

Securing Linux Systems with AppArmor

Crispin Cowan, PhD

Director of Software Engineering
Security Architect, SUSE Linux

February 18, 2007



Novell.[®]

Agenda



Overview

A Closer Look at AppArmor

Deployment Scenarios

Demonstration of AppArmor

Competitive Positioning

AppArmor Futures

Overview of AppArmor

AppArmor intrusion prevention

- Creates “firewall” around applications
- Protects even against **unknown** application vulnerabilities
- No security expertise required
- Comprehensive wizard-based tool set, integrated in YaST
- Default profiles for standard applications

A closer look at AppArmor

Security Model

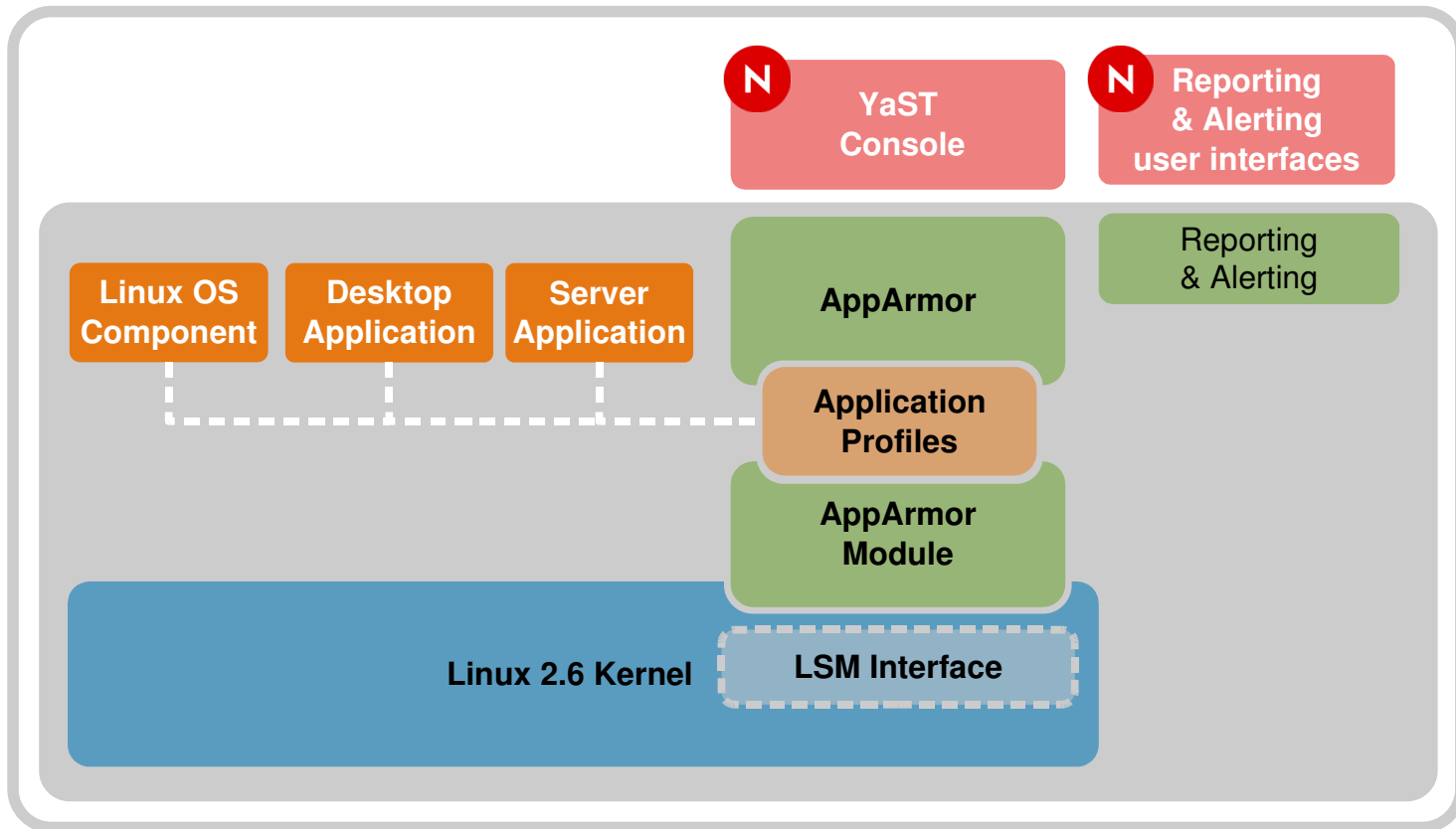
- Proactive “whitelist” approach, **no** attack signature database
- Profiles grant access to the **minimal** list of files/directories and POSIX capabilities required by the application
- Complete kernel-level mediation through Linux Security Module

Automated workflow for security profile generation

- Autoscans for open network ports
- Find applications listening to ports and check for existing profile
- Auto-generate profile template based on static analysis
- Auto learn mode: Automatically expands profile while running the application through normal operation
- Interactive optimizer assists in simplifying the profiles with regular expressions and foundation classes

AppArmor A Closer Look

AppArmor Architecture



Critical Issue #1: Complete Mediation

Must not be possible to bypass HIPS system

- Must be in the kernel

AppArmor uses LSM interface in 2.6 kernel

- LSM (Linux Security Module) provides in-kernel mediation without having to maintain a patched kernel
- Provides an open standard API for access control module
- Precise information on application behavior, accuracy, performance
- Provides highest quality non-bypassable mediation

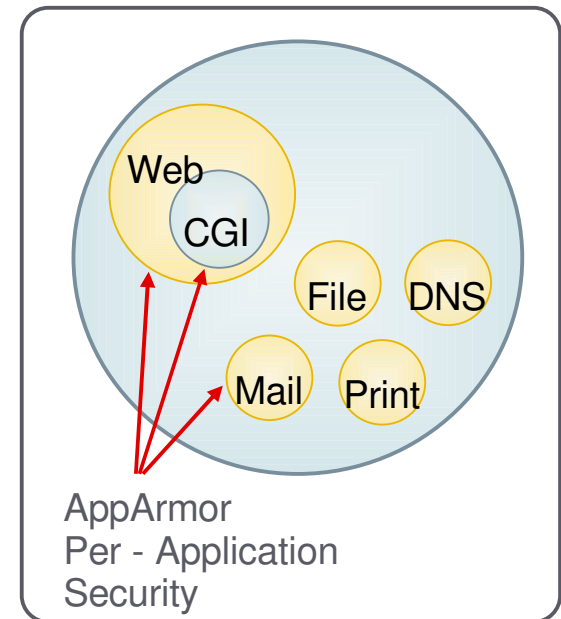
Critical Issue #2: Security Model

Misuse prevention vs. anomaly prevention

- Misuse prevention easier to manage
- Anomaly prevention much more secure, traditionally **hard** to use

AppArmor is easy anomaly prevention for application security

- Focus on *application* security
- Name-based access control for ease of understanding policy
- Hybrid white list/black list
 - White list *within* an application profile
 - Black list system-wide



AppArmor Security Profile

Whenever a protected program runs regardless of UID, AppArmor controls:

- The POSIX capabilities it can have (even if it is running as root)
- The directories/files it can read/write/execute

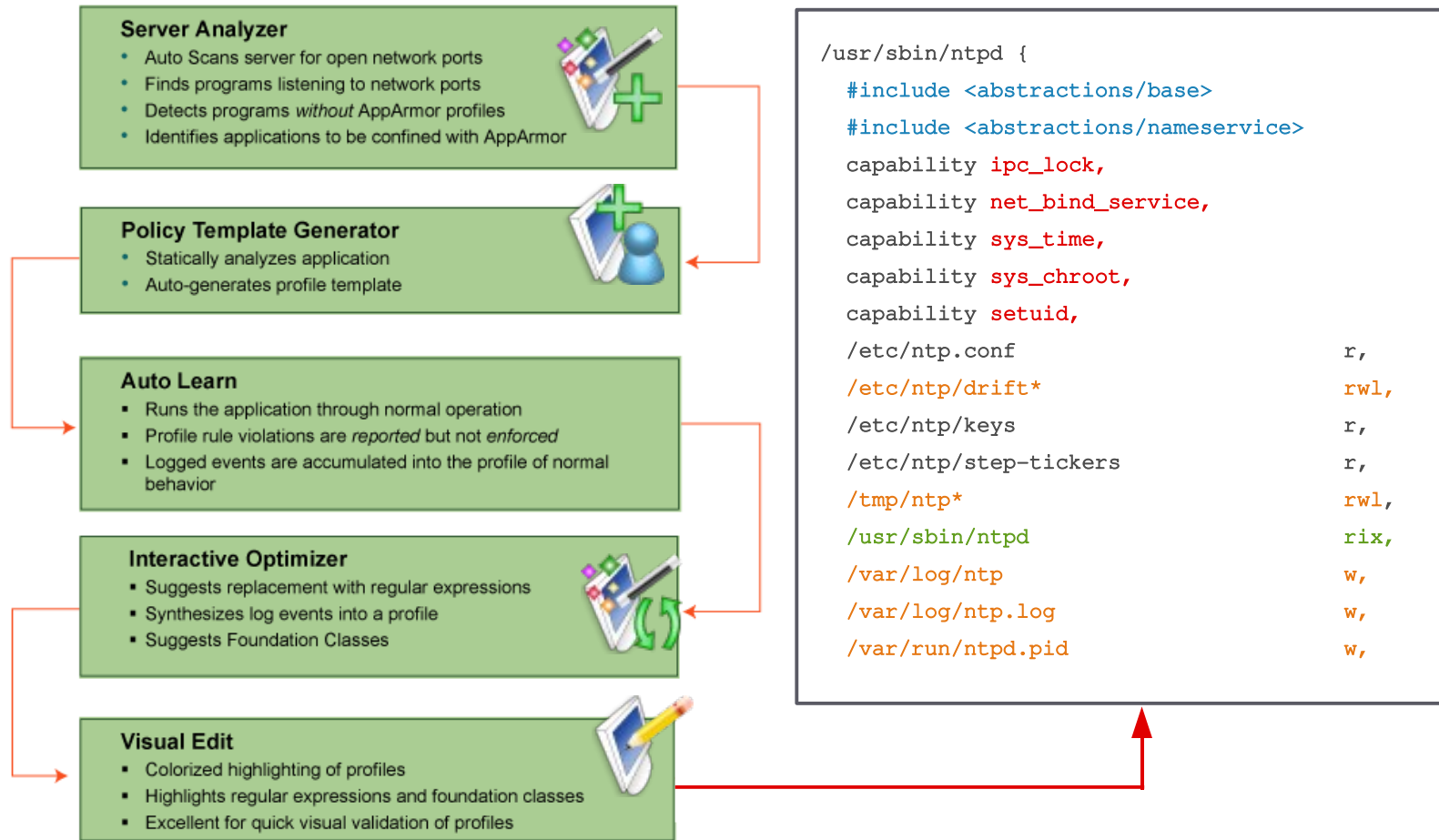
```
/usr/sbin/ntpd {
  #include <abstractions/base>
  #include <abstractions/nameservice>

  capability ipc_lock,
  capability net_bind_service,
  capability sys_time,
  capability sys_chroot,
  capability setuid,

  /etc/ntp.conf                r,
  /etc/ntp/drift*              rwl,
  /etc/ntp/keys                r,
  /etc/ntp/step-tickers        r,
  /tmp/ntp*                    rwl,
  /usr/sbin/ntpd               rix,
  /var/log/ntp                 w,
  /var/log/ntp.log             w,
  /var/run/ntpd.pid            w,
  /var/lib/ntp/drift           rwl,
  /var/lib/ntp/drift.TEMP      rwl,
  /var/lib/ntp/var/run/ntp/ntpd.pid w,
  /var/lib/ntp/drift/ntp.drift r,
  /drift/ntp.drift.TEMP       rwl,
  /drift/ntp.drift            rwl,
}
```

