

4.2 Models: Role Compatibility (RC) II

- Separation of Administration Duties
  - Admin Roles
  - Assign Roles
  - Additional access rights: Admin, Assign, Access Control, Supervisor
- Lifetime limits for all compatibility settings.

4.3 Models: Access Control Lists (ACL)

- What subject may access which object with which requests
- Subjects:
  - RC roles (!)
  - Users
  - ACL Groups
- ACL Groups of users:
  - All users can have individual groups
  - Private and global groups
- Inheritance with masks (similar to Netware 3.xx)
- Default ACLs on top of hierarchy.

4.3 Models: Access Control Lists II

- Special Rights for administration:
  - Access Control
  - Forward
  - Supervisor
- Lifetime limits for all ACL entries and group memberships
- New in 1.2.3: ACL learning mode.

4.4 Models: File Flags (FF)

- Inheritable FILE, DIR, FIFO and SYMLINK attributes
- Valid for all users
- e.g. read-only, no-execute, secure-delete, append-only.

4.5 Models: Linux Capabilities (CAP)

- Minimum and maximum capability sets for users and programs
- Applied at CHANGE_OWNER on processes (setuid) and EXECUTE
- Precedence of Minimum over Maximum Sets
- Precedence of Program over User Sets
- Limit rights of root programs or extend rights of normal user programs
- E.g. limit mail server to never change network settings.

4.6 Models: Process Jails (JAIL)

- Preconfigured process encapsulation
- Sealed chroot jails
- No contact to processes outside the jail
- Many further restictions, some optional
- Specially limits administration and network accesses.

4.7 Models: Resource Control (RES)

- Minimum and maximum resource limits for users and programs
- Applied at CHANGE_OWNER on process (setuid) and EXECUTE
- Precedence of Minimum over Maximum Sets
- Precedence of Program over User Sets
- Only management of existing Linux process attributes
- Max. file size, number of processes, memory usage, etc.

5 Installation under Linux

5.1 Linux Kernel
5.2 Administration tools
5.3 First Boot

4.8 Models: Pageexec (PAX)

- Management of process attributes for PaX kernel security extension
- PaX protects from common attack types against buggy programs
- Special protection against inserted program code
- More info: pax.grsecurity.net.

5 Installation for Linux

- Linux Kernel (pre-patched)
  - Extract kernel source tar archive
  - Configure, touch Makefile, compile and install
  - RSBAC normal and maint kernels / Soft Mode
- Linux Kernel (patch yourself)
  - Extract RSBAC tar archive in kernel dir
  - Patch kernel (with patch-x.y.z-va.b.c.gz)
  - Apply bugfixes
  - Configure, touch Makefile, compile and install
  - RSBAC normal and maint kernels / Soft Mode
- Administration tools
  - Extract tar archive
  - ./configure && make && make install

5 Installation for Linux II

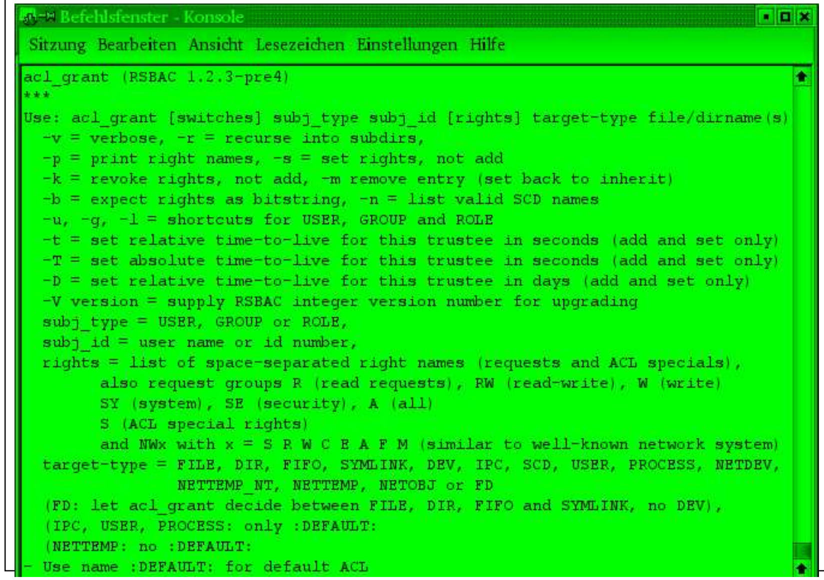
- First Boot
  - Kernel parameter rsbac_auth_enable_login
  - Add user 400 (Security Officer etc.)
  - Adjust AUTH capabilities for failed services or use AUTH learning mode.

