

# Novell AppArmor

2.1

[www.novell.com](http://www.novell.com)

April 15, 2011

Novell AppArmor Administration Guide



# ***Novell AppArmor Administration Guide***

All content is copyright © Novell, Inc.

## Legal Notice

This manual is protected under Novell intellectual property rights. By reproducing, duplicating or distributing this manual you explicitly agree to conform to the terms and conditions of this license agreement.

This manual may be freely reproduced, duplicated and distributed either as such or as part of a bundled package in electronic and/or printed format, provided however that the following conditions are fulfilled:

That this copyright notice and the names of authors and contributors appear clearly and distinctively on all reproduced, duplicated and distributed copies. That this manual, specifically for the printed format, is reproduced and/or distributed for noncommercial use only. The express authorization of Novell, Inc must be obtained prior to any other use of any manual or part thereof.

For Novell trademarks, see the Novell Trademark and Service Mark list <http://www.novell.com/company/legal/trademarks/tmlist.html>. \* Linux is a registered trademark of Linus Torvalds. All other third party trademarks are the property of their respective owners. A trademark symbol (®, ™ etc.) denotes a Novell trademark; an asterisk (\*) denotes a third party trademark.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither Novell, Inc., SUSE LINUX Products GmbH, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

# Contents

<b>About This Guide</b>	<b>v</b>
<b>1 Immunizing Programs</b>	<b>1</b>
1.1 Introducing the AppArmor Framework . . . . .	2
1.2 Determining Programs to Immunize . . . . .	4
1.3 Immunizing cron Jobs . . . . .	5
1.4 Immunizing Network Applications . . . . .	5
<b>2 Profile Components and Syntax</b>	<b>11</b>
2.1 Breaking a Novell AppArmor Profile into Its Parts . . . . .	12
2.2 #include Statements . . . . .	14
2.3 Capability Entries (POSIX.1e) . . . . .	15
2.4 Using the Local AppArmor Profile Repository . . . . .	15
<b>3 Building and Managing Profiles with YaST</b>	<b>17</b>
3.1 Adding a Profile Using the Wizard . . . . .	18
3.2 Manually Adding a Profile . . . . .	25
3.3 Editing Profiles . . . . .	26
3.4 Deleting a Profile . . . . .	31
3.5 Updating Profiles from Log Entries . . . . .	32
3.6 Managing Novell AppArmor and Security Event Status . . . . .	33
<b>4 Building Profiles from the Command Line</b>	<b>37</b>
4.1 Checking the AppArmor Module Status . . . . .	37
4.2 Building AppArmor Profiles . . . . .	39
4.3 Adding or Creating an AppArmor Profile . . . . .	40

4.4	Editing an AppArmor Profile . . . . .	40
4.5	Deleting an AppArmor Profile . . . . .	40
4.6	Two Methods of Profiling . . . . .	41
4.7	Paths and Globbing . . . . .	60
4.8	File Permission Access Modes . . . . .	62
4.9	Important Filenames and Directories . . . . .	66
<b>5</b>	<b>Profiling Your Web Applications Using ChangeHat</b>	<b>69</b>
5.1	Apache ChangeHat . . . . .	70
5.2	Configuring Apache for mod_apparmor . . . . .	77
<b>6</b>	<b>Managing Profiled Applications</b>	<b>81</b>
6.1	Monitoring Your Secured Applications . . . . .	81
6.2	Configuring Security Event Notification . . . . .	82
6.3	Configuring Reports . . . . .	85
6.4	Reacting to Security Event Rejections . . . . .	104
6.5	Maintaining Your Security Profiles . . . . .	105
<b>7</b>	<b>Support</b>	<b>107</b>
7.1	Updating Novell AppArmor Online . . . . .	107
7.2	Using the Man Pages . . . . .	107
7.3	For More Information . . . . .	109
7.4	Troubleshooting . . . . .	110
7.5	Reporting Bugs for AppArmor . . . . .	112
<b>A</b>	<b>Background Information on AppArmor Profiling</b>	<b>115</b>
	<b>Glossary</b>	<b>117</b>

# About This Guide

Novell® AppArmor is designed to provide easy-to-use application security for both servers and workstations. Novell AppArmor is an access control system that lets you specify per program which files the program may read, write, and execute. AppArmor secures applications by enforcing good application behavior without relying on attack signatures, so it can prevent attacks even if they are exploiting previously unknown vulnerabilities.

Novell AppArmor consists of:

- A library of AppArmor profiles for common Linux\* applications describing what files the program needs to access.
- A library of AppArmor profile foundation classes (profile building blocks) needed for common application activities, such as DNS lookup and user authentication.
- A tool suite for developing and enhancing AppArmor profiles, so that you can change the existing profiles to suit your needs and create new profiles for your own local and custom applications.
- Several specially modified applications that are AppArmor enabled to provide enhanced security in the form of unique subprocess confinement, including Apache and Tomcat.
- The Novell AppArmor-loadable kernel module and associated control scripts to enforce AppArmor policies on your SUSE Linux Enterprise® system.

This guide covers the following topics:

## *Immunizing Programs*

Describes the operation of Novell AppArmor and describes the types of programs that should have Novell AppArmor profiles created for them.

## *Profile Components and Syntax*

Introduces the profile components and syntax.

## *Building and Managing Profiles with YaST*

Describes how to use the AppArmor YaST modules to build, maintain and update profiles.

### *Building Profiles from the Command Line*

Describes how to use the AppArmor command line tools to build, maintain and update profiles.

### *Profiling Your Web Applications Using ChangeHat*

Enables you to create subprofiles for the Apache Web server that allow you to tightly confine small sections of Web application processing.

### *Managing Profiled Applications*

Describes how to perform Novell AppArmor profile maintenance, which involves tracking common issues and concerns.

### *Support*

Indicates support options for this product.

### Glossary

Provides a list of terms and their definitions.

## 1 Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

## 2 Documentation Conventions

The following typographical conventions are used in this manual:

- `/etc/passwd`: filenames and directory names
- *placeholder*: replace *placeholder* with the actual value
- `PATH`: the environment variable `PATH`
- `ls, --help`: commands, options, and parameters
- `user`: users or groups

- **Alt, Alt + F1**: a key to press or a key combination; keys are shown in uppercase as on a keyboard
- *File, File > Save As*: menu items, buttons
- This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block.

This paragraph is only relevant for the specified architectures. The arrows mark the beginning and the end of the text block.

- *Dancing Penguins* (Chapter *Penguins*, ↑Another Manual): This is a reference to a chapter in another manual.



# Immunizing Programs

Novell® AppArmor provides immunization technologies that protect applications from the inherent vulnerabilities they possess. After installing Novell AppArmor, setting up Novell AppArmor profiles, and rebooting the computer, your system becomes immunized because it begins to enforce the Novell AppArmor security policies. Protecting programs with Novell AppArmor is referred to as *immunizing*.

Novell AppArmor sets up a collection of default application profiles to protect standard Linux services. To protect other applications, use the Novell AppArmor tools to create profiles for the applications that you want protected. This chapter introduces the philosophy of immunizing programs. Proceed to Chapter 2, *Profile Components and Syntax* (page 11), Chapter 3, *Building and Managing Profiles with YaST* (page 17), or Chapter 4, *Building Profiles from the Command Line* (page 37) if you are ready to build and manage Novell AppArmor profiles.

Novell AppArmor provides streamlined access control for network services by specifying which files each program is allowed to read, write, and execute. This ensures that each program does what it is supposed to do and nothing else. Novell AppArmor quarantines programs to protect the rest of the system from being damaged by a compromised process.

Novell AppArmor is a host intrusion prevention or mandatory access control scheme. Previously, access control schemes were centered around users because they were built for large timeshare systems. Alternatively, modern network servers largely do not permit users to log in, but instead provide a variety of network services for users, such as Web, mail, file, and print servers. Novell AppArmor controls the access given to network services and other programs to prevent weaknesses from being exploited.

---

**TIP: Background Information for Novell AppArmor**

---

To get a more in-depth overview of AppArmor and the overall concept behind it, refer to Appendix A, *Background Information on AppArmor Profiling* (page 115).

---

## 1.1 Introducing the AppArmor Framework

This section provides a very basic understanding of what is happening “behind the scenes” (and under the hood of the YaST interface) when you run AppArmor.

An AppArmor profile is a plain text file containing path entries and access permissions. See Section 2.1, “Breaking a Novell AppArmor Profile into Its Parts” (page 12) for a detailed reference profile. The directives contained in this text file are then enforced by the AppArmor routines to quarantine the process or program.

The following tools interact in the building and enforcement of AppArmor profiles and policies:

### aa-unconfined

aa-unconfined detects any application running on your system that listens for network connections and is not protected by an AppArmor profile. Refer to Section “aa-unconfined—Identifying Unprotected Processes” (page 60) for detailed information about this tool.

### aa-autodep

aa-autodep creates a basic skeleton of a profile that needs to be fleshed out before it is put to productive use. The resulting profile is loaded and put into complain mode, reporting any behavior of the application that is not (yet) covered by AppArmor rules. Refer to Section “aa-autodep—Creating Approximate Profiles” (page 44) for detailed information about this tool.

### aa-genprof

aa-genprof generates a basic profile and asks you to refine this profile by executing the application, generating log events that need to be taken care of by AppArmor policies. You are guided through a series of questions to deal with the log events

that have been triggered during the application's execution. After the profile has been generated, it is loaded and put into enforce mode. Refer to Section “aa-genprof—Generating Profiles” (page 47) for detailed information about this tool.

#### aa-logprof

aa-logprof interactively scans and reviews the log entries generated by an application that is confined by an AppArmor profile in complain mode. It assists you in generating new entries in the profile concerned. Refer to Section “aa-logprof—Scanning the System Log” (page 54) for detailed information about this tool.

#### aa-complain

aa-complain toggles the mode of an AppArmor profile from enforce to complain. Exceptions to rules set in a profile are logged, but the profile is not enforced. Refer to Section “aa-complain—Entering Complain or Learning Mode” (page 45) for detailed information about this tool.

#### aa-enforce

aa-enforce toggles the mode of an AppArmor profile from complain to enforce. Exceptions to rules set in a profile are logged, but not permitted—the profile is enforced. Refer to Section “aa-enforce—Entering Enforce Mode” (page 46) for detailed information about this tool.

Once a profile has been built and is loaded, there are two ways in which it can get processed:

#### complain

In complain 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. To improve the profile, turn complain mode on, run the program through a suite of tests to generate log events that characterize the program's access needs, then postprocess the log with the AppArmor tools (YaST or aa-logprof) to transform log events into improved profiles.

#### enforce

In enforce mode, violations of AppArmor profile rules, such as the profiled program accessing files not permitted by the profile, are detected. The violations are logged and not permitted. The default is for enforce mode to be enabled. To log the violations only, but still permit them, use complain mode. Enforce toggles with complain mode.

# 1.2 Determining Programs to Immunize

Now that you have familiarized yourself with AppArmor, start selecting the applications for which to build profiles. Programs that need profiling are those that mediate privilege. The following programs have access to resources that the person using the program does not have, so they grant the privilege to the user when used:

## cron Jobs

Programs that are run periodically by cron. Such programs read input from a variety of sources and can run with special privileges, sometimes with as much as `root` privilege. For example, cron can run `/usr/sbin/logrotate` daily to rotate, compress, or even mail system logs. For instructions for finding these types of programs, refer to Section 1.3, “Immunizing cron Jobs” (page 5).

## Web Applications

Programs that can be invoked through a Web browser, including CGI Perl scripts, PHP pages, and more complex Web applications. For instructions for finding these types of programs, refer to Section 1.4.1, “Immunizing Web Applications” (page 7).

## Network Agents

Programs (servers and clients) that have open network ports. User clients, such as mail clients and Web browsers mediate privilege. These programs run with the privilege to write to the user's home directory and they process input from potentially hostile remote sources, such as hostile Web sites and e-mailed malicious code. For instructions for finding these types of programs, refer to Section 1.4.2, “Immunizing Network Agents” (page 9).

Conversely, unprivileged programs do not need to be profiled. For instance, a shell script might invoke the `cp` program to copy a file. Because `cp` does not have its own profile, it inherits the profile of the parent shell script, so can copy any files that the parent shell script's profile can read and write.

## 1.3 Immunizing cron Jobs

To find programs that are run by cron, inspect your local cron configuration. Unfortunately, cron configuration is rather complex, so there are numerous files to inspect. Periodic cron jobs are run from these files:

```
/etc/crontab
/etc/cron.d/*
/etc/cron.daily/*
/etc/cron.hourly/*
/etc/cron.monthly/*
/etc/cron.weekly/*
```

For `root`'s cron jobs, edit the tasks with `crontab -e` and list `root`'s cron tasks with `crontab -l`. You must be `root` for these to work.

Once you find these programs, you can use the *Add Profile Wizard* to create profiles for them. Refer to Section 3.1, “Adding a Profile Using the Wizard” (page 18).

## 1.4 Immunizing Network Applications

An automated method for finding network server daemons that should be profiled is to use the `aa-unconfined` tool. You can also simply view a report of this information in the YaST module (refer to Section “Application Audit Report” (page 91) for instructions).

The `aa-unconfined` tool uses the command `netstat -nlp` to inspect your open ports from inside your computer, detect the programs associated with those ports, and inspect the set of Novell AppArmor profiles that you have loaded. `aa-unconfined` then reports these programs along with the Novell AppArmor profile associated with each program or reports “none” if the program is not confined.

---

### NOTE

If you create a new profile, you must restart the program that has been profiled to have it be effectively confined by AppArmor.

---

Below is a sample aa-unconfined output:

```
2325 /sbin/portmap not confined
3702❶ /usr/sbin/sshd❷ confined
    by '/usr/sbin/sshd❸ (enforce) '
4040 /usr/sbin/ntpd confined by '/usr/sbin/ntpd (enforce) '
4373 /usr/lib/postfix/master confined by '/usr/lib/postfix/master (enforce) '

4505 /usr/sbin/httpd2-prefork confined by '/usr/sbin/httpd2-prefork (enforce) '
5274 /sbin/dhcpd not confined
5592 /usr/bin/ssh not confined
7146 /usr/sbin/cupsd confined by '/usr/sbin/cupsd (complain) '
```

- ❶ The first portion is a number. This number is the process ID number (PID) of the listening program.
- ❷ The second portion is a string that represents the absolute path of the listening program
- ❸ The final portion indicates the profile confining the program, if any.

---

#### NOTE

aa-unconfined requires `root` privileges and should not be run from a shell that is confined by an AppArmor profile.

---

aa-unconfined does not distinguish between one network interface and another, so it reports all unconfined processes, even those that might be listening to an internal LAN interface.

Finding user network client applications is dependent on your user preferences. The aa-unconfined tool detects and reports network ports opened by client applications, but only those client applications that are running at the time the aa-unconfined analysis is performed. This is a problem because network services tend to be running all the time, while network client applications tend only to be running when the user is interested in them.

Applying Novell AppArmor profiles to user network client applications is also dependent on user preferences. Therefore, we leave profiling of user network client applications as an exercise for the user.

To aggressively confine desktop applications, the `aa-unconfined` command supports a `paranoid` option, which reports all processes running and the corresponding AppArmor profiles that might or might not be associated with each process. The user can then decide whether each of these programs needs an AppArmor profile.

If you have new or modified profiles, you can submit them to the `apparmor-general@forge.novell.com` [<mailto:apparmor-general@forge.novell.com>] mailing list along with a use case for the application behavior that you exercised. The AppArmor team reviews and may submit the work into SUSE Linux Enterprise. We cannot guarantee that every profile will be included, but we make a sincere effort to include as much as possible so that end users can contribute to the security profiles that ship in SUSE Linux Enterprise.

## 1.4.1 Immunizing Web Applications

To find Web applications, investigate your Web server configuration. The Apache Web server is highly configurable and Web applications can be stored in many directories, depending on your local configuration. SUSE Linux Enterprise, by default, stores Web applications in `/srv/www/cgi-bin/`. To the maximum extent possible, each Web application should have an Novell AppArmor profile.

Once you find these programs, you can use the AppArmor *Add Profile Wizard* to create profiles for them. Refer to Section 3.1, “Adding a Profile Using the Wizard” (page 18).

Because CGI programs are executed by the Apache Web server, the profile for Apache itself, `usr/sbin/httpd2-prefork` for Apache2 on SUSE Linux Enterprise, must be modified to add execute permissions to each of these programs. For instance, adding the line `/srv/www/cgi-bin/my_hit_counter.pl rpx` grants Apache permission to execute the Perl script `my_hit_counter.pl` and requires that there be a dedicated profile for `my_hit_counter.pl`. If `my_hit_counter.pl` does not have a dedicated profile associated with it, the rule should say `/srv/www/cgi-bin/my_hit_counter.pl rix` to cause `my_hit_counter.pl` to inherit the `usr/sbin/httpd2-prefork` profile.

Some users might find it inconvenient to specify execute permission for every CGI script that Apache might invoke. Instead, the administrator can grant controlled access to collections of CGI scripts. For instance, adding the line

```
/srv/www/cgi-bin/*.{pl,py,pyc} rix
```

allows Apache to execute all files in `/srv/www/cgi-bin/` ending in `.pl` (Perl scripts) and `.py` or `.pyc` (Python scripts). As above, the `ix` part of the rule causes Python scripts to inherit the Apache profile, which is appropriate if you do not want to write individual profiles for each Python script.

---

## NOTE

If you want the subprocess confinement module (`apache2-mod-apparmor`) functionality when Web applications handle Apache modules (`mod_perl` and `mod_php`), use the ChangeHat features when you add a profile in YaST or at the command line. To take advantage of the subprocess confinement, refer to Section 5.1, “Apache ChangeHat” (page 70).

---

Profiling Web applications that use `mod_perl` and `mod_php` requires slightly different handling. In this case, the “program” is a script interpreted directly by the module within the Apache process, so no `exec` happens. Instead, the Novell AppArmor version of Apache calls `change_hat()` using a subprofile (a “hat”) corresponding to the name of the URI requested.

---

## NOTE

The name presented for the script to execute might not be the URI, depending on how Apache has been configured for where to look for module scripts. If you have configured your Apache to place scripts in a different place, the different names appear in log file when Novell AppArmor complains about access violations. See Chapter 6, *Managing Profiled Applications* (page 81).

---

For `mod_perl` and `mod_php` scripts, this is the name of the Perl script or the PHP page requested. For example, adding this subprofile allows the `localtime.php` page to execute and access the local system time:

```
/usr/bin/httpd2-prefork {
# ...
^/cgi-bin/localtime.php {
    /etc/localtime           r,
    /srv/www/cgi-bin/localtime.php r,
    /usr/lib/locale/**      r,
}
}
```

If no subprofile has been defined, the Novell AppArmor version of Apache applies the `DEFAULT_URI` hat. This subprofile is basically sufficient to display an HTML Web page. The `DEFAULT_URI` hat that Novell AppArmor provides by default is the following:

```
^DEFAULT_URI {
    /usr/sbin/suexec2 ixr,
    /var/log/apache2/** rwl,
    /home/*/public_html/** r,
    /srv/www/htdocs/** r,
    /srv/www/icons/*.{gif,jpg,png} r,
    /usr/share/apache2/** r,
}
```

To use a single Novell AppArmor profile for all Web pages and CGI scripts served by Apache, a good approach is to edit the `DEFAULT_URI` subprofile.

## 1.4.2 Immunizing Network Agents

To find network server daemons and network clients (such as `fetchmail`, `Firefox`, `amaroK` or `Banshee`) that should be profiled, you should inspect the open ports on your machine, consider the programs that are answering on those ports, and provide profiles for as many of those programs as possible. If you provide profiles for all programs with open network ports, an attacker cannot get to the file system on your machine without passing through a Novell AppArmor profile policy.

Scan your server for open network ports manually from outside the machine using a scanner, such as `nmap`, or from inside the machine using the `netstat --inet -n`

-p command. Then inspect the machine to determine which programs are answering on the discovered open ports.

---

**TIP**

Refer to the man page of the `netstat` command for a detailed reference of all possible options.

---

# Profile Components and Syntax

# 2

You are ready to build Novell AppArmor profiles after you select the programs to profile. To do so, it is important to understand the components and syntax of profiles. AppArmor profiles contain several building blocks that help build simple and reusable profile code: `#include` files, abstractions, program chunks, and capability entries. `#include` statements are used to pull in parts of other AppArmor profiles to simplify the structure of new profiles. Abstractions are `#include` statements grouped by common application tasks. Program chunks are chunks of profiles that are specific to program suites. Capability entries are profile entries for any of the POSIX.1e Linux capabilities.

