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

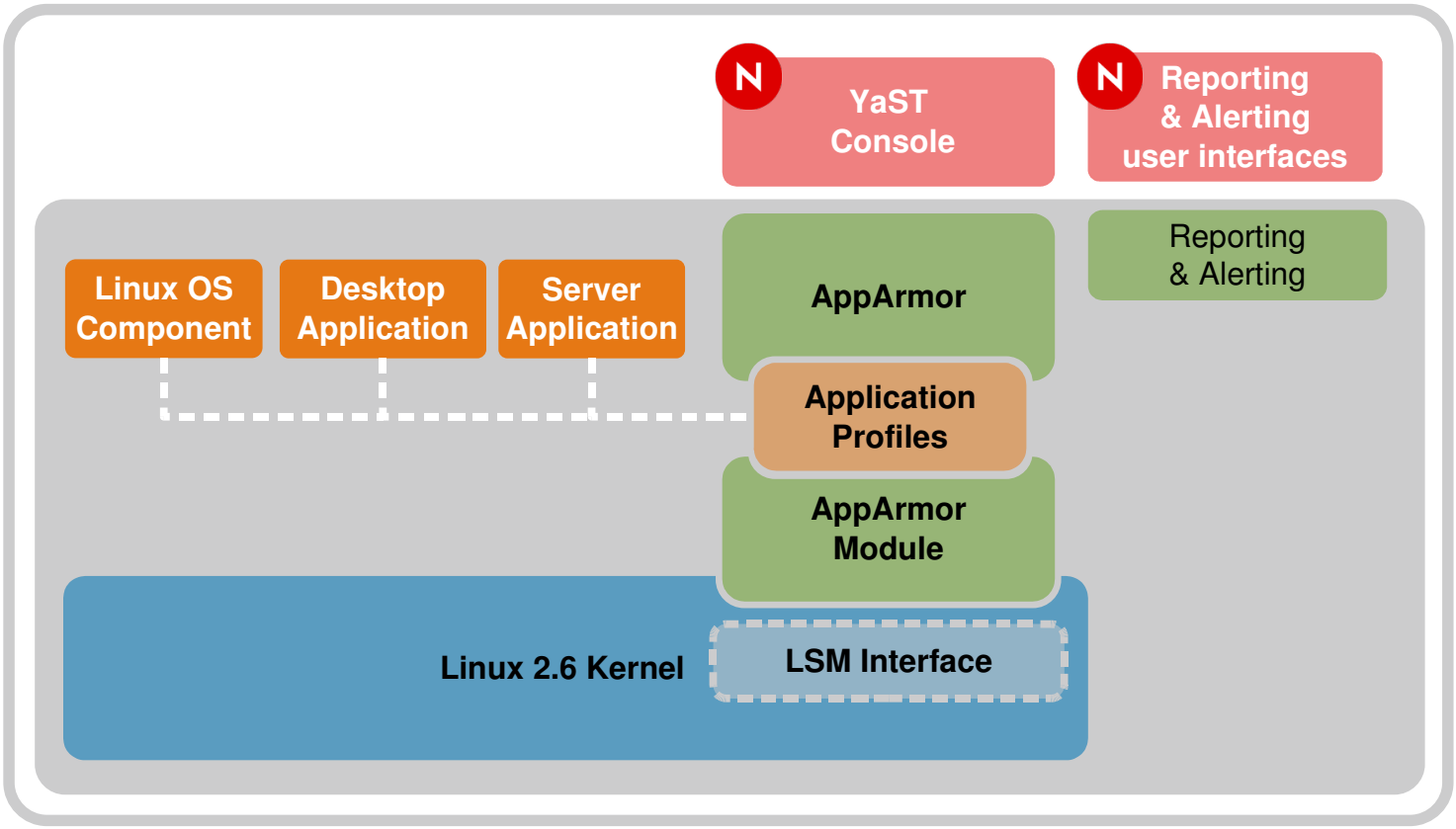
- Proactive “whitelist” approach, **no** attack signature database
- Profiles grant access to the **minimal** list of files/directories and POSIX capabilities required by the application
- Complete kernel-level mediation through Linux Security Module

Automated workflow for security profile generation

- Autoscan 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

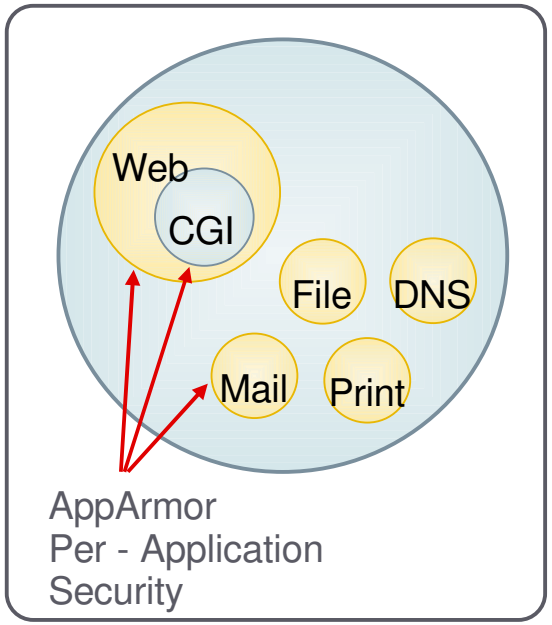
Critical Issue #2: Security Model

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

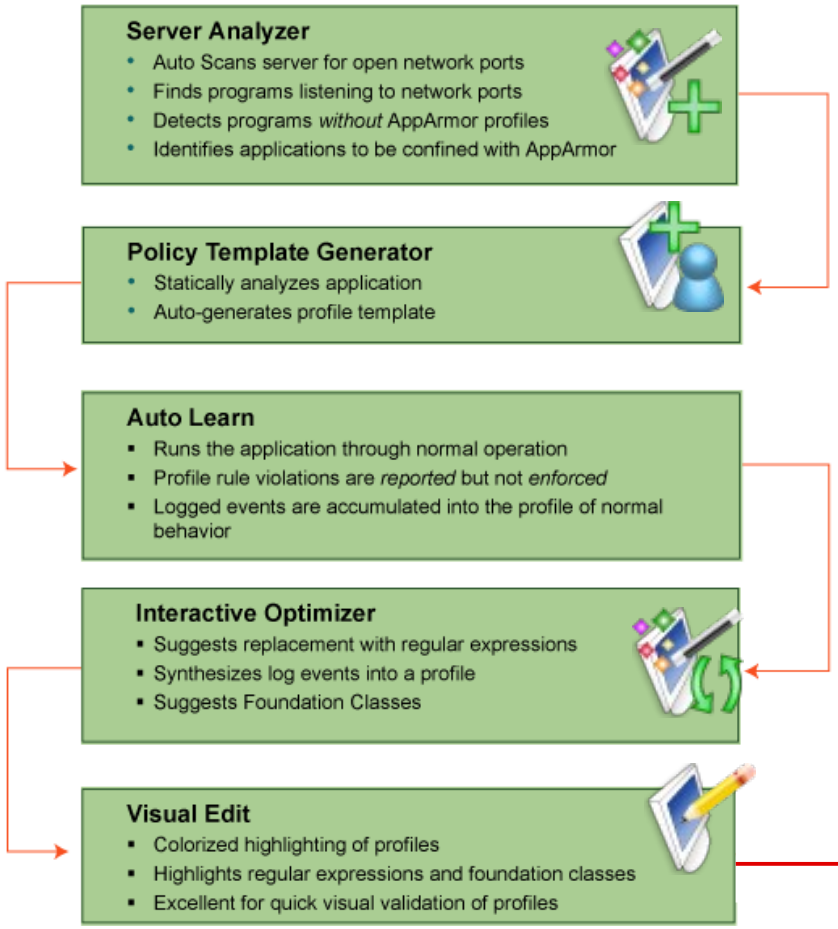
Example security profile for `ntpd`

```
/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,
}
```

Automated Workflow




```
/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,
```

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)