6 Administration

6.1 Command Line Tools
6.2 Menues

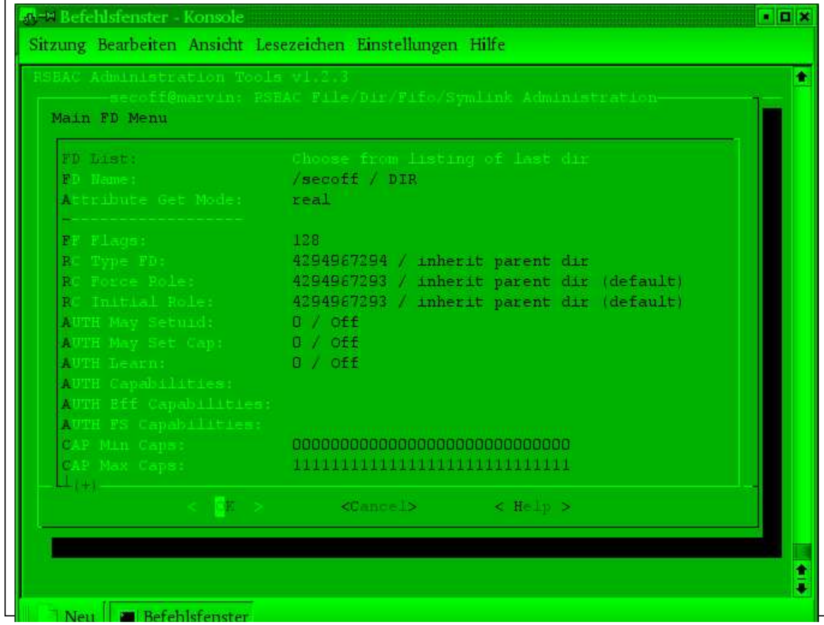
6.1 Administration: Command Line

- General and Model specific (RC, AUTH, ACL)



```
acl_grant (RSBAC 1.2.3-pre4)
***
Use: acl_grant [switches] subj_type subj_id [rights] target-type file/dirname(s)
  -v = verbose, -r = recurse into subdirs,
  -p = print right names, -s = set rights, not add
  -k = revoke rights, not add, -m remove entry (set back to inherit)
  -b = expect rights as bitstring, -n = list valid SCD names
  -u, -g, -l = shortcuts for USER, GROUP and ROLE
  -t = set relative time-to-live for this trustee in seconds (add and set only)
  -T = set absolute time-to-live for this trustee in seconds (add and set only)
  -D = set relative time-to-live for this trustee in days (add and set only)
  -V version = supply RSBAC integer version number for upgrading
  subj_type = USER, GROUP or ROLE,
  subj_id = user name or id number,
  rights = list of space-separated right names (requests and ACL specials),
        also request groups R (read requests), RW (read-write), W (write)
        SY (system), SE (security), A (all)
        S (ACL special rights)
        and NWx with x = S R W C E A F M (similar to well-known network system)
  target-type = FILE, DIR, FIFO, SYMLINK, DEV, IPC, SCD, USER, PROCESS, NETDEV,
                NETTEMP_NT, NETTEMP, NETOBJ or FD
  (FD: let acl_grant decide between FILE, DIR, FIFO and SYMLINK, no DEV),
  (IPC, USER, PROCESS: only :DEFAULT:
  (NETTEMP: no :DEFAULT:
- Use name :DEFAULT: for default ACL
```

