

Novell AppArmor (1.2)

Quick Start

NOVELL® QUICK START CARD **

This document helps you understand the main concepts behind Novell® AppArmor—the content of AppArmor profiles. Learn how to create or modify AppArmor profiles. There are three ways to do this:

1. With YaST (graphical or in ncurses mode)
2. Using command line tools
3. Using a text editor (especially for fine-tuning)

Subdomain versus AppArmor

Originally “AppArmor” and the corresponding Linux kernel module were called “Subdomain”. In SLES 9, the kernel module, RPM packages, and tools still have “Subdomain” in their name, but for SLES 10 the move to “AppArmor” will be complete (the kernel module will be renamed to `apparmor.ko`).

AppArmor Modes

Complain

In complain or learning mode, violations of AppArmor profile rules, such as the profiled program accessing files not permitted by the profile, are detected. The violations are permitted, but also logged. This mode is convenient for developing profiles.

Manually activating complain mode (using the command line) adds a flag to the top of the profile so that `/bin/foo` becomes `/bin/foo flags=(complain)`.

enforce

Loading a profile in enforcement mode enforces the policy defined in the profile as well as reports policy violation attempts to syslogd.

Starting and Stopping AppArmor

Use the `rcsubdomain` command with one of the following parameters:

start

Load the kernel module, mount securityfs, parse and load profiles, and start event-log.

stop

Stop event-log, unmount securityfs, and invalidate profiles.

reload

Reload profiles.

status

If AppArmor is enabled, output how many profiles are loaded in complain or enforce mode.

AppArmor Command Line Tools

autodep

Guess basic AppArmor profile requirements. `autodep` creates an approximate profile for the program or application examined. The resulting profile is called “approximate” because it does not necessarily contain all of the profile entries that the program needs to be confined properly.

complain

Set an AppArmor profile to complain mode.

enforce

Set an AppArmor profile to enforce mode from complain mode.

genprof

Generate a profile. When running, you must specify a program to profile. If the specified program is not an absolute path, `genprof` searches the `$PATH` variable. If

a profile does not exist, genprof creates one using autodep.

logprof

Manage AppArmor profiles. logprof is an interactive tool used to review the learning or complain mode output found in the AppArmor syslog entries and to generate new entries in AppArmor profiles.

unconfined

Output a list of processes with tcp or udp ports that do not have AppArmor profiles loaded.

Methods of Profiling

Stand-Alone Profiling

Using genprof. Suitable for profiling small applications.

Systemic Profiling

Suitable for profiling large numbers of programs all at once and for profiling applications that may run “forever.”

To apply systemic profiling, proceed as follows:

1. Create profiles for the individual programs that make up your application (autodep).
2. Put relevant profiles into learning or complain mode.
3. Exercise your application.
4. Analyze the log (logprof).
5. Repeat Steps 3-4.
6. Edit the profiles.
7. Return to enforce mode.
8. Rescan all profiles (`rcsubdomain restart`).

Learning Mode

When using genprof, logprof, or YaST in learning mode, you get several options for how to proceed:

Allow

Grant access.

Deny

Prevent access.

Glob

Modify the directory path to include all files in the suggested directory.

Glob w/Ext

Modify the original directory path while retaining the filename extension. This allows the program to access all files in the suggested directories that end with the specified extension.

Edit

Enable editing of the highlighted line. The new (edited) line appears at the bottom of the list. This option is called *New* in the logprof and genprof command line tools.

Abort

Abort logprof or YaST, losing all rule changes entered so far and leaving all profiles unmodified.

Finish

Close logprof or YaST, saving all rule changes entered so far and modifying all profiles.

Example Profile

```
# a comment about foo.
/usr/bin/foo {
    /bin/mount                ux,
    /dev/{,u}random           r,
    /etc/ld.so.cache           r,
    /etc/foo.conf              r,
    /etc/foo/*                 r,
    /lib/ld-*.so*              x,
    /lib/lib*.so*              r,
    /proc/[0-9]**             r,
    /usr/lib/**                r,
    /tmp/foo.pid               wr,
    /tmp/foo.*                 lrw,
}
# a comment about foo's subprofile, bar.
/usr/bin/foo^bar {
    /lib/ld-*.so*              x,
    /usr/bin/bar               x,
    /var/spool/*               rwl,
}
```

Structure of a Profile

Profiles are simple text files in the `/etc/subdomain.d` directory. They consist of several parts: `#include`, capability entries, rules, and “hats.”