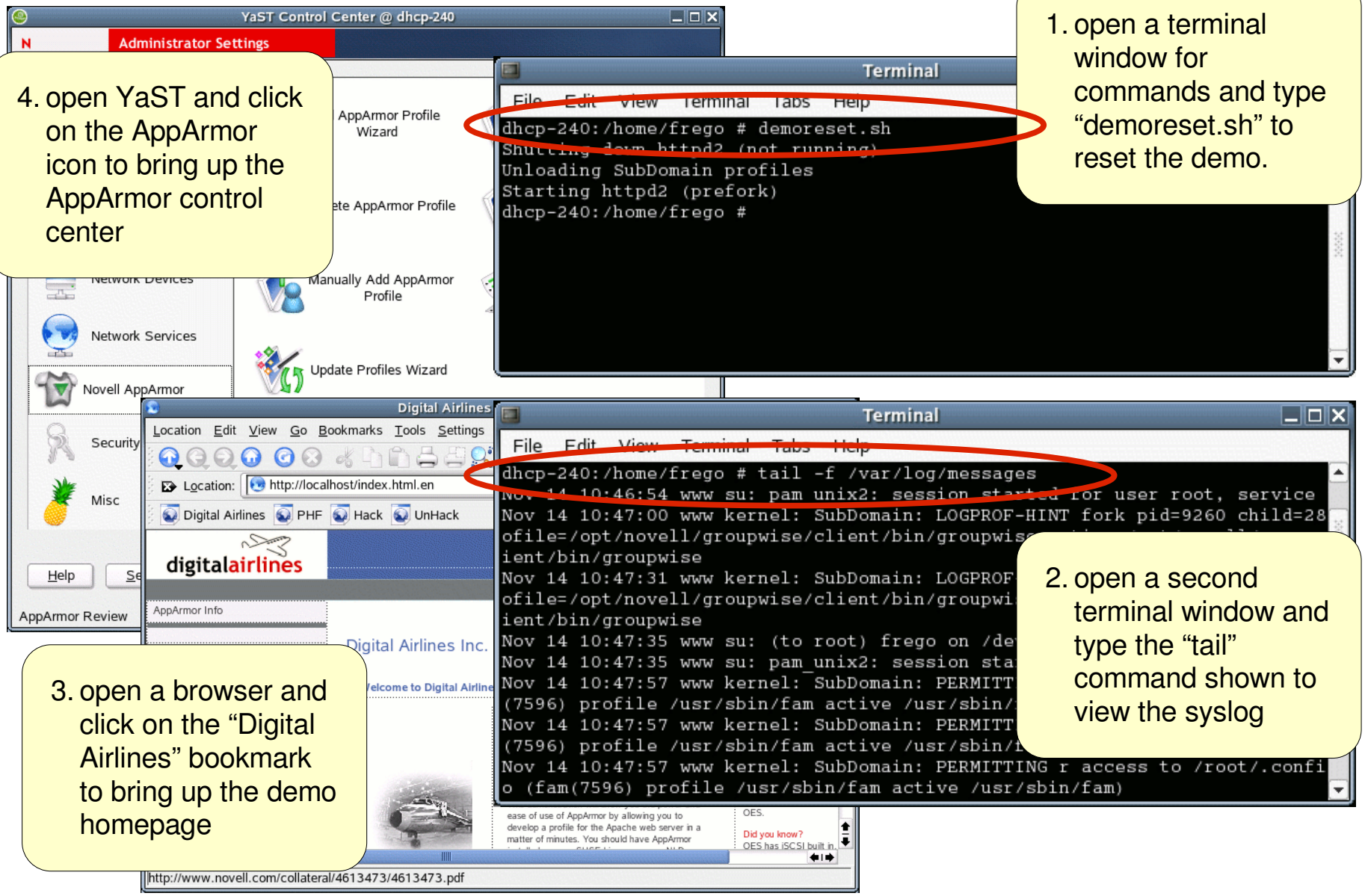
- firefox
- opera
- evolution
- gaim
- realplay
- postfix
- acroread
- mysqld
- ethereal
- postfix
- sendmail
- many more...

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



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

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

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

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

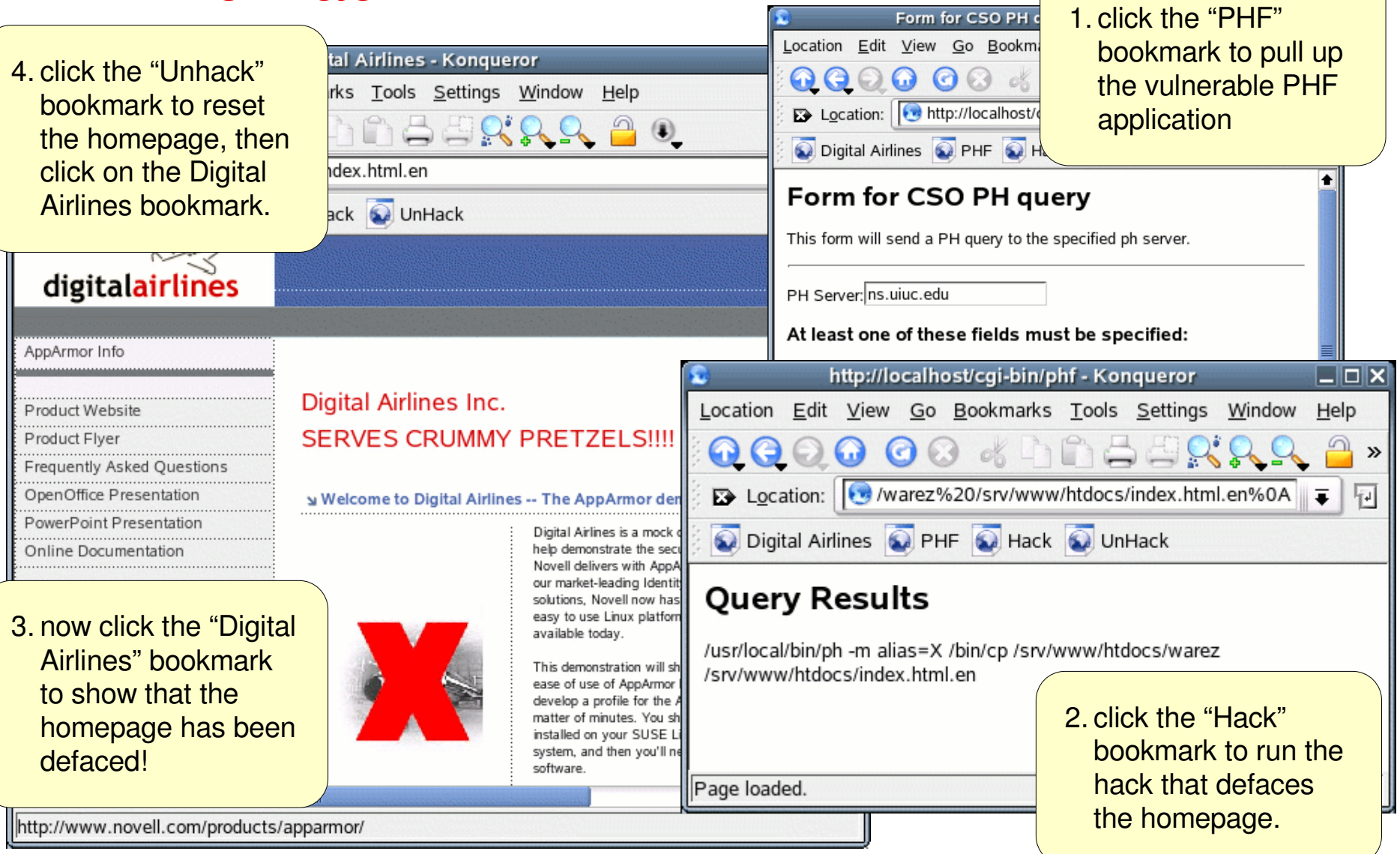
The Hack

1. click the “PHF” bookmark to pull up the vulnerable PHF application

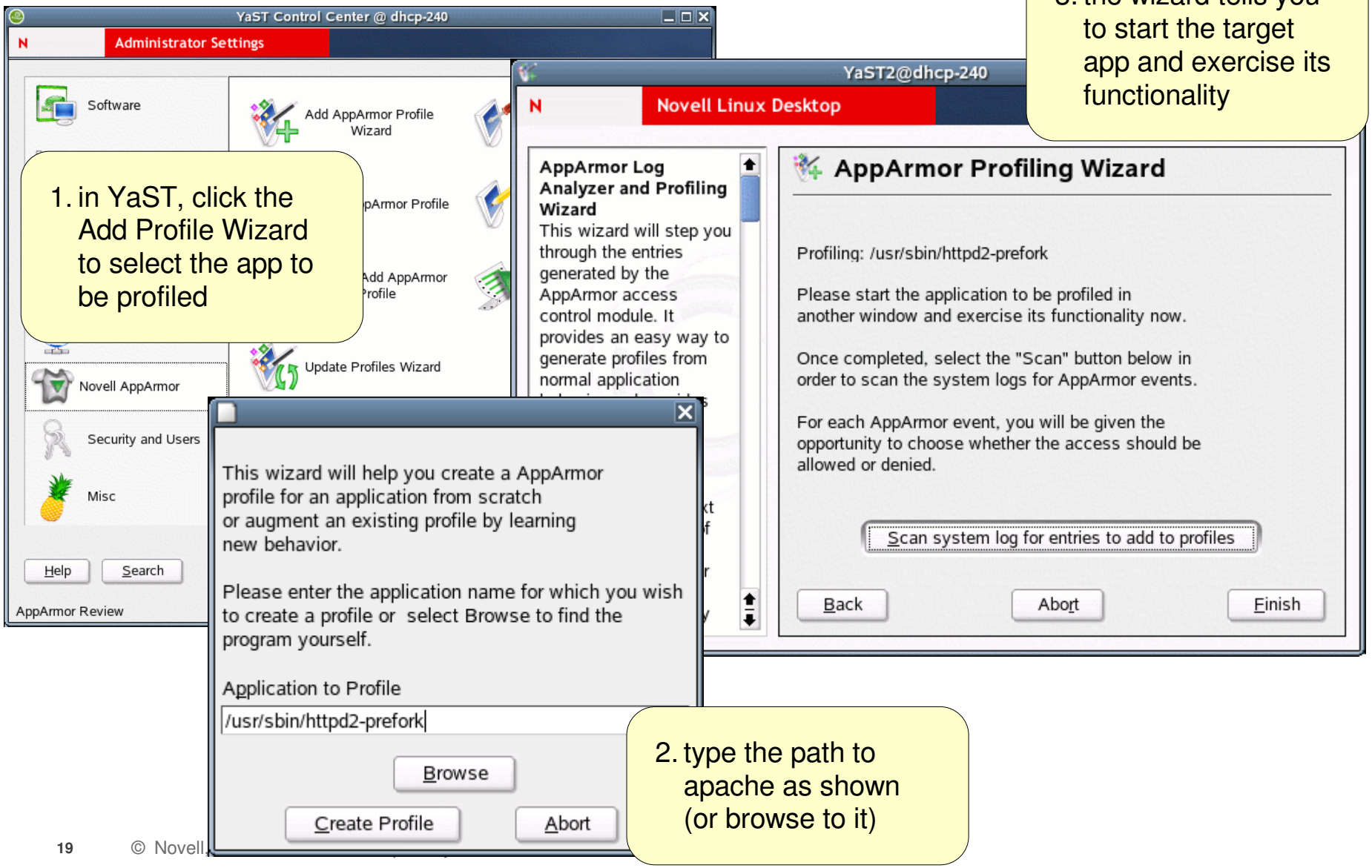
2. click the “Hack” bookmark to run the hack that defaces the homepage.