Example security profile for ntpd

Automated Workflow



Native Unix Syntax, Semantics

AppArmor access controls reflect classic Unix permission patterns

- Complements Unix permissions rather than overlaying a new paradigm

Regular expressions in AppArmor rules

- `/dev/{,u}random` matches `/dev/random` and `/dev/urandom`
- `/lib/ld-*.so*` matches most of the libraries in `/lib`
- `/home/*/plan` matches everyone's `.plan` file
- `/home/*/public_html/**` matches everyone's public HTML directory tree

Profile Building Blocks

A set of “foundation class” rules that can be #include'd in your profiles

- base: needed by nearly all programs
- authentication: program will authenticate users
- console: program interacts with TTY consoles
- kerberos: uses Kerberos cryptography
- nameservice: program needs to look up domain names
- wutmp: program updates user login logs

Includes Default Set of Policies

/etc/apparmor.d

(default loaded)

- netstat
- ping
- klogd
- syslog
- ldd
- squid
- traceroute
- identd
- mdnsd
- named
- nscd
- ntpd

/etc/apparmor/extras

(not loaded, but available)

- firefox
- opera
- evolution
- gaim
- realplay
- postfix
- acroread
- mysqld
- ethereal
- postfix
- sendmail
- many more...

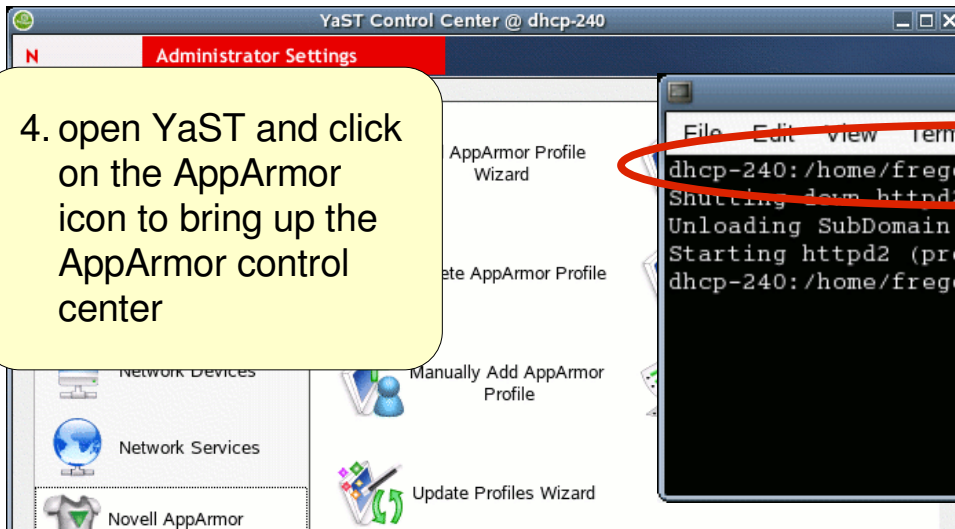
AppArmor Demo

Apache Profile – YaST Toolset

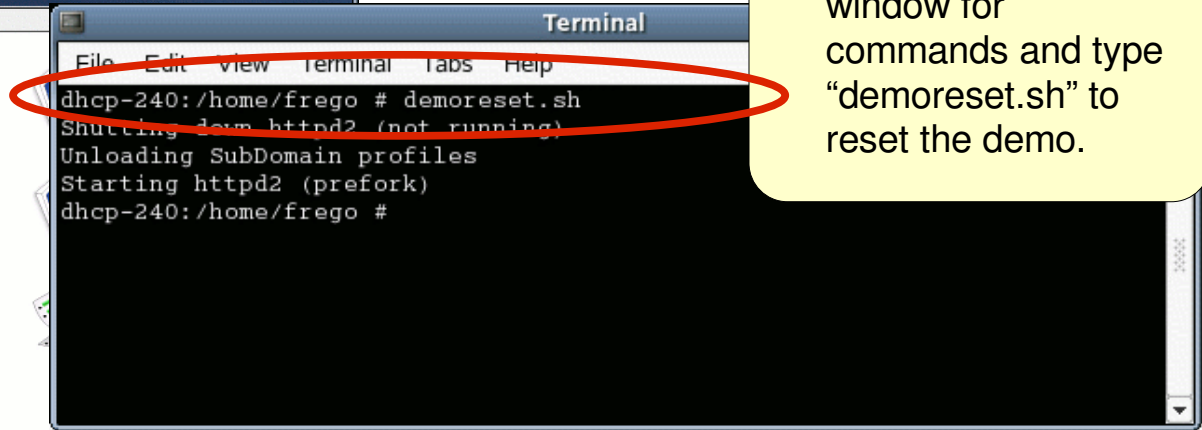
1. Local Apache web server running vulnerable PHF script
2. Exploit PHF vulnerability; deface web page
3. Develop profiles for Apache and PHF app
4. Try hack again; hack fails

The Setup

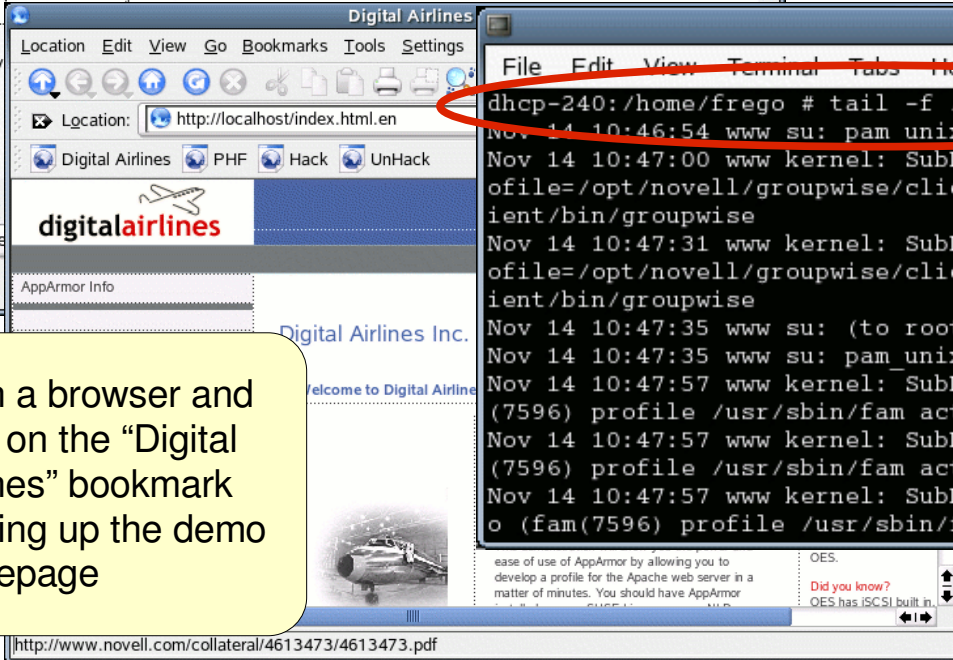
4. open YaST and click on the AppArmor icon to bring up the AppArmor control center



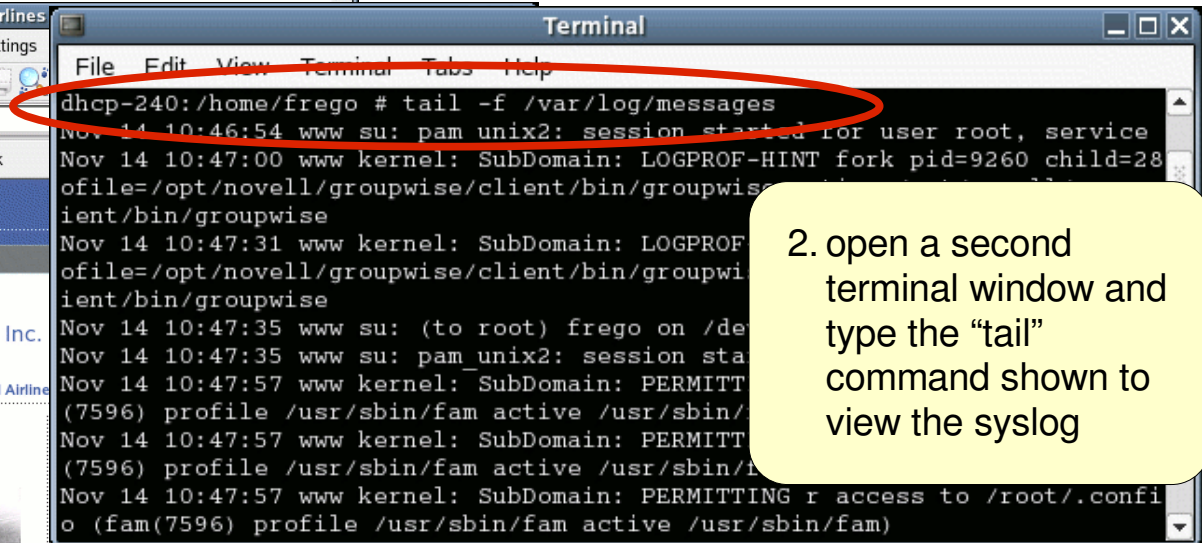
1. open a terminal window for commands and type "demoreset.sh" to reset the demo.



