Package ‘RAppArmor’

May 17, 2016

**Type** Package

**Title** Bindings to AppArmor and Security Related Linux Tools

**Version** 2.0.2

**Author** Jeroen Ooms

**Maintainer** Jeroen Ooms <jeroen.ooms@stat.ucla.edu>

**License** Apache License (== 2.0)

**URL** http://www.jstatsoft.org/v55/i07/
  http://github.com/jeroenooms/RAppArmor#readme

**BugReports** http://github.com/jeroenooms/RAppArmor/issues

**OS_type** unix

**SystemRequirements** linux (>= 3.0), libapparmor-dev (optional)

**Description** Bindings to various methods in the kernel for enforcing security restrictions. AppArmor can apply mandatory access control (MAC) policies on a given task (process) via security profiles with detailed ACL definitions. In addition the package has kernel bindings for setting the process hardware resource limits (rlimit), uid, gid, affinity and priority. The high level R function 'eval.secure' builds on these methods to do dynamic sandboxing: it evaluates a single R expression within a temporary fork which acts as a sandbox by enforcing fine grained restrictions without affecting the main R process. Recent versions on this package can also be installed on systems without libapparmor, in which case some features are automatically disabled.

**VignetteBuilder** R.rsp

**Suggests** testthat, R.rsp

**Depends** tools

**Imports** parallel

**RoxygenNote** 5.0.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2016-05-17 21:34:04
Description

A hat is a subprofile which name starts with a '^'. The difference between hats and profiles is that one can escape (revert) from the hat using the token. Hence this provides more limited security than a profile.

Usage

aa_change_hat(subprofile, magic_token, verbose = TRUE)

Arguments

subprofile character string identifying the subprofile (hat) name (without the "^")
magic_token a number that will be the key to revert out of the hat.
verbose print some C output (TRUE/FALSE)

References


See Also

Other apparmor: aa_change_profile, aa_find_mountpoint, aa_is_enabled
aa_change_profile

Examples

```r
## Not run:
aa_change_profile("testprofile");
aa_getcon();
test <- read.table("/etc/group");
aa_change_hat("testhat", 13337);
aa_getcon();
test <- read.table("/etc/group");
aa_revert_hat(13337);
test <- read.table("/etc/group");

## End(Not run)
```

Description

This function changes the current R process to an AppArmor profile. Note that this generally is a one way process: most profiles explicitly prevent switching into another profile, otherwise it would defeat the purpose.

Usage

```r
aa_change_profile(profile, verbose = TRUE)
```

Arguments

- `profile`: character string with the name of the profile.
- `verbose`: print some C output (TRUE/FALSE)

References


See Also

Other apparmor: `aa_change_hat, aa_find_mountpoint, aa_is_enabled`
aa_find_mountpoint

Examples

```r
## Not run:
test <- read.table("/etc/passwd");
aa_change_profile("testprofile");
aa_getcon();
test <- read.table("/etc/passwd");

## End(Not run)
```

Description

Find the apparmor mountpoint

Usage

```r
aa_find_mountpoint(verbos = TRUE)
```

Arguments

- `verbose`: print some C output (TRUE/FALSE)

References


Ubuntu Manpage: `aa_find_mountpoint` - find where the apparmor interface filesystem is mounted. [http://manpages.ubuntu.com/manpages/precise/man2/aa_find_mountpoint.2.html](http://manpages.ubuntu.com/manpages/precise/man2/aa_find_mountpoint.2.html)

See Also

Other apparmor: `aa_change_hat, aa_change_profile, aa_is_enabled`
aa_getcon

Get AppArmor confinement context for the current task

Description
We can use this function to see if there is an AppArmor profile associated with the current process, and in which mode it current is set (enforce, complain, disable).