For help determining the programs to profile, refer to Section 1.2, “Determining Programs to Immunize” (page 4). To start building AppArmor profiles with YaST, proceed to Chapter 3, *Building and Managing Profiles with YaST* (page 17). To build profiles using the AppArmor command line interface, proceed to Chapter 4, *Building Profiles from the Command Line* (page 37).

## 2.1 Breaking a Novell AppArmor Profile into Its Parts

Novell AppArmor profile components are called Novell AppArmor rules. Currently there are two main types of Novell AppArmor rules, path entries and capability entries. Path entries specify what the process can access in the file system and capability entries provide a more fine-grained control over what a confined process is allowed to do through other system calls that require privileges. Includes are a type of meta rule or directives that pull in path and capability entries from other files.

The easiest way of explaining what a profile consists of and how to create one is to show the details of a sample profile, in this case for a hypothetical application called `/usr/bin/foo`:

```
#include <tunables/global>❶

# a comment naming the application to confine
/usr/bin/foo❷
{❸
    #include <abstractions/base>❹

    capability setgid❺,

    /bin/mount          ux,
    /dev/{,u}❻random    r,
    /etc/ld.so.cache    r,
    /etc/foo.conf       r,
    /etc/foo/*          r,
    /lib/ld-*.so*       mr,
    /lib/lib*.so*       mr,
    /proc/[0-9]**       r,
    /usr/lib/**         mr,
    /tmp/❼              r,
    /tmp/foo.pid        wr,
    /tmp/foo.*          lrw,
    /@{HOME}❽/.foo_file rw,
    /@{HOME}/.foo_lock w,

    # a comment about foo's subprofile, bar.
    ^bar❾ {
        /lib/ld-*.so*    mr,
        /usr/bin/bar     px,
        /var/spool/*     rwl,
    }
}
```

- ❶ This loads a file containing variable definitions.
- ❷ The normalized path to the program that is confined.
- ❸ The curly braces ( { } ) serve as a container for include statements, subprofiles, path entries, and capability entries.
- ❹ This directive pulls in components of AppArmor profiles to simplify profiles.
- ❺ Capability entry statements enable each of the 29 POSIX.1e draft capabilities.
- ❻ The curly braces ( { } ) make this rule apply to the path both with and without the content enclosed by the braces.
- ❼ A path entry specifying what areas of the file system the program can access. The first part of a path entry specifies the absolute path of a file (including regular expression globbing) and the second part indicates permissible access modes (r for read, w for write, and x for execute). A whitespace of any kind (spaces or tabs) can precede pathnames or separate the pathname from the access modes. Spaces between the access mode and the trailing comma is optional. Find a comprehensive overview of the available access modes in Section 4.8, “File Permission Access Modes” (page 62).
- ❽ This variable expands to a value that can be changed without changing the entire profile.
- ❾ This section references a subprofile of the application, also known as a “hat”. For more details on AppArmor's ChangeHat feature, refer to Chapter 5, *Profiling Your Web Applications Using ChangeHat* (page 69).

---

### TIP: Using Variables in Profiles

With the current AppArmor tools, variables as presented in the above example can only be used when manually editing and maintaining a profile.

A typical example when variables come in handy are network scenarios in which user home directories are not mounted in the standard location

`/home/username`, but under a custom location. Find the variable definitions for this use case (`@{HOME}` and `@{HOMEDIRS}`) in the `/etc/apparmor.d/tunables/home` file.

---

When a profile is created for a program, the program can access only the files, modes, and POSIX capabilities specified in the profile. These restrictions are in addition to the native Linux access controls.

**Example:** To gain the capability `CAP_CHOWN`, the program must have both access to `CAP_CHOWN` under conventional Linux access controls (typically, be a `root`-owned process) and have the capability `chown` in its profile. Similarly, to be able to write to the file `/foo/bar` the program must have both the correct user ID and mode bits set in the files attributes (see the `chmod` and `chown` man pages) and have `/foo/bar` `w` in its profile.

Attempts to violate Novell AppArmor rules are recorded in `/var/log/audit/audit.log` if the `audit` package is installed or otherwise in `/var/log/messages`. In many cases, Novell AppArmor rules prevent an attack from working because necessary files are not accessible and, in all cases, Novell AppArmor confinement restricts the damage that the attacker can do to the set of files permitted by Novell AppArmor.

## 2.2 #include Statements

`#include` statements are directives that pull in components of other Novell AppArmor profiles to simplify profiles. Include files fetch 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.

By default, AppArmor adds `/etc/apparmor.d` to the path in the `#include` statement. AppArmor expects the include files to be located in `/etc/apparmor.d`. Unlike other profile statements (but similar to C programs), `#include` lines do not end with a comma.

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

### 2.2.1 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. Files listed in these abstractions are specific to the named task. Programs that require one of these files usually require some of the other files listed in the abstraction file (depending on the local configuration

as well as the specific requirements of the program). Find abstractions in `/etc/apparmor.d/abstractions`.

## 2.2.2 Program Chunks

The `program-chunks` directory (`/etc/apparmor.d/program-chunks`) contains some chunks of profiles that are specific to program suites and not generally useful outside of the suite, thus are never suggested for use in profiles by the profile wizards (`aa-logprof` and `aa-genprof`). Currently program chunks are only available for the postfix program suite.

## 2.3 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.

## 2.4 Using the Local AppArmor Profile Repository

AppArmor ships a set of profiles enabled by default and created by the AppArmor developers and kept under the `/etc/apparmor.d`. In addition to these profiles, SUSE Linux Enterprise ships profiles for individual applications together with the respective application. These profiles are not enabled by default and reside under another directory than the standard AppArmor profiles, `/etc/apparmor/profiles/extras`.

The AppArmor tools, both YaST and `aa-genprof` and `aa-logprof`, support the use of a local repository. Whenever you start to create a new profile from scratch and there already is one inactive profile in your local repository, you are asked whether you would like to use the existing inactive one from `/etc/apparmor/profiles/extras` and whether you want to base your efforts on it. If you decide to use this profile, it gets copied over to the directory of profiles enabled by default (`/etc/apparmor.d`) and loaded whenever AppArmor is started. Any further further adjustments will be done to the active profile under `/etc/apparmor.d`.



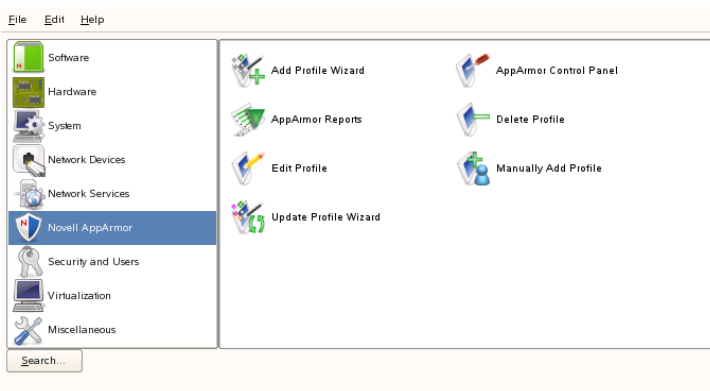
# Building and Managing Profiles with YaST

# 3

YaST provides an easy way to build profiles and manage Novell® AppArmor. It provides two interfaces: a fully graphical one and a text-based one. The text-based interface consumes less resources and bandwidth, making it a better choice for remote administration or for times when a local graphical environment is inconvenient. Although the interfaces have differing appearances, they offer the same functionality in similar ways. Another alternative is to use AppArmor commands, which can control AppArmor from a terminal window or through remote connections. The command line tools are described in Chapter 4, *Building Profiles from the Command Line* (page 37).

Start YaST from the main menu and enter your `root` password when prompted for it. Alternatively, start YaST by opening a terminal window, logging in as `root`, and entering `yast2` for the graphical mode or `yast` for the text-based mode.

**Figure 3.1** *YaST Controls for AppArmor*



The right frame shows the AppArmor options:

#### Add Profile Wizard

For detailed steps, refer to Section 3.1, “Adding a Profile Using the Wizard” (page 18).

#### Manually Add Profile

Add a Novell AppArmor profile for an application on your system without the help of the wizard. For detailed steps, refer to Section 3.2, “Manually Adding a Profile” (page 25).

#### Edit Profile

Edits an existing Novell AppArmor profile on your system. For detailed steps, refer to Section 3.3, “Editing Profiles” (page 26).

#### Delete Profile

Deletes an existing Novell AppArmor profile from your system. For detailed steps, refer to Section 3.4, “Deleting a Profile” (page 31).

#### Update Profile Wizard

For detailed steps, refer to Section 3.5, “Updating Profiles from Log Entries” (page 32).

#### AppArmor Reports

For detailed steps, refer to Section 6.3, “Configuring Reports” (page 85).

#### AppArmor Control Panel

For detailed steps, refer to Section 3.6, “Managing Novell AppArmor and Security Event Status” (page 33).

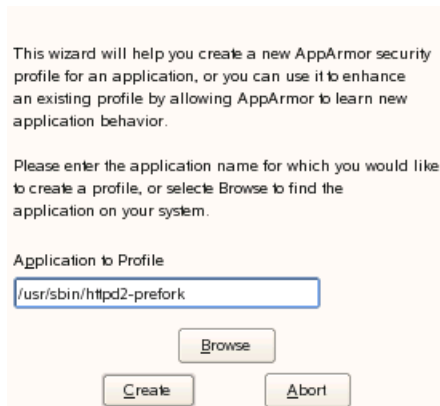
## 3.1 Adding a Profile Using the Wizard

*Add Profile Wizard* is designed to set up Novell AppArmor profiles using the AppArmor profiling tools, *aa-genprof* (generate profile) and *aa-logprof* (update profiles from learning mode log file). For more information about these tools, refer to Section 4.6.3, “Summary of Profiling Tools” (page 44).

- 1 Stop the application before profiling it to ensure that application start-up is included in the profile. To do this, make sure that the application or daemon is not running.

For example, enter `rcPROGRAM stop` (or `/etc/init.d/PROGRAM stop`) in a terminal window while logged in as `root`, replacing `PROGRAM` with the name of the program to profile.

- 2 Start YaST and select *Novell AppArmor > Add Profile Wizard*.



- 3 Enter the name of the application or browse to the location of the program.
- 4 Click *Create*. This runs an AppArmor tool named `aa-autodep`, which performs a static analysis of the program to profile and loads an approximate profile into the AppArmor module. For more information about `aa-autodep`, refer to Section “`aa-autodep`—Creating Approximate Profiles” (page 44).

Depending on whether the profile you are about to create already exists in the local profile repository (see Section 2.4, “Using the Local AppArmor Profile Repository” (page 15)) or whether it does not exist yet, proceed with one of the following options:

- Determine whether you want to use or fine-tune an already existing profile from your local profile repository, as outlined in Step 5 (page 19).
  - Create the profile from scratch and proceed with Step 6 (page 20) and beyond.
- 5 If the profile already exists in the local profile repository under `/etc/apparmor/profiles/extra`, YaST informs you that there is an inactive

profile which you can either use as a base for your own efforts or which you can just accept as is.

Alternatively, you can choose not to use the local version at all and start creating the profile from scratch. In any case, proceed with Step 6 (page 20).

- 6** Run the application to profile.
- 7** Perform as many of the application functions as possible so learning mode can log the files and directories to which the program requires access to function properly. Be sure to include restarting and stopping the program in the exercised functions. AppArmor needs to handle these events as well as any other program function.
- 8** Click *Scan system log for AppArmor events* to parse the learning mode log files. This generates a series of questions that you must answer to guide the wizard in generating the security profile.

If requests to add hats appear, proceed to Chapter 5, *Profiling Your Web Applications Using ChangeHat* (page 69).

The questions fall into two categories:

- A resource is requested by a profiled program that is not in the profile (see Figure 3.2, “Learning Mode Exception: Controlling Access to Specific Resources” (page 21)). Allow or deny access to a specific resource.
- A program is executed by the profiled program and the security domain transition has not been defined (see Figure 3.3, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 21)). Define execute permissions for an entry.

Each of these cases results in a series of questions that you must answer to add the resource to the profile or to add the program to the profile. For an example of each case, see Figure 3.2, “Learning Mode Exception: Controlling Access to Specific Resources” (page 21) and Figure 3.3, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 21). Subsequent steps describe your options in answering these questions.

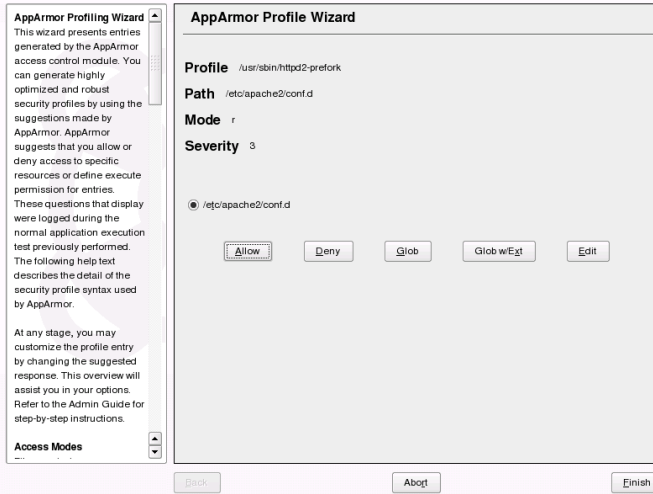
---

## NOTE: Varying Processing Options

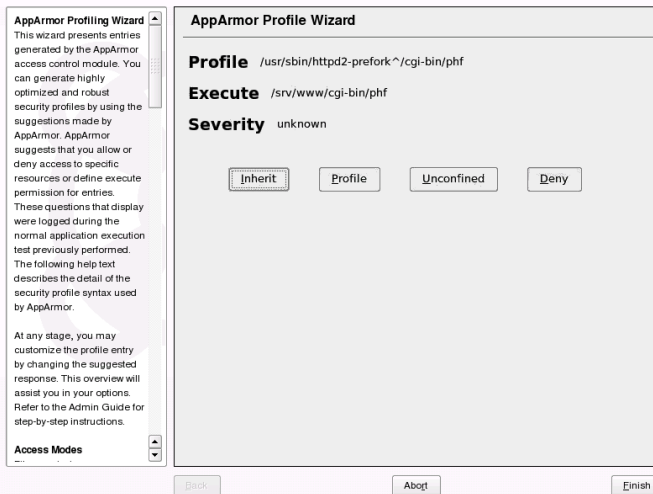
Depending on the type of entry processed, the available options vary.

---

**Figure 3.2** *Learning Mode Exception: Controlling Access to Specific Resources*



**Figure 3.3** *Learning Mode Exception: Defining Execute Permissions for an Entry*



9 The *Add Profile Wizard* begins suggesting directory path entries that have been accessed by the application profiled (as seen in Figure 3.2, “Learning Mode Exception: Controlling Access to Specific Resources” (page 21)) or requires you to define execute permissions for entries (as seen in Figure 3.3, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 21)).

- For Figure 3.2: Learning Mode Exception: Controlling Access to Specific Resources: Select the option that satisfies the request for access, which could be a suggested include, a particular globbed version of the path, or the actual pathname. Depending on the situation, these options are available:

`#include`

The section of a Novell AppArmor profile that refers to an include file. Include files give 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.

Globbed Version

Accessed by clicking *Glob*. For information about globbing syntax, refer to Section 4.7, “Paths and Globbing” (page 60).

Actual Pathname

Literal path that the program needs to access to run properly.

After selecting a directory path, process it as an entry to the Novell AppArmor profile by clicking *Allow* or *Deny*. If you are not satisfied with the directory path entry as it is displayed, you can also *Glob* or *Edit* it.

The following options are available to process the learning mode entries and build the profile:

Allow

Grant the program access to the specified directory path entries. The *Add Profile Wizard* suggests file permission access. For more information about this, refer to Section 4.8, “File Permission Access Modes” (page 62).

Deny

Click *Deny* to prevent the program from accessing the specified paths.

### Glob

Clicking this modifies the directory path (using wild cards) to include all files in the suggested directory. Double-clicking it grants access to all files and subdirectories beneath the one shown. For more information about globbing syntax, refer to Section 4.7, “Paths and Globbing” (page 60).

### Glob w/Ext

Modify the original directory path while retaining the filename extension. A single click causes `/etc/apache2/file.ext` to become `/etc/apache2/*.ext`, adding the wild card (asterisk) in place of the filename. This allows the program to access all files in the suggested directories that end with the `.ext` extension. When you double-click it, access is granted to all files with the particular extension and subdirectories beneath the one shown.

### Edit

Edit the highlighted line. The new edited line appears at the bottom of the list.

### Abort

Abort `aa-logprof`, losing all rule changes entered so far and leaving all profiles unmodified.

### Finish

Close `aa-logprof`, saving all rule changes entered so far and modifying all profiles.

Click *Allow* or *Deny* for each learning mode entry. These help build the Novell AppArmor profile.

---

### NOTE

The number of learning mode entries corresponds to the complexity of the application.

---

- For Figure 3.3: Learning Mode Exception: Defining Execute Permissions for an Entry: From the following options, select the one that satisfies the request for access. For detailed information about the options available, refer to Section 4.8, “File Permission Access Modes” (page 62).

### Inherit

Stay in the same security profile (parent's profile).

### Profile

Require a separate profile to exist for the executed program. When selecting this option, also select whether AppArmor should sanitize the environment when switching profiles by removing certain environment variables that can modify the execution behavior of the child process. Unless these variables are absolutely required to properly execute the child process, always choose the more secure, sanitized option.

### Unconfined

Execute the program without a security profile. When prompted, have AppArmor sanitize the environment to avoid adding security risks by inheriting certain environment variables from the parent process.

---

#### **WARNING: Risks of Running Unconfined**

Unless absolutely necessary, do not run unconfined. Choosing the *Unconfined* option executes the new program without any protection from AppArmor.

---

### Deny

Click *Deny* to prevent the program from accessing the specified paths.

### Abort

Abort aa-logprof, losing all rule changes entered so far and leaving all profiles unmodified.

### Finish

Close aa-logprof, saving all rule changes entered so far and modifying all profiles.

- 10 Repeat the previous steps if you need to execute more functionality of the application.

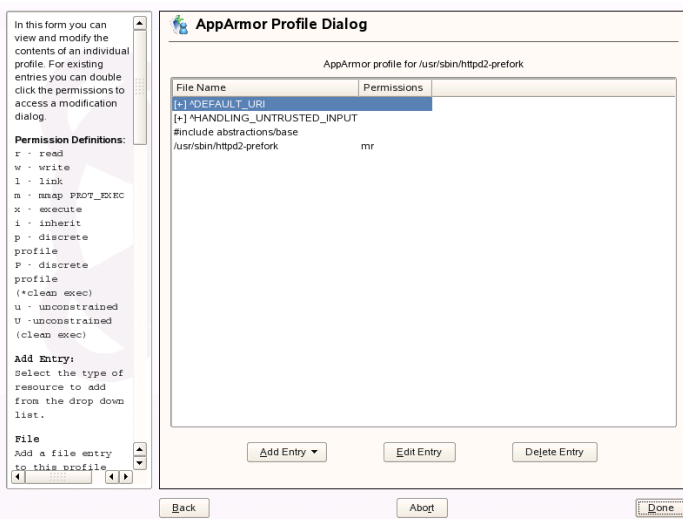
When you are done, click *Finish*. Choose to apply your changes to the local profile set.

As soon as you exit the *Profile Creation Wizard*, the profile is saved locally. The profile is then loaded into the AppArmor module.

## 3.2 Manually Adding a Profile

Novell AppArmor enables you to create a Novell AppArmor profile by manually adding entries into the profile. Select the application for which to create a profile then add entries.

- 1 Start YaST and select *Novell AppArmor > Manually Add Profile*.
- 2 Browse your system to find the application for which to create a profile.
- 3 When you find the application, select it and click *Open*. A basic, empty profile appears in the *AppArmor Profile Dialog* window.