#include

This is the section of an AppArmor profile that refers to an include file, which procures access permissions for programs. By using an include, you can give the program access to directory paths or files that are also required by other programs. Using includes can reduce the size of a profile. It is good practice to select includes when suggested.

To assist you in profiling your applications, AppArmor provides two classes of `#includes`: abstractions and program chunks.

Capability Entries (POSIX.1e)

Capabilities statements are simply the word “capability” followed by the name of the POSIX.1e capability as defined in the `capabilities(7)` man page.

Rules: General Options for Files and Directories

Option	File	Description
read	r	

Option	File	Description
write	w	
link	l	(necessary for deleting=unlink)

Rules: Defining Execute Permissions

For executables that may be called from the confined programs, the profile creating tools ask you for an appropriate mode, which is also reflected directly in the profile itself:

Option	File	Description
Inherit	ix	Stay in the same (parent's) profile.
Profile	px	Requires that a separate profile exists for the executed program.
Unconstrained	ux	Executes the program without a profile. Avoid running programs in unconstrained or unconfined mode for security reasons.

Rules: Paths and Globbing

Glob	Description
*	Substitutes for any number of characters, except /.
**	Substitutes for any number of characters, including /.
?	Substitutes for any single character, except /.
[abc]	Substitutes for the single character a, b, or c.
[a-c]	Substitutes for the single character a, b, or c.
{ ab,cd }	Expand to one rule to match ab and another to match cd.

Hats

An AppArmor profile represents a security policy for an individual program instance or process. It applies to an executable program, but if a portion of the program needs different access permissions than other portions, the program can “change hats” to use a different security context, distinctive from the access of the main program. This is known as a hat or subprofile.

A profile can have an arbitrary number of subprofiles, but there are only two levels: a subprofile cannot have sub-subprofiles. A subprofile is written as a separate profile and named as the containing profile followed by the subprofile name, separated by a ^. Subprofiles must be stored in the same file as the parent profile.

The AppArmor ChangeHat feature requires an application modification to call the `change_hat` function. The supplied

Apache module `mod-change-hat` provides this functionality for the Apache2 Web server that ships with SLES 9 SP3.

Helpful Additions

Subdomain.vim

Subdomain.vim colors:

Blue	#include lines that pull in other Novell AppArmor rules and comments that begin with #
White	Ordinary read access lines
Brown	Capability statements and complain flags
Yellow	Lines that grant write access
Green	Lines that grant execute permission (either ix or px)
Red	Lines that grant unconfined access (ux)
Red Background	Syntax errors that are not loading properly into the AppArmor modules

Autodocumentation

The tool “sitar” provided with SLES 9 SP 3 gathers all system configuration information available from your system and creates comprehensive system documentation. It can be used to document all new and changed profiles.

Logging and Auditing

All AppArmor events are logged using the system's syslog-interface. On top of this infrastructure, event notification can be configured. Currently this feature should be configured using YaST. It is based on severity levels according to `/etc/apparmor/severity.db`. Notification frequency and type of notification (such as e-mail) can be configured.

Use YaST for generating reports in CSV or HTML format.

Directories and Files

<code>/subdomain/</code>	Read access to the kernel module. Get the status of Subdomain or AppArmor.
<code>/etc/apparmor/</code>	Contains a number of configuration files for fine-tuning the behavior of AppArmor.
<code>/etc/subdomain.d/</code>	Location of profiles named with the convention of replacing the / in paths with . (except for the root /) so profiles are easier to manage. For example, the profile for the program <code>/usr/sbin/ntpd</code> is named <code>usr.sbin.ntpd</code> .

/etc/subdomain.d/abstractions/

Abstractions are #includes that are grouped by common application tasks. These tasks include access to authentication mechanisms, access to name service routines, common graphics requirements, and system accounting, for example, base, consoles, kerberosclient, perl, user-mail, user-tmp, authentication, bash, and nameservice.

/etc/subdomain.d/program-chunks/

Program chunks are access controls for specific programs that a system administrator might want to control based on local site policy. Each chunk is used by a single program.

Novell.

Copyright © 2006 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher. Novell is a registered trademark of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners. A trademark symbol (®), TM, etc.) denotes a Novell trademark; an asterisk (*) denotes a third-party trademark.

Created by SUSE® with XSL-FO