Usage
aa_getcon(verbose = TRUE)

Arguments
verbose print some C output (TRUE/FALSE)

Details
Note that in order for this function to do its work, it needs read access to the attributes of the current process. If aa_getcon fails with a permission denied error, it might actually mean that the current process is being confined with a very restrictive profile.

References


aa_is_enabled
Check if AppArmor is Enabled

Description
This function tries to lookup the status of AppArmor in the kernel. However, some confined profiles might not have enough privileges to lookup this status. Also see aa_getcon().

Usage
aa_is_enabled(verbose = TRUE)

Arguments
verbose print some C output (TRUE/FALSE)
References


See Also

Other apparmor: aa_change_hat, aa_change_profile, aa_find_mountpoint

eval.secure

Secure evaluation

Description

Evaluate in a sandboxed environment.

Usage

eval.secure(..., uid, gid, priority, profile, timeout = 60, silent = FALSE, verbose = FALSE, affinity, closeAllConnections = FALSE, RLIMIT_AS, RLIMIT_CORE, RLIMIT_CPU, RLIMIT_DATA, RLIMIT_FSIZE, RLIMIT_MEMLOCK, RLIMIT_MSGQUEUE, RLIMIT_NICE, RLIMIT_Nofile, RLIMIT_NPROC, RLIMIT_RTPrIO, RLIMIT_RtTIME, RLIMIT_SIGPENDING, RLIMIT_STACK)

Arguments

... arguments passed on to eval.
uid integer or name of linux user. See setuid.
gid integer or name of linux group. See setgid.
priority priority. Value between -20 and 20. See setpriority.
profile AppArmor security profile. Has to be preloaded by Linux. See aa_change_profile.
timeout timeout in seconds.
silent suppress output on stdout. See mcparallel.
verbose print some C output (TRUE/FALSE)
affinity which cpu(s) to use. See setaffinity.
closeAllConnections closes (and destroys) all user connections. See closeAllConnections.
RLIMIT_AS hard limit passed on to rlimit_as.
RLIMIT_CORE hard limit passed on to rlimit_core.
RLIMIT_CPU hard limit passed on to rlimit_cpu.
RLIMIT_DATA hard limit passed on to rlimit_data.
eval.secure

- **RLIMIT_FSIZE**: hard limit passed on to `rlimit_fsize`.
- **RLIMIT_MEMLOCK**: hard limit passed on to `rlimit_memlock`.
- **RLIMIT_MSGQUEUE**: hard limit passed on to `rlimit_msgqueue`.
- **RLIMIT_NICE**: hard limit passed on to `rlimit_nice`.
- **RLIMIT_NOFILE**: hard limit passed on to `rlimit_nofile`.
- **RLIMIT_NPROC**: hard limit passed on to `rlimit_nproc`.
- **RLIMIT_RTPRIO**: hard limit passed on to `rlimit_rtprio`.
- **RLIMIT_RTTIME**: hard limit passed on to `rlimit_rtttime`.
- **RLIMIT_SIGPENDING**: hard limit passed on to `rlimit_sigpending`.
- **RLIMIT_STACK**: hard limit passed on to `rlimit_stack`.

**Details**

This function creates a fork, then sets rlimits, uid, gid, priority, apparmor profile where specified, and then evaluates the expression inside the fork. The return object of the evaluation is copied to the parent process and returned by `eval.secure`. After evaluation is done, the fork is immediately killed. If the timeout is reached the fork is also killed and an error is raised.

Evaluation of an expression using `eval.secure` has no side effects on the current R session. Any assignments to the global environment, changes in options, or library loadings done by the evaluation will get lost, as we explicitly want to prevent this. Only the return value of the expression will be copied to the main process. Files saved to disk by the sandboxed evaluation (where allowed by apparmor profile, etc) will also persist.

Note that if the initial process does not have superuser rights, rlimits can only be decreased and setuid/setgid might not work. In this case, specifying an RLIMIT higher than the current value will result in an error. Some of the rlimits can also be specified inside of the apparmor profile. When a rlimit is set both in the profile and through R, the more restrictive one will be effective.