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

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



Choosing the Application

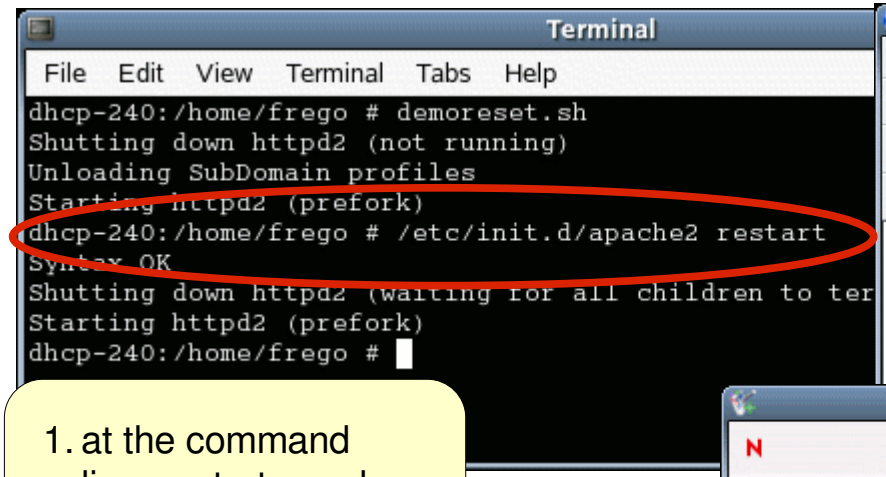


1. in YaST, click the Add Profile Wizard to select the app to be profiled

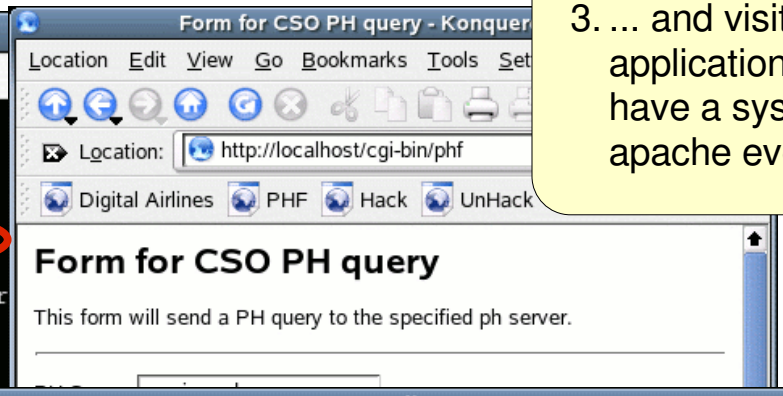
2. type the path to apache as shown (or browse to it)

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

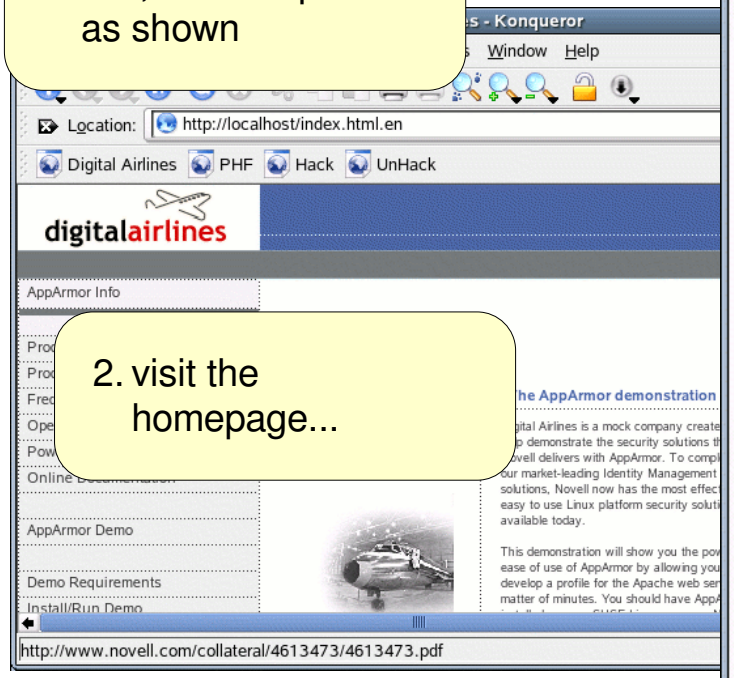
Exercising Apache



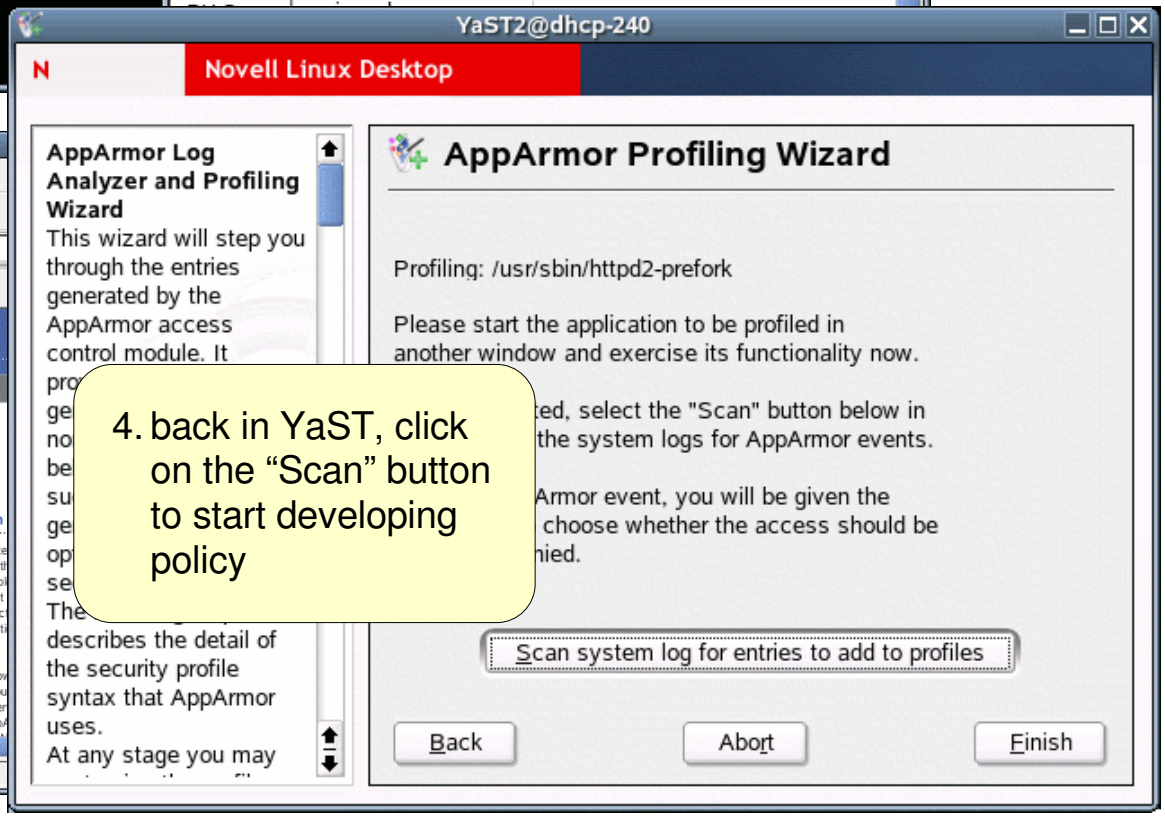
1. at the command line, restart apache as shown