6.2 Administration: Menues



```
RSBAC Administration Tools v1.2.3
      secoff@marvin: RSBAC File/Dir/Fifo/Symlink Administration
 Main FD Menu

  FD List:                    Choose from listing of last dir
  FD Name:                    /secoff / DIR
  Attribute Get Mode:         real
  --------------------
  FF Flags:                   128
  RC Type FD:                 4294967294 / inherit parent dir
  RC Force Role:              4294967293 / inherit parent dir (default)
  RC Initial Role:            4294967293 / inherit parent dir (default)
  AUTH May Setuid:            0 / off
  AUTH May Set Cap:           0 / off
  AUTH Learn:                 0 / off
  AUTH Capabilities:
  AUTH Eff Capabilities:
  AUTH FS Capabilities:
  CAP Min Caps:               00000000000000000000000000000000
  CAP Max Caps:               11111111111111111111111111111111
 (+)
```

7 Areas of use

7.1 Workstations
7.2 Server systems

7.1 Areas of use: Workstations

- Protection against unwanted configuration changes
- Malware protection
- Reduced administration work.

7.2 Areas of use: Server Systems

- Encapsulation of services
- Need-to-Know principle
- Malware protection
- Firewalls
  - DNS, Proxies, etc.
  - Advanced Protection of base system
- (Virtual) Webservers
  - Apache, Zope etc.
  - Separation of domains
  - Protection of critical data
  - Encapsulation of CGIs.

7.2 Areas of use: Server Systems II

- (Virtual) mail servers
  - sendmail, postfix, qmail, POP3, IMAP, Mailing Lists etc.
  - Separation of mail areas
- File servers
  - Samba, Coda, etc.
  - Separation of organizational areas
- Application servers
  - Separation between user accounts
  - Protection against user attacks
- Other servers.

8 Practical Experience

8.1 Running Systems
8.2 Stability
8.3 Performance

8.1 Experience: Running Systems

- Linux distributions Adamantix and Gentoo Hardened with RSBAC
- m-privacy TightGate-Pro
  - Extensive use of RSBAC
  - Application server system for secure Internet access
  - Strong encapsulation of all network services and users
  - Uses most of the models mentioned
- Many other stable production systems worldwide.

8.2 Practical Experience: Stability

- More than four years of very high stability
- SMP systems more than three years of high stability
- Few people reported problems with v1.2.3 on 2.6 kernels

8.3 Practical Experience: Performance

- Performance influences
  - Number and dynamic change of attribute objects
  - Number and type of decision modules
  - Logging
- Benchmarks
  - Celeron 333 system, 2.4.19 kernel, RSBAC 1.2.1
  - Three Linux kernel compile runs each
  - Runtime with framework only: +0.68% (Kernel +11.33%)
  - Runtime with RC, AUTH, network, logging enabled: +2.30% (kernel +43.02%)
  - Runtime with REG, FF, RC, AUTH, ACL, CAP, network (def. config): +4.21% (kernel +82.47%).

9 Online Ressources

- RSBAC Homepage: http://www.rsbac.org
- Mailing List
  - Requests: rsbac-request@rsbac.org
  - Mails: rsbac@rsbac.org
  - Archive available (see contact page)
- Adamantix
  - http://www.adamantix.org
- Gentoo Hardened Subproject RSBAC
  - http://hardened.gentoo.org/rsbac

10 How to Identify Security Requirements on a Server

10.1 System Base

10.2 Services

10.3 Users, User IDs and Paths

10.4 Logging

10.1 Requirements: System Base

- Filesystem Structure
  - Modification often leads to denial of service
  - -> Find critical elements, e.g. /bin, /etc, /boot, /var
- Executables
  - Liable to virus or trojan infection, possible denial of service
  - -> Identify all (dirs with) executables in the system to be protected
    - /bin, /usr/bin, /sbin, /usr/sbin, several dirs under /usr/lib, ...
  - -> Specify, what files should *not* be executed
    - What is not protected should never be executed, so best chose 'everything else'
- Libraries
  - Same as executables, but different access patterns
  - Files *.so*, some subdirs, e.g. /usr/lib/apache.