3. open a browser and click on the "Digital Airlines" bookmark to bring up the demo homepage



2. open a second terminal window and type the "tail" command shown to view the syslog



The Hack

4. click the "Unhack" bookmark to reset the homepage, then click on the Digital Airlines bookmark.

3. now click the "Digital Airlines" bookmark to show that the homepage has been defaced!

1. click the "PHF" bookmark to pull up the vulnerable PHF application

2. click the "Hack" bookmark to run the hack that defaces the homepage.

The screenshot shows a Konqueror web browser window displaying the Digital Airlines homepage. The page has a blue header with the 'digitalairlines' logo. A sidebar on the left contains a menu with items like 'AppArmor Info', 'Product Website', 'Product Flyer', 'Frequently Asked Questions', 'OpenOffice Presentation', 'PowerPoint Presentation', and 'Online Documentation'. The main content area features the text 'Digital Airlines Inc. SERVES CRUMMY PRETZELS!!!!' and a large red 'X' over a small image. Below this, there is a section titled 'Welcome to Digital Airlines -- The AppArmor demo' with introductory text about AppArmor. The browser's address bar shows 'http://www.novell.com/products/apparmor/'.

Overlaid on the screenshot are two smaller browser windows. The top one is titled 'Form for CSO PH query' and shows a form with a 'PH Server' field containing 'ns.uiuc.edu'. The bottom one is titled 'Query Results' and shows the output of a command: '/usr/local/bin/ph -m alias=X /bin/cp /srv/www/htdocs/warez /srv/www/htdocs/index.html.en'. Both windows have bookmarks for 'Digital Airlines', 'PHF', 'Hack', and 'UnHack'.

Choosing the Application

1. in YaST, click the Add Profile Wizard to select the app to be profiled

2. type the path to apache as shown (or browse to it)

3. the wizard tells you to start the target app and exercise its functionality

AppArmor Log Analyzer and Profiling Wizard
This wizard will step you through the entries generated by the AppArmor access control module. It provides an easy way to generate profiles from normal application

AppArmor Profiling Wizard
Profiling: /usr/sbin/httpd2-prefork
Please start the application to be profiled in another window and exercise its functionality now.
Once completed, select the "Scan" button below in order to scan the system logs for AppArmor events.
For each AppArmor event, you will be given the opportunity to choose whether the access should be allowed or denied.

Scan system log for entries to add to profiles

Back Abort Finish

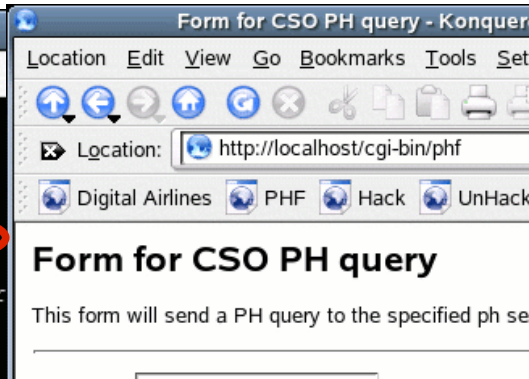
This wizard will help you create a AppArmor profile for an application from scratch or augment an existing profile by learning new behavior.
Please enter the application name for which you wish to create a profile or select Browse to find the program yourself.
Application to Profile
/usr/sbin/httpd2-prefork
Browse
Create Profile Abort

Exercising Apache

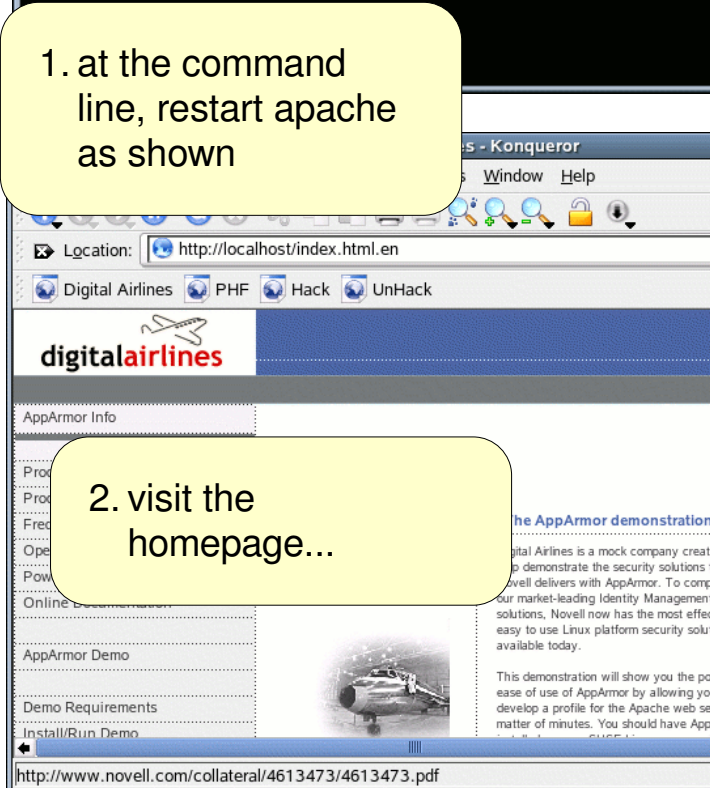
```

Terminal
File Edit View Terminal Tabs Help
dhcp-240:/home/freg0 # demoreset.sh
Shutting down httpd2 (not running)
Unloading SubDomain profiles
Starting httpd2 (prefork)
dhcp-240:/home/freg0 # /etc/init.d/apache2 restart
Syntax OK
Shutting down httpd2 (waiting for all children to terminate)
Starting httpd2 (prefork)
dhcp-240:/home/freg0 #
  
```

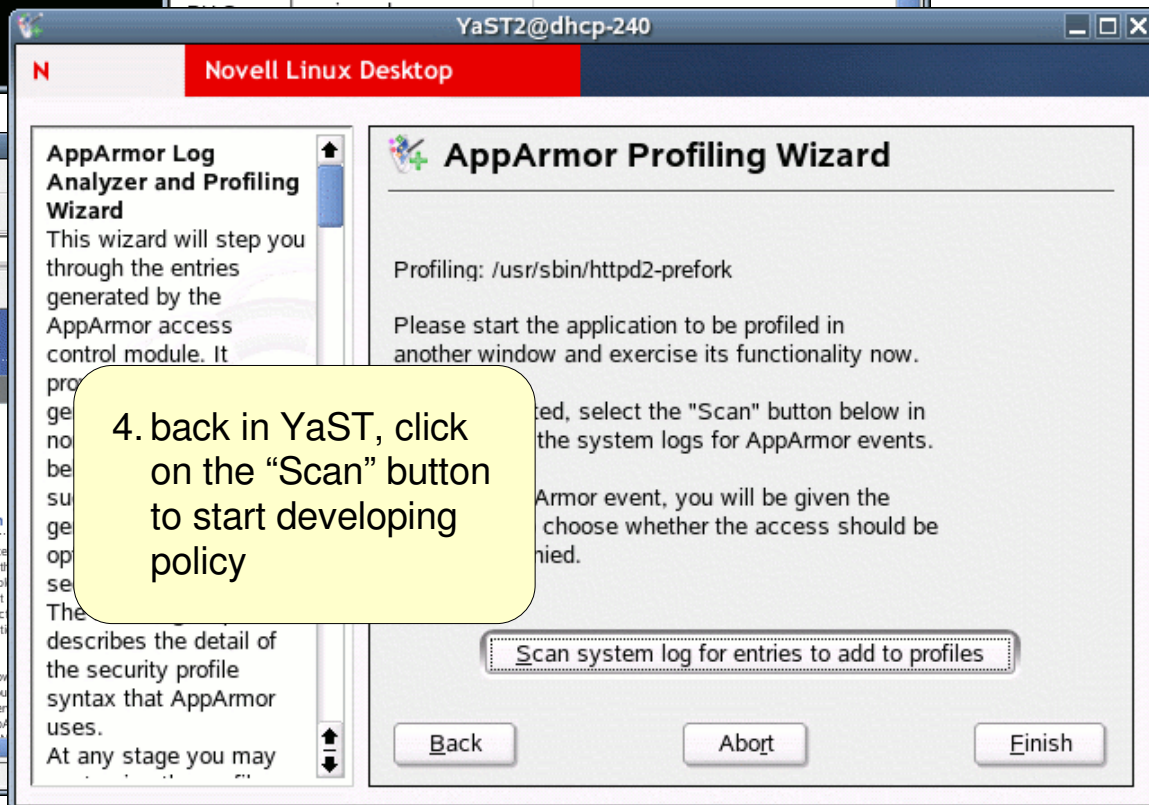
1. at the command line, restart apache as shown



3. ... and visit the PHF application. Now we have a syslog full of apache events.



2. visit the homepage...



4. back in YaST, click on the "Scan" button to start developing policy

Creating AppArmor Policy

The image shows two sequential screenshots of the AppArmor Profiling Wizard in a Novell Linux Desktop environment. The window title is 'YaST2@dhcp-240' and the desktop name is 'Novell Linux Desktop'.

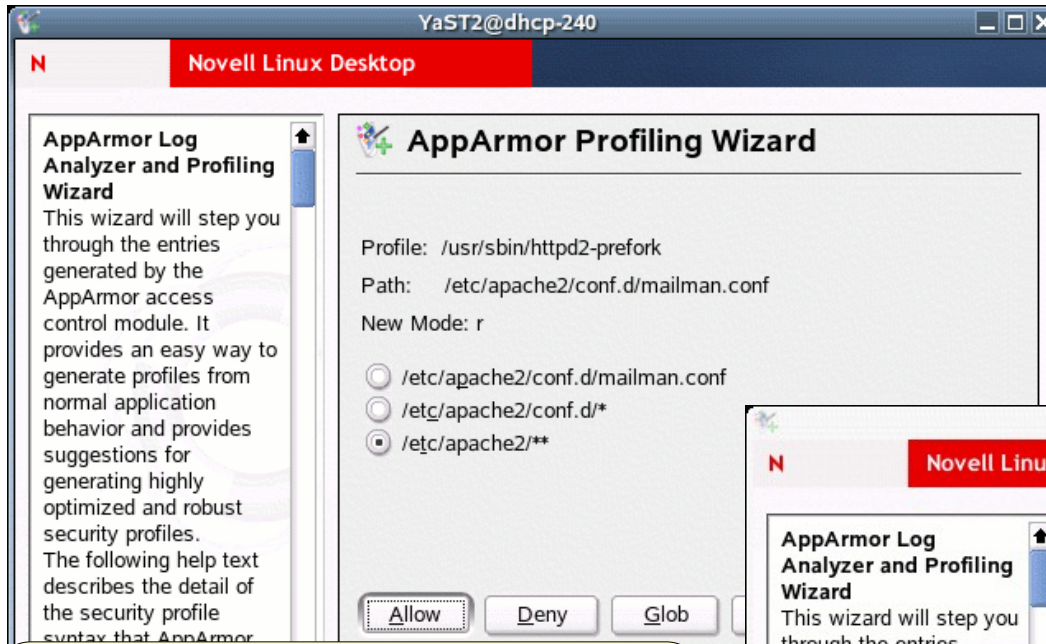
First Screenshot: The wizard is titled 'AppArmor Profiling Wizard'. It shows the profile path as '/usr/sbin/httpd2-prefork' and the application path as '/srv/www/cgi-bin/phf'. Three radio buttons are visible: 'Inherit' (unselected), 'Profile' (selected), and 'Unconfined' (unselected). An 'Allow' button is at the bottom right.