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

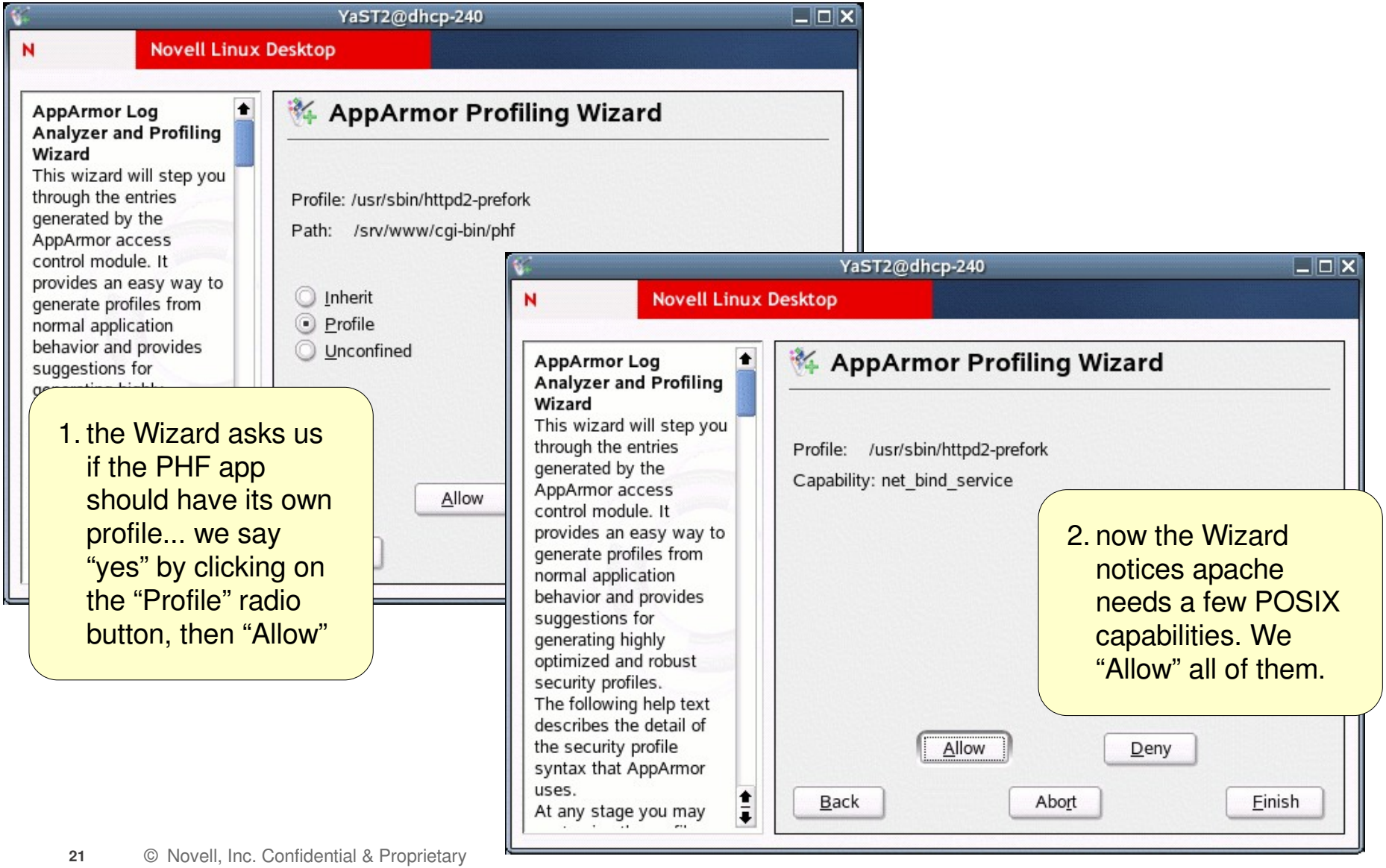


2. visit the homepage...



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

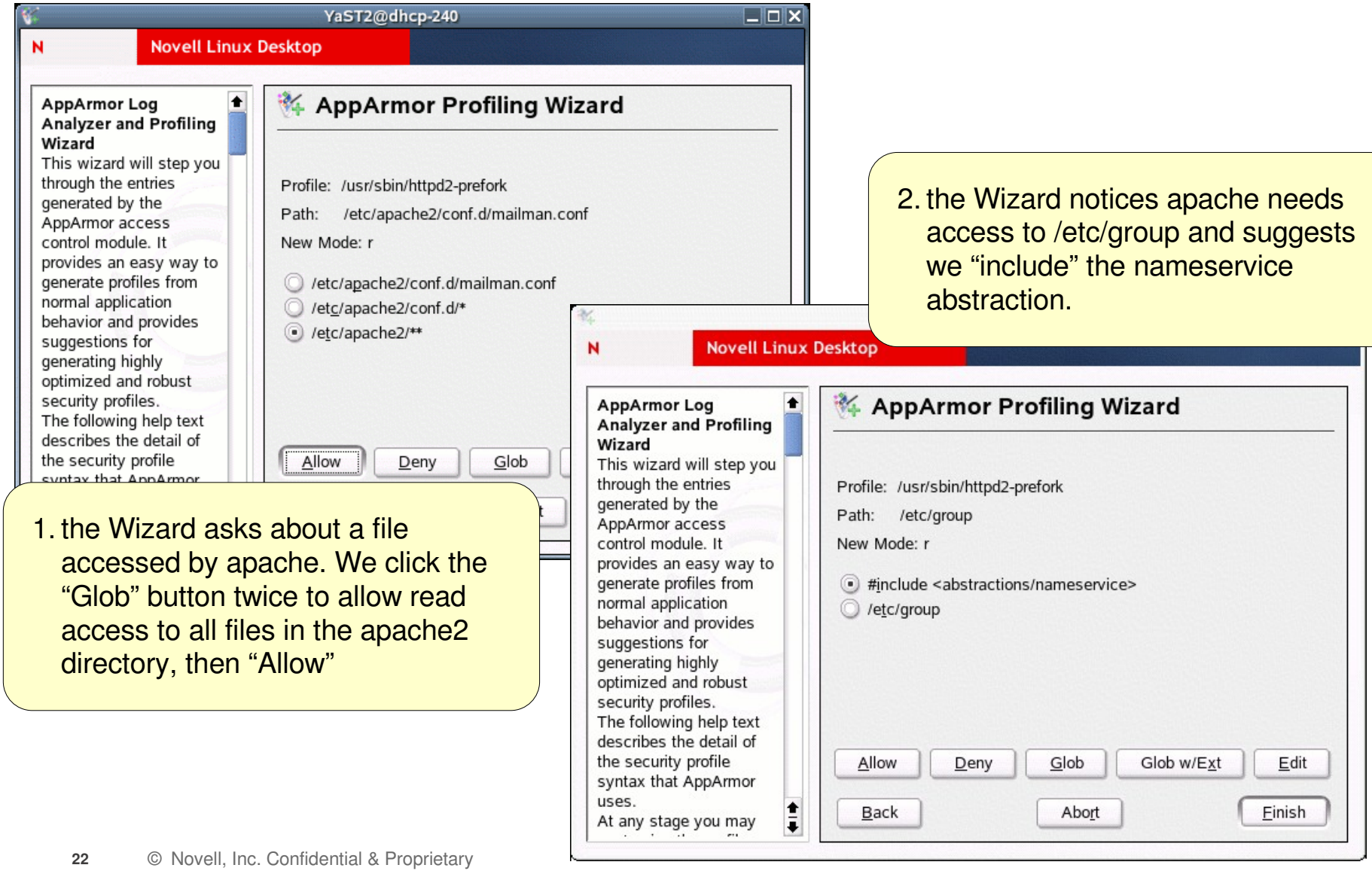
Creating AppArmor Policy



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”

2. now the Wizard notices apache needs a few POSIX capabilities. We “Allow” all of them.

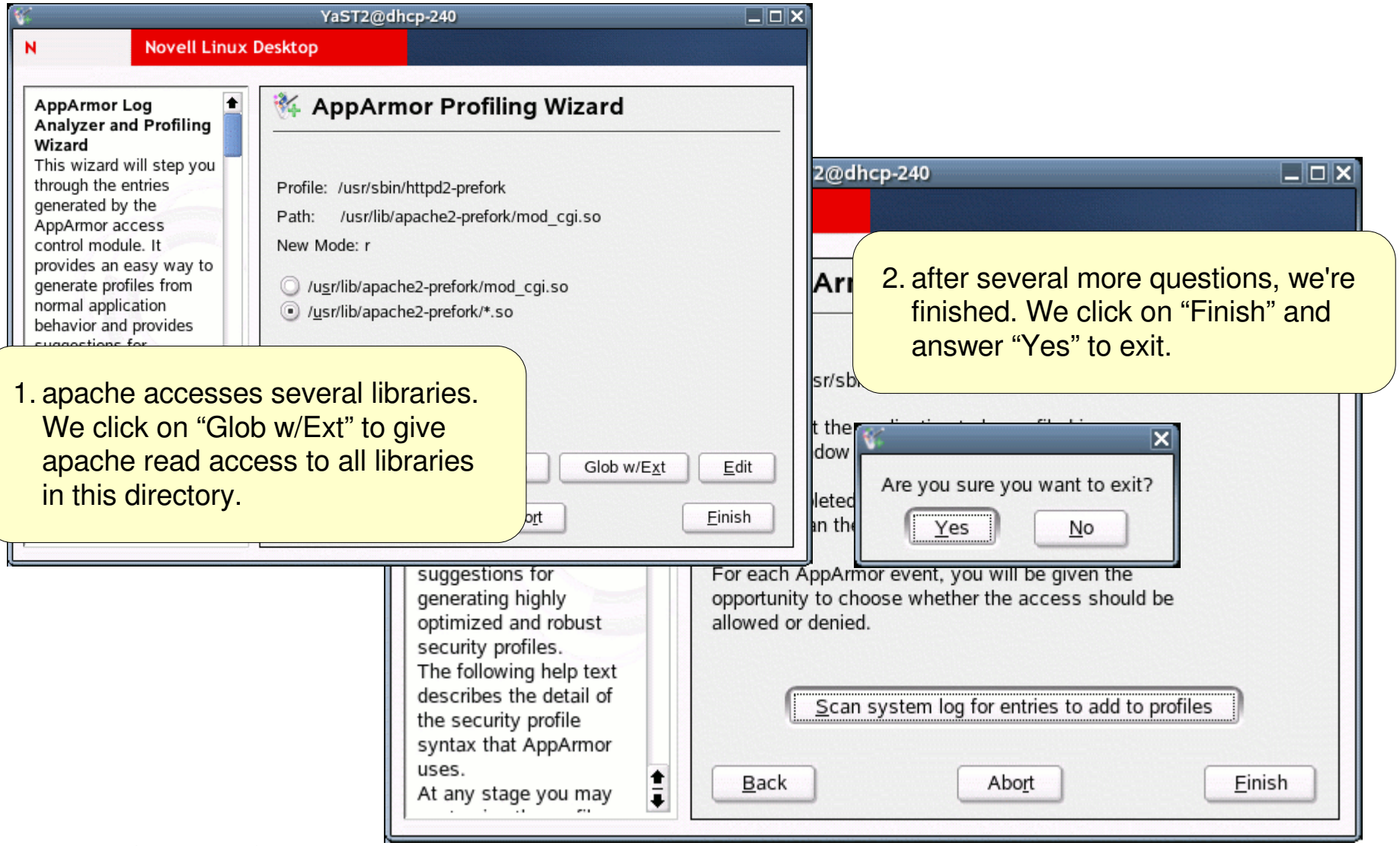
Creating AppArmor Policy 2



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.