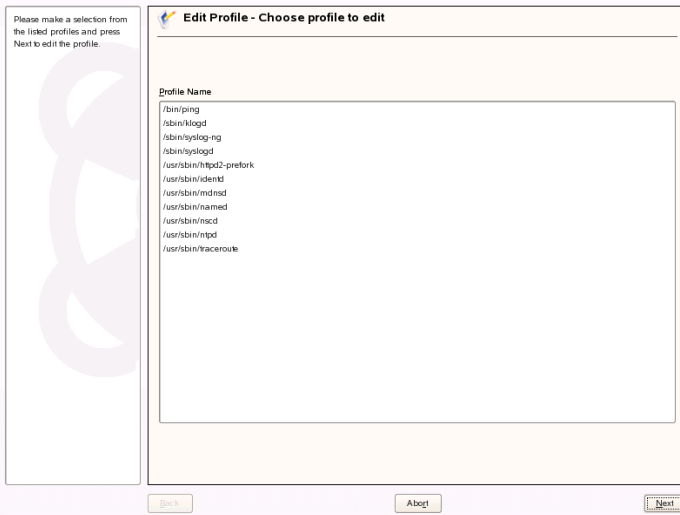
- 4 In *AppArmor Profile Dialog*, add, edit, or delete AppArmor profile entries by clicking the corresponding buttons and referring to Section 3.3.1, “Adding an Entry” (page 28), Section 3.3.2, “Editing an Entry” (page 30), or Section 3.3.3, “Deleting an Entry” (page 31).

5 When finished, click *Done*.

## 3.3 Editing Profiles

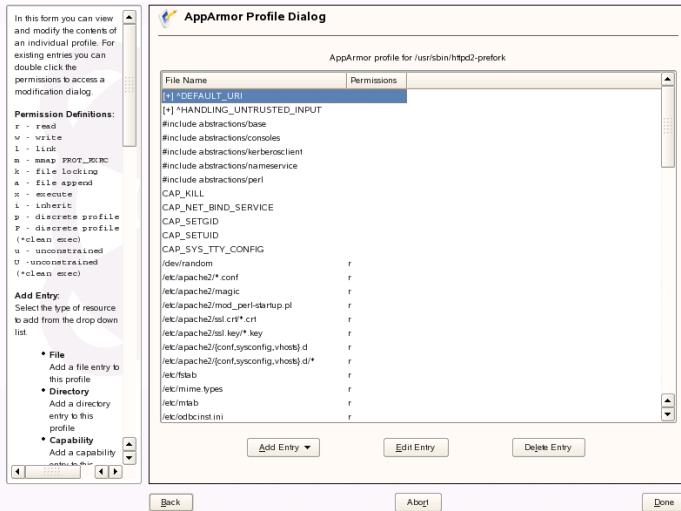
AppArmor enables you to edit Novell AppArmor profiles manually by adding, editing, or deleting entries. To edit a profile, proceed as follows:

1 Start YaST and select *Novell AppArmor > Edit Profile*.



2 From the list of profiled applications, select the profile to edit.

3 Click *Next*. The *AppArmor Profile Dialog* window displays the profile.



- 4 In the *AppArmor Profile Dialog* window, add, edit, or delete Novell AppArmor profile entries by clicking the corresponding buttons and referring to Section 3.3.1, “Adding an Entry” (page 28), Section 3.3.2, “Editing an Entry” (page 30), or Section 3.3.3, “Deleting an Entry” (page 31).
- 5 When you are finished, click *Done*.
- 6 In the pop-up that appears, click *Yes* to confirm your changes to the profile and reload the AppArmor profile set.

---

### TIP: Syntax Checking in AppArmor

AppArmor contains a syntax check that notifies you of any syntax errors in profiles you are trying to process with the YaST AppArmor tools. If an error occurs, edit the profile manually as `root` and reload the profile set with `rcapparmor reload`.

---

## 3.3.1 Adding an Entry

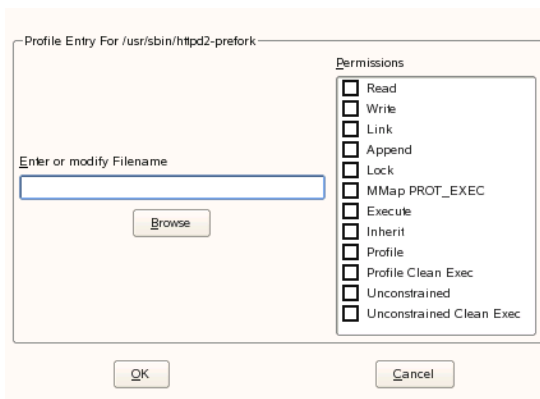
The *Add Entry* option can be found in Section 3.2, “Manually Adding a Profile” (page 25) or Section 3.3, “Editing Profiles” (page 26). When you select *Add Entry*, a list shows the types of entries you can add to the Novell AppArmor profile.

From the list, select one of the following:

### File

In the pop-up window, specify the absolute path of a file, including the type of access permitted. When finished, click *OK*.

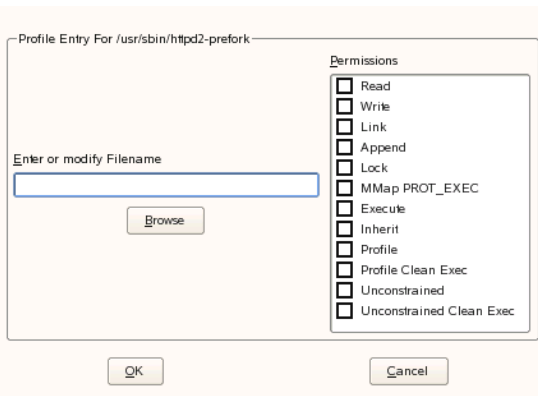
You can use globbing if necessary. For globbing information, refer to Section 4.7, “Paths and Globbing” (page 60). For file access permission information, refer to Section 4.8, “File Permission Access Modes” (page 62).



### Directory

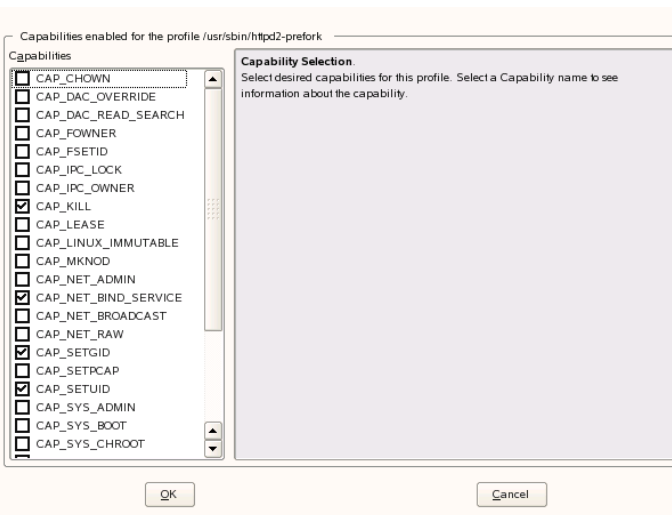
In the pop-up window, specify the absolute path of a directory, including the type of access permitted. You can use globbing if necessary. When finished, click *OK*.

For globbing information, refer to Section 4.7, “Paths and Globbing” (page 60). For file access permission information, refer to Section 4.8, “File Permission Access Modes” (page 62).



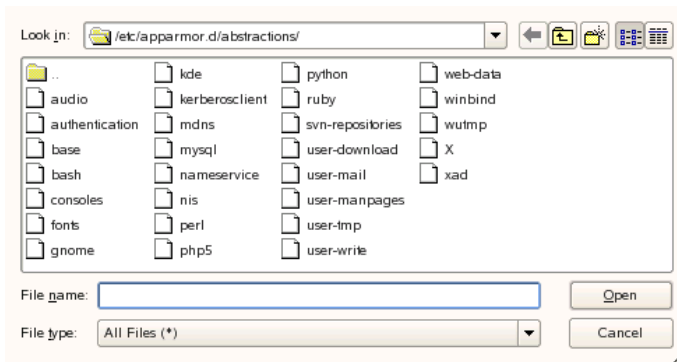
## Capability

In the pop-up window, select the appropriate capabilities. These are statements that enable each of the 32 POSIX.1e capabilities. Refer to Section 2.1, “Breaking a Novell AppArmor Profile into Its Parts” (page 12) for more information about capabilities. When finished making your selections, click *OK*.



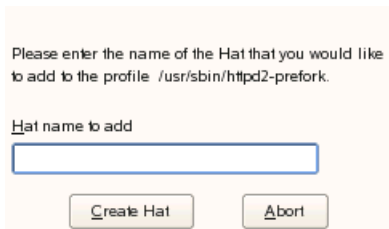
## Include

In the pop-up window, browse to the files to use as includes. Includes are directives that pull in components of other Novell AppArmor profiles to simplify profiles. For more information, refer to Section 2.2, “#include Statements” (page 14).



## Hat

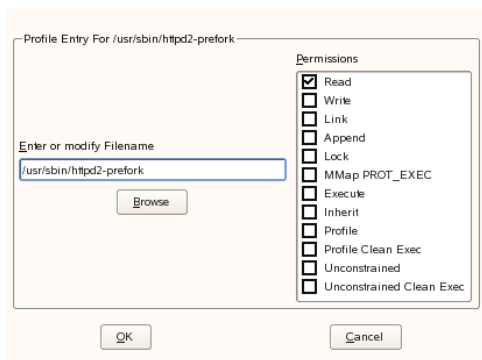
In the pop-up window, specify the name of the subprofile (*hat*) to add to your current profile and click *Create Hat*. For more information, refer to Chapter 5, *Profiling Your Web Applications Using ChangeHat* (page 69).



## 3.3.2 Editing an Entry

When you select *Edit Entry*, the file browser pop-up window opens. From here, edit the selected entry.

In the pop-up window, specify the absolute path of a file, including the type of access permitted. You can use globbing if necessary. When finished, click *OK*.



For globbing information, refer to Section 4.7, “Paths and Globbing” (page 60). For file access permission information, refer to Section 4.8, “File Permission Access Modes” (page 62).

### 3.3.3 Deleting an Entry

To delete an entry in a given profile, select *Delete Entry*. AppArmor removes the selected profile entry.

## 3.4 Deleting a Profile

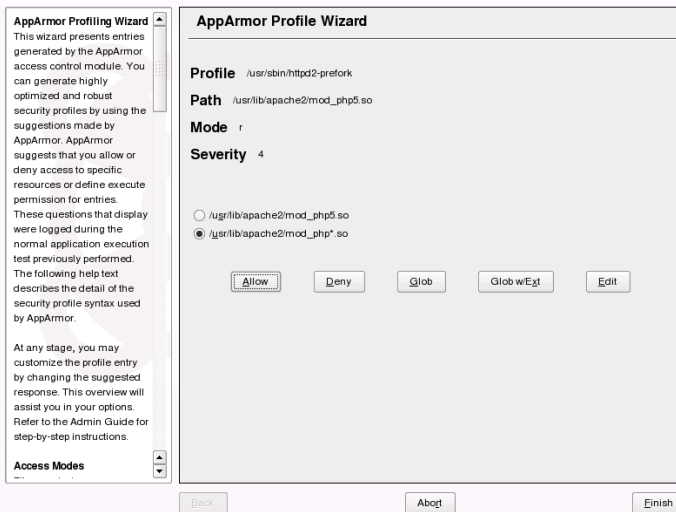
AppArmor enables you to delete an AppArmor profile manually. Simply select the application for which to delete a profile then delete it as follows:

- 1 Start YaST and select *Novell AppArmor > Delete Profile*.
- 2 Select the profile to delete.
- 3 Click *Next*.
- 4 In the pop-up that opens, click *Yes* to delete the profile and reload the AppArmor profile set.

## 3.5 Updating Profiles from Log Entries

The Novell AppArmor profile wizard uses `aa-logprof`, the tool that scans log files and enables you to update profiles. `aa-logprof` tracks messages from the Novell AppArmor module that represent exceptions for all profiles running on your system. These exceptions represent the behavior of the profiled application that is outside of the profile definition for the program. You can add the new behavior to the relevant profile by selecting the suggested profile entry.

- 1 Start YaST and select *Novell AppArmor > Update Profile Wizard*.



Running *Update Profile Wizard* (`aa-logprof`) parses the learning mode log files. This generates a series of questions that you must answer to guide `aa-logprof` to generate the security profile. The exact procedure is the same as with creating a new profile. Refer to Step 8 (page 20) in Section 3.1, “Adding a Profile Using the Wizard” (page 18) for details.

- 2 When you are done, click *Finish*. In the following pop-up, click *Yes* to exit the *Add Profile Wizard*. The profile is saved and loaded into the Novell AppArmor module.

## 3.6 Managing Novell AppArmor and Security Event Status

You can change the status of AppArmor by enabling or disabling it. Enabling AppArmor protects your system from potential program exploitation. Disabling AppArmor, even if your profiles have been set up, removes protection from your system. You can determine how and when you are notified when system security events occur.

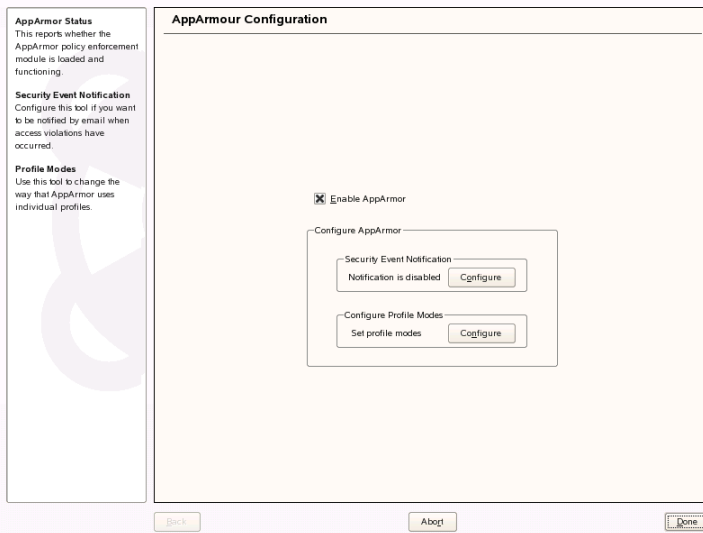
---

### NOTE

For event notification to work, you must set up a mail server on your system that can send outgoing mail using the single mail transfer protocol (SMTP), such as postfix or exim.

---

To configure event notification or change the status of AppArmor, start YaST and select *Novell AppArmor > Novell AppArmor Control Panel*.



From the *AppArmor Configuration* screen, determine whether Novell AppArmor and security event notification are running by looking for a status message that reads *enabled* or configure the mode of individual profiles.

To change the status of Novell AppArmor, continue as described in Section 3.6.1, “Changing Novell AppArmor Status” (page 34). To change the mode of individual profiles, continue as described in Section 3.6.2, “Changing the Mode of Individual Profiles” (page 34). To configure security event notification, continue as described in Section 6.2, “Configuring Security Event Notification” (page 82).

## 3.6.1 Changing Novell AppArmor Status

When you change the status of AppArmor, set it to enabled or disabled. When AppArmor is enabled, it is installed, running, and enforcing the AppArmor security policies.

- 1 Start YaST and select *Novell AppArmor > AppArmor Control Panel*.
- 2 Enable AppArmor by checking *Enable AppArmor* or disable AppArmor by deselecting it.
- 3 Click *Done* in the *AppArmor Configuration* window.
- 4 Click *File > Quit* in the YaST Control Center.

## 3.6.2 Changing the Mode of Individual Profiles

AppArmor can apply profiles in two different modes. 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 and is used by the AppArmor tools for generating profiles. Loading a profile in *enforce* mode enforces the policy defined in the profile and reports policy violation attempts to syslogd.

The *Profile Modes* dialog allows you to view and edit the mode of currently loaded AppArmor profiles. This feature is useful for determining the status of your system during profile development. During the course of systemic profiling (see Section 4.6.2, “Systemic Profiling” (page 42)), you can use this tool to adjust and monitor the scope of the profiles for which you are learning behavior.

To edit an application's profile mode, proceed as follows:

- 1 Start YaST and select *Novell AppArmor > AppArmor Control Panel*.
- 2 In the *Configure Profile Modes* section, select *Configure*.
- 3 Select the profile for which to change the mode.
- 4 Select *Toggle Mode* to set this profile to *complain* mode or to *enforce* mode.
- 5 Apply your settings and leave YaST with *Done*.

To change the mode of all profiles, use *Set All to Enforce* or *Set All to Complain*.

---

**TIP: Listing the Profiles Available**

By default, only active profiles are listed—any profile that has a matching application installed on your system. To set up a profile before installing the respective application, click *Show All Profiles* and select the profile to configure from the list that appears.

---



# Building Profiles from the Command Line

Novell® AppArmor provides the ability to use a command line interface rather than a graphical interface to manage and configure your system security. Track the status of Novell AppArmor and create, delete, or modify AppArmor profiles using the AppArmor command line tools.

---

## TIP: Background Information

Before starting to manage your profiles using the AppArmor command line tools, check out the general introduction to AppArmor given in Chapter 1, *Immunizing Programs* (page 1) and Chapter 2, *Profile Components and Syntax* (page 11).

---

## 4.1 Checking the AppArmor Module Status

An AppArmor module can be in any one of three states:

### Unloaded

The AppArmor module is not loaded into the kernel.

### Running

The AppArmor module is loaded into the kernel and is enforcing AppArmor program policies.

## Stopped

The AppArmor module is loaded into the kernel, but no policies are enforced.

Detect the state of the AppArmor module by inspecting `/sys/kernel/security/apparmor/profiles`. If `cat /sys/kernel/security/apparmor/profiles` reports a list of profiles, AppArmor is running. If it is empty and returns nothing, AppArmor is stopped. If the file does not exist, AppArmor is unloaded.

Manage AppArmor through the script `rcapparmor`, which can perform the following operations:

`rcapparmor start`

Behavior depends on the AppArmor module state. If it is unloaded, `start` loads the module and starts it, putting it in the running state. If it is stopped, `start` causes the module to rescan the AppArmor profiles usually found in `/etc/apparmor.d` and puts the module in the running state. If the module is already running, `start` reports a warning and takes no action.

`rcapparmor stop`

Stops the AppArmor module if it is running by removing all profiles from kernel memory, effectively disabling all access controls, and putting the module into the stopped state. If the AppArmor module is unloaded or already stopped, `stop` tries to unload the profiles again, but nothing happens.

`rcapparmor restart`

Causes the AppArmor module to rescan the profiles in `/etc/apparmor.d` without unconfining running processes. Freshly created profiles are enforced and recently deleted ones are removed from the `/etc/apparmor.d` directory.

`rcapparmor kill`

Unconditionally removes the AppArmor module from the kernel. This is unsafe, because unloading modules from the Linux kernel is unsafe. This command is provided only for debugging and emergencies when the module might need to be removed.

---

## WARNING

AppArmor is a powerful access control system and it is possible to lock yourself out of your own machine to the point where you must boot the machine from a rescue medium (such as the first medium of SUSE Linux Enterprise) to regain control.

To prevent such a problem, always ensure that you have a running, unconfined, `root` login on the machine being configured when you restart the AppArmor module. If you damage your system to the point where logins are no longer possible (for example, by breaking the profile associated with the SSH daemon), you can repair the damage using your running `root` prompt then restart the AppArmor module.

---

## 4.2 Building AppArmor Profiles

The AppArmor module profile definitions are stored in the `/etc/apparmor.d` directory as plain text files. For a detailed description of the syntax of these files, refer to Chapter 2, *Profile Components and Syntax* (page 11).

All files in the `/etc/apparmor.d` directory are interpreted as profiles and are loaded as such. Renaming files in that directory is not an effective way of preventing profiles from being loaded. You must remove profiles from this directory to prevent them from being read and evaluated effectively.

You can use a text editor, such as `vim`, to access and make changes to these profiles. The following options contain detailed steps for building profiles:

### Adding or Creating AppArmor Profiles

Refer to Section 4.3, “Adding or Creating an AppArmor Profile” (page 40)

### Editing AppArmor Profiles

Refer to Section 4.4, “Editing an AppArmor Profile” (page 40)

### Deleting AppArmor Profiles

Refer to Section 4.5, “Deleting an AppArmor Profile” (page 40)

## 4.3 Adding or Creating an AppArmor Profile

To add or create an AppArmor profile for an application, you can use a systemic or stand-alone profiling method, depending on your needs. Learn more about these two approaches in Section 4.6, “Two Methods of Profiling” (page 41).