Second Screenshot: The wizard is in a later step, showing the profile path as '/usr/sbin/httpd2-prefork' and the capability as 'net_bind_service'. At the bottom, there are four buttons: 'Back', 'Allow' (highlighted with a dashed border), 'Deny', and 'Finish'.

Callout 1: A yellow box on the left contains the text: "1. the Wizard asks us if the PHF app should have its own profile... we say 'yes' by clicking on the 'Profile' radio button, then 'Allow'".

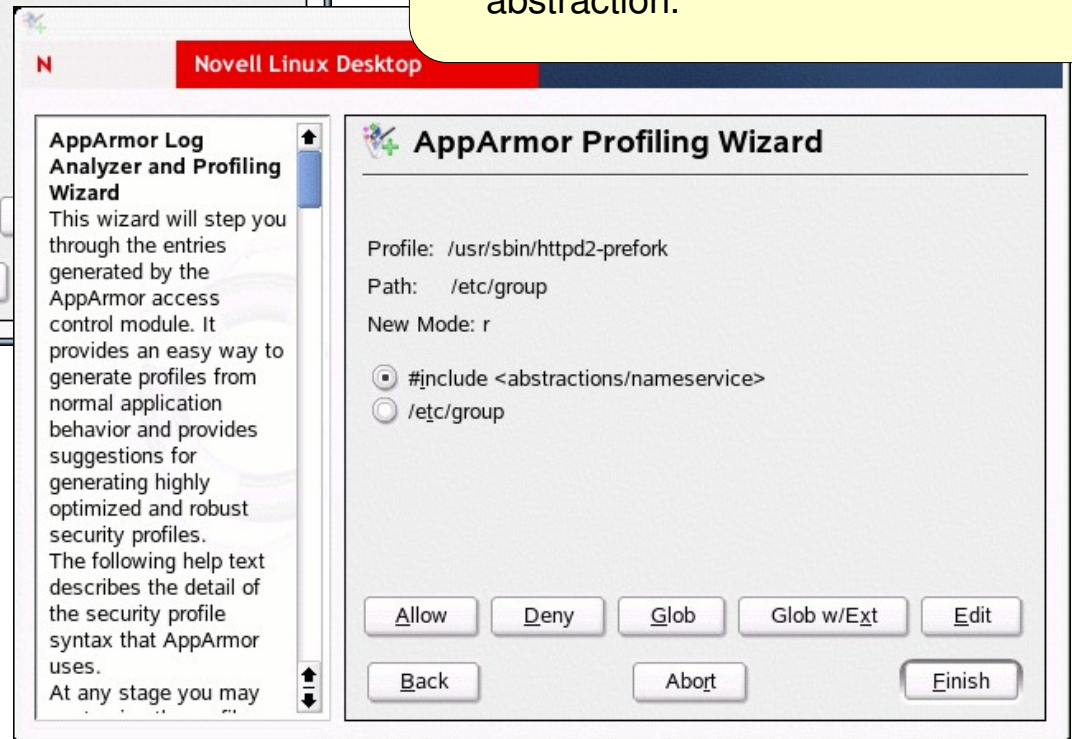
Callout 2: A yellow box on the right contains the text: "2. now the Wizard notices apache needs a few POSIX capabilities. We 'Allow' all of them."

Creating AppArmor Policy 2

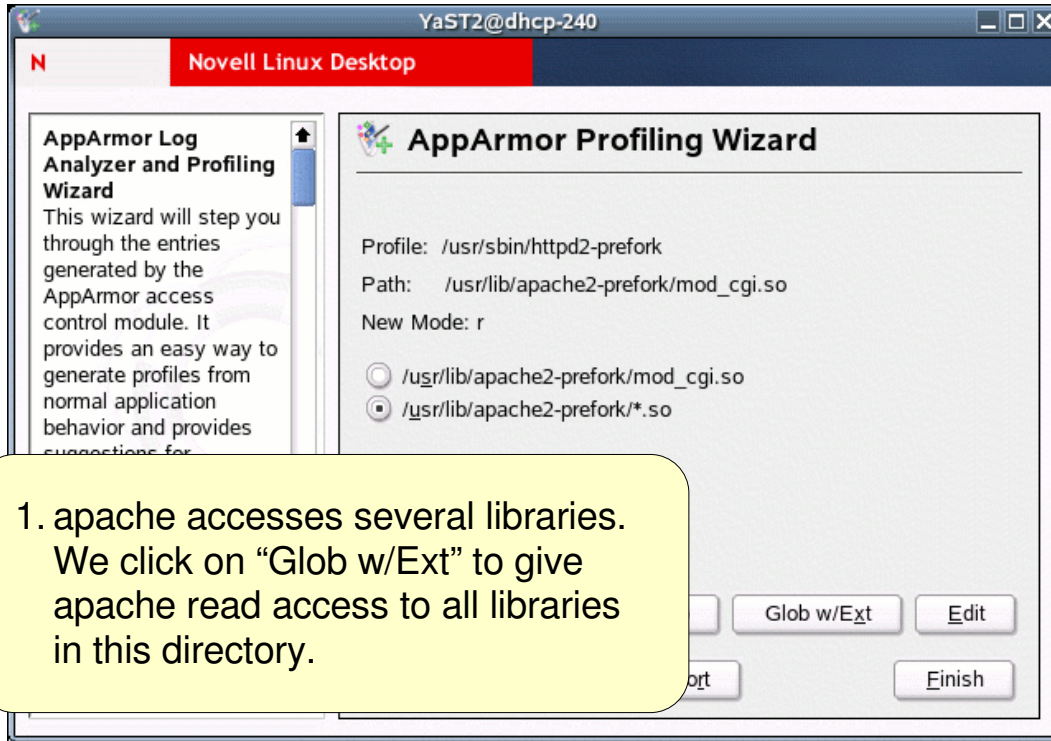


1. the Wizard asks about a file accessed by apache. We click the "Glob" button twice to allow read access to all files in the apache2 directory, then "Allow"

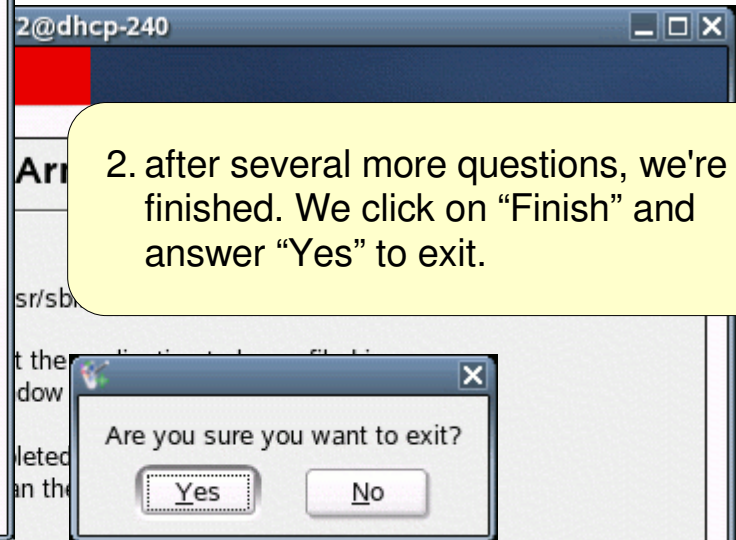
2. the Wizard notices apache needs access to /etc/group and suggests we "include" the nameservice abstraction.



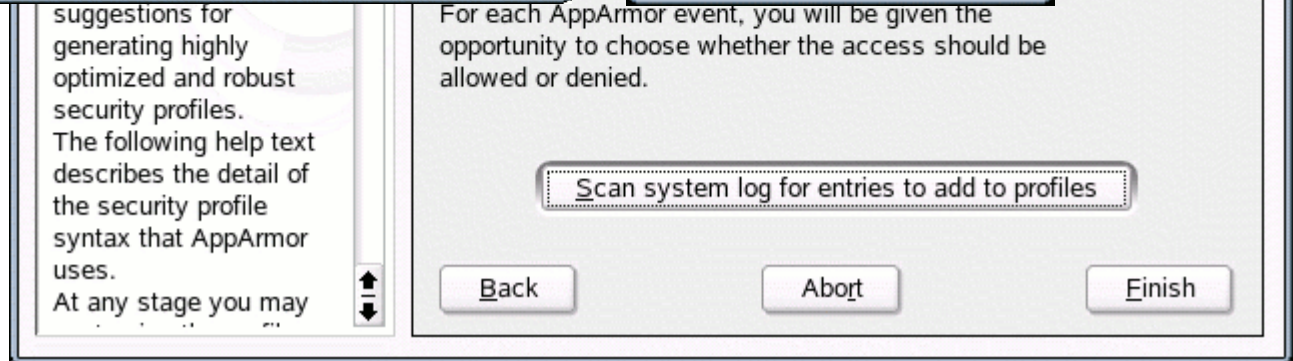
Creating Apache Policy 3



1. apache accesses several libraries. We click on "Glob w/Ext" to give apache read access to all libraries in this directory.