**References**


**Examples**

```r
## Not run:
## Restricting file access ##
eval.secure(list.files("/"))
eval.secure(list.files("/"), profile="r-base")

eval.secure(system("ls /", intern=TRUE))
eval.secure(system("ls /", intern=TRUE), profile="r-base")

## Limiting CPU time ##
cputest <- function()
```
A <- matrix(rnorm(1e7), 1e3);
B <- svd(A);
}

## setTimeLimit doesn't always work:
setTimeLimit(5);
cputest();
setTimeLimit();

#timeout does work:
eval.secure(cputest(), timeout=5)

## Limiting memory ##
A <- matrix(rnorm(1e8), 1e4);
B <- eval.secure(matrix(rnorm(1e8), 1e4), RLIMIT_AS = 100*1024*1024)

## Limiting procs ##
forkbomb <- function(){
  repeat{
    parallel::mcpall(forkbomb());
  }
}

## Forkbomb is mitigated ##
eval.secure(forkbomb(), RLIMIT_NPROC=10)

## End(Not run)

---

**kill**

*Unix Process Utilities*

**Description**

Read or set process properties.

**Usage**

```r
ekil(pid, signal = SIGTERM, verbose = FALSE)
setuid(uid, verbose = FALSE)
getuid()
setgid(gid, verbose = FALSE)
getgid()
setpgid(verbose = FALSE)
```
getpgid()

setpriority(prio, verbose = FALSE)

getpriority()

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>process ID</td>
</tr>
<tr>
<td>signal</td>
<td>kill signal</td>
</tr>
<tr>
<td>verbose</td>
<td>emit some debugging output in C</td>
</tr>
<tr>
<td>uid</td>
<td>user ID</td>
</tr>
<tr>
<td>gid</td>
<td>group ID</td>
</tr>
<tr>
<td>prio</td>
<td>priority value</td>
</tr>
</tbody>
</table>

References


Description

Get and set RLIMIT values of the current process.

Usage

```c
rlimit_as(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_core(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_cpu(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_data(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_fsize(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_memlock(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
rlimit_msgqueue(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)
```
rlimit_nice(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_nofile(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_nproc(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_rtprio(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_rttime(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_sigpending(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

rlimit_stack(hardlim = NULL, softlim = hardlim, pid = 0, verbose = FALSE)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hardlim</td>
<td>hard limit in bytes</td>
</tr>
<tr>
<td>softlim</td>
<td>soft limit in bytes</td>
</tr>
<tr>
<td>pid</td>
<td>id of the target process.</td>
</tr>
<tr>
<td>verbose</td>
<td>print some C output (TRUE/FALSE)</td>
</tr>
</tbody>
</table>

**References**


**Examples**

```r
## Not run:
#load lib
library(RAppArmor)

#current limit
rlimit_as();

#set hard limit
rlimit_as(1e9);

#set separate hard and soft limit.
rlimit_as(1e9, 1e8);
```
# soft limits can be elevated
rlimit_as(soft = 1e7);
rlimit_as(soft = 1e9);

# set other limits
rlimit_core(1e9);
rlimit_data(1e9);
rlimit_fsize(1e9);
rlimit_memlock(10000);
rlimit_msgqueue(1e5);
rlimit_nofile(10);
rlimit_nproc(100);
rlimit_rtttime(1e9);
rlimit_sigpending(1e4);
rlimit_stack(1000);
rlimit_cpu(60);

## End(Not run)
## Not run: testfun <- function(){
  Sys.sleep(3);
  repeat{
    svd(matrix(rnorm(1e6,1e3)));
  }
};
# will be killed after 8 seconds (3s idle, 5s CPU):
system.time(eval_secure(testfun(), RLIMIT_CPU=5));

# will be killed after 5 seconds
system.time(eval_secure(testfun(), timeout=5));
## End(Not run)

---

setaffinity

**Process affinity manipulation**

**Description**

Function to get/set the process’s CPU affinity mask. Affinity mask allows binding a process to a specific core(s) within the machine.