## 4.4 Editing an AppArmor Profile

The following steps describe the procedure for editing an AppArmor profile:

- 1 If you are not currently logged in as `root`, enter `su` in a terminal window.
- 2 Enter the `root` password when prompted.
- 3 Go to the profile directory with `cd /etc/apparmor.d/`.
- 4 Enter `ls` to view all profiles currently installed.
- 5 Open the profile to edit in a text editor, such as `vim`.
- 6 Make the necessary changes then save the profile.
- 7 Restart AppArmor by entering `rcapparmor restart` in a terminal window.

## 4.5 Deleting an AppArmor Profile

The following steps describe the procedure for deleting an AppArmor profile.

- 1 If you are not currently logged in as `root`, enter `su` in a terminal window.
- 2 Enter the `root` password when prompted.
- 3 Go to the AppArmor directory with `cd /etc/apparmor.d/`.

- 4 Enter `ls` to view all the AppArmor profiles that are currently installed.
- 5 Delete the profile with `rm profilename`.
- 6 Restart AppArmor by entering `rcapparmor restart` in a terminal window.

## 4.6 Two Methods of Profiling

Given the syntax for AppArmor profiles in Chapter 2, *Profile Components and Syntax* (page 11), you could create profiles without using the tools. However, the effort involved would be substantial. To avoid such a hassle, use the AppArmor tools to automate the creation and refinement of profiles.

There are two ways to approach AppArmor profile creation. Tools are available for both methods.

### Stand-Alone Profiling

A method suitable for profiling small applications that have a finite run time, such as user client applications like mail clients. For more information, refer to Section 4.6.1, “Stand-Alone Profiling” (page 42).

### Systemic Profiling

A method suitable for profiling large numbers of programs all at once and for profiling applications that may run for days, weeks, or continuously across reboots, such as network server applications like Web servers and mail servers. For more information, refer to Section 4.6.2, “Systemic Profiling” (page 42).

Automated profile development becomes more manageable with the AppArmor tools:

- 1 Decide which profiling method suits your needs.
- 2 Perform a static analysis. Run either `aa-genprof` or `aa-autodep`, depending on the profiling method chosen.
- 3 Enable dynamic learning. Activate learning mode for all profiled programs.

## 4.6.1 Stand-Alone Profiling

Stand-alone profile generation and improvement is managed by a program called `aa-genprof`. This method is easy because `aa-genprof` takes care of everything, but is limited because it requires `aa-genprof` to run for the entire duration of the test run of your program (you cannot reboot the machine while you are still developing your profile).

To use `aa-genprof` for the stand-alone method of profiling, refer to Section “`aa-genprof`—Generating Profiles” (page 47).

## 4.6.2 Systemic Profiling

This method is called *systemic profiling* because it updates all of the profiles on the system at once, rather than focusing on the one or few targeted by `aa-genprof` or stand-alone profiling. With systemic profiling, profile construction and improvement are somewhat less automated, but more flexible. This method is suitable for profiling long-running applications whose behavior continues after rebooting or a large number of programs all at once.

Build an AppArmor profile for a group of applications as follows:

- 1 Create profiles for the individual programs that make up your application.

Although this approach is systemic, AppArmor only monitors those programs with profiles and their children. To get AppArmor to consider a program, you must at least have `aa-autodep` create an approximate profile for it. To create this approximate profile, refer to Section “`aa-autodep`—Creating Approximate Profiles” (page 44).

- 2 Put relevant profiles into learning or complain mode.

Activate learning or complain mode for all profiled programs by entering `aa-complain /etc/apparmor.d/*` in a terminal window while logged in as `root`. This functionality is also available through the YaST Profile Mode module, described in Section 3.6.2, “Changing the Mode of Individual Profiles” (page 34).

When in learning mode, access requests are not blocked even if the profile dictates that they should be. This enables you to run through several tests (as shown in Step 3 (page 43)) and learn the access needs of the program so it runs properly. With this information, you can decide how secure to make the profile.

Refer to Section “aa-complain—Entering Complain or Learning Mode” (page 45) for more detailed instructions for using learning or complain mode.

### 3 Exercise your application.

Run your application and exercise its functionality. How much to exercise the program is up to you, but you need the program to access each file representing its access needs. Because the execution is not being supervised by aa-genprof, this step can go on for days or weeks and can span complete system reboots.

### 4 Analyze the log.

In systemic profiling, run aa-logprof directly instead of letting aa-genprof run it (as in stand-alone profiling). The general form of aa-logprof is:

```
aa-logprof [ -d /path/to/profiles ] [ -f /path/to/logfile ]
```

Refer to Section “aa-logprof—Scanning the System Log” (page 54) for more information about using aa-logprof.

### 5 Repeat Step 3 (page 43) and Step 4 (page 43).

This generates optimum profiles. An iterative approach captures smaller data sets that can be trained and reloaded into the policy engine. Subsequent iterations generate fewer messages and run faster.

### 6 Edit the profiles.

You might want to review the profiles that have been generated. You can open and edit the profiles in `/etc/apparmor.d/` using vim.

### 7 Return to enforce mode.

This is when the system goes back to enforcing the rules of the profiles, not just logging information. This can be done manually by removing the `flags=(complain)` text from the profiles or automatically by using the `aa-enforce` command, which works identically to the `aa-complain` command, except it sets the profiles to enforce mode. This functionality is also available through the YaST Profile Mode module, described in Section 3.6.2, “Changing the Mode of Individual Profiles” (page 34).

To ensure that all profiles are taken out of complain mode and put into enforce mode, enter `aa-enforce /etc/apparmor.d/*`.

## 8 Rescan all profiles.

To have AppArmor rescan all of the profiles and change the enforcement mode in the kernel, enter `rcapparmor restart`.

## 4.6.3 Summary of Profiling Tools

All of the AppArmor profiling utilities are provided by the `apparmor-utils` RPM package and are stored in `/usr/sbin`. Each tool has a different purpose.

### aa-autodep—Creating Approximate Profiles

This creates an approximate profile for the program or application selected. You can generate approximate profiles for binary executables and interpreted script programs. The resulting profile is called “approximate” because it does not necessarily contain all of the profile entries that the program needs to be properly confined by AppArmor. The minimum `aa-autodep` approximate profile has at least a base include directive, which contains basic profile entries needed by most programs. For certain types of programs, `aa-autodep` generates a more expanded profile. The profile is generated by recursively calling `ldd(1)` on the executables listed on the command line.

To generate an approximate profile, use the `aa-autodep` program. The program argument can be either the simple name of the program, which `aa-autodep` finds by searching your shell's path variable, or it can be a fully qualified path. The program itself can be of any type (ELF binary, shell script, Perl script, etc.). `aa-autodep` generates an approximate profile to improve through the dynamic profiling that follows.

The resulting approximate profile is written to the `/etc/apparmor.d` directory using the AppArmor profile naming convention of naming the profile after the absolute path of the program, replacing the forward slash (`/`) characters in the path with period (`.`) characters. The general form of `aa-autodep` is to enter the following in a terminal window when logged in as `root`:

```
aa-autodep [ -d /path/to/profiles ] [program1 program2...]
```

If you do not enter the program name or names, you are prompted for them.

`/path/to/profiles` overrides the default location of `/etc/apparmor.d`, should you keep profiles in a location other than the default.

To begin profiling, you must create profiles for each main executable service that is part of your application (anything that might start without being a child of another program that already has a profile). Finding all such programs depends on the application in question. Here are several strategies for finding such programs:

### Directories

If all the programs to profile are in one directory and there are no other programs in that directory, the simple command `aa-autodep`  
`/path/to/your/programs/*` creates basic profiles for all programs in that directory.

### ps command

You can run your application and use the standard Linux `ps` command to find all processes running. Then manually hunt down the location of these programs and run the `aa-autodep` for each one. If the programs are in your path, `aa-autodep` finds them for you. If they are not in your path, the standard Linux command `find` might be helpful in finding your programs. Execute `find / -name 'my_application' -print` to determine an application's path (`my_application` being an example application). You may use wild cards if appropriate.

## aa-complain—Entering Complain or Learning Mode

The complain or learning mode tool (`aa-complain`) detects violations of AppArmor profile rules, such as the profiled program accessing files not permitted by the profile. The violations are permitted, but also logged. To improve the profile, turn complain mode on, run the program through a suite of tests to generate log events that characterize the program's access needs, then postprocess the log with the AppArmor tools to transform log events into improved 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)`. To use complain mode, open a terminal window and enter one of the following lines as `root`:

- If the example program (`program1`) is in your path, use:

```
aa-complain [program1 program2 ...]
```

- If the program is not in your path, specify the entire path as follows:

```
aa-complain /sbin/program1
```

- If the profiles are not in `/etc/apparmor.d`, use the following to override the default location:

```
aa-complain /path/to/profiles/ program1
```

- Specify the profile for `program1` as follows:

```
aa-complain /etc/apparmor.d/sbin.program1
```

Each of the above commands activates the complain mode for the profiles or programs listed. If the program name does not include its entire path, `aa-complain` searches `$PATH` for the program. For instance, `aa-complain /usr/sbin/*` finds profiles associated with all of the programs in `/usr/sbin` and puts them into complain mode.

`aa-complain /etc/apparmor.d/*` puts all of the profiles in `/etc/apparmor.d` into complain mode.

---

### TIP: Toggling Profile Mode with YaST

YaST offers a graphical front-end for toggling complain and enforce mode. See Section 3.6.2, “Changing the Mode of Individual Profiles” (page 34) for information.

---

## aa-enforce—Entering Enforce Mode

The enforce mode detects violations of AppArmor profile rules, such as the profiled program accessing files not permitted by the profile. The violations are logged and not permitted. The default is for enforce mode to be enabled. To log the violations only, but still permit them, use complain mode. Enforce toggles with complain mode.

Manually activating enforce mode (using the command line) adds a flag to the top of the profile so that `/bin/foo` becomes `/bin/foo flags=(enforce)`. To use enforce mode, open a terminal window and enter one of the following lines as `root`.

- If the example program (*program1*) is in your path, use:

```
aa-enforce [program1 program2 ...]
```

- If the program is not in your path, specify the entire path, as follows:

```
aa-enforce /sbin/program1
```

- If the profiles are not in */etc/apparmor.d*, use the following to override the default location:

```
aa-enforce /path/to/profiles/program1
```

- Specify the profile for *program1* as follows:

```
aa-enforce /etc/apparmor.d/sbin.program1
```

Each of the above commands activates the enforce mode for the profiles and programs listed.

If you do not enter the program or profile names, you are prompted to enter one.

*/path/to/profiles* overrides the default location of */etc/apparmor.d*.

The argument can be either a list of programs or a list of profiles. If the program name does not include its entire path, *aa-enforce* searches *\$PATH* for the program.

---

### **TIP: Toggling Profile Mode with YaST**

YaST offers a graphical front-end for toggling complain and enforce mode. See Section 3.6.2, “Changing the Mode of Individual Profiles” (page 34) for information.

---

## **aa-genprof—Generating Profiles**

*aa-genprof* is AppArmor's profile generating utility. It runs *aa-autodep* on the specified program, creating an approximate profile (if a profile does not already exist for it), sets it to complain mode, reloads it into AppArmor, marks the log, and prompts the user to execute the program and exercise its functionality. Its syntax is as follows:

```
aa-genprof [ -d /path/to/profiles ] program
```

To create a profile for the the Apache Web server program `httpd2-prefork`, do the following as `root`:

- 1 Enter `rcapache2 stop`.
- 2 Next, enter `aa-genprof httpd2-prefork`.

Now `aa-genprof` does the following:

1. Resolves the full path of `httpd2-prefork` using your shell's path variables. You can also specify a full path. On SUSE Linux Enterprise, the default full path is `/usr/sbin/httpd2-prefork`.
2. Checks to see if there is an existing profile for `httpd2-prefork`. If there is one, it updates it. If not, it creates one using the `aa-autodep` as described in Section 4.6.3, “Summary of Profiling Tools” (page 44).
3. Puts the profile for this program into learning or complain mode so that profile violations are logged but are permitted to proceed. A log event looks like this (see `/var/log/audit/audit.log`):

```
type=APPARMOR_ALLOWED msg=audit(1189682639.184:20816):
operation="file_mmap" requested_mask="r" denied_mask="r"
name="/srv/www/htdocs/index.html" pid=27471
profile="null-complain-profile"
```

If you are not running the audit daemon, the AppArmor events are logged to `/var/log/messages`:

```
Mar 13 13:20:30 K23 kernel: audit(1189682430.672:20810):
operation="file_mmap" requested_mask="r" denied_mask="r"
name="/srv/www/htdocs/phpsysinfo/templates/bulix/form.tpl" pid=30405
profile="/usr/sbin/httpd2-prefork//phpsysinfo/"
```

They also can be viewed using the `dmesg` command:

```
audit(1189682430.672:20810): operation="file_mmap" requested_mask="r"
denied_mask="r" name="/srv/www/htdocs/phpsysinfo/templates/bulix/form.tpl"
pid=30405 profile="/usr/sbin/httpd2-prefork//phpsysinfo/"
```

4. Marks the log with a beginning marker of log events to consider. For example:

```
Sep 13 17:48:52 figwit root: GenProf: e2ff78636296f16d0b5301209a04430d
```

- 3 When prompted by the tool, run the application to profile in another terminal window and perform as many of the application functions as possible. Thus, the learning mode can log the files and directories to which the program requires access in order to function properly. For example, in a new terminal window, enter `rcapache2 start`.
- 4 Select from the following options that are available in the `aa-logprof` terminal window after you have executed the program function:
  - `S` runs `aa-logprof` on the system log from where it was marked when `aa-genprof` was started and reloads the profile. If system events exist in the log, `AppArmor` parses the learning mode log files. This generates a series of questions that you must answer to guide `aa-genprof` in generating the security profile.
  - `F` exits the tool and returns to the main menu.

---

**NOTE**

If requests to add hats appear, proceed to Chapter 5, *Profiling Your Web Applications Using ChangeHat* (page 69).

---

- 5 Answer two types of questions:
  - A resource is requested by a profiled program that is not in the profile (see Example 4.1, “Learning Mode Exception: Controlling Access to Specific Resources” (page 50)).
  - A program is executed by the profiled program and the security domain transition has not been defined (see Example 4.2, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 51)).

Each of these categories results in a series of questions that you must answer to add the resource or program to the profile. Example 4.1, “Learning Mode Exception: Controlling Access to Specific Resources” (page 50) and Example 4.2, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 51) provide examples of each one. Subsequent steps describe your options in answering these questions.

- Dealing with execute accesses is complex. You must decide how to proceed with this entry regarding which execute permission type to grant to this entry:

### Example 4.1 Learning Mode Exception: Controlling Access to Specific Resources

```
Reading log entries from /var/log/audit/audit.log.  
Updating AppArmor profiles in /etc/apparmor.d.
```

```
Profile: /usr/sbin/xinetd  
Program: xinetd  
Execute: /usr/lib/cups/daemon/cups-lpd  
Severity: unknown
```

```
[(I)nherit] / (P)rofile / (U)nconfined / (D)eny / Abo(r)t / (F)inish
```

#### Inherit (ix)

The child inherits the parent's profile, running with the same access controls as the parent. This mode is useful when a confined program needs to call another confined program without gaining the permissions of the target's profile or losing the permissions of the current profile. This mode is often used when the child program is a *helper application*, such as the `/usr/bin/mail` client using `less` as a pager or the Mozilla\* Web browser using Adobe Acrobat\* to display PDF files.

#### Profile (px)

The child runs using its own profile, which must be loaded into the kernel. If the profile is not present, attempts to execute the child fail with permission denied. This is most useful if the parent program is invoking a global service, such as DNS lookups or sending mail with your system's MTA.

Choose the *profile with clean exec* (Px) option to scrub the environment of environment variables that could modify execution behavior when passed to the child process.

#### Unconfined (ux)

The child runs completely unconfined without any AppArmor profile applied to the executed resource.

Choose the *unconfined with clean exec* (Ux) option to scrub the environment of environment variables that could modify execution behavior when passed to the child process. This option introduces a security vulnerability that could be used to exploit AppArmor. Only use it as a last resort.

#### mmap (m)

This permission denotes that the program running under the profile can access the resource using the `mmap` system call with the flag `PROT_EXEC`. This

means that the data mapped in it can be executed. You are prompted to include this permission if it is requested during a profiling run.

#### Deny

Prevents the program from accessing the specified directory path entries. AppArmor then continues to the next event.

#### Abort

Aborts aa-logprof, losing all rule changes entered so far and leaving all profiles unmodified.

#### Finish

Closes aa-logprof, saving all rule changes entered so far and modifying all profiles.

- Example 4.2, “Learning Mode Exception: Defining Execute Permissions for an Entry” (page 51) shows AppArmor suggesting directory path entries that have been accessed by the application being profiled. It might also require you to define execute permissions for entries.

### **Example 4.2** *Learning Mode Exception: Defining Execute Permissions for an Entry*

```
Adding /bin/ps ix to profile.
```

```
Profile: /usr/sbin/xinetd
Path: /etc/hosts.allow
New Mode: r
```

```
[1 - /etc/hosts.allow]
```

```
[(A)llow] / (D)eny / (N)ew / (G)lob / Glob w/(E)xt / Abo(r)t / (F)inish
```

AppArmor provides one or more paths or includes. By entering the option number, select the desired options then proceed to the next step.

---

#### **NOTE**

All of these options are not always presented in the AppArmor menu.

---

#### `#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.

#### Globbered Version

This is accessed by selecting *Glob* as described in the next step. For information about globbing syntax, refer to Section 4.7, “Paths and Globbing” (page 60).

#### Actual Path

This is the literal path to which the program needs access so that it can run properly.

After you select the path or include, process it as an entry into the AppArmor profile by selecting *Allow* or *Deny*. If you are not satisfied with the directory path entry as it is displayed, you can also *Glob* it.

The following options are available to process the learning mode entries and build the profile:

#### Select Enter

Allows access to the selected directory path.

#### Allow

Allows access to the specified directory path entries. AppArmor suggests file permission access. For more information, refer to Section 4.8, “File Permission Access Modes” (page 62).

#### Deny

Prevents the program from accessing the specified directory path entries. AppArmor then continues to the next event.

#### New

Prompts you to enter your own rule for this event, allowing you to specify a regular expression. If the expression does not actually satisfy the event that prompted the question in the first place, AppArmor asks for confirmation and lets you reenter the expression.

#### Glob

Select a specific path or create a general rule using wild cards that match a broader set of paths. To select any of the offered paths, enter the number that

is printed in front of the path then decide how to proceed with the selected item.

For more information about globbing syntax, refer to Section 4.7, “Paths and Globbing” (page 60).

#### Glob w/Ext

This modifies the original directory path while retaining the filename extension. For example, `/etc/apache2/file.ext` becomes `/etc/apache2/*.ext`, adding the wild card (asterisk) in place of the filename. This allows the program to access all files in the suggested directory that end with the `.ext` extension.

#### Abort

Aborts `aa-logprof`, losing all rule changes entered so far and leaving all profiles unmodified.

#### Finish

Closes `aa-logprof`, saving all rule changes entered so far and modifying all profiles.

- 6** To view and edit your profile using `vim`, enter `vim /etc/apparmor.d/profilename` in a terminal window.
- 7** Restart AppArmor and reload the profile set including the newly created one using the `rcapparmor restart` command.

Like the graphical front-end for building AppArmor profiles, the YaST Add Profile Wizard, `aa-genprof` also supports the use of the local profile repository under `/etc/apparmor/profiles/extras`

To use a profile from the local repository, proceed as follows:

- 1** Start `aa-genprof` as described above.

If `aa-genprof` finds an inactive local profile, the following lines appear on your terminal window:

```
Profile: /usr/lib/firefox/firefox.sh
[1 - Inactive local profile for /usr/bin/opera]
```

[ (V)iew Profile ] / (U)se Profile / (C)reate New Profile / Abo(r)t / (F)inish

- 2 If you want to just use this profile, hit U (*Use Profile*) and follow the profile generation procedure outlined above.

If you want to examine the profile before activating it, hit V (*View Profile*).

If you want to ignore the existing profile, hit C (*Create New Profile*) and follow the profile generation procedure outlined above to create the profile from scratch.

- 3 Leave aa-genprof by hitting F (*Finish*) when you are done and save your changes.