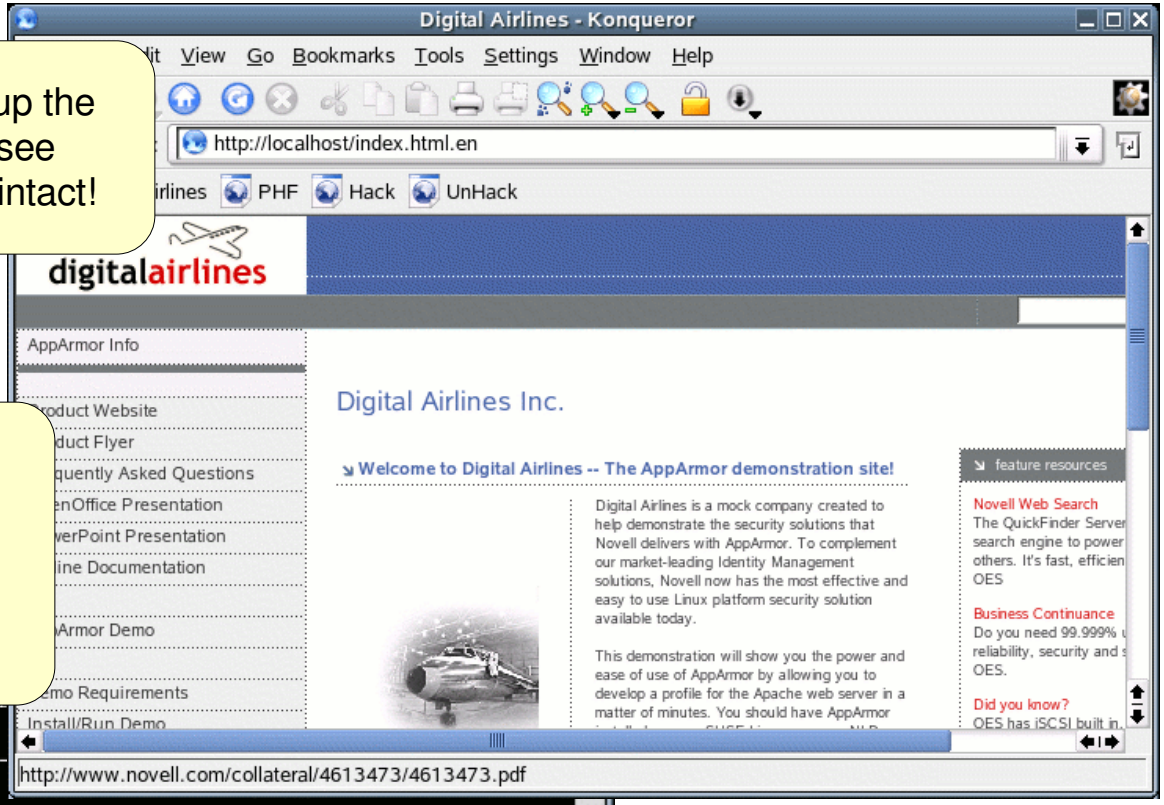
Creating Apache Policy 3

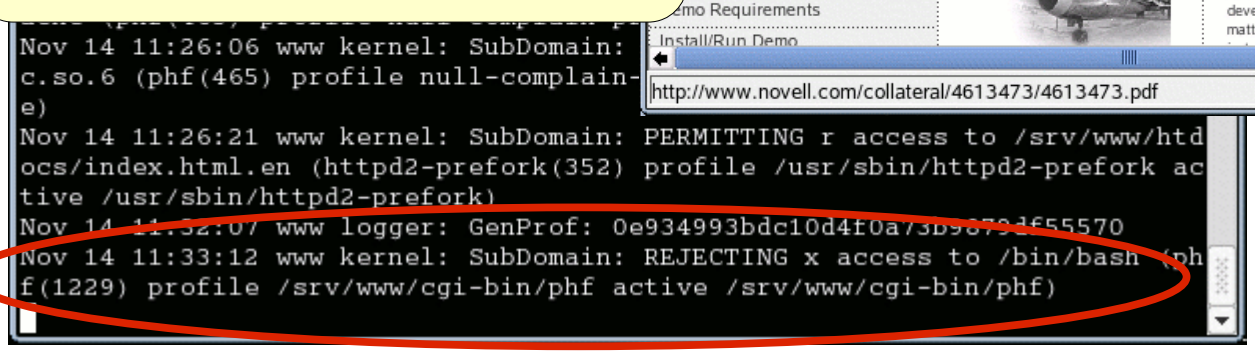


Blocking the Attack

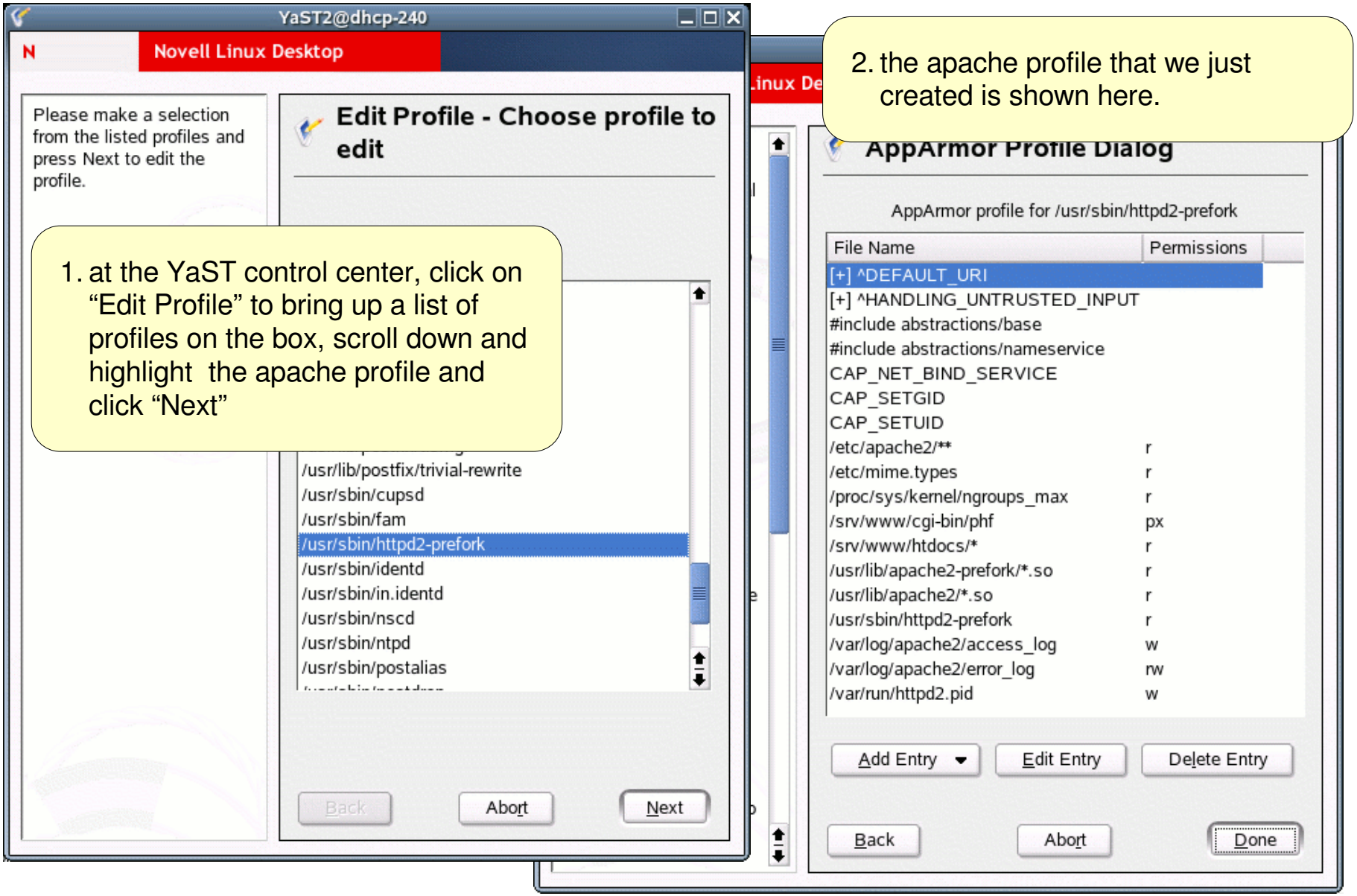
1. back at our website, we pull up the homepage, try the hack and see that the home page remains intact!