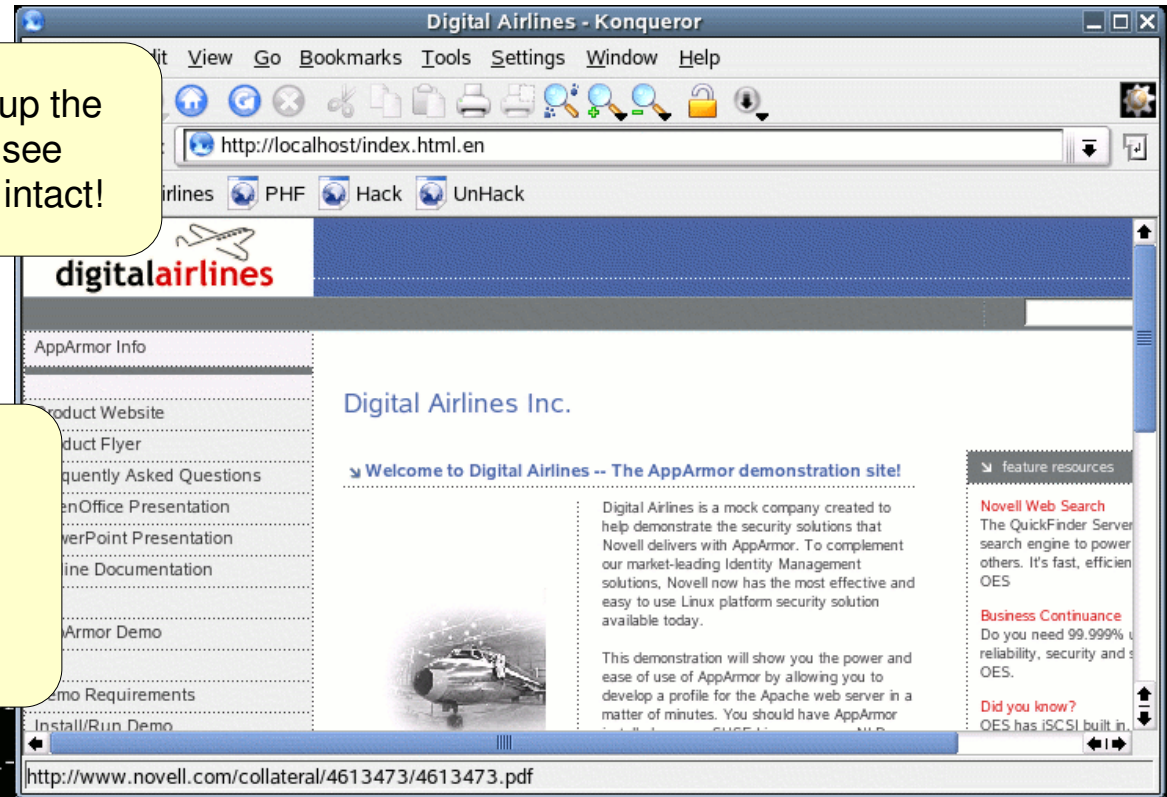
2. after several more questions, we're finished. We click on "Finish" and answer "Yes" to exit.



Blocking the Attack

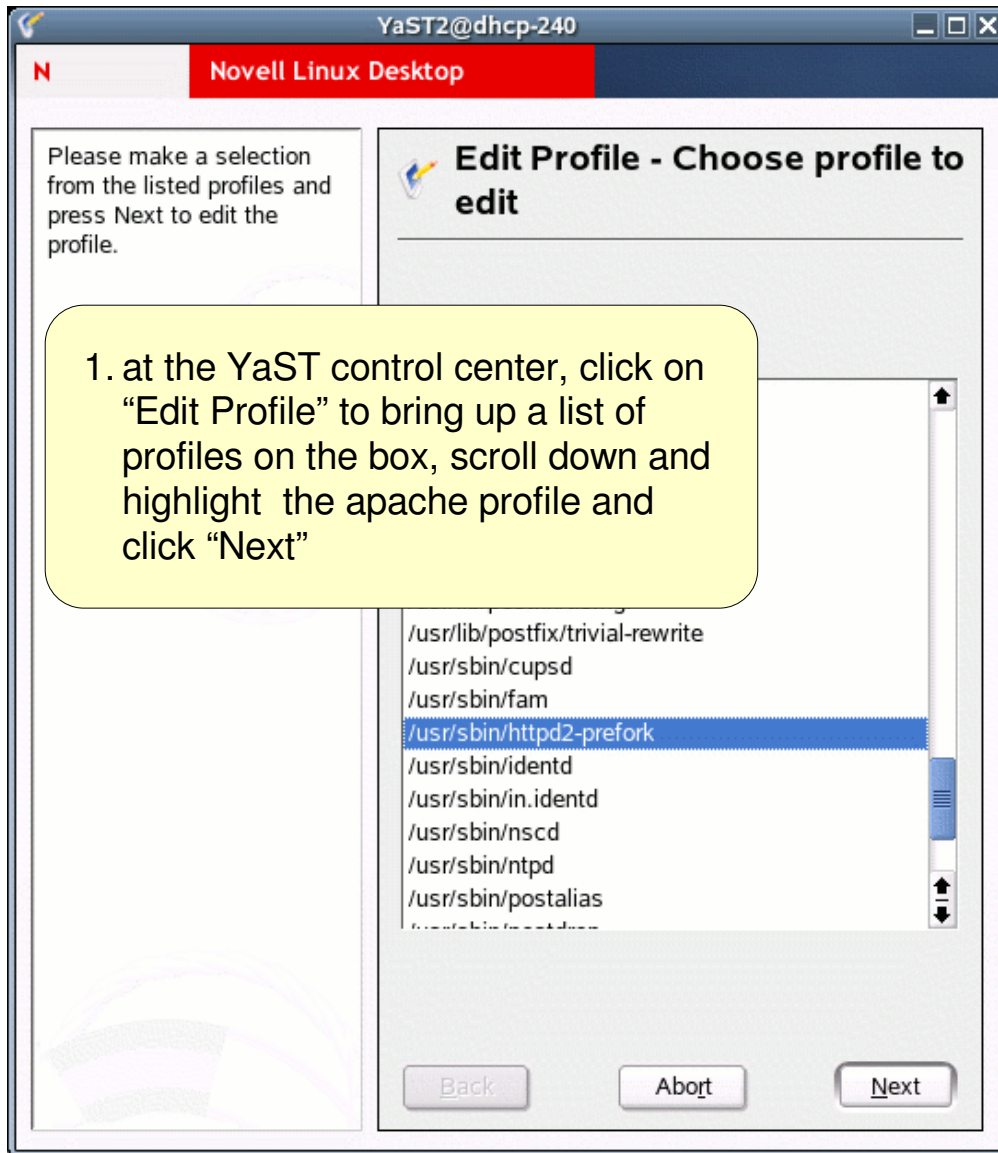
1. back at our website, we pull up the homepage, try the hack and see that the home page remains intact!

2. looking at the syslog, we see a "REJECT" entry telling us an attempted attack via the phf application was blocked by the newly created AppArmor profiles.

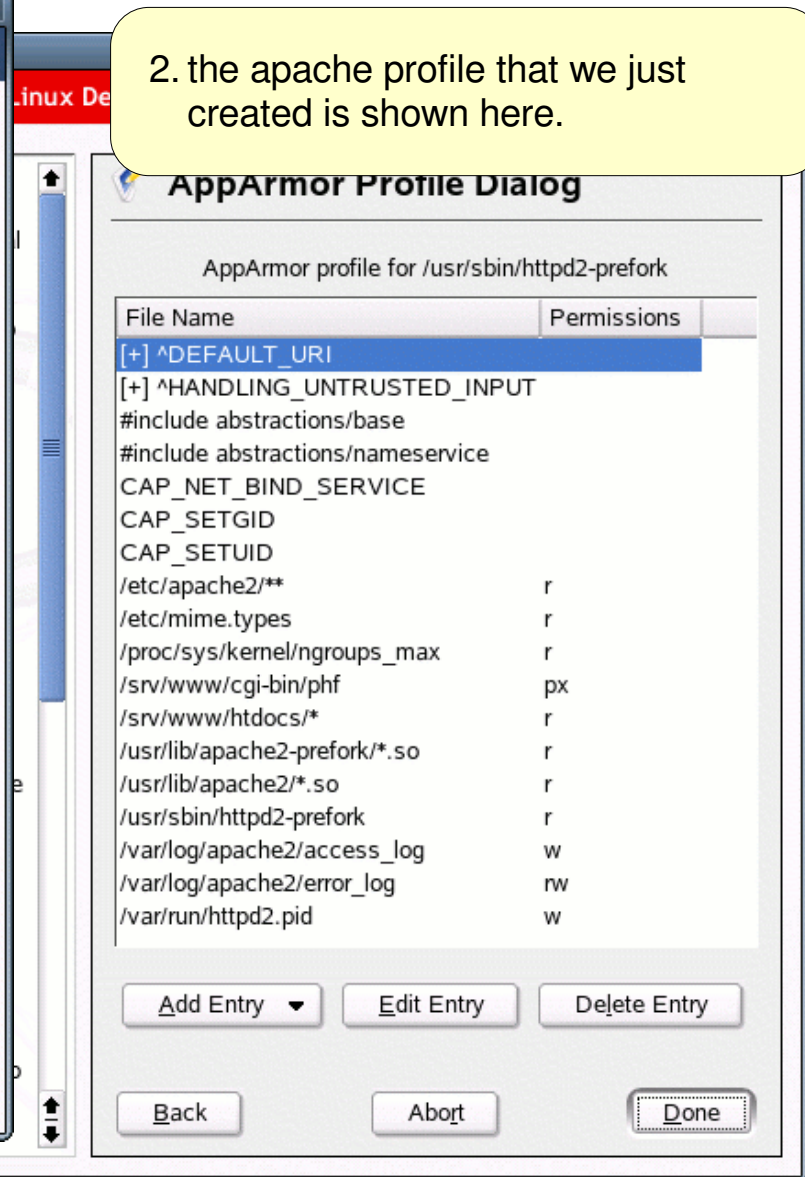


```
Nov 14 11:26:06 www kernel: SubDomain: c.so.6 (phf(465) profile null-complain-
e)
Nov 14 11:26:21 www kernel: SubDomain: PERMITTING r access to /srv/www/htd
ocs/index.html.en (httpd2-prefork(352) profile /usr/sbin/httpd2-prefork ac
tive /usr/sbin/httpd2-prefork)
Nov 14 11:32:07 www logger: GenProf: 0e934993bdc10d4f0a/3b9879df55570
Nov 14 11:33:12 www kernel: SubDomain: REJECTING x access to /bin/bash (ph
f(1229) profile /srv/www/cgi-bin/phf active /srv/www/cgi-bin/phf)
```


Reviewing our Apache Policy



2. the apache profile that we just created is shown here.



What Else Can I Do?