## aa-logprof—Scanning the System Log

aa-logprof is an interactive tool used to review the learning or complain mode output found in the log entries in `/var/log/audit/audit.log` or `/var/log/messages` (if auditd is not running) and generate new entries in AppArmor security profiles.

When you run aa-logprof, it begins to scan the log files produced in learning or complain mode and, if there are new security events that are not covered by the existing profile set, it gives suggestions for modifying the profile. The learning or complain mode traces program behavior and enters it in the log. aa-logprof uses this information to observe program behavior.

If a confined program forks and executes another program, aa-logprof sees this and asks the user which execution mode should be used when launching the child process. The execution modes *ix*, *px*, *Px*, *ux*, and *Ux* are options for starting the child process. If a separate profile exists for the child process, the default selection is *px*. If one does not exist, the profile defaults to *ix*. Child processes with separate profiles have aa-autodep run on them and are loaded into AppArmor, if it is running.

When aa-logprof exits, profiles are updated with the changes. If the AppArmor module is running, the updated profiles are reloaded and, if any processes that generated security events are still running in the null-complain-profile, those processes are set to run under their proper profiles.

To run aa-logprof, enter `aa-logprof` into a terminal window while logged in as `root`. The following options can be used for aa-logprof:

```
aa-logprof -d /path/to/profile/directory/
```

Specifies the full path to the location of the profiles if the profiles are not located in the standard directory, `/etc/apparmor.d/`.

```
aa-logprof -f /path/to/logfile/
```

Specifies the full path to the location of the log file if the log file is not located in the default directory, `/var/log/audit/audit.log` or `/var/log/messages` (if `auditd` is not running).

```
aa-logprof -m "string marker in logfile"
```

Marks the starting point for `aa-logprof` to look in the system log. `aa-logprof` ignores all events in the system log before the specified mark. If the mark contains spaces, it must be surrounded by quotes to work correctly. For example:

```
aa-logprof -m"17:04:21"
```

or

```
logprof -m e2ff78636296f16d0b5301209a04430d
```

`aa-logprof` scans the log, asking you how to handle each logged event. Each question presents a numbered list of AppArmor rules that can be added by pressing the number of the item on the list.

By default, `aa-logprof` looks for profiles in `/etc/apparmor.d/` and scans the log in `/var/log/messages`. In many cases, running `aa-logprof` as `root` is enough to create the profile.

However, there might be times when you need to search archived log files, such as if the program exercise period exceeds the log rotation window (when the log file is archived and a new log file is started). If this is the case, you can enter `zcat -f `ls -ltr /var/log/messages*` | aa-logprof -f -`.

## aa-logprof Example 1

The following is an example of how `aa-logprof` addresses `httpd2-prefork` accessing the file `/etc/group`. `[]` indicates the default option.

In this example, the access to `/etc/group` is part of `httpd2-prefork` accessing name services. The appropriate response is `1`, which includes a predefined set of AppArmor rules. Selecting `1` to `#include` the name service package resolves all of the future

questions pertaining to DNS lookups and also makes the profile less brittle in that any changes to DNS configuration and the associated name service profile package can be made just once, rather than needing to revise many profiles.

```
Profile: /usr/sbin/httpd2-prefork
Path: /etc/group
New Mode: r

[1 - #include <abstractions/nameservice>]
 2 - /etc/group
[(A)llow] / (D)eny / (N)ew / (G)lob / Glob w/(E)xt / Abo(r)t / (F)inish
```

Select one of the following responses:

#### Select Enter

Triggers the default action, which is, in this example, allowing access to the specified directory path entry.

#### Allow

Allows access to the specified directory path entries. AppArmor suggests file permission access. For more information about this, refer to Section 4.8, “File Permission Access Modes” (page 62).

#### Deny

Prevents the program from accessing the specified directory path entries. AppArmor then continues to the next event.

#### New

Prompts you to enter your own rule for this event, allowing you to specify whatever form of regular expression you want. If the expression entered does not actually satisfy the event that prompted the question in the first place, AppArmor asks for confirmation and lets you reenter the expression.

#### Glob

Select either a specific path or create a general rule using wild cards that matches on a broader set of paths. To select any of the offered paths, enter the number that is printed in front of the paths then decide how to proceed with the selected item.

For more information about globbing syntax, refer to Section 4.7, “Paths and Globbing” (page 60).

## Glob w/Ext

This modifies the original directory path while retaining the filename extension. For example, `/etc/apache2/file.ext` becomes `/etc/apache2/*.ext`, adding the wild card (asterisk) in place of the filename. This allows the program to access all files in the suggested directory that end with the `.ext` extension.

## Abort

Aborts `aa-logprof`, losing all rule changes entered so far and leaving all profiles unmodified.

## Finish

Closes `aa-logprof`, saving all rule changes entered so far and modifying all profiles.

# aa-logprof Example 2

For example, when profiling `vsftpd`, see this question:

```
Profile: /usr/sbin/vsftpd
Path:    /y2k.jpg
New Mode: r
```

```
[1 - /y2k.jpg]
```

```
(A)llow / [(D)eny] / (N)ew / (G)lob / Glob w/(E)xt / Abo(r)t / (F)inish
```

Several items of interest appear in this question. First, note that `vsftpd` is asking for a path entry at the top of the tree, even though `vsftpd` on SUSE Linux Enterprise serves FTP files from `/srv/ftp` by default. This is because `httpd2-prefork` uses `chroot` and, for the portion of the code inside the `chroot` jail, `AppArmor` sees file accesses in terms of the `chroot` environment rather than the global absolute path.

The second item of interest is that you might want to grant FTP read access to all JPEG files in the directory, so you could use *Glob w/Ext* and use the suggested path of `/*.jpg`. Doing so collapses all previous rules granting access to individual `.jpg` files and forestalls any future questions pertaining to access to `.jpg` files.

Finally, you might want to grant more general access to FTP files. If you select *Glob* in the last entry, `aa-logprof` replaces the suggested path of `/y2k.jpg` with `/*`. Alternatively, you might want to grant even more access to the entire directory tree, in which case you could use the *New path* option and enter `/**/*.jpg` (which would grant access

to all `.jpg` files in the entire directory tree) or `/**` (which would grant access to all files in the directory tree).

These items deal with read accesses. Write accesses are similar, except that it is good policy to be more conservative in your use of regular expressions for write accesses. Dealing with execute accesses is more complex. Find an example in Example 4.1, “Learning Mode Exception: Controlling Access to Specific Resources” (page 50).

In the following example, the `/usr/bin/mail` mail client is being profiled and `aa-logprof` has discovered that `/usr/bin/mail` executes `/usr/bin/less` as a helper application to “page” long mail messages. Consequently, it presents this prompt:

```
/usr/bin/nail -> /usr/bin/less
(I)nherit / (P)rofile / (U)nconfined / (D)eny
```

---

### TIP

The actual executable file for `/usr/bin/mail` turns out to be `/usr/bin/nail`, which is not a typographical error.

---

The program `/usr/bin/less` appears to be a simple one for scrolling through text that is more than one screen long and that is in fact what `/usr/bin/mail` is using it for. However, `less` is actually a large and powerful program that makes use of many other helper applications, such as `tar` and `rpm`.

---

### TIP

Run `less` on a tar file or an RPM file and it shows you the inventory of these containers.

---

You do not want to run `rpm` automatically when reading mail messages (that leads directly to a Microsoft\* Outlook-style virus attack, because `rpm` has the power to install and modify system programs), so, in this case, the best choice is to use *Inherit*. This results in the `less` program executed from this context running under the profile for `/usr/bin/mail`. This has two consequences:

- You need to add all of the basic file accesses for `/usr/bin/less` to the profile for `/usr/bin/mail`.

- You can avoid adding the helper applications, such as tar and rpm, to the `/usr/bin/mail` profile so that when `/usr/bin/mail` runs `/usr/bin/less` in this context, the less program is far less dangerous than it would be without AppArmor protection.

In other circumstances, you might instead want to use the *Profile* option. This has two effects on `aa-logprof`:

- The rule written into the profile uses `px`, which forces the transition to the child's own profile.
- `aa-logprof` constructs a profile for the child and starts building it, in the same way that it built the parent profile, by assigning events for the child process to the child's profile and asking the `aa-logprof` user questions.

If a confined program forks and executes another program, `aa-logprof` sees this and asks the user which execution mode should be used when launching the child process. The execution modes of `inherit`, `profile`, `unconfined` or an option to deny the execution are presented.

If a separate profile exists for the child process, the default selection is `profile`. If a profile does not exist, the default is `inherit`. The `inherit` option, or `ix`, is described in Section 4.8, “File Permission Access Modes” (page 62).

The `profile` option indicates that the child program should run in its own profile—a secondary question asks whether to sanitize the environment that the child program inherits from the parent. If you choose to sanitize the environment, this places the execution modifier `Px` in your AppArmor profile. If you select not to sanitize, `px` is placed in the profile and no environment sanitizing occurs. The default for the execution mode is `px` if you select `profile` execution mode.

The `unconfined` execution mode is not recommended and should only be used in cases where there is no other option to generate a profile for a program reliably. Selecting `unconfined` opens a warning dialog asking for confirmation of the choice. If you are sure and choose *Yes*, a second dialog asks whether to sanitize the environment. Choosing *Yes* uses the execution mode `Ux` in your profile. Choosing *No* uses the execution mode `ux` for your profile. The default value selected is `Ux` for `unconfined` execution mode.

---

**IMPORTANT: Running Unconfined**

Choosing `ux` is very dangerous and provides no enforcement of policy from a security perspective of resulting execution behavior of the child program.

---

## aa-unconfined—Identifying Unprotected Processes

The `aa-unconfined` command examines open network ports on your system, compares that to the set of profiles loaded on your system, and reports network services that do not have AppArmor profiles. It requires `root` privileges and that it not be confined by an AppArmor profile.

`aa-unconfined` must be run as `root` to retrieve the process executable link from the `/proc` file system. This program is susceptible to the following race conditions:

- An unlinked executable is mishandled
- A process that dies between `netstat(8)` and further checks is mishandled

---

**NOTE**

This program lists processes using TCP and UDP only. In short, this program is unsuitable for forensics use and is provided only as an aid to profiling all network-accessible processes in the lab.

---

## 4.7 Paths and Globbing

Globbing (or regular expression matching) is when you modify the directory path using wild cards to include a group of files or subdirectories. File resources can be specified with a globbing syntax similar to that used by popular shells, such as `csh`, `Bash`, and `zsh`.

---

*	Substitutes for any number of any characters, except /.  Example: An arbitrary number of a path element, including entire directories.
**	Substitutes for any number of characters, including /.  Example: an arbitrary number of path elements, including entire directories.
?	Substitutes for any single character, except /.
[abc]	Substitutes for the single character a, b, or c.  Example: a rule that matches <code>/home[01]/*/.plan</code> allows a program to access <code>.plan</code> files for users in both <code>/home0</code> and <code>/home1</code> .
[a-c]	Substitutes for the single character a, b, or c.
{ab, cd}	Expands to one rule to match <code>ab</code> and one rule to match <code>cd</code> .  Example: a rule that matches <code>{usr, www}/pages/**</code> grants access to Web pages in both <code>/usr/pages</code> and <code>/www/pages</code> .

---

## 4.8 File Permission Access Modes

File permission access modes consist of combinations of the following nine modes:

---

r	Read mode
w	Write mode
px	Discrete profile execute mode
Px	Discrete profile execute mode—clean exec
ux	Unconstrained execute mode
Ux	Unconstrained execute mode—clean exec
ix	Inherit execute mode
m	Allow <code>PROT_EXEC</code> with <code>mmap(2)</code> calls
l	Link mode

---

### Read Mode (r)

Allows the program to have read access to the resource. Read access is required for shell scripts and other interpreted content and determines if an executing process can core dump or be attached to with `ptrace(2)` (`ptrace(2)` is used by utilities like `strace(1)`, `ltrace(1)`, and `gdb(1)`).

### Write Mode (w)

Allows the program to have write access to the resource. Files must have this permission if they are to be unlinked (removed).

### Discrete Profile Execute Mode (px)

This mode requires that a discrete security profile is defined for a resource executed at an AppArmor domain transition. If there is no profile defined, the access is denied.

---

**WARNING: Using the Discrete Profile Execute Mode**

---

`px` does not scrub the environment of variables such as `LD_PRELOAD`. As a result, the calling domain may have an undue amount of influence over the called item.

---

Incompatible with `Ux`, `ux`, `Px`, and `ix`.

**Discrete Profile Execute Mode (Px)—Clean Exec**

`Px` allows the named program to run in `px` mode, but AppArmor invokes the Linux kernel's `unsafe_exec` routines to scrub the environment, similar to `setuid` programs. See `ld.so(8)` for some information about `setuid` and `setgid` environment scrubbing.

Incompatible with `Ux`, `ux`, `px`, and `ix`.

**Unconstrained Execute Mode (ux)**

Allows the program to execute the resource without any AppArmor profile applied to the executed resource. Requires listing `execute mode` as well.

This mode is useful when a confined program needs to be able to perform a privileged operation, such as rebooting the machine. By placing the privileged section in another executable and granting unconstrained execution rights, it is possible to bypass the mandatory constraints imposed on all confined processes. For more information about what is constrained, see the `apparmor(7)` man page.

---

**WARNING: Using Unconstrained Execute Mode (ux)**

Use `ux` only in very special cases. It enables the designated child processes to be run without any AppArmor protection. `ux` does not scrub the environment of variables such as `LD_PRELOAD`. As a result, the calling domain may have an undue amount of influence over the called resource. Use this mode only if the child absolutely must be run unconfined and `LD_PRELOAD` must be used. Any profile using this mode provides negligible security. Use at your own risk.

---

This mode is incompatible with `Ux`, `px`, `Px`, and `ix`.

**Unconstrained Execute Mode (Ux)—Clean Exec**

`Ux` allows the named program to run in `ux` mode, but AppArmor invokes the Linux kernel's `unsafe_exec` routines to scrub the environment, similar to `setuid` programs. See `ld.so(8)` for some information about `setuid` and `setgid` environment scrubbing.

---

**WARNING: Using Unconstrained Execute Mode (Ux)**

Use `Ux` only in very special cases. It enables the designated child processes to run without any AppArmor protection. Use this mode only if the child absolutely must be run unconfined. Use at your own risk.

---

Incompatible with `ux`, `px`, `Px`, and `ix`.

**Inherit Execute Mode (ix)**

`ix` prevents the normal AppArmor domain transition on `execve(2)` when the profiled program executes the named program. Instead, the executed resource inherits the current profile.

This mode is useful when a confined program needs to call another confined program without gaining the permissions of the target's profile or losing the permissions of the current profile. There is no version to scrub the environment because `ix` executions do not change privileges.

Incompatible with `Ux`, `ux`, `Px`, and `px`. Implies `m`.

## Allow Executable Mapping (m)

This mode allows a file to be mapped into memory using `mmap(2)`'s `PROT_EXEC` flag. This flag marks the pages executable. It is used on some architectures to provide nonexecutable data pages, which can complicate exploit attempts. AppArmor uses this mode to limit which files a well-behaved program (or all programs on architectures that enforce nonexecutable memory access controls) may use as libraries, to limit the effect of invalid `-L` flags given to `ld(1)` and `LD_PRELOAD`, `LD_LIBRARY_PATH`, given to `ld.so(8)`.

## Link Mode

The link mode mediates access to hard links. When a link is created, the target file must have the same access permissions as the link created (with the exception that the destination does not need link access).

When choosing one of the `Ux` or `Px` file permission access modes, take into account that the following environment variables are removed from the environment before the child process inherits it. As a consequence, applications or processes relying on any of these variables do not work anymore if the profile applied to them carries `Ux` or `Px` flags:

- `GCONV_PATH`
- `GETCONF_DIR`
- `HOSTALIASES`
- `LD_AUDIT`
- `LD_DEBUG`
- `LD_DEBUG_OUTPUT`
- `LD_DYNAMIC_WEAK`
- `LD_LIBRARY_PATH`
- `LD_ORIGIN_PATH`
- `LD_PRELOAD`

- LD\_PROFILE
- LD\_SHOW\_AUXV
- LD\_USE\_LOAD\_BIAS
- LOCALDOMAIN
- LOCPATH
- MALLOC\_TRACE
- NLSPATH
- RESOLV\_HOST\_CONF
- RES\_OPTIONS
- TMPDIR
- TZDIR

## 4.9 Important Filenames and Directories

The following list contains the most important files and directories used by the AppArmor framework. If you intend to manage and troubleshoot your profiles manually, make sure that you know about these files and directories:

`/sys/kernel/security/apparmor/profiles`

Virtualized file representing the currently loaded set of profiles.

`/etc/apparmor/`

Location of AppArmor configuration files.

`/etc/apparmor.d/`

Location of profiles, named with the convention of replacing the `/` in paths with `.` (not for the root `/`) so profiles are easier to manage. For example, the profile for the program `/usr/sbin/ntpd` is named `usr.sbin.ntpd`.

`/etc/apparmor.d/abstractions/`

Location of abstractions.

`/etc/apparmor.d/program-chunks/`

Location of program chunks.

`/proc/*/attr/current`

Check this file to review the confinement status of a process and the profile that is used to confine the process. The `ps auxZ` command retrieves this information automatically.



# Profiling Your Web Applications Using ChangeHat

A Novell® AppArmor profile represents the 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*.

ChangeHat enables programs to change to or from a *hat* within a Novell AppArmor profile. It enables you to define security at a finer level than the process. This feature requires that each application be made “ChangeHat aware” meaning that it is modified to make a request to the Novell AppArmor module to switch security domains at arbitrary times during the application execution. Two examples for ChangeHat-aware applications are the Apache Web server and Tomcat.

A profile can have an arbitrary number of subprofiles, but there are only two levels: a subprofile cannot have further 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.

Note that the security of hats is considerably weaker than that of full profiles. That is to say, if an attacker can find just the right kind of bug in a program, they may be able to escape from a hat into the containing profile. This is because the security of hats is determined by a secret key handled by the containing process, and the code running in the hat must not have access to the key. Thus `change_hat` is most useful in conjunction with application servers, where a language interpreter (such as PERL, PHP, or Java) is isolating pieces of code such that they do not have direct access to the memory of the containing process.

The rest of this chapter describes using `change_hat` in conjunction with Apache, to contain web server components run using `mod_perl` and `mod_php`. Similar approaches can be used with any application server by providing an application module similar to the `mod_apparmor` described next in Section 5.2.2, “Location and Directory Directives” (page 78).

---

**NOTE: For More Information**

For more information, see the `change_hat` man page.

---

## 5.1 Apache ChangeHat

Novell AppArmor provides a `mod_apparmor` module (package `apache2-mod-apparmor`) for the Apache program (only included in SUSE Linux Enterprise Server). This module makes the Apache Web server ChangeHat aware. Install it along with Apache.

When Apache is ChangeHat aware, it checks for the following customized Novell AppArmor security profiles in the order given for every URI request that it receives.

- URI-specific hat (for example, `^phpsysinfo/templates/classic/images/bar_left.gif`)
- `DEFAULT_URI`
- `HANDLING_UNTRUSTED_INPUT`

---

**NOTE: Apache Configuration**

If you install `apache2-mod-apparmor`, make sure the module gets loaded in Apache by executing the following command:

```
a2enmod apparmor
```

---

## 5.1.1 Managing ChangeHat-Aware Applications

As with most of the Novell AppArmor tools, you can use two methods for managing ChangeHat, YaST or the command line interface. Managing ChangeHat-aware applications from the command line is much more flexible, but the process is also more complicated. Both methods allow you to manage the hats for your application and populate them with profile entries.

The following steps are a demonstration that adds hats to an Apache profile using YaST. In the *Add Profile Wizard*, the Novell AppArmor profiling utilities prompt you to create new hats for distinct URI requests. Choosing to create a new hat allows you to create individual profiles for each URI. You can create very tight rules for each request.

If the URI that is processed does not represent significant processing or otherwise does not represent a significant security risk, safely select *Use Default Hat* to process this URI in the default hat, which is the default security profile.

This example creates a new hat for the URI `phpsysinfo` and its subsequent accesses. Using the profiling utilities, delegate what to add to this new hat. The resulting hat becomes a tight-security container that encompasses all the processing on the server that occurs when the `phpsysinfo` URI is passed to the Apache Web server.