**Usage**

setaffinity(cpus = 1:ncores(), verbose = FALSE)

getaffinity_count(verbos = FALSE)

getaffinity(verbos = FALSE)

ncores()
Arguments

- **cpus**: Which cpu cores to restrict to. Must be vector of integers between 1 and ncores.
- **verbose**: Print some C output (TRUE/FALSE)

Details

Setting a process affinity allows for restricting the process to only use certain cores in the machine. The cores are indexed by the operating system as 1 to ncores. One can lookup ncores using ncores(). Calling setaffinity with no arguments resets the process to use any of the available cores.

Note that setaffinity is different from setting r_limit values in the sense that it is not a one-way process. An unprivileged user can change the process affinity to any value. In order to 'lock' an affinity value, one would have to manipulate Linux capability value for CAP_SYS_NICE.

References


Examples

```r
## Not run:
ncores();
getaffinity();
getaffinity_count();
setaffinity(1); #restricts the process to core number 1.
getaffinity();
setaffinity(); #reset
getaffinity();

## End(Not run)
```

Description

This function loads the 'testthat' package and runs a number of unit tests for RAppArmor. Note that the tests assume that the main process is unconfined. Try running it both as root and as a regular user to cover both cases.

Usage

```r
unittests()
```
Details

Occasionaly, one or two tests might fail due to random fluctuations in available memory, cpu, etc. If this happens, try running the tests again, possibly with less other programs running in the background.

userinfo

Lookup user info

Description

Function looks up uid, gid, and userinfo for a given username.

Usage

userinfo(username, uid, gid)

Arguments

username character name identifying the loginname of the user.
uid integer specifying the uid of the user to lookup.
gid integer specifying the gid to lookup.

Value

a parsed row from /etc/passwd

References


Index

aa_change_hat, 2, 3, 4, 6
aa_change_profile, 2, 3, 4, 6
aa_find_mountpoint, 2, 3, 4, 6
aa_getcon, 5
aa_is_enabled, 2–4, 5
aa_revert_hat (aa_change_hat), 2
affinity (setaffinity), 11

closeAllConnections, 6

eval, 6
eval_secure, 6, 7

getaffinity (setaffinity), 11
getaffinity_count (setaffinity), 11
getgid (kill), 8
getpgid (kill), 8
getpriority (kill), 8
getuid (kill), 8

kill, 8

mcpool, 6

ncore (setaffinity), 11

rlimit_as, 6, 9
rlimit_core, 6
rlimit_core (rlimit_as), 9
rlimit_cpu, 6
rlimit_cpu (rlimit_as), 9
rlimit_data, 6
rlimit_data (rlimit_as), 9
rlimit_fsize, 7
rlimit_fsize (rlimit_as), 9
rlimit_memlock, 7
rlimit_memlock (rlimit_as), 9
rlimit_msgqueue, 7
rlimit_msgqueue (rlimit_as), 9
rlimit_nice, 7
rlimit_nice (rlimit_as), 9

rlimit_nofile, 7
rlimit_nofile (rlimit_as), 9
rlimit_nproc, 7
rlimit_nproc (rlimit_as), 9
rlimit_rtprio, 7
rlimit_rtprio (rlimit_as), 9
rlimit_rtttime, 7
rlimit_rtttime (rlimit_as), 9
rlimit_sigpending, 7
rlimit_sigpending (rlimit_as), 9
rlimit_stack, 7
rlimit_stack (rlimit_as), 9

setaffinity, 6, 11
setgid, 6
setgid (kill), 8
setpgid (kill), 8
setpriority, 6
setpriority (kill), 8
setuid, 6
setuid (kill), 8

unittests, 12
userinfo, 13