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.





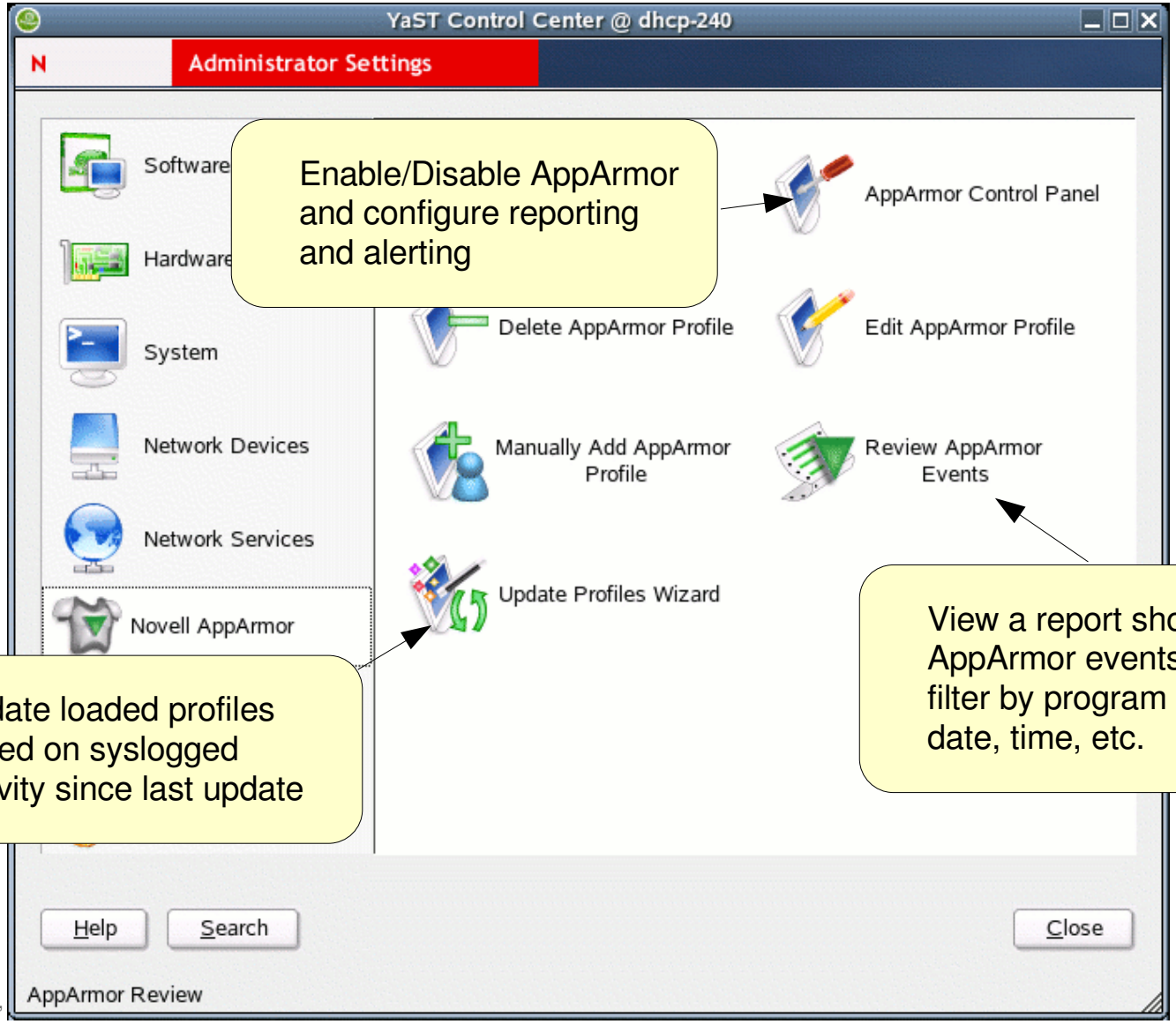
Reviewing our Apache Policy



1. at the YaST control center, click on “Edit Profile” to bring up a list of profiles on the box, scroll down and highlight the apache profile and click “Next”

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

What Else Can I Do?



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 scrpt 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