The URI runs the application `phpsysinfo` (refer to <http://phpsysinfo.sourceforge.net> for more information). The `phpsysinfo` package is assumed to be installed in `/srv/www/htdocs/phpsysinfo` in a clean (new) installation of SUSE Linux Enterprise and AppArmor.

- 1 Once `phpsysinfo` is installed, you are ready to add hats to the Apache profile. From the Novell AppArmor GUI, select *Add Profile Wizard*.

This wizard will help you create a new AppArmor security profile for an application, or you can use it to enhance an existing profile by allowing AppArmor to learn new application behavior.

Please enter the application name for which you would like to create a profile, or select **Browse** to find the application on your system.

Application to Profile

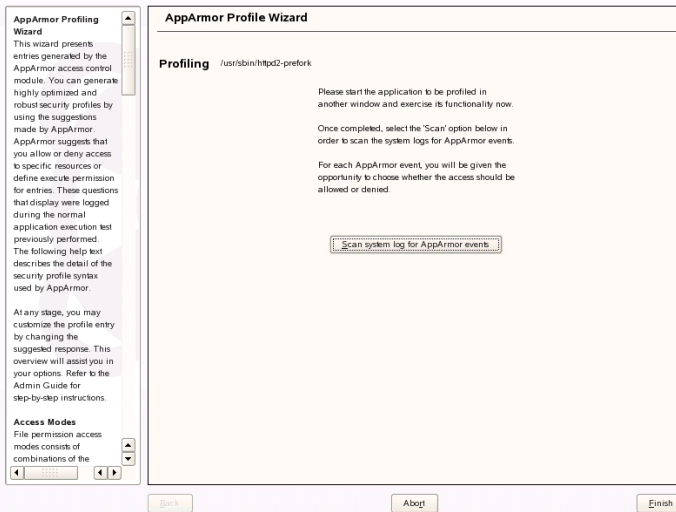
**Browse**

**Create**

**Abort**

2 In *Application to Profile*, enter `/usr/sbin/httpd2-prefork`.

3 Click *Create*.



4 Restart Apache by entering `rcapache2 restart` in a terminal window.

Restart any program you are profiling at this point.

- 5 Open `http://localhost/phpsysinfo/` in a Web browser window. The browser window should display network usage and system information.

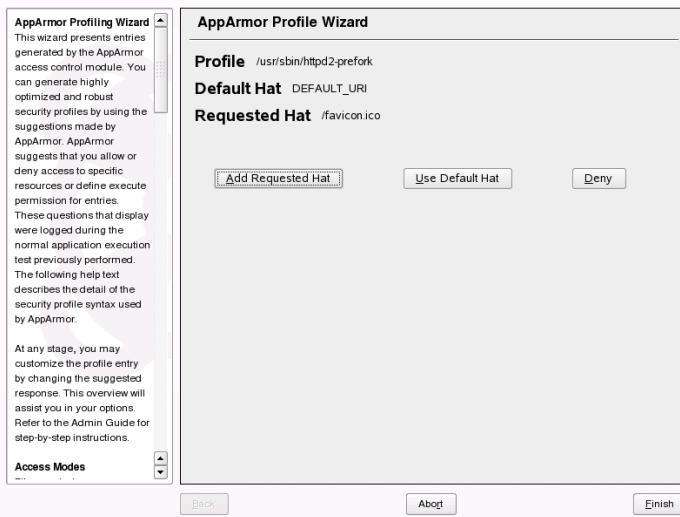
---

**NOTE: Data Caching**

To ensure that this request is processed by the server and you do not review cached data in your browser, refresh the page. To do this, click the browser *Refresh* button to make sure that Apache processes the request for the `phpsysinfo` URI.

---

- 6 Click *Scan System Log for Entries to Add to Profiles*. Novell AppArmor launches the `aa-logprof` tool, which scans the information learned in the previous step. It begins to prompt you with profile questions.
- 7 `aa-logprof` first prompts with *Add Requested Hat* or *Use Default Hat* because it noticed that the `phpsysinfo` URI was accessed. Select *Add Requested Hat*.



- 8 Click *Allow*.

Choosing *Add Requested Hat* in the previous step creates a new hat in the profile and specifies that the results of subsequent questions about the script's actions are added to the newly created hat rather than the default hat for this application.

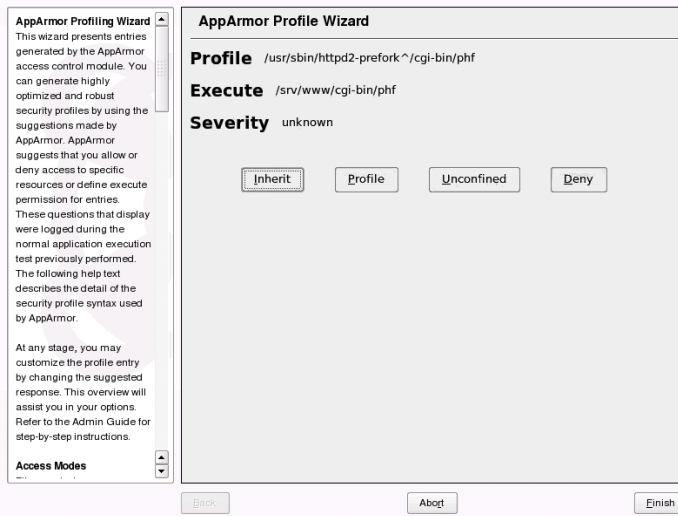
In the next screen, Novell AppArmor displays an external program that the script executed. You can specify that the program should run confined by the `phpsysinfo` hat (choose *Inherit*), confined by a separate profile (choose *Profile*), or that it should run unconfined or without any security profile (choose *Unconfined*). For the case of the *Profile* option, a new profile is created for the program if one does not already exist.

---

## NOTE: Security Considerations

Selecting *Unconfined* can create a significant security hole and should be done with caution.

---



**8a** Select *Inherit* for the `/bin/bash` path. This adds `/bin/bash` (accessed by Apache) to the `phpsysinfo` hat profile with the necessary permissions.

**8b** Click *Allow*.

**9** The remaining questions prompt you to generate new hats and add entries to your profile and its hats. The process of adding entries to profiles is covered in detail in the Section 3.1, “Adding a Profile Using the Wizard” (page 18).

When all profiling questions are answered, click *Finish* to save your changes and exit the wizard.

The following is an example `phpsysinfo` hat.

**Example 5.1** *Example phpsysinfo Hat*

```
/usr/sbin/httpd2-prefork {
...
^phpsysinfo {
  #include <abstractions/bash>
  #include <abstractions/namespace>

  /bin/basename          ixr,
  /bin/bash              ixr,
  /bin/df                ixr,
  /bin/grep              ixr,
  /bin/mount             Ux,
  /bin/sed               ixr,
  /dev/bus/usb/          r,
  /dev/bus/usb/**       r,
  /dev/null              w,
  /dev/tty               rw,
  /dev/urandom           r,
  /etc/SuSE-release      r,
  /etc/ld.so.cache       r,
  /etc/lsb-release       r,
  /etc/lsb-release.d/   r,
  /lib/ld-2.6.1.so       ixr,
  /proc/**               r,
  /sbin/lspci            ixr,
  /srv/www/htdocs/phpsysinfo/** r,
  /sys/bus/pci/**        r,
  /sys/bus/scsi/devices/ r,
  /sys/devices/**        r,
  /usr/bin/cut           ixr,
  /usr/bin/getopt        ixr,
  /usr/bin/head          ixr,
  /usr/bin/lsb_release   ixr,
  /usr/bin/lsscsi        ixr,
  /usr/bin/tr            ixr,
  /usr/bin/who           ixr,
  /usr/lib/lib*so*       mr,
  /usr/lib/locale/**     r,
  /usr/sbin/lsub         ixr,
  /usr/share/locale/**   r,
  /usr/share/pki.ids     r,
  /usr/share/usb.ids     r,
  /var/log/apache2/access_log w,
  /var/run/utmp          r,
}
}
```

---

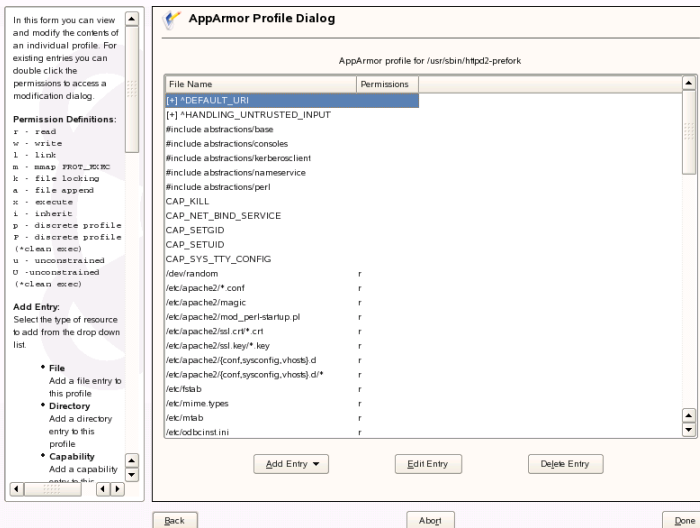
## NOTE: Hat and Parent Profile Relationship

The profile `^phpsysinfo` is only valid in the context of a process running under the parent profile `httpd2-prefork`.


---

## 5.1.2 Adding Hats and Entries to Hats

When you use the *Edit Profile* dialog (for instructions, refer to Section 3.3, “Editing Profiles” (page 26)) or when you add a new profile using *Manually Add Profile* (for instructions, refer to Section 3.2, “Manually Adding a Profile” (page 25)), you are given the option of adding hats (subprofiles) to your Novell AppArmor profiles. Add a *ChangeHat* subprofile from the *AppArmor Profile Dialog* window as in the following.



- 1 From the *AppArmor Profile Dialog* window, click *Add Entry* then select *Hat*. The *Enter Hat Name* dialog box opens:



Please enter the name of the Hat that you would like to add to the profile /usr/sbin/httpd2-prefork.

Hat name to add

Create Hat

Abort

- 2 Enter the name of the hat to add to the Novell AppArmor profile. The name is the URI that, when accessed, receives the permissions set in the hat.
- 3 Click *Create Hat*. You are returned to the *AppArmor Profile Dialog* screen.
- 4 After adding the new hat, click *Done*.

---

#### NOTE: For More Information

For an example of an Novell AppArmor profile, refer to Example 5.1, “Example phpsysinfo Hat” (page 75).

---

## 5.2 Configuring Apache for mod\_apparmor

Apache is configured by placing directives in plain text configuration files. The main configuration file is usually `httpd.conf`. When you compile Apache, you can indicate the location of this file. Directives can be placed in any of these configuration files to alter the way Apache behaves. When you make changes to the main configuration files, you need to start or restart Apache so the changes are recognized.

### 5.2.1 Virtual Host Directives

Virtual host directives control whether requests that contain trailing pathname information following an actual filename or that refer to a nonexistent file in an existing directory are accepted or rejected. For Apache documentation on virtual host directives, refer

to <http://httpd.apache.org/docs-2.2/mod/core.html#virtualhost>.

The ChangeHat-specific configuration keyword is `AADefaultHatName`. It is used similarly to `AAHatName`, for example, `AADefaultHatName My_Funky_Default_Hat`.

The configuration option is actually based on a server directive, which enables you to use the keyword outside of other options, setting it for the default server. Virtual hosts are considered internally within Apache to be separate “servers,” so you can set a default hat name for the default server as well as one for each virtual host, if desired.

When a request comes in, the following steps reflect the sequence in which `mod_apparmor` attempts to apply hats.

1. A location or directory hat as specified by the `AAHatName` keyword
2. A hat named by the entire URI path
3. A default server hat as specified by the `AADefaultHatName` keyword
4. `DEFAULT_URI` (if none of those exist, it goes back to the “parent” Apache hat)

## 5.2.2 Location and Directory Directives

Location and directory directives specify hat names in the program configuration file so the program calls the hat regarding its security. For Apache, you can find documentation about the location and directory directives at <http://httpd.apache.org/docs-2.0/sections.html>.

The location directive example below specifies that, for a given location, `mod_apparmor` should use a specific hat:

```
<Location /foo/> AAHatName MY_HAT_NAME </Location>
```

This tries to use `MY_HAT_NAME` for any URI beginning with `/foo/` (`/foo/`, `/foo/bar`, `/foo/cgi/path/blah_blah/blah`, etc.).

The directory directive works similarly to the location directive, except it refers to a path in the file system as in the following example:

```
<Directory "/srv/www/www.immunix.com/docs">
  # Note lack of trailing slash
  AAHatName immunix.com
</Directory>
```

**Example:** The program `phpsysinfo` is used to illustrate a location directive in the following example. The tarball can be downloaded from <http://phpsysinfo.sourceforge.net>.

- 1 After downloading the tarball, install it into `/srv/www/htdocs/phpsysinfo`.
- 2 Create `/etc/apache2/conf.d/phpsysinfo.conf` and add the following text to it:

```
<Location "/phpsysinfo">
  AAHatName phpsysinfo
</Location>
```

The following hat should then work for `phpsysinfo`:

```
/usr/sbin/httpd2-prefork {
  ...
  ^phpsysinfo {
    #include <abstractions/bash>
    #include <abstractions/nameservice>

    /bin/basename          ixr,
    /bin/bash              ixr,
    /bin/df                ixr,
    /bin/grep              ixr,
    /bin/mount             Ux,
    /bin/sed               ixr,
    /dev/bus/usb/          r,
    /dev/bus/usb/**        r,
    /dev/null              w,
    /dev/tty               rw,
    /dev/urandom           r,
    /etc/SuSE-release      r,
    /etc/ld.so.cache       r,
    /etc/lsb-release       r,
    /etc/lsb-release.d/   r,
    /lib/ld-2.6.1.so       ixr,
    /proc/**               r,
    /sbin/lspci            ixr,
    /srv/www/htdocs/phpsysinfo/** r,
```

```

/sys/bus/pci/**                r,
/sys/bus/scsi/devices/         r,
/sys/devices/**                r,
/usr/bin/cut                    ixr,
/usr/bin/getopt                 ixr,
/usr/bin/head                   ixr,
/usr/bin/lsb_release           ixr,
/usr/bin/lsscsi                 ixr,
/usr/bin/tr                     ixr,
/usr/bin/who                    ixr,
/usr/lib/lib*so*               mr,
/usr/lib/locale/**             r,
/usr/sbin/lsusb                 ixr,
/usr/share/locale/**           r,
/usr/share/pci.ids              r,
/usr/share/usb.ids              r,
/var/log/apache2/access_log     w,
/var/run/utmp                   r,
}
}

```

- 3** Reload Novell AppArmor profiles by entering `rcapparmor restart` at a terminal window as `root`.
- 4** Restart Apache by entering `rcapache2 restart` at a terminal window as `root`.
- 5** Enter `http://hostname/phpsysinfo/` into a browser to receive the system information that `phpsysinfo` delivers.
- 6** Locate configuration errors by going to `/var/log/audit/audit.log` or running `dmesg` and looking for any rejections in the output.

# Managing Profiled Applications

After creating profiles and immunizing your applications, SUSE Linux Enterprise® becomes more efficient and better protected if you perform Novell® AppArmor profile maintenance, which involves analyzing log files and refining your profiles as well as backing up your set of profiles and keeping it up-to-date. You can deal with these issues before they become a problem by setting up event notification by e-mail, running periodic reports, updating profiles from system log entries by running the `aa-logprof` tool through YaST, and dealing with maintenance issues.

## 6.1 Monitoring Your Secured Applications

Applications that are confined by Novell AppArmor security profiles generate messages when applications execute in unexpected ways or outside of their specified profile. These messages can be monitored by event notification, periodic report generation, or integration into a third-party reporting mechanism.

For reporting and alerting, AppArmor uses a userspace daemon (`/usr/sbin/aa-eventd`). This daemon monitors log traffic, sends out notifications, and runs scheduled reports. It does not require any end user configuration and it is started automatically as part of the security event notification through the YaST AppArmor Control Panel or by the configuration of scheduled reports in the YaST AppArmor Reports module.

Apart from transparently enabling and disabling aa-eventd with the YaST modules, you can manually toggle its status with the `rcaaeventd` init script. The AppArmor event daemon is not required for proper functioning of the profiling process (such as enforcement or learning). It is just required for reporting.

Find more details on security event notification in Section 6.2, “Configuring Security Event Notification” (page 82) and on scheduled reports in Section 6.3, “Configuring Reports” (page 85).

## 6.2 Configuring Security Event Notification

Security event notification is a Novell AppArmor feature that informs you when systemic Novell AppArmor activity occurs. Activate it by selecting a notification frequency (receiving daily notification, for example). Enter an e-mail address, so you can be notified by e-mail when Novell AppArmor security events occur. Select one of the following notification types:

### Terse

Terse notification summarizes the total number of system events without providing details. For example:

```
jupiter.example.com has had 41 security events since Mon Sep 10 14:53:16
2007.
```

### Summary Notification

Summary notification displays the logged Novell AppArmor security events and lists the number of individual occurrences, including the date of the last occurrence. For example:

```
AppArmor: PERMITTING access to capability 'setgid' (httpd2-prefork(6347)
profile /usr/sbin/httpd2-prefork active /usr/sbin/httpd2-prefork) 2 times,
the latest at Sat Oct 9 16:05:54 2004.
```

### Verbose Notification

Verbose notification displays unmodified, logged Novell AppArmor security events. It tells you every time an event occurs and writes a new line in the verbose log.

These security events include the date and time the event occurred, when the application profile permits and rejects access, and the type of file permission access that is permitted or rejected. Verbose notification also reports several messages that the

aa-logprof tool (see Section “aa-logprof—Scanning the System Log” (page 54)) uses to interpret profiles. For example:

```
type=APPARMOR_DENIED msg=audit(1189428793.218:2880):
operation="file_permission" requested_mask="w" denied_mask="w"
name="/var/log/apache2/error_log" pid=22969
profile="/usr/sbin/httpd2-prefork"
```

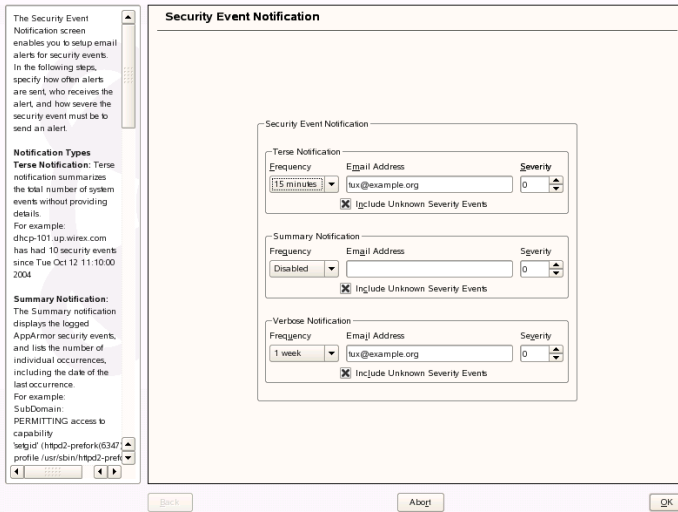
---

## NOTE

You must set up a mail server that can send outgoing mail using the SMTP protocol (for example, postfix or exim) for event notification to work.

---

- 1 In the *Enable Security Event Notification* section of the *AppArmor Configuration* window, click *Configure*.



- 2 In the *Security Event Notification* window, enable *Terse*, *Summary*, or *Verbose* event notification.

- 2a In each applicable notification type section, enter the e-mail addresses of those who should receive notification in the field provided. If notification is enabled, you must enter an e-mail address. Separate multiple e-mail addresses with commas.

**2b** For each notification type enabled, select the frequency of notification.

Select a notification frequency from the following options:

- Disabled
- 1 minute
- 5 minutes
- 10 minutes
- 15 minutes
- 30 minutes
- 1 hour
- 1 day
- 1 week

**2c** For each selected notification type, select the lowest severity level for which a notification should be sent. Security events are logged and the notifications are sent at the time indicated by the interval when events are equal to or greater than the selected severity level. If the interval is *1 day*, the notification is sent daily, if security events occur.

---

**NOTE: Severity Levels**

Novell AppArmor sends out event messages for things that are in the severity database and above the level selected. Severity levels are numbered 1 through 10, with 10 being the most severe security incident. The `/etc/severity.db` file defines the severity level of potential security events. The severity levels are determined by the importance of different security events, such as certain resources accessed or services denied.