10.1 Requirements: System Base II

- Configuration Files
  - Modification can lead to illegal accesses or denial of service
  - -> identify all (dirs with) important configuration files
- Kernel Objects
  - Kernel Images
  - Kernel Module Files
    - Allow only those to be loaded
  - System.map
  - Raw Memory
    - Should never be accessed
- Devices
  - Raw access can bypass access control and lead to almost any problem
  - -> Identify all devices, which can be used to compromise the system (/dev/hda, /dev/mem, ...).

10.1 Requirements: System Base III

- Authentication data
  - Essential for security
  - -> Identify programs which may read or even modify for all users
  - -> e.g. /bin/login, /usr/bin/passwd, /usr/sbin/user{add|mod|del}
  - -> Optional: 'Account Manager' user who may read or even modify
- Network Resources
  - Prevent local service program replacements
  - Limit possible attacks from this on other systems
  - -> Identify local resources, which must only be served by one program
  - -> Identify remote resources needed by services and users

10.1 Requirements: System Base IV

- Other Objects
  - boot files
  - ioports / direct hardware access (X server etc.)
  - log files
  - ...

10.2 Requirements: Services

- Protection of and against all services
- Local services maintain functionality
  - Identify all local services you need (and turn all others off)
- Network services make servers, but are their main vulnerability
  - Identify all network services you need (and turn all others off)
- Identify objects and access patterns for each service
  - Don't worry: a rough approximation gives a good start.

10.3 Requirements: Users, User IDs and Paths

- Identify all user types of the system
  - Local and remote users
  - What services do they use?
- Find all user IDs needed by each service
  - Service users and running IDs (wwwdata, ssh etc.)
  - Ranges of IDs usable
- Identify the user ID paths
  - User login paths (who logs in through which service)
  - Chains of IDs used by services.

10.4 Requirements: Logging

- Detect attacks
- Provide user accountability (who did what)
- Provide a modification history etc.
- -> Identify the users, programs, objects and accesses you would like to know about.

11 Selecting a Security Model Combination

11.1 General Criteria
11.2 Model Specifics
11.3 Experiences

11.1 Model Selection: General Criteria

- Only consider models you really understand
- Think how each model could meet your requirements *before* choosing
  - -> Feedback from requirement break down to models
- Keep it simple:
  - Choose only those models that really give you a benefit
  - Try to keep models distinct - overlaps can be confusing
- Develop a personal order in which to apply each model from easiest to most difficult.

11.2 Model Selection: Model Specifics

- Authentication Enforcement (AUTH)
  - Use for all user ID related things, e.g. to restrict login paths
  - Quite simple
  - Essential
- File Flags (FF)
  - Use for filesystem object protection which is common for all users
  - Pretty simple
  - Recommended for directory structure protection
- Process Jails (JAIL)
  - Easy to use service encapsulation
  - Pretty simple
  - Recommended for all services which need no administrative privileges.

11.2 Model Selection: Model Specs II

- Role Compatibility (RC)
  - Use for all users and objects, which can be generalized into roles and types
  - Use for program based administration
  - Medium level
  - Strongly recommended because of role/type abstraction
- Access Control Lists (ACL)
  - Use whenever you need rights for individual users or objects
  - Use, if you also need discretionary control or individual user groups
  - Medium level, but difficult to keep setup overview
  - Recommended for uses named above

11.3 Model Selection: Personal Experiences

- Typical Combination: AUTH, RC, JAIL, a bit of FF, CAP, RES
- Optional: PAX, DAZ
- ACL mostly unused.

11.2 Model Selection: Model Specs III

- Other Models: CAP, RES, PAX, DAZ, MAC, FC, SIM, PM
  - Only use for specific needs
  - MAC, FC, SIM, PM in most cases not recommended
  - Not treated here.

12 Breaking the Requirements into Model Specific Designs

12.1 Base Protection and Service Encapsulation

12.2 AUTH
12.3 FF
12.4 JAIL
12.5 RC
12.6 ACL
12.7 Logging

12.8 Special RSBAC Goodies

12.1 Base Protection and Service Encapsulation

- Base Protection: Service independent protection of the system base
  - Protect identified system base (see 10.1: Base requirements)
  - Infrastructure and ’fallback’ for service encapsulation
  - Strongly recommended
- Service Encapsulation: ’Sandbox’ around each individual service
  - Minimum access rights
  - For remote access and root account services strongly recommended
  - Other services optional
- No strict separation
  - Service encapsulation uses Base Protection infrastructure.