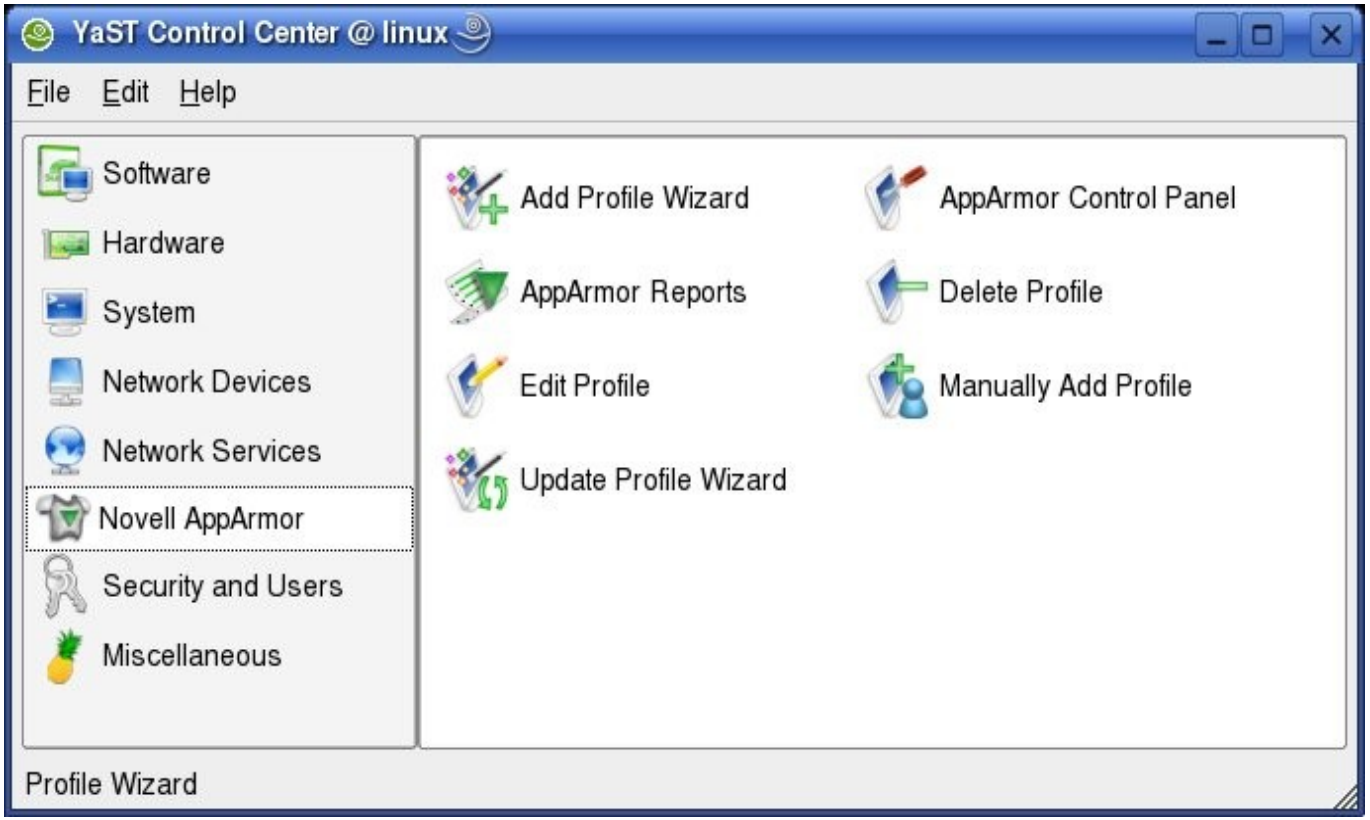
Changing In to a Subprofile

- 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

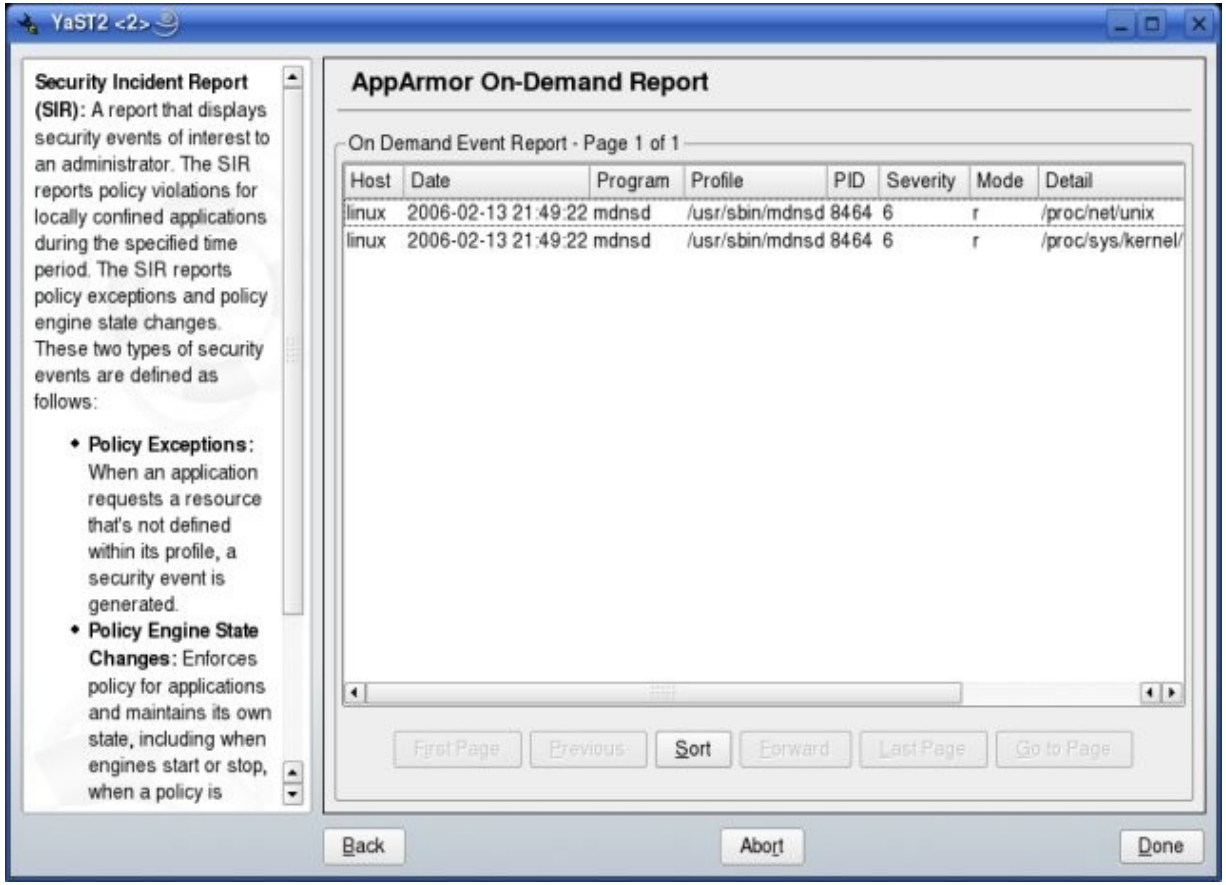
- for those of us allergic to mice :-)

Reporting and Alerting

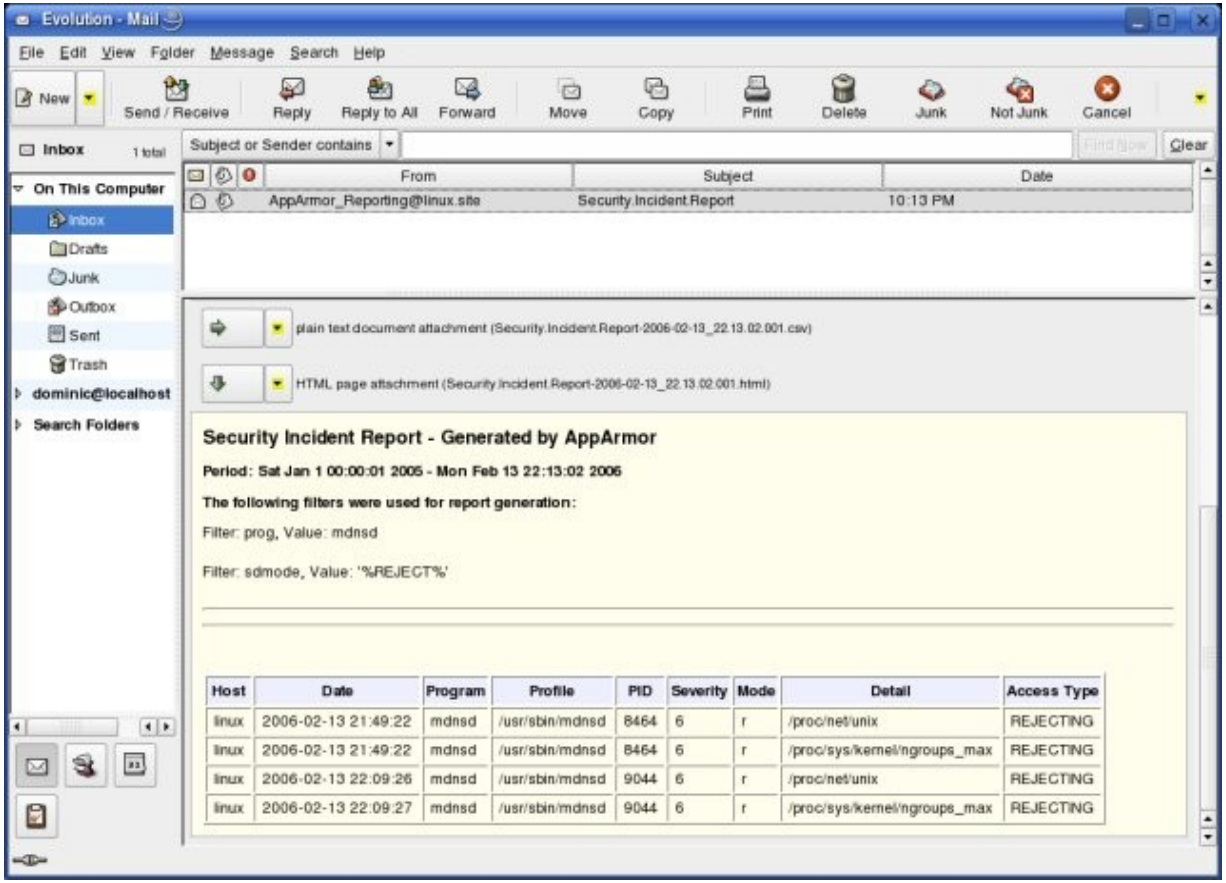
- Report on AppArmor events
- Scheduled reporting for tracking data over time
- Audit reports identify unconfined processes
- Data that can integrate into an enterprise security plan

Configuring Notification

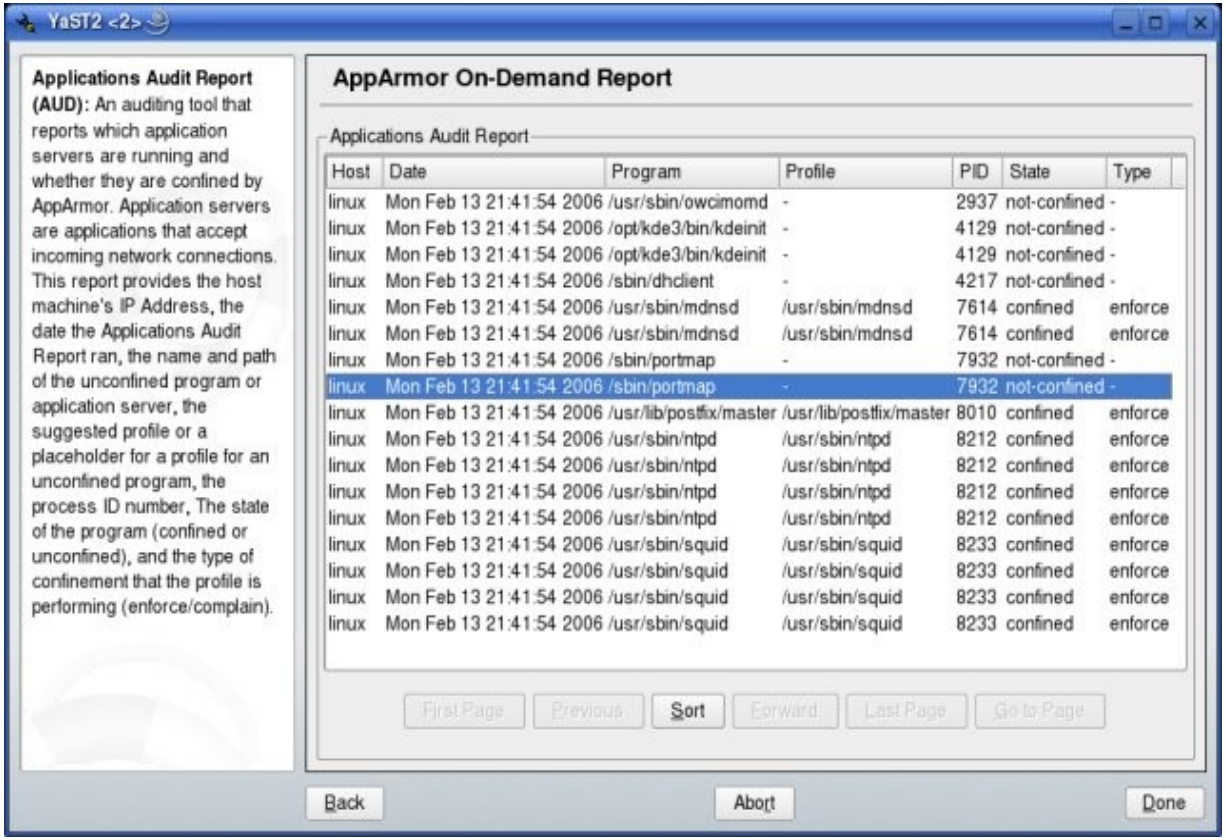
Security Incident Report



Security Incident Report – via Email



Application Audit Report



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(ftpd, `, auth_chkpwd')
type etc_ftpd_t, file_type, sysadmfile;

can_network(ftpd_t)
can_ypbind(ftpd_t)
allow ftpd_t self:unix_dgram_socket create_socket_perms;
allow ftpd_t self:unix_stream_socket create_socket_perms;
allow ftpd_t self:process {getcap setcap};
allow ftpd_t self:fifo_file rw_file_perms;