---

**3** Click *OK*.

4 Click *Done* in the *Novell AppArmor Configuration* window.

5 Click *File > Quit* in the YaST Control Center.

After configuring security event notification, read the reports and determine whether events require follow up. Follow up may include the procedures outlined in Section 6.4, “Reacting to Security Event Rejections” (page 104).

## 6.3 Configuring Reports

Novell AppArmor's reporting feature adds flexibility by enhancing the way users can view security event data. The reporting tool performs the following:

- Creates on-demand reports
- Exports reports
- Schedules periodic reports for archiving
- E-mails periodic reports
- Filters report data by date
- Filters report data by other options, such as program name

Using reports, you can read important Novell AppArmor security events reported in the log files without manually sifting through the messages only useful to the `aa-logprof` tool. Narrow down the size of the report by filtering by date range or program name. You can also export an `html` or `csv` file.

The following are the three types of reports available in Novell AppArmor:

### Executive Security Summary

A combined report, consisting of one or more security incident reports from one or more machines. This report can provide a single view of security events on multiple machines. For more details, refer to Section “Executive Security Summary” (page 94).

## Application Audit Report

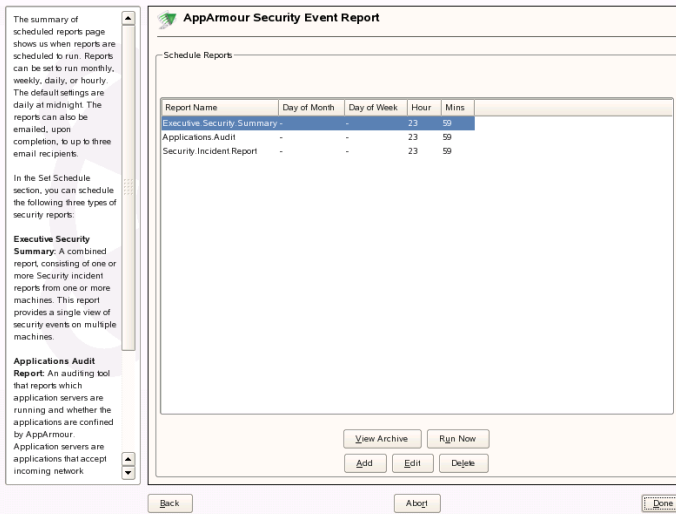
An auditing tool that reports which application servers are running and whether the applications are confined by AppArmor. Application servers are applications that accept incoming network connections. For more details, refer to Section “Application Audit Report” (page 91).

## Security Incident Report

A report that displays application security for a single host. It reports policy violations for locally confined applications during a specific time period. You can edit and customize this report or add new versions. For more details, refer to Section “Security Incident Report” (page 92).

To use the Novell AppArmor reporting features, proceed with the following steps:

- 1 Open *YaST* > *Novell AppArmor*.
- 2 In *Novell AppArmor*, click *AppArmor Reports*. The *AppArmor Security Event Reports* window appears. From the *Reports* window, select an option and proceed to the respective section for instructions:



### View Archive

Displays all reports that have been run and stored in `/var/log/apparmor/reports-archived/`. Select the report you want to see in detail and click

*View*. For *View Archive* instructions, proceed to Section 6.3.1, “Viewing Archived Reports” (page 87).

#### Run Now

Produces an instant version of the selected report type. If you select a security incident report, it can be further filtered in various ways. For *Run Now* instructions, proceed to Section 6.3.2, “Run Now: Running On-Demand Reports” (page 95).

#### Add

Creates a scheduled security incident report. For *Add* instructions, proceed to Section 6.3.3, “Adding New Reports” (page 98).

#### Edit

Edits a scheduled security incident report.

#### Delete

Deletes a scheduled security incident report. All stock or canned reports cannot be deleted.

#### Back

Returns you to the Novell AppArmor main screen.

#### Abort

Returns you to the Novell AppArmor main screen.

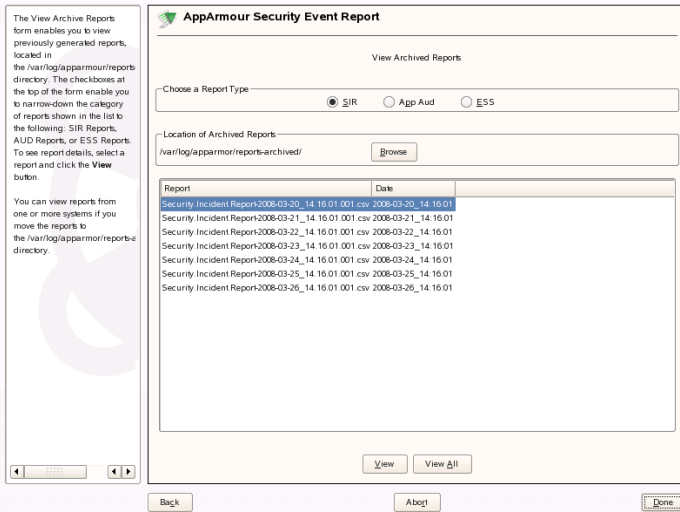
#### Next

Performs the same function as the *Run Now* button.

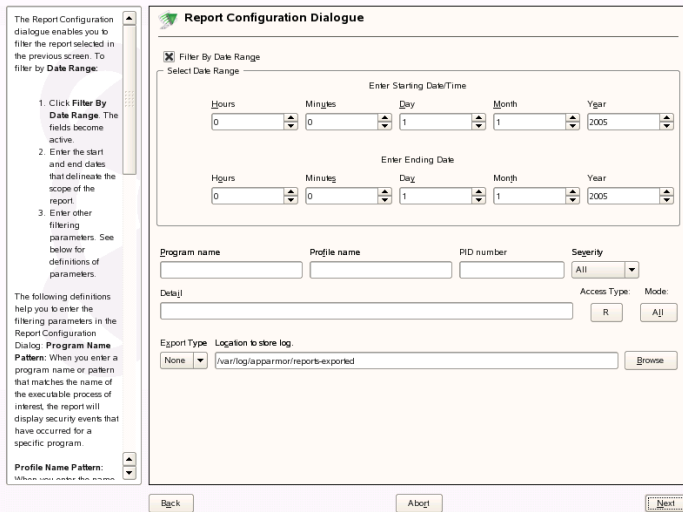
## 6.3.1 Viewing Archived Reports

*View Reports* enables you to specify the location of a collection of reports from one or more systems, including the ability to filter by date or names of programs accessed and display them all together in one report.

- 1 From the *AppArmor Security Event Report* window, select *View Archive*.



- 2 Select the report type to view. Toggle between the different types: *SIR* (Security Incident Report), *App Aud* (Application Audit), and *ESS* (Executive Security Summary).
- 3 You can alter the directory location of the archived reports in *Location of Archived Reports*. Select *Accept* to use the current directory or select *Browse* to find a new report location. The default directory is `/var/log/apparmor/reports-archived`.
- 4 To view all the reports in the archive, select *View All*. To view a specific report, select a report file listed in the *Report* field then select *View*.
- 5 For *Application Audit* and *Executive Security Summary* reports, proceed to Step 9 (page 90).
- 6 The *Report Configuration Dialog* opens for *Security Incident* reports.



7 The *Report Configuration* dialog enables you to filter the reports selected in the previous screen. Enter the desired filter details. The fields are:

### Date Range

To display reports for a certain time period, select *Filter By Date Range*. Enter the start and end dates that define the scope of the report.

### Program Name

When you enter a program name or pattern that matches the name of the binary executable of the program of interest, the report displays security events that have occurred for a specific program.

### Profile Name

When you enter the name of the profile, the report displays the security events that are generated for the specified profile. You can use this to see what is being confined by a specific profile.

### PID Number

*PID number* is a number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

### Severity

Select the lowest severity level for security events to include in the report. The selected severity level and above are then included in the reports.

### Detail

A source to which the profile has denied access. This includes capabilities and files. You can use this field to report the resources to which profiles prevent access.

### Access Type

The access type describes what is actually happening with the security event. The options are PERMITTING, REJECTING, or AUDITING.

### Mode

The *Mode* is the permission that the profile grants to the program or process to which it is applied. The options are `all` (all modes without filtering), `r` (read), `w` (write), `l` (link), `x` (execute), and `m` (mmap).

### Export Type

Enables you to export a CSV (comma separated values) or HTML file. The CSV file separates pieces of data in the log entries with commas using a standard data format for importing into table-oriented applications. You can enter a path for your exported report by typing the full path in the field provided.

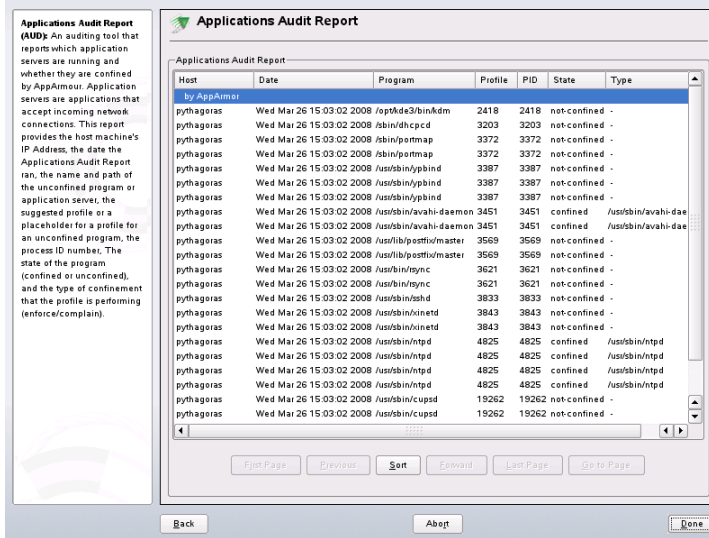
### Location to Store Log

Enables you to change the location at which to store the exported report. The default location is `/var/log/apparmor/reports-exported`. When you change this location, select *Accept*. Select *Browse* to browse the file system.

- 8** To see the report, filtered as desired, select *Next*. One of the three reports displays.
- 9** Refer the following sections for detailed information about each type of report.
  - For the application audit report, refer to Section “Application Audit Report” (page 91).
  - For the security incident report, refer to Section “Security Incident Report” (page 92).
  - For the executive summary report, refer to Section “Executive Security Summary” (page 94).

# Application Audit Report

An application audit report is an auditing tool that reports which application servers are running and whether they are confined by AppArmor.



The following fields are provided in an application audit report:

## Host

The machine protected by AppArmor for which the security events are reported.

## Date

The date during which security events occurred.

## Program

The name and path of the executing process.

## Profile

The absolute name of the security profile that is applied to the process.

## PID

A number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

### State

This field reveals whether the program listed in the program field is confined. If it is not confined, you might consider creating a profile for it.

### Type

This field reveals the type of confinement the security event represents. It says either complain or enforce. If the application is not confined (state), no type of confinement is reported.

## Security Incident Report

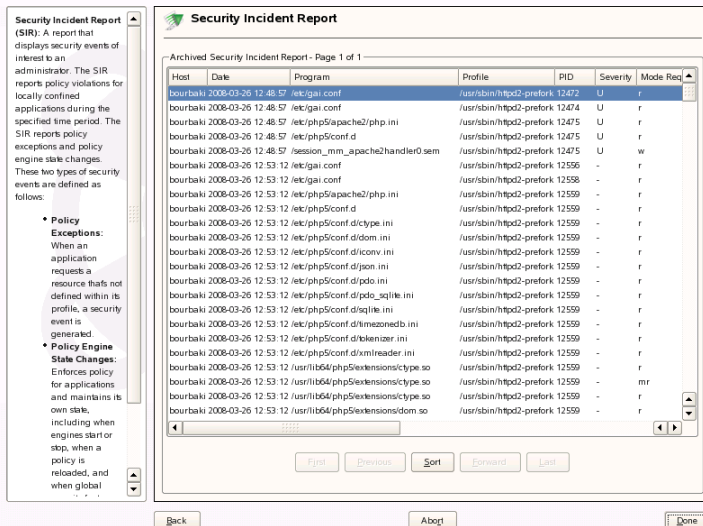
A security incident report displays security events of interest to an administrator. The SIR reports policy violations for locally confined applications during the specified time period. It also reports policy exceptions and policy engine state changes. These two types of security events are defined as follows:

### Policy Exceptions

When an application requests a resource that is not defined within its profile, a security event is triggered. A report is generated that displays security events of interest to an administrator. The SIR reports policy violations for locally confined applications during the specified time period. The SIR reports policy exceptions and policy engine state changes.

### Policy Engine State Changes

Enforces policy for applications and maintains its own state, including when engines start or stop, when a policy is reloaded, and when global security features are enabled or disabled.



The fields in the SIR report have the following meanings:

### Host

The machine protected by AppArmor for which the security events are reported.

### Date

The date during which security events occurred.

### Program

The name of the executing process.

### Profile

The absolute name of the security profile that is applied to the process.

### PID

A number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

### Severity

Severity levels of events are reported from the severity database. The severity database defines the importance of potential security events and numbers them 1 through 10, 10 being the most severe security incident. The severity levels are de-

terminated by the threat or importance of different security events, such as certain resources accessed or services denied.

## Mode

The mode is the permission that the profile grants to the program or process to which it is applied. The options are `r` (read), `w` (write), `l` (link), and `x` (execute).

## Detail

A source to which the profile has denied access. This includes capabilities and files. You can use this field to report the resources to which the profile prevents access.

## Access Type

The access type describes what is actually happening with the security event. The options are `PERMITTING`, `REJECTING`, or `AUDITING`.

# Executive Security Summary

A combined report consisting of one or more high-level reports from one or more machines. This report can provide a single view of security events on multiple machines if each machine's data is copied to the report archive directory, which is `/var/log/apparmor/reports-archived`. One line of the ESS report represents a range of SIR reports.

**Executive Security Summary**  
(ESS): A combined report, consisting of one or more high-level reports from one or more machines. This report can provide a single view of security events on multiple machines if each machine's data is copied to the reports archive directory, which is `/var/log/apparmor/reports-archived`. This report provides the host machine's IP address, the start and end dates of the polled events, total number of rejects, total number of events, average of severity levels reported, and the highest severity level reported. One line of the ESS report represents a range of SIR reports.

**Executive Security Summary Report**

Host	Start Date	End Date	Num Rejects	Num Events	Ave. Sev	High Sev
bourbaki.suse.de	2005-01-01 09:00:01	2008-03-26 14:16:02	5	1457	-1.00	-1

First Page Previous Sort Forward Last Page Go to Page

Back Abort Done

The following fields are provided in an executive security summary:

**Host**

The machine protected by AppArmor for which the security events are reported.

**Start Date**

The first date in a range of dates during which security events are reported.

**End Date**

The last date in a range of dates during which security events are reported.

**Num Rejects**

In the date range given, the total number of security events that are rejected access attempts.

**Num Events**

In the date range given, the total number of security events.

**Ave. Sev**

This is the average of the severity levels reported in the date range given. Unknown severities are disregarded in this figure.

**High Sev**

This is the severity of the highest severity event reported in the date range given.

## 6.3.2 Run Now: Running On-Demand Reports

The *Run Now* report feature enables you to instantly extract report information from the Novell AppArmor event logs without waiting for scheduled events. If you need help navigating to the main report screen, see Section 6.3, “Configuring Reports” (page 85). Perform the following steps to run a report from the list of reports:

- 1 Select the report to run instantly from the list of reports in the *Schedule Reports* window.
- 2 Select *Run Now* or *Next*. The next screen depends on which report you selected in the previous step. As an example, select a security incident report.

### 3 The *Report Configuration Dialog* opens for security incident reports.

The Report Configuration dialog enables you to filter the report selected in the previous screen. To filter by Date Range:

1. Click **Filter By Date Range**. The fields become active.
2. Enter the start and end dates that delineate the scope of the report.
3. Enter other filtering parameters. See below for definitions of parameters.

The following definitions help you to enter the filtering parameters in the Report Configuration Dialog:

- Program Name Pattern:** When you enter a program name or pattern that matches the name of the executable process of interest, the report will display security events that have occurred for a specific program.
- Profile Name Pattern:** When you enter the name of the profile, the report displays the security events that are generated for the specified profile. You can use this to see what is confined by a specific profile.

**Report Configuration Dialog**

Filter By Date Range

Select Date Range

Enter Starting Date/Time

Hours: 0 Minutes: 0 Day: 1 Month: 1 Year: 2005

Enter Ending Date

Hours: 0 Minutes: 0 Day: 1 Month: 1 Year: 2005

Program name: Profile name: PID number: Severity: All

Detail: Access Type: Mode: R All

Export Type: Logon to store log. None /var/log/apparmor/reports-exported Browse

Back About Next

### 4 The *Report Configuration Dialog* enables you to filter the reports selected in the previous screen. Enter the desired filter details. The following filter options are available:

#### Date Range

To limit reports to a certain time period, select *Filter By Date Range*. Enter the start and end dates that determine the scope of the report.

#### Program Name

When you enter a program name or pattern that matches the name of the binary executable for the program of interest, the report displays security events that have occurred for the specified program only.

#### Profile Name

When you enter the name of the profile, the report displays the security events that are generated for the specified profile. You can use this to see what is confined by a specific profile.

#### PID Number

A number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

### Severity

Select the lowest severity level for security events to include in the report. The selected severity level and above are included in the reports.

### Detail

A source to which the profile has denied access. This includes capabilities and files. You can use this field to report the resources to which profiles prevent access.

### Access Type

The access type describes what is actually happening with the security event. The options are `PERMITTING`, `REJECTING`, or `AUDITING`.

### Mode

The mode is the permission that the profile grants to the program or process to which it is applied. The options are `r` (read), `w` (write), `l` (link), and `x` (execute).

### Export Type

Enables you to export a CSV (comma separated values) or HTML file. The CSV file separates pieces of data in the log entries with commas using a standard data format for importing into table-oriented applications. Enter a path for your exported report by typing in the full path in the field provided.

### Location to Store Log

Enables you to change the location that the exported report is stored. The default location is `/var/log/apparmor/reports-exported`. When you change this location, select *Accept*. Select *Browse* to browse the file system.

**5** To see the report, filtered as desired, select *Next*. One of the three reports displays.

Refer the following sections for detailed information about each type of report.

- For the application audit report, refer to Section “Application Audit Report” (page 91).
- For the security incident report, refer to Section “Security Incident Report” (page 92).
- For the executive summary report, refer to Section “Executive Security Summary” (page 94).

## 6.3.3 Adding New Reports

Adding new reports enables you to create a scheduled security incident report that displays Novell AppArmor security events according to your preset filters. When a report is set up in *Schedule Reports*, it periodically launches a report of Novell AppArmor security events that have occurred on the system.

You can configure a daily, weekly, monthly, or hourly report to run for a specified period. You can set the report to display rejections for certain severity levels or to filter by program name, profile name, severity level, or denied resources. This report can be exported to an HTML (Hypertext Markup Language) or CSV (Comma Separated Values) file format.

---

### NOTE

Return to the beginning of this section if you need help navigating to the main report screen (see Section 6.3, “Configuring Reports” (page 85)).

---

To add a new scheduled security incident report, proceed as follows:

- 1 Click *Add* to create a new security incident report. The first page of *Add Scheduled SIR* opens.

The screenshot shows a web form titled "Add Scheduled SIR". It contains the following fields and controls:

- Report Name:** A text input field containing "My Report".
- Scheduling:** Four dropdown menus: "Day of Month" (All), "Day of Week" (Sun), "Hour" (0), and "Minute" (5).
- Email Targets:** Three text input fields labeled "Email Target 1", "Email Target 2", and "Email Target 3". The first field contains "tux@example.com".
- Export Settings:** A dropdown menu for "Export Type" (None) and a text input field for "Location to store log" containing "/var/log/apparmor/reports-exported". A "Browse" button is next to the location field.
- Navigation:** "Cancel" and "Next" buttons at the bottom.

- 2 Fill in the fields with the following filtering information, as necessary:

#### Report Name

Specify the name of the report. Use names that easily distinguish different reports.

#### Day of Month

Select any day of the month to activate monthly filtering in reports. If you select ALL, monthly filtering is not performed.

#### Day of Week

Select the day of the week on which to schedule weekly reports, if desired. If you select ALL, weekly filtering is not performed. If monthly reporting is selected, this field defaults to ALL.

#### Hour and Minute

Select the time. This specifies the hour and minute that you would like the reports to run. If you do not change the time, selected reports runs at midnight. If neither month nor day of week are selected, the report runs daily at the specified time.