12.2 Requirements to AUTH: User ID paths

- Define setuid capabilities for all programs
- Follows directly from 10.3: User ID requirements.

12.3 Requirements to FF: Base protection only

- Filesystem infrastructure
  - Set no_rename_or_delete on all important dirs and files (not inherited), e.g. /etc, /bin, /usr/bin, /boot, ...
- Protect executables, libraries, configuration files, kernel objects and boot files
  - Set flags search_only (only applied on dirs) and read_only
  - Optional: set execute_only on binary executables (scripts need READ_OPEN etc.)
- Protect against execution of uncontrolled files
  - Unset flag add_inherited on all objects named above
  - Set flag no_execute on / (or e.g. /home only)
- Service encapsulation not possible.

12.4 Requirements to JAIL: Service encapsulation only

- Start each service in a JAIL
  - Use rsbac_jail wrapper program
  - Replace chroot() calls with rsbac_jail() in source
- Allow only required Linux capabilities
- Create sub-jails whenever useful
- Can be used for almost all services.

12.5 Requirements to RC

- Protect executables, libraries, configuration files, kernel objects, boot files and /tmp dirs
  - Define one RC file/dir type for each group
  - Remove unneccessary rights to these types from all defined roles
  - Optional: Define new role ’Configuration’
    - Only role with write access to configuration files
    - Assign to config user or make System Admin role compatible with it
  - Optional: Define new role ’Module Loader’
    - Only role allowed to load modules
    - Can only read libraries and type ’Modules’
    - Set as initial role for insmod etc.
  - Set types for the protected objects

12.5 Requirements to RC II

- Protect against execution of uncontrolled files
  - Remove EXECUTE right to all types except executables
  - Remove MAP_EXEC right to all types except executables and libraries.
- Protect devices
  - Define RC device types, e.g. ’Raw Disk’
  - Define RC roles for specific tasks, e.g. ’Raw Disk Access’ for fsck
  - Remove unneccessary rights to these types from all defined roles
  - Assign specific task roles to programs
  - Set types for the protected objects

12.5 Requirements to RC III

- Authentication data
  - Define RC file/dir types ’Account Data’ and ’Auth Data’
  - Define RC roles ’Authenticate’ and ’Change Auth Data’
  - Set rights:
    - All roles may read account data (e.g. /etc/passwd)
    - Role ’Authenticate’ may also read ’Auth Data’
    - ’Change Auth Data’ may read and write ’Account Data’ and ’Auth Data’
  - Assign roles to identified programs as initial roles or forced roles
- Protect network resources
  - Define network templates for all identified local and remote network endpoints
  - Define RC NETOBJ types, e.g. ’http-remote’
  - Assign network rights to all roles as desired
  - Assign RC NETOBJ types to templates.

12.5 Requirements to RC IV

- Service encapsulation
  - Define RC role(s) for service
    - Copy existing role, e.g. ’General User’
  - Define RC file/dir types for service specific data
    - Log dirs, data, file server areas etc.
  - Define Network Templates and RC NETOBJ types for service specific network resources
  - Set role rights:
    - Access own types as necessary
    - SEARCH, READ_OPEN, READ, CLOSE and MAP_EXEC libraries
    - Only SEARCH ’General Type’ for path resolution
    - Optional: read and write on /tmp dirs (try to avoid)
    - No access to other FD types
    - Device and NETOBJ type access as required

12.5 Requirements to RC V

- Service encapsulation (cont.)
  - Assign roles to service users or program file (root services)
    - User’s default role or program file initial / forced role
  - Optional: Define default process create type for role
    - Protect against signals and tracing by others.