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

ifdef(`crond.te', `
system_crond_entry(ftpd_exec_t, ftpd_t)
can_exec(ftpd_t, { sbin_t shell_exec_t })
')

allow ftpd_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(ftpd_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
ifdef(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftpd_t { tmpfs_t initrc_t }:shm { read write unix_read unix_write associate };
')

# Use capabilities.
allow ftpd_t ftpd_t:capability { net_bind_service setuid setgid fowner fsetid chown sys_resource sys_chroot };

# 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(ftpd_t, var_log_t, xferlog_t, file)
# Execute /bin/ls (can comment this out for proftpd)
# also may need rules to allow tar etc...
can_exec(ftpd_t, ls_exec_t)

allow { ftpd_t initrc_t } etc_ftpd_t:file r_file_perms;
allow ftpd_t { etc_t resolv_conf_t etc_runtime_t }:file { getattr read };
allow ftpd_t proc_t:file { getattr read };


')dnl end if ftp_home_dir
```

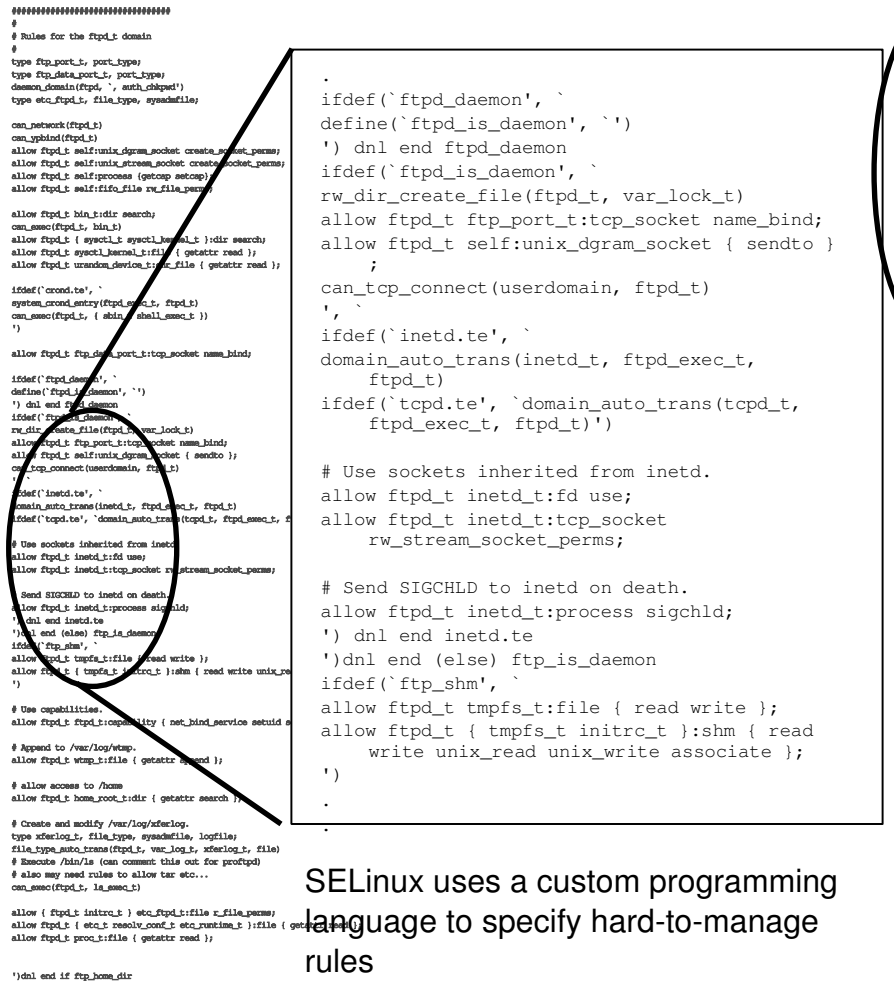
- AppArmor

```
/usr/sbin/in.ftpd {
  #include <immunix-standard/base>
  #include <immunix-standard/nameservice>
  #include <immunix-standard/authentication>
  #include <user-custom/ftpd>
  /                                 r,
  /dev/urandom                      r,
  /etc/fstab                        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,
  /usr/share/ssl/.rnd               w,
  /var/log/xferlog                  w,
  /var/run                          wr,
  /var/run/ftp.{pids,rips}-all      wr,
}
```

AppArmor profile for the *same program* is about 4x smaller

AppArmor vs. SELinux: Compare Resulting Policy

- SELinux



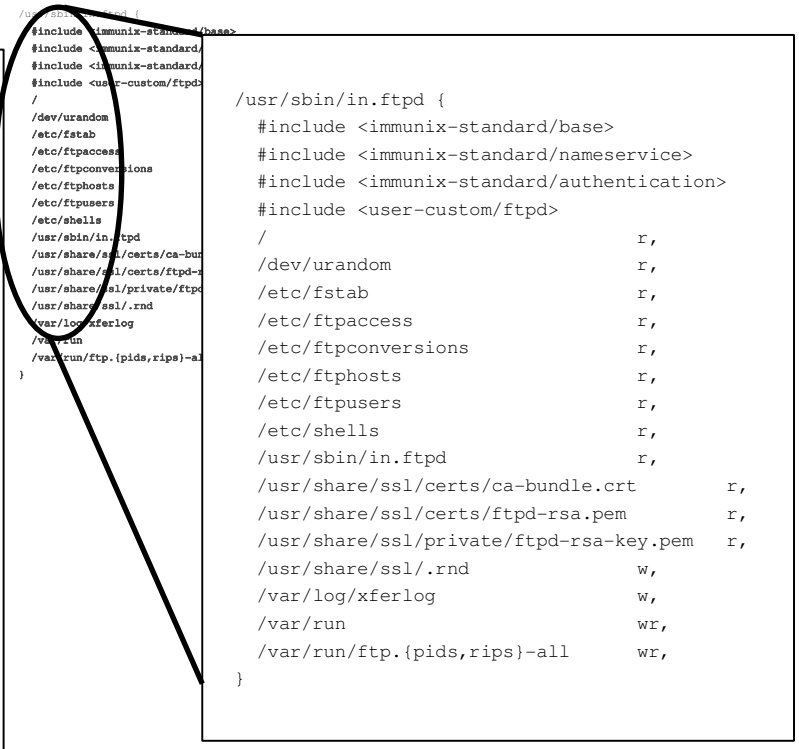
```
.
ifdef(`ftpd_daemon', `
define(`ftpd_is_daemon', `')
') dnl end ftpd_daemon
ifdef(`ftpd_is_daemon', `
rw_dir_create_file(ftpd_t, var_lock_t)
allow ftpd_t ftp_port_t:tcp_socket name_bind;
allow ftpd_t self:unix_dgram_socket { sendto }
    ;
can_tcp_connect(userdomain, ftpd_t)
', `
ifdef(`inetd.te', `
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
ifdef(`ftp_shm', `
allow ftpd_t tmpfs_t:file { read write };
allow ftpd_t { tmpfs_t initrc_t }:shm { read
    write unix_read unix_write associate };
')
.
.
```

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

- AppArmor



```
/usr/sbin/in.ftpd {
  #include <immunix-standard/base>
  #include <immunix-standard/nameservice>
  #include <immunix-standard/authentication>
  #include <user-custom/ftpd>
  /                                    r,
  /dev/urandom                         r,
  /etc/fstab                           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,
  /usr/share/ssl/.rnd                  w,
  /var/log/xferlog                     w,
  /var/run                             wr,
  /var/run/ftp.{pids,rips}-all         wr,
}
```

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

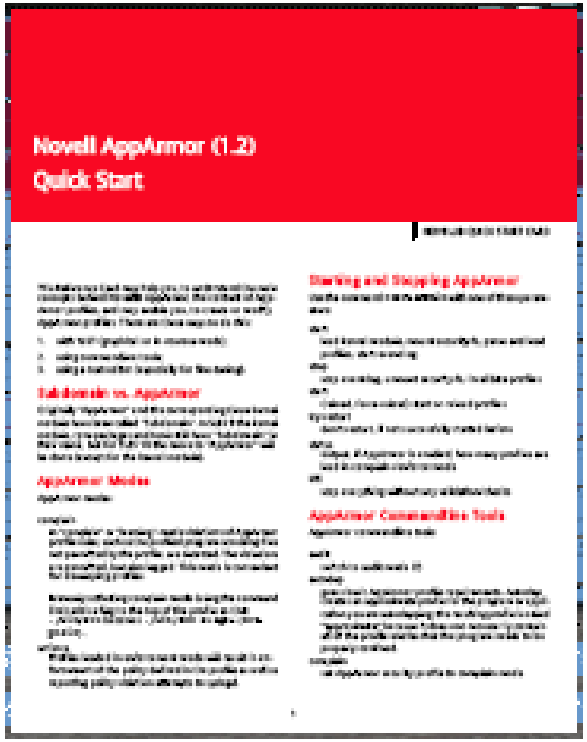
AppArmor Future Development

- **DB Armor** – access controls for database tables and files
- **Default Policy** – system level list of resources that can *only* be accesses 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®