#### E-Mail Target

You have the ability to send the scheduled security incident report via e-mail to up to three recipients. Just enter the e-mail addresses for those who require the security incident information.

#### Export Type

This option enables you to export a CSV (comma separated values) or HTML file. The CSV file separates pieces of data in the log entries with commas using a standard data format for importing into table-oriented applications. Enter a path for your exported report by typing in the full path in the field provided.

#### Location to Store Log

Enables you to change the location that the exported report is stored. The default location is `/var/log/apparmor/reports-exported`. When you change this location, select *Accept*. Select *Browse* to browse the file system.

- 3 Click *Next* to proceed to the second page of *Add Scheduled SIR*.

The image shows a configuration window with the following elements:

- Program name:** A text input field containing "httpd2-prefork".
- Profile name:** An empty text input field.
- PID number:** An empty text input field.
- Detail:** An empty text input field.
- Severity:** A dropdown menu currently showing "All".
- Access Type:** A button labeled "R".
- Mode:** A button labeled "All".
- Buttons:** "Cancel" and "Save" buttons at the bottom.

**4** Fill in the fields with the following filtering information, as necessary:

**Program Name**

You can specify a program name or pattern that matches the name of the binary executable for the program of interest. The report displays security events that have occurred for the specified program only.

**Profile Name**

You can specify the name of the profile for which the report should display security events. You can use this to see what is being confined by a specific profile.

**PID Number**

A number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

**Detail**

A source to which the profile has denied access. This includes capabilities and files. You can use this field to create a report of resources to which profiles prevent access.

**Severity**

Select the lowest severity level of security events to include in the report. The selected severity level and above are included in the reports.

**Access Type**

The access type describes what is actually happening with the security event. The options are PERMITTING, REJECTING, or AUDITING.

## Mode

The mode is the permission that the profile grants to the program or process to which it is applied. The options are `r` (read), `w` (write), `l` (link), and `x` (execute).

- 5 Click *Save* to save this report. Novell AppArmor returns to the *Scheduled Reports* main window where the newly scheduled report appears in the list of reports.

## 6.3.4 Editing Reports

From the AppArmor *Reports* screen, you can select and edit a report. The three pre-configured reports (*stock reports*) cannot be edited or deleted.

---

### NOTE

Return to the beginning of this section if you need help navigating to the main report screen (see Section 6.3, “Configuring Reports” (page 85)).

---

Perform the following steps to modify a report from the list of reports:

- 1 From the list of reports in the *Schedule Reports* window, select the report to edit. This example assumes that you have selected a security incident report.
- 2 Click *Edit* to edit the security incident report. The first page of the *Edit Scheduled SIR* displays.

Dialog box titled "Edit Report Schedule for Executive Security Summary".

Day of Month: All | Day of Week: All | Hour: 23 | Minute: 59

Email Target 1: | Email Target 2: | Email Target 3: |

Export Type: Both | Location to store log: /var/log/apparmor/reports-archived | Browse

Buttons: Cancel, Save

- 3 Modify the following filtering information, as necessary:

### Day of Month

Select any day of the month to activate monthly filtering in reports. If you select `All`, monthly filtering is not performed.

### Day of Week

Select the day of the week on which to schedule the weekly reports. If you select `All`, weekly filtering is not performed. If monthly reporting is selected, this defaults to `All`.

### Hour and Minute

Select the time. This specifies the hour and minute that you would like the reports to run. If you do not change the time, the selected report runs at midnight. If neither the day of the month nor day of the week is selected, the report runs daily at the specified time.

### E-Mail Target

You have the ability to send the scheduled security incident report via e-mail to up to three recipients. Just enter the e-mail addresses for those who require the security incident information.

### Export Type

This option enables you to export a CSV (comma separated values) or HTML file. The CSV file separates pieces of data in the log entries with commas using a standard data format for importing into table-oriented applications. Enter a path for your exported report by typing the full path in the field provided.

### Location to Store Log

Enables you to change the location where the exported report is stored. The default location is `/var/log/apparmor/reports-exported`. When you change this location, select *Accept*. Select *Browse* to browse the file system.

- 4 Click *Next* to proceed to the next *Edit Scheduled SIR* page. The second page of *Edit Scheduled Reports* opens.

The image shows a configuration dialog box with the following elements:

- Program name:** A text input field containing "httpd2-prefork".
- Profile name:** An empty text input field.
- PID number:** An empty text input field.
- Detail:** An empty text input field.
- Severity:** A dropdown menu currently set to "All".
- Access Type:** A button labeled "R".
- Mode:** A button labeled "All".
- Buttons:** "Cancel" and "Save" buttons at the bottom.

## 5 Modify the fields with the following filtering information, as necessary:

### Program Name

You can specify a program name or pattern that matches the name of the binary executable for the program of interest. The report displays security events that have occurred for the specified program only.

### Profile Name

You can specify the name of the profile for which to display security events. You can use this to see what is being confined by a specific profile.

### PID Number

Process ID number is a number that uniquely identifies one specific process or running program (this number is valid only during the lifetime of that process).

### Detail

A source to which the profile has denied access. This includes capabilities and files. You can use this field to create a report of resources to which profiles prevent access.

### Severity

Select the lowest severity level for security events to include in the report. The selected severity level and above are included in the reports.

### Access Type

The access type describes what is actually happening with the security event. The options are PERMITTING, REJECTING, or AUDITING.

## Mode

The mode is the permission that the profile grants to the program or process to which it is applied. The options are `r` (read), `w` (write), `l` (link), and `x` (execute).

- 6 Select *Save* to save the changes to this report. Novell AppArmor returns to the *Scheduled Reports* main window where the scheduled report appears in the list of reports.

## 6.3.5 Deleting Reports

*Delete a Report* enables you to permanently remove a report from the list of Novell AppArmor scheduled reports. To delete a report, follow these instructions:

- 1 To remove a report from the list of reports, highlight the report and click *Delete*.
- 2 From the confirmation pop-up, select *Cancel* if you do not want to delete the selected report. If you are sure you want to remove the report permanently from the list of reports, select *Delete*.

## 6.4 Reacting to Security Event Rejections

When you receive a security event rejection, examine the access violation and determine if that event indicated a threat or was part of normal application behavior. Application-specific knowledge is required to make the determination. If the rejected action is part of normal application behavior, run `aa-logprof` at the command line or the *Update Profile Wizard* in Novell AppArmor to update your profile.

If the rejected action is not part of normal application behavior, this access should be considered a possible intrusion attempt (that was prevented) and this notification should be passed to the person responsible for security within your organization.

## 6.5 Maintaining Your Security Profiles

In a production environment, you should plan on maintaining profiles for all of the deployed applications. The security policies are an integral part of your deployment. You should plan on taking steps to back up and restore security policy files, plan for software changes, and allow any needed modification of security policies that your environment dictates.

### 6.5.1 Backing Up Your Security Profiles

Because you take the time to make profiles, it makes sense to back them up. Backing up profiles might save you from having to reprofile all your programs after a disk crash. Also, if profiles are changed, you can easily restore previous settings by using the backed up files.

Back up profiles by copying the profile files to a specified directory.

- 1 You should first archive the files into one file. To do this, open a terminal window and enter the following as `root`:

```
tar zcJpf profiles.tgz /etc/apparmor.d
```

The simplest method to ensure that your security policy files are regularly backed up is to include the directory `/etc/apparmor.d` in the list of directories that your backup system archives.

- 2 You can also use `scp` or a file manager like Konqueror or Nautilus to store the files on some kind of storage media, the network, or another computer.

### 6.5.2 Changing Your Security Profiles

Maintenance of security profiles includes changing them if you decide that your system requires more or less security for its applications. To change your profiles in Novell AppArmor, refer to Section 3.3, “Editing Profiles” (page 26).

## 6.5.3 Introducing New Software into Your Environment

When you add a new application version or patch to your system, you should always update the profile to fit your needs. You have several options that depend on your company's software deployment strategy. You can deploy your patches and upgrades into a test or production environment. The following explains how to do this with each method.

If you intend to deploy a patch or upgrade in a test environment, the best method for updating your profiles is one of the following:

- Run the profiling wizard by selecting *Add Profile Wizard* in YaST. This creates a new profile for the added or patched application. For step-by-step instructions, refer to Section 3.1, “Adding a Profile Using the Wizard” (page 18).
- Run `aa-genprof` by typing `aa-genprof` in a terminal while logged in as `root`. For detailed instructions, refer to Section “aa-genprof—Generating Profiles” (page 47).

If you intend to deploy a patch or upgrade directly into a production environment, the best method for updating your profiles is one of the following:

- Monitor the system frequently to determine if any new rejections should be added to the profile and update as needed using `aa-logprof`. For detailed instructions, refer to Section “aa-logprof—Scanning the System Log” (page 54).
- Run the YaST *Update Profile Wizard* to learn the new behavior (high security risk as all accesses are allowed and logged, not rejected). For step-by-step instructions, refer to Section 3.5, “Updating Profiles from Log Entries” (page 32).

# Support

This chapter outlines maintenance-related tasks. Learn how to update Novell® AppArmor and get a list of available man pages providing basic help for using the command line tools provided by Novell AppArmor. Use the troubleshooting section to learn about some common problems encountered with Novell AppArmor and their solutions. Report defects or enhancement requests for Novell AppArmor following the instructions in this chapter.

## 7.1 Updating Novell AppArmor Online

Updates for Novell AppArmor packages are provided in the same way as any other update for SUSE Linux Enterprise. Retrieve and apply them exactly like for any other package that ships as part of SUSE Linux Enterprise.

## 7.2 Using the Man Pages

There are man pages available for your use. In a terminal, enter `man apparmor` to open the apparmor man page. Man pages are distributed in sections numbered 1 through 8. Each section is specific to a category of documentation:

**Table 7.1** *Man Pages: Sections and Categories*

---

<b>Section</b>	<b>Category</b>
1	User commands
2	System calls
3	Library functions
4	Device driver information
5	Configuration file formats
6	Games
7	High level concepts
8	Administrator commands

---

The section numbers are used to distinguish man pages from each other. For example, `exit(2)` describes the `exit` system call, while `exit(3)` describes the `exit` C library function.

The Novell AppArmor man pages are:

- `unconfined(8)`
- `autodep(1)`
- `complain(1)`
- `enforce(1)`
- `genprof(1)`
- `logprof(1)`
- `change_hat(2)`

- `logprof.conf(5)`
- `apparmor.conf(5)`
- `apparmor.d(5)`
- `apparmor.vim(5)`
- `apparmor(7)`
- `apparmor_parser(8)`

## 7.3 For More Information

Find more information about the AppArmor product on the Novell AppArmor product page at Novell: <http://www.novell.com/products/apparmor/>. Find the product documentation for Novell AppArmor, including this document, at <http://www.novell.com/documentation/apparmor/> or in the installed system in `/usr/share/doc/manual`.

There are specific mailing lists for AppArmor that users can post to or join to communicate with developers.

`apparmor-general@forge.novell.com` [ <mailto:apparmor-general@forge.novell.com> ]

This is a mailing list for end users of AppArmor. It is a good place for questions about how to use AppArmor to protect your applications.

`apparmor-dev@forge.novell.com` [ <mailto:apparmor-dev@forge.novell.com> ]

This is a developer mailing list for AppArmor developers and community members. This list is for questions about development of core AppArmor features—the kernel module and the profiling tools. If you are interested in reviewing the code for AppArmor and contributing reviews or patches, this would be the list for you.

`apparmor-announce@forge.novell.com` [ <mailto:apparmor-announce@forge.novell.com> ]

This is a low traffic list announcing the availability of new releases or features.

## 7.4 Troubleshooting

This section lists the most common problems and error messages that may occur using Novell AppArmor.

### 7.4.1 How to React to odd Application Behavior?

If you notice odd application behavior or any other type of application problem, you should first check the reject messages in the log files to see if AppArmor is too closely constricting your application. To check reject messages, start *YaST* > *Novell AppArmor* and go to *AppArmor Reports*. Select *View Archive* and *App Aud* for the application audit report. You can filter dates and times to narrow down the specific periods when the unexpected application behavior occurred.

If you detect reject messages that indicate that your application or service is too closely restricted by AppArmor, update your profile to properly handle your use case of the application. Do this with the *Update Profile Profile Wizard* in YaST, as described in Section 3.5, “Updating Profiles from Log Entries” (page 32).

If you decide to run your application or service without AppArmor protection, remove the application's profile from `/etc/apparmor.d` or move it to another location.

### 7.4.2 How to Resolve Issues with Apache?

Apache is not starting properly or it is not serving Web pages and you just installed a new module or made a configuration change. When you install additional Apache modules (like `apache2-mod_apparmor`) or make configuration changes to Apache, you should profile Apache again to catch any additional rules that need to be added to the profile.

## 7.4.3 Why are the Reports not Sent by E-Mail?

When the reporting feature generates an HTML or CSV file that exceeds the default size, the file is not sent. Mail servers have a default, hard limit for e-mail size. This limitation can impede AppArmor's ability to send e-mails that are generated for reporting purposes. If your mail is not arriving, this could be why. Consider the mail size limits and check the archives if e-mails have not been received.

## 7.4.4 How to Exclude Certain Profiles from the List of Profiles Used?

AppArmor always loads and applies all profiles that are available in its profile directory (`/etc/apparmor.d/`). If you decide not to apply a profile to a certain application, delete the appropriate profile or move it to another location where AppArmor would not check for it.

## 7.4.5 Can I Manage Profiles for Applications not Installed on my System?

Managing profiles with AppArmor requires you to have access to a the system's log the application is running on. So you do not need to run the application on your profile build host as long as you have access to the machine that runs the application. You can run the application on one system, transfer the logs (`/var/log/audit.log` or, if `audit` is not installed, `/var/log/messages`) to your profile build host and run `aa-logprof -f path_to_logfile`.

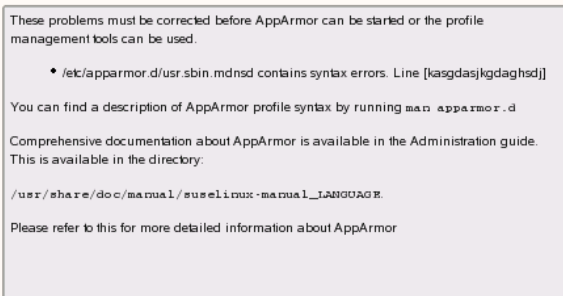
## 7.4.6 How to Spot and fix AppArmor Syntax Errors?

Manually editing Novell AppArmor profiles can introduce syntax errors. If you attempt to start or restart AppArmor with syntax errors in your profiles, error results are shown. This example shows the syntax of the entire parser error.

```
localhost:~ # rcapparmor start
Loading AppArmor profiles
AppArmor parser error, line 2: Found unexpected character: 'h'
Profile /etc/apparmor.d/usr.sbin.squid failed to load
failed
```

Using the AppArmor YaST tools, a graphical error message indicates which profile contained the error and requests you to fix it.

#### Errors found in AppArmor profiles



To fix a syntax error, log in to a terminal window as `root`, open the profile, and correct the syntax. Reload the profile set with `rcapparmor reload`.

## 7.5 Reporting Bugs for AppArmor

The developers of AppArmor are eager to deliver products of the highest quality. Your feedback and your bug reports help us keep the quality high. Whenever you encounter a bug in AppArmor, file a bug report against this product:

- 1 Use your Web browser to go to <https://bugzilla.novell.com/index.cgi>.
- 2 Enter the account data of your Novell account and click *Login*

*or*

Create a new Novell account as follows:

**2a** Click *Create New Account* on the *Login to Continue* page.

**2b** Provide a username and password and additional address data and click *Create Login* to immediately proceed with the login creation.

*or*

Provide data on which other Novell accounts you maintain to sync all these to one account.

- 3** Check whether a problem similar to yours has already been reported by clicking *Search Reports*. Use a quick search against a given product and keyword or use the *Advanced Search*.
- 4** If your problem has already been reported, check this bug report and add extra information to it, if necessary.
- 5** If your problem has not been reported yet, select *New* from the top navigation bar and proceed to the *Enter Bug* page.
- 6** Select the product against which to file the bug. In your case, this would be your product's release. Click *Submit*.
- 7** Select the product version, component (AppArmor in this case), hardware platform, and severity.
- 8** Enter a brief headline describing your problem and add a more elaborate description including log files. You may create attachments to your bug report for screen shots, log files, or test cases.
- 9** Click *Submit* after you have entered all the details to send your report to the developers.



# Background Information on AppArmor Profiling



For more information about the science and security of Novell AppArmor, refer to the following papers:

*SubDomain: Parsimonious Server Security* by Crispin Cowan, Steve Beattie, Greg Kroah-Hartman, Calton Pu, Perry Wagle, and Virgil Gligor

Describes the initial design and implementation of Novell AppArmor. Published in the proceedings of the USENIX LISA Conference, December 2000, New Orleans, LA. This paper is now out of date, describing syntax and features that are different from the current Novell AppArmor product. This paper should be used only for scientific background and not for technical documentation.

*Defcon Capture the Flag: Defending Vulnerable Code from Intense Attack* by Crispin Cowan, Seth Arnold, Steve Beattie, Chris Wright, and John Viega

A good guide to strategic and tactical use of Novell AppArmor to solve severe security problems in a very short period of time. Published in the Proceedings of the DARPA Information Survivability Conference and Expo (DISCEX III), April 2003, Washington, DC.

*AppArmor for Geeks* by Seth Arnold

This document tries to convey a better understanding of the technical details of AppArmor. It is available at [http://en.opensuse.org/SDB:AppArmor\\_geeks](http://en.opensuse.org/SDB:AppArmor_geeks).



# Glossary

## Apache

Apache is a freely available UNIX-based Web server. It is currently the most commonly used Web server on the Internet. Find more information about Apache at the Apache Web site at <http://www.apache.org>.

## application firewalling

Novell AppArmor contains applications and limits the actions they are permitted to take. It uses privilege confinement to prevent attackers from using malicious programs on the protected server and even using trusted applications in unintended ways.

## attack signature

Pattern in system or network activity that signals a possible virus or hacker attack. Intrusion detection systems might use attack signatures to distinguish between legitimate and potentially malicious activity.

By not relying on attack signatures, Novell AppArmor provides "proactive" instead of "reactive" defense from attacks. This is better because there is no window of vulnerability where the attack signature must be defined for Novell AppArmor as it does for products using attack signatures to secure their networks.

## GUI

Graphical user interface. Refers to a software front-end meant to provide an attractive and easy-to-use interface between a computer user and application. Its elements include such things as windows, icons, buttons, cursors, and scroll bars.

## globbing

Filename substitution.

## HIP

Host intrusion prevention. Works with the operating system kernel to block abnormal application behavior in the expectation that the abnormal behavior represents an unknown attack. Blocks malicious packets on the host at the network level before they can "hurt" the application they target.

### mandatory access control

A means of restricting access to objects that is based on fixed security attributes assigned to users, files, and other objects. The controls are mandatory in the sense that they cannot be modified by users or their programs.

### profile foundation classes

Profile building blocks needed for common application activities, such as DNS lookup and user authentication.

### RPM

The RPM Package Manager. An open packaging system available for anyone to use. It works on Red Hat Linux, SUSE Linux Enterprise, and other Linux and UNIX systems. It is capable of installing, uninstalling, verifying, querying, and updating computer software packages. See <http://www.rpm.org/> for more information.

### SSH

Secure Shell. A service that allows you to access your server from a remote computer and issue text commands through a secure connection.

### streamlined access control

Novell AppArmor provides streamlined access control for network services by specifying which files each program is allowed to read, write, and execute. This ensures that each program does what it is supposed to do and nothing else.

### URI

Universal resource identifier. The generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI.

### URL

Uniform Resource Locator. The global address of documents and other resources on the World Wide Web.

The first part of the address indicates what protocol to use and the second part specifies the IP address or the domain name where the resource is located.

For example, in `http://www.novell.com`, `http` is the protocol to use.

### vulnerabilities

An aspect of a system or network that leaves it open to attack. Characteristics of computer systems that allow an individual to keep it from correctly operating or

that allows unauthorized users to take control of the system. Design, administrative, or implementation weaknesses or flaws in hardware, firmware, or software. If exploited, a vulnerability could lead to an unacceptable impact in the form of unauthorized access to information or disruption of critical processing.