12.6 Requirements to ACL

- Protect executables, libraries, configuration files, kernel objects, boot files and /tmp dirs
  - Set inheritance mask to filter out unneccessary rights to these objects
- Protect against execution of uncontrolled files
  - Explicitly grant SEARCH, READ_OPEN, READ, CLOSE and EXECUTE right for group ’Everyone’ to all executables and libraries
  - Remove EXECUTE right from FD :DEFAULT:
- Protect devices
  - Set inheritance mask to filter out unneccessary rights to these objects
  - Explicitly grant necessary accesses for special task users (or groups / RC roles), e.g. for fsck.

12.6 Requirements to ACL II

- Authentication data
  - Only user, group or RC role based protection possible
  - Set inheritance mask to filter out unneccessary rights to these objects
  - Explicitly grant necessary accesses for special task users (or RC roles)
- Protect network resources
  - Define network templates for all identified local and remote network endpoints
  - Inheritance from NETOBJ to matching template to NETOBJ default ACL
  - -> Set template’s inheritance mask to filter out unneccessary rights to the network objects covered by each template
  - -> Set ACL entries on the templates for all subjects as desired.

12.6 Requirements to ACL III

- Service encapsulation
  - Only user, group or RC role based protection possible
  - Group everyone might have to be replaced by a controlled group
  - Set service user rights:
    - Access own dirs/files as necessary
    - SEARCH, READ_OPEN, READ, CLOSE and MAP_EXEC libraries
    - Only SEARCH :DEFAULT: for path resolution
    - Optional: read and write on /tmp dirs (try to avoid)
    - No access to other FD objects
    - Device access as required.

12.7 Requirements to Logging Setup

- Set individual logging for identified objects and requests
- Set individual user and program logging for identified requests
- Use RSBAC own logging source at /proc/rsbac-info/rmsg for untamperable logging.

12.8 Special RSBAC Goodies II

- Secure delete for sensitive data
  - Use FF flag secure_delete or RC FD type attribute
- AUTH learning mode
  - Let system learn required AUTH capabilities.
- ACL learning mode
  - Add missing ACLs for single users and objects automatically
- Separate logging source
  - Use rsbac_nosyslog and rklogd to log invisible from root
- TTL-Settings
  - Use lifetime limits for many AUTH, RC and ACL settings.

12.8 Special RSBAC Goodies

- Softmode
  - Optimize your setup without locking yourself out
  - Global and individual module softmode
- Individual user (RC role) /tmp dirs with symlink redirection
  - mkdir /tmpdir
  - mkdir /tmpdir/tmp<uid> (mkdir /tmpdir/tmp<role-nr>)
  - rmdir /tmp
  - ln -s /tmpdir/tmp /tmp
- Allow security admins to browse all dirs without suid root
  - Use CAP model to set user min_cap DAC_READ_SEARCH
- Hide other user’s processes
  - Use CAP module’s process hiding
  - Kernel parameter rsbac_cap_process_hiding.

13 Sample System

13.1 Select Simple Server Type:

- Webserver, Proxy Server, Mail or File Server?

13.2 Specify Requirements

- Filesystem Structure
- Executables
- Libraries
- Configuration Files
- Kernel Objects
- Devices
- Authentication data
- Network Resources
- Other Objects.

13 Sample Configuration

13.3 Select Models

13.4 Design a Configuration

13.5 Implement It.

14 Improvement Discussion

14.1 Outlook on v1.2.4
14.2 ???

14.1 Outlook on v1.2.4

- Kernel space user management
  - Full passwd/shadow compatible
  - Fine grained access control by all modules
  - Checking and account logic in kernel only
  - PAM and NSS modules for easy usage
  - Authentication enforcement: only setuid to authenticated uids
  - => Finally taking user control away from ordinary programs
- AUTH daemon for more secure network authentication
  - Alternative to kernel based user management
- Improved learning modes
- Many small changes (see online to-do list)
- ???

14.2 Improvements: ???

- ???

15 Ending It Up

15.1 Conclusion: What We Learned

15.2 How to Go On

15.3 Final Questions.

Rule Set Based Access Control (RSBAC)

Linux Kongress 2004 - One Day Workshop

Amon Ott <ao@rsbac.org>

Thank you!