YaST Control Center @ dhcp-240

Administrator Settings

Software

Hardware

System

Network Devices

Network Services

Novell AppArmor

AppArmor Control Panel

Delete AppArmor Profile

Manually Add AppArmor Profile

Update Profiles Wizard

Edit AppArmor Profile

Review AppArmor Events

Enable/Disable AppArmor and configure reporting and alerting

Update loaded profiles based on syslogged activity since last update

View a report showing AppArmor events and filter by program name, date, time, etc.

Help Search Close

AppArmor Review

Sub-process Confinement

Apache mod_perl and mod_php scripts

- Apache mod_apparmor applies new protection before interpreting scripts
- If a specific profile for that script exists, it is used
- If no specific profile exists, then a default script profile is used
- Impact: don't need to run all CGIs with the full privilege of Apache just to get mod_perl efficiency
- The only known way to defend PHP code

Login Authentication

- Add a similar module to PAM: pam_armor
- Pre-authentication sshd and logind are in a restrictive profile

Subprocess Confinement with Change_Hat

Changing In to a Subprofile

- An AppArmor profile applies to an executable program
- If a portion of the program needs different access permissions than other portions, the program can "change hats" to a different role
- To change into a new hat, program calls the `change_hat()` function
- Passes in a pointer to the subprofile and a 32bit `magic_token`.

Changing Out of a Subprofile

- To return to original profile, program calls `change_hat()` with a pointer to `NULL` as the subprofile, and the original `magic_token` value.
- If `magic_token` does not match the original `magic_token`, change back will not happen, and current task will be killed.
- If `magic_token` matches the original token, then the process will change back to the original profile.

YaST Integration



- Reporting
- Alerting
- Profile Development
- Service Configuration

Command-line Interface

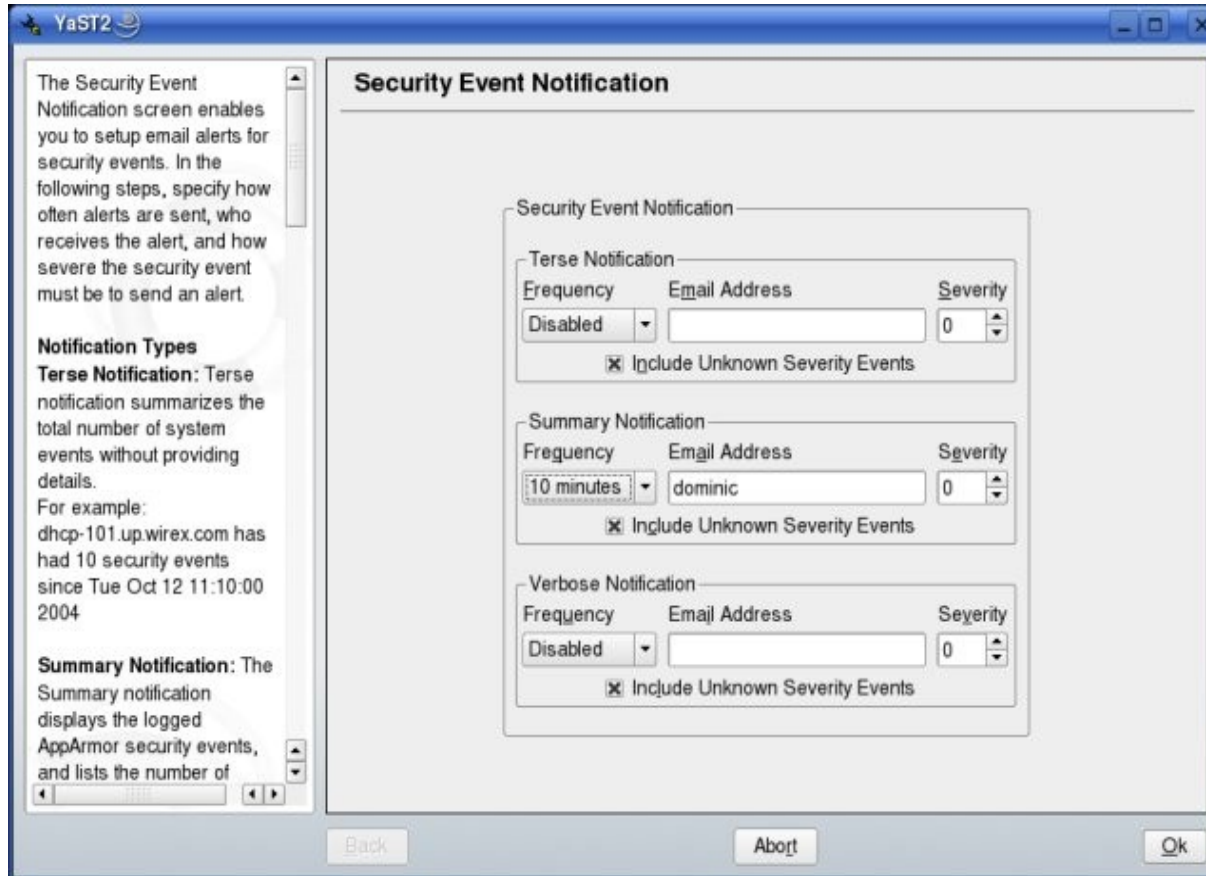
There is also a command-line interface

- for those of us allergic to mice :-)

Reporting and Alerting

- Report on AppArmor events
- Scheduled reporting for tracking data over time
- Audit reports identify unconfined processes
- Data that can integrate into an enterprise security plan

Configuring Notification



Security Incident Report

The screenshot shows a window titled "YaST2 <2>" with a "Security Incident Report" section on the left and an "AppArmor On-Demand Report" section on the right. The left section contains a definition of a Security Incident Report (SIR) and two bullet points: "Policy Exceptions" and "Policy Engine State Changes". The right section displays an "On Demand Event Report - Page 1 of 1" with a table of events.

Security Incident Report (SIR): A report 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. These two types of security events are defined as follows:

- **Policy Exceptions:** When an application requests a resource that's not defined within its profile, a security event is generated.
- **Policy Engine State Changes:** Enforces policy for applications and maintains its own state, including when engines start or stop, when a policy is

AppArmor On-Demand Report

On Demand Event Report - Page 1 of 1

Host	Date	Program	Profile	PID	Severity	Mode	Detail
linux	2006-02-13 21:49:22	mdnsd	/usr/sbin/mdnsd	8464	6	r	/proc/net/unix
linux	2006-02-13 21:49:22	mdnsd	/usr/sbin/mdnsd	8464	6	r	/proc/sys/kernel/

Navigation buttons: Back, Abort, Done, First Page, Previous, Sort, Forward, Last Page, Go to Page.

Security Incident Report – via Email

Evolution - Mail

File Edit View Folder Message Search Help

New Send / Receive Reply Reply to All Forward Move Copy Print Delete Junk Not Junk Cancel

Inbox 1 total Subject or Sender contains Find Now Clear

On This Computer

- Inbox
- Drafts
- Junk
- Outbox
- Sent
- Trash

dominic@localhost

Search Folders

From: AppArmor_Reporting@linux.site Subject: Security Incident Report Date: 10:13 PM

plain text document attachment (Security Incident Report-2006-02-13_22:13:02.001.csv)

HTML page attachment (Security Incident Report-2006-02-13_22:13:02.001.html)

Security Incident Report - Generated by AppArmor

Period: Sat Jan 1 00:00:01 2005 - Mon Feb 13 22:13:02 2006

The following filters were used for report generation:

Filter: prog, Value: mdnsd

Filter: smode, Value: '%REJECT%'

Host	Date	Program	Profile	PID	Severity	Mode	Detail	Access Type
linux	2006-02-13 21:49:22	mdnsd	/usr/sbin/mdnsd	8464	6	r	/proc/net/unix	REJECTING
linux	2006-02-13 21:49:22	mdnsd	/usr/sbin/mdnsd	8464	6	r	/proc/sys/kernel/ngroups_max	REJECTING
linux	2006-02-13 22:09:26	mdnsd	/usr/sbin/mdnsd	9044	6	r	/proc/net/unix	REJECTING
linux	2006-02-13 22:09:27	mdnsd	/usr/sbin/mdnsd	9044	6	r	/proc/sys/kernel/ngroups_max	REJECTING

Application Audit Report

YaST2 <2>

Applications Audit Report (AUD): An auditing tool that reports which application servers are running and whether they are confined by AppArmor. Application servers are applications that accept incoming network connections. This report provides the host machine's IP Address, the date the Applications Audit Report ran, the name and path of the unconfined program or application server, the suggested profile or a placeholder for a profile for an unconfined program, the process ID number, the state of the program (confined or unconfined), and the type of confinement that the profile is performing (enforce/complain).

AppArmor On-Demand Report

Applications Audit Report

Host	Date	Program	Profile	PID	State	Type
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/owcimomd	-	2937	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/opt/kde3/bin/kdeinit	-	4129	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/opt/kde3/bin/kdeinit	-	4129	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/sbin/dhclient	-	4217	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/mdnsd	/usr/sbin/mdnsd	7614	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/mdnsd	/usr/sbin/mdnsd	7614	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/sbin/portmap	-	7932	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/sbin/portmap	-	7932	not-confined	-
linux	Mon Feb 13 21:41:54 2006	/usr/lib/postfix/master	/usr/lib/postfix/master	8010	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/ntpd	/usr/sbin/ntpd	8212	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/ntpd	/usr/sbin/ntpd	8212	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/ntpd	/usr/sbin/ntpd	8212	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/ntpd	/usr/sbin/ntpd	8212	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/squid	/usr/sbin/squid	8233	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/squid	/usr/sbin/squid	8233	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/squid	/usr/sbin/squid	8233	confined	enforce
linux	Mon Feb 13 21:41:54 2006	/usr/sbin/squid	/usr/sbin/squid	8233	confined	enforce

Best Uses For AppArmor

Best Targets for AppArmor



Any Company whose networked servers are running mission critical applications

Any organization with a high cost associated with compromised data

Any organization faced with regulatory compliance

...

Any Linux application is *exposed to attack* and that *matters* :)

Best Targets for AppArmor



Networked Servers

- Isolate all programs interacting with outside world
- Auto-scan tool finds applications that should be profiled
- Profiles represent your total exposure – auditable policy

Business Applications

- Complex, not easily auditable for security
- May be closed source
- Prevents attacks on one component from spreading to other components or systems

Corporate Desktop

- Profiles for desktop applications that process external data
- Separates these programs from other applications/data on the system
- Protects high-risk programs

POS Terminals, Kiosks

- Isolate all programs interacting with outside world
- Comprehensive profile set defined for specific uses
- Limits misuse of machines
- AppArmor profiles for user session and executable apps

Comparisons

AppArmor vs. SELinux: Creating Policy

SELinux audit2allow

1. Create a file at `$SELINUX_SRC/domains/program/foo.te`.
2. Put the daemon domain macro call in the file.
3. Create the file contexts file.
4. Put the first list of file contexts in `file.fc`.
5. Load the new policy with `make load`.
6. Label the foo files.
7. Start the daemon, `service foo start`.
8. Examine your audit log for denial messages.
9. Familiarize yourself with the errors the daemon is generating.
10. Use `audit2allow` to start the first round of policy rules
11. Look to see if the `foo_t` domain tries to create a network socket
12. Continue to iterate through the basic steps to generate all the rules you need.
13. If the domain tries to access `port_t`, which relates to `tclass=tcp_socket` or `tclass=udp_socket` in the AVC log message, you need to determine what port number foo needs to use.
14. Iterate through the remaining AVC denials. When they are resolved with new policy, you can configure the unique port requirements for the `foo_t` domain.
15. With the daemon started, determine which port foo is using.
16. Remove the generic `port_t` rule, replacing it with a specific rule for a new port type based on the `foo_t` domain.

AppArmor

1. Open YaST Control Center
2. Run Server Analyzer to determine which programs to profile
3. Run the Profile Wizard to generate a profile template
4. Run the application through normal operation
5. Run the interactive optimizer to synthesize log events into a profile

Network Storage

SELinux can only do all/nothing access control for NFS-mounted volumes

- SELinux depends on labels, which are stored in extended attributes, which are not supported in NFS2 or NFS3
- Applies a single label to the mount point
- Policies either grant or deny access to the **entire** NFS volume

AppArmor does not use extended attributes

- Can write fine-grained profiles that grant access to individual files that reside on NFS volumes

AppArmor vs. SELinux:

Compare Resulting Policy

SELinux

```
#####
#
# Rules for the ftpd_t domain
#
type ftp_port_t port_type;
type ftp_data_port_t, port_type;
daemon_domain(ftp_d, 'auth_chkpwd')
type etc_ftpd_t, file_type, sysdeffile;

com_network(ftp_d);
com_ybind(ftp_d);
allow ftp_d self:unix_dgram_socket create_socket_perms;
allow ftp_d self:unix_stream_socket create_socket_perms;
allow ftp_d self:process (getcap setcap);
allow ftp_d self:file_file rw_file_perms;

allow ftp_d bin_t:dir search;
com_ssoe(ftp_d, bin_t);
allow ftp_d { ssoe_t ssoe_t_journal_t }:dir search;
allow ftp_d ssoe_t_journal_t:file { getattr read };
allow ftp_d urandom_device_t:chr_file { getattr read };

ifdef('cronD.te',
system_cronD_entry(ftp_ssoe_t, ftp_d)
com_ssoe(ftp_d, { sbin_t shall_ssoe_t })
)

allow ftp_d ftp_data_port_t:tcp_socket name_bind;

ifdef('ftpd_daemon',
define('ftpd_daemon',
) dn1 end ftpd_daemon
ifdef('ftpd_daemon',
rw_dir_create_file(ftp_d, var_lock_t)
allow ftp_d ftp_port_t:tcp_socket name_bind;
allow ftp_d self:unix_dgram_socket { sendto };
com_tcp_connect(userdadmin, ftp_d)
,
ifdef('inetd.te',
domain_auto_trans(inetd_t, ftpd_ssoe_t, ftp_d)
ifdef('tcpd.te', 'domain_auto_trans(tcpd_t, ftpd_ssoe_t, ftp_d)')

# Use sockets inherited from inetd.
allow ftp_d inetd_t:fd use;
allow ftp_d inetd_t:tcp_socket rw_stream_socket_perms;

# Send SIGCHLD to inetd on death.
allow ftp_d inetd_t:process sigchld;
) dn1 end inetd.te
)dn1 end (else) ftp_daemon
ifdef('ftp_shm',
allow ftp_d tmpfs_t:file { read write };
allow ftp_d { tmpfs_t initx_t };shm { read write unix_read unix_write associate };
)

# Use capabilities.
allow ftp_d ftp_d:capability { net_bind_service setuid setgid former fsuid chown sys_resource sys_chroot };

# Append to /var/log/wtmp.
allow ftp_d wtmp_t:file { getattr append };

# allow access to /home
allow ftp_d home_root_t:dir { getattr search };

# Create and modify /var/log/xferlog.
file_xferlog_t, file_type, sysdeffile, logfile;
file_type_auto_trans(ftp_d, var_log_t, xferlog_t, file)
# Remove /bin/lis (can comment this out for protftpd)
# also may need rules to allow tar etc...
com_ssoe(ftp_d, ia_ssoe_t)

allow { ftp_d initx_t } etc_ftpd_t:file_r_file_perms;
allow ftp_d { etc_t resolv_conf_t etc_runtime_t }:file { getattr read };
allow ftp_d proc_t:file { getattr read };

)dn1 end if ftp_home_dir
```

AppArmor

```
/usr/sbin/in.ftpd {
#include <immunix-standard/base>
#include <immunix-standard/nameservice>
#include <immunix-standard/authentication>
#include <user-custom/ftpd>
/
/dev/urandom r,
/etc/etab r,
/etc/ftpaccess r,
/etc/ftpconversions r,
/etc/ftphosts r,
/etc/ftpusers r,
/etc/shells r,
/usr/sbin/in.ftpd r,
/usr/share/ssl/certs/ca-bundle.crt r,
/usr/share/ssl/certs/ftpd-rsa.pem r,
/usr/share/ssl/private/ftpd-rsa-key.pem r,
/var/log/xferlog w,
/var/run w,
/var/run/ftp.{pids,rips}-all wr,
}
```

AppArmor profile for the same program is about 4x smaller

AppArmor vs. SELinux:

Compare Resulting Policy

SELinux

```

#####
#
# Rules for the ftpd_t domain
#
type ftp_port_t, port_type;
type ftp_data_port_t, port_type;
domain_domain(ftp_t, self::selfp);
type etc_ftpd_t, file_type, sysadmfile;

can_network(ftp_t);
can_ybind(ftp_t);
allow ftp_t self::unix_dgram_socket create_socket_perms;
allow ftp_t self::unix_stream_socket create_socket_perms;
allow ftp_t self::process (getcap setcap);
allow ftp_t self::file rw_file_perms;

allow ftp_t bin_t:dir search;
can_exec(ftp_t, bin_t);
allow ftp_t ( sysctl_t sysctl_kernel_t ):dir search;
allow ftp_t sysctl_kernel_t:file { getattr read };
allow ftp_t urandom_device_t:file { getattr read };

ifdef(`crond.te', `
systemd_cored_entry(ftp_exec_t, ftp_t)
can_exec(ftp_t, (sbin_t sbin_t ssmc_t))
')

allow ftp_t ftp_data_port_t:tcp_socket name_bind;

ifdef(`ftpd_daemon', `
define(`ftpd_daemon', `')
define(`ftpd_daemon', `')
') dnl end ftpd_daemon
ifdef(`ftpd_daemon', `
rw_dir_create_file(ftp_t, var_lock_t)
allow ftp_t ftp_port_t:tcp_socket name_bind;
allow ftp_t self::unix_dgram_socket { sendto };
can_tcp_connect(userdomain, ftp_t)
')

ifdef(`inetd.te', `
domain_auto_trans(inetd_t, ftp_exec_t,
ftp_t)
ifdef(`tcpd.te', `domain_auto_trans(tcpd_t,
ftp_exec_t, ftpd_t)')

# Use sockets inherited from inetd.
allow ftpd_t inetd_t:fd use;
allow ftpd_t inetd_t:tcp_socket
rw_stream_socket_perms;

# Send SIGCHLD to inetd on death.
allow ftpd_t inetd_t:process sigchld;
') dnl end inetd.te
')dnl end (else) ftp_is_daemon
endif(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftp_t { tmpfs_t intrtc_t }:shm { read write unix_read
write unix_read write associate };
')

# Use capabilities.
allow ftpd_t ftp_t:capability { net_bind_service setuid };

# Append to /var/log/wtmp.
allow ftpd_t wtmp_t:file { getattr append };

# allow access to /home
allow ftpd_t home_root_t:dir { getattr search };

# Create and modify /var/log/xferlog.
type xferlog_t, file_type, sysadmfile, logfile;
file_type_auto_trans(ftp_t, var_log_t, xferlog_t, file)
# Escoute /bin/lis (can comment this out for proftpd)
# also may need rules to allow tar etc...
can_exec(ftp_t, ls_exec_t)

allow { ftp_t intrtc_t } etc_ftpd_t:file_t:file_perms;
allow ftp_t { etc_t resolv_conf_t etc_runtime_t }:file { getattr read };
allow ftp_t proc_t:file { getattr read };

')dnl end if ftp_home_dir

```

```

.
ifdef(`ftpd_daemon', `
define(`ftpd_is_daemon', `')
') dnl end ftpd_daemon
ifdef(`ftpd_is_daemon', `
rw_dir_create_file(ftp_t, var_lock_t)
allow ftpd_t ftp_port_t:tcp_socket name_bind;
allow ftpd_t self::unix_dgram_socket { sendto };
;
can_tcp_connect(userdomain, ftpd_t)
')
ifdef(`inetd.te', `
domain_auto_trans(inetd_t, ftpd_exec_t,
ftpd_t)
ifdef(`tcpd.te', `domain_auto_trans(tcpd_t,
ftpd_exec_t, ftpd_t)')

# Use sockets inherited from inetd.
allow ftpd_t inetd_t:fd use;
allow ftpd_t inetd_t:tcp_socket
rw_stream_socket_perms;

# Send SIGCHLD to inetd on death.
allow ftpd_t inetd_t:process sigchld;
') dnl end inetd.te
')dnl end (else) ftp_is_daemon
endif(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftpd_t { tmpfs_t intrtc_t }:shm { read
write unix_read unix_write associate };
')
.

```

SELinux uses a custom programming language to specify hard-to-manage rules

AppArmor

```

#####
#
# Rules for the ftpd_t domain
#
type ftp_port_t, port_type;
type ftp_data_port_t, port_type;
domain_domain(ftp_t, self::selfp);
type etc_ftpd_t, file_type, sysadmfile;

can_network(ftp_t);
can_ybind(ftp_t);
allow ftp_t self::unix_dgram_socket create_socket_perms;
allow ftp_t self::unix_stream_socket create_socket_perms;
allow ftp_t self::process (getcap setcap);
allow ftp_t self::file rw_file_perms;

allow ftp_t bin_t:dir search;
can_exec(ftp_t, bin_t);
allow ftp_t ( sysctl_t sysctl_kernel_t ):dir search;
allow ftp_t sysctl_kernel_t:file { getattr read };
allow ftp_t urandom_device_t:file { getattr read };

ifdef(`crond.te', `
systemd_cored_entry(ftp_exec_t, ftp_t)
can_exec(ftp_t, (sbin_t sbin_t ssmc_t))
')

allow ftp_t ftp_data_port_t:tcp_socket name_bind;

ifdef(`ftpd_daemon', `
define(`ftpd_daemon', `')
define(`ftpd_daemon', `')
') dnl end ftpd_daemon
ifdef(`ftpd_daemon', `
rw_dir_create_file(ftp_t, var_lock_t)
allow ftp_t ftp_port_t:tcp_socket name_bind;
allow ftp_t self::unix_dgram_socket { sendto };
can_tcp_connect(userdomain, ftp_t)
')

ifdef(`inetd.te', `
domain_auto_trans(inetd_t, ftpd_exec_t,
ftpd_t)
ifdef(`tcpd.te', `domain_auto_trans(tcpd_t,
ftpd_exec_t, ftpd_t)')

# Use sockets inherited from inetd.
allow ftpd_t inetd_t:fd use;
allow ftpd_t inetd_t:tcp_socket
rw_stream_socket_perms;

# Send SIGCHLD to inetd on death.
allow ftpd_t inetd_t:process sigchld;
') dnl end inetd.te
')dnl end (else) ftp_is_daemon
endif(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftpd_t { tmpfs_t intrtc_t }:shm { read
write unix_read unix_write associate };
')

# Use capabilities.
allow ftpd_t ftp_t:capability { net_bind_service setuid };

# Append to /var/log/wtmp.
allow ftpd_t wtmp_t:file { getattr append };

# allow access to /home
allow ftpd_t home_root_t:dir { getattr search };

# Create and modify /var/log/xferlog.
type xferlog_t, file_type, sysadmfile, logfile;
file_type_auto_trans(ftp_t, var_log_t, xferlog_t, file)
# Escoute /bin/lis (can comment this out for proftpd)
# also may need rules to allow tar etc...
can_exec(ftp_t, ls_exec_t)

allow { ftp_t intrtc_t } etc_ftpd_t:file_t:file_perms;
allow ftp_t { etc_t resolv_conf_t etc_runtime_t }:file { getattr read };
allow ftp_t proc_t:file { getattr read };

')dnl end if ftp_home_dir

```

```

#####
#
# Rules for the ftpd_t domain
#
type ftp_port_t, port_type;
type ftp_data_port_t, port_type;
domain_domain(ftp_t, self::selfp);
type etc_ftpd_t, file_type, sysadmfile;

can_network(ftp_t);
can_ybind(ftp_t);
allow ftp_t self::unix_dgram_socket create_socket_perms;
allow ftp_t self::unix_stream_socket create_socket_perms;
allow ftp_t self::process (getcap setcap);
allow ftp_t self::file rw_file_perms;

allow ftp_t bin_t:dir search;
can_exec(ftp_t, bin_t);
allow ftp_t ( sysctl_t sysctl_kernel_t ):dir search;
allow ftp_t sysctl_kernel_t:file { getattr read };
allow ftp_t urandom_device_t:file { getattr read };

ifdef(`crond.te', `
systemd_cored_entry(ftp_exec_t, ftp_t)
can_exec(ftp_t, (sbin_t sbin_t ssmc_t))
')

allow ftp_t ftp_data_port_t:tcp_socket name_bind;

ifdef(`ftpd_daemon', `
define(`ftpd_is_daemon', `')
') dnl end ftpd_daemon
ifdef(`ftpd_is_daemon', `
rw_dir_create_file(ftp_t, var_lock_t)
allow ftpd_t ftp_port_t:tcp_socket name_bind;
allow ftpd_t self::unix_dgram_socket { sendto };
;
can_tcp_connect(userdomain, ftpd_t)
')
ifdef(`inetd.te', `
domain_auto_trans(inetd_t, ftpd_exec_t,
ftpd_t)
ifdef(`tcpd.te', `domain_auto_trans(tcpd_t,
ftpd_exec_t, ftpd_t)')

# Use sockets inherited from inetd.
allow ftpd_t inetd_t:fd use;
allow ftpd_t inetd_t:tcp_socket
rw_stream_socket_perms;

# Send SIGCHLD to inetd on death.
allow ftpd_t inetd_t:process sigchld;
') dnl end inetd.te
')dnl end (else) ftp_is_daemon
endif(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftpd_t { tmpfs_t intrtc_t }:shm { read
write unix_read unix_write associate };
')

# Use capabilities.
allow ftpd_t ftp_t:capability { net_bind_service setuid };

# Append to /var/log/wtmp.
allow ftpd_t wtmp_t:file { getattr append };

# allow access to /home
allow ftpd_t home_root_t:dir { getattr search };

# Create and modify /var/log/xferlog.
type xferlog_t, file_type, sysadmfile, logfile;
file_type_auto_trans(ftp_t, var_log_t, xferlog_t, file)
# Escoute /bin/lis (can comment this out for proftpd)
# also may need rules to allow tar etc...
can_exec(ftp_t, ls_exec_t)

allow { ftp_t intrtc_t } etc_ftpd_t:file_t:file_perms;
allow ftp_t { etc_t resolv_conf_t etc_runtime_t }:file { getattr read };
allow ftp_t proc_t:file { getattr read };

')dnl end if ftp_home_dir

```

Classical Linux syntax with read/write/execute permissions: No new jargon

AppArmor Roadmap

AppArmor Near Term Development

- **Network Access Control** – TCP/UDP based network access control per process
- **Profile Merge Tool** – allows two profiles to be merged into a single profile consisting of union set of both
- **Profile Sharing** – tools and portal for community sharing of AppArmor profiles
- **Tomcat Support** – AppArmor containment for Java servlets
- **PAM change_hat** – strengthens security of AppArmor's role-based shell functionality for applications that use PAM (e.g., sshd, gdm, ftp)
- **CIM Providers** – Standards based CIM instrumentation for Reporting, Alerting, Profile State

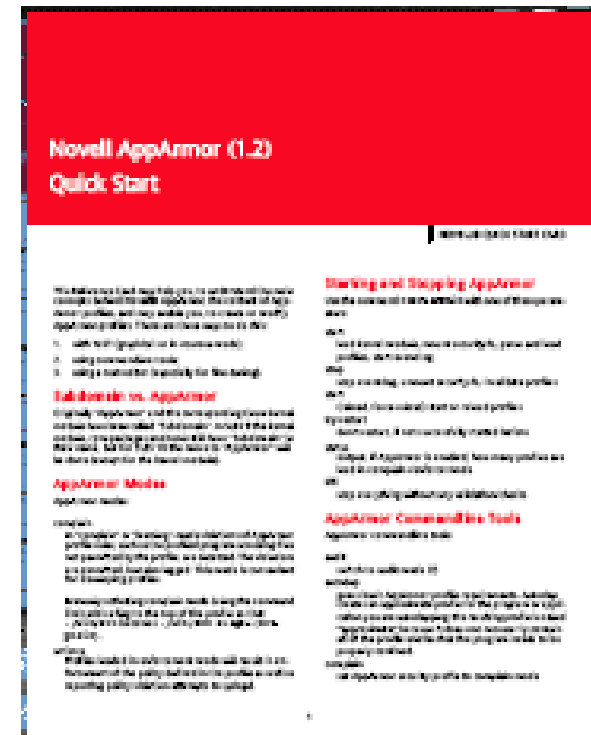
AppArmor Future Development

- **DB Armor** – access controls for database tables and files
- **Default Policy** – system level list of resources that can *only* be accessed through an AppArmor profile
- **DBUS Event Advertising** – report security events via DBUS
- **DBUS / HAL Event Mediation** – containment for hardware abstraction layer
- **IPC Mediation** – mediate inter-process communication
- **Enterprise Management** – integration with Novell enterprise management system
- **Profile Lint** – tool for analyzing profiles for dangerous rules
- **Resource Limits Mediation**
- **Centralized Profile Development**

AppArmor Resources

Where to get AppArmor information

- Documentation
 - <http://www.novell.com/documentation/apparmor>
 - AppArmor Quickstart
 - AppArmor Users Guide
 - AppArmor Reference Card
- Websites
 - <http://www.novell.com/linux/security/apparmor/>
 - <http://www.opensuse.org/apparmor>



Availability

AppArmor bundled with:

- SLES10
- SLED10
- SUSE Linux 10.1

AppArmor is open source: GPL

- <http://opensuse.org/AppArmor>
- Mailing lists: apparmor-announce, apparmor-general, apparmor-dev

Contact:

- Crispin Cowan, Security Architect crispin@novell.com

AppArmor for Everyone

AppArmor's ease of use makes it a good idea for a de facto Linux security standard

Need ports to many distros

Note: openSUSE build service will build packages for many different distros

- http://en.opensuse.org/Build_Service

Use this to build AppArmor (or anything else) for your distro of choice

AppArmor for Debian

- AppArmor has already been ported to Ubuntu by Magnus Runesson
 - <http://www.linuxalert.org/ubuntu/apparmor/>
 - In discussion for mainstream inclusion in future Ubuntu releases
- and to Gentoo by Mathew Snelham
 - http://sigalrm.com/apparmor/apparmor-ebuilds_2006
- Debian:
 - Should be easy to generate from Ubuntu port
 - Need a maintainer

AppArmor for Red Hat

AppArmor has been ported to RH variants multiple times

- But the people doing the work didn't want to be public maintainers, so no public repository

Steve Beattie @ SUSE ported to RHEL5

- http://developer.novell.com/wiki/index.php/Special:Downloads/apparmor/Development_-_RHEL5_beta_2_packages/
- http://software.opensuse.org/download/home:/steve-beattie/Fedora_Extras_6/

Seeking a RH/Fedora user to maintain the

Novell.[®]

Unpublished Work of Novell, Inc. All Rights Reserved.

This work is an unpublished work and contains confidential, proprietary, and trade secret information of Novell, Inc. Access to this work is restricted to Novell employees who have a need to know to perform tasks within the scope of their assignments. No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of Novell, Inc. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

