

Securing Linux Systems with AppArmor

Crispin Cowan, PhD

Director of Software Engineering
Security Architect, SUSE Linux

Novell.[®]

AppArmor:

~~Easy-to-use Security for Ubuntu Linux~~

Crispin Cowan, PhD
Security Architect, SUSE

*What Is This 'AppArmor' Thing
and Why Should I Care?*

Novell.[®]

Agenda



Overview

A Closer Look at AppArmor

Deployment Scenarios

Demonstration of AppArmor

Competitive Positioning

AppArmor Futures

Software Security Problem

Problem: Imperfect software :-)

- Reliable software does what it is supposed to do
- *Secure* software does what it is supposed to do, *and nothing else*

Solution: only use perfect software

... slight supply problem :-)

AppArmor Solution

Enforce that applications only get to do what they are supposed to do

What means “do”?

- At ultimate detail, this is the code itself
- *But we clearly can't get that right :-)*
- Need something simpler, more abstract

Resources:

- Restrict the application to only access the OS resources it should need

What Would You Do With That?

Make a server *network secure*:

- Confine all programs with open network ports
- If all open ports lead to confined processes, then you have completely defined policy for what a network user or attacker can do
- Yet *far* from having created policy for the whole system

Is that really secure?

Hard to say

Security is semi-decidable

- You can only tell when something is *insecure*
- Hence all the Defcon talks on breaking something, and few on securing something

So lets put it to a practical test

- Put it in competition at Defcon and let people beat on it

Defcon CtF 2002-5 a la Ghettohackers



Some real-world red teaming

Play an Immunix server in the Defcon
Capture the Flag (CtF) games

Almost no holds barred:

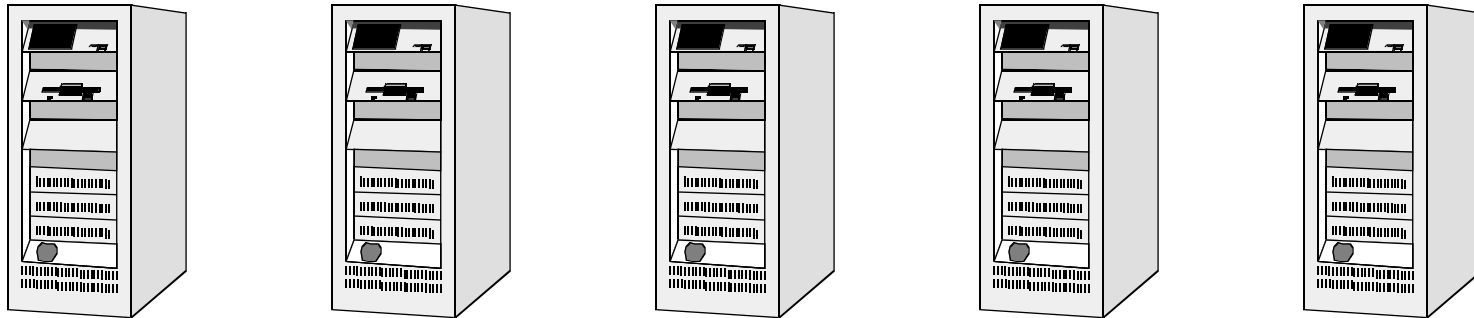
- No flooding
- No physical attacks

New gaming rig designed by the
Ghettohackers



ghettohackers.net

Basic Defcon CtF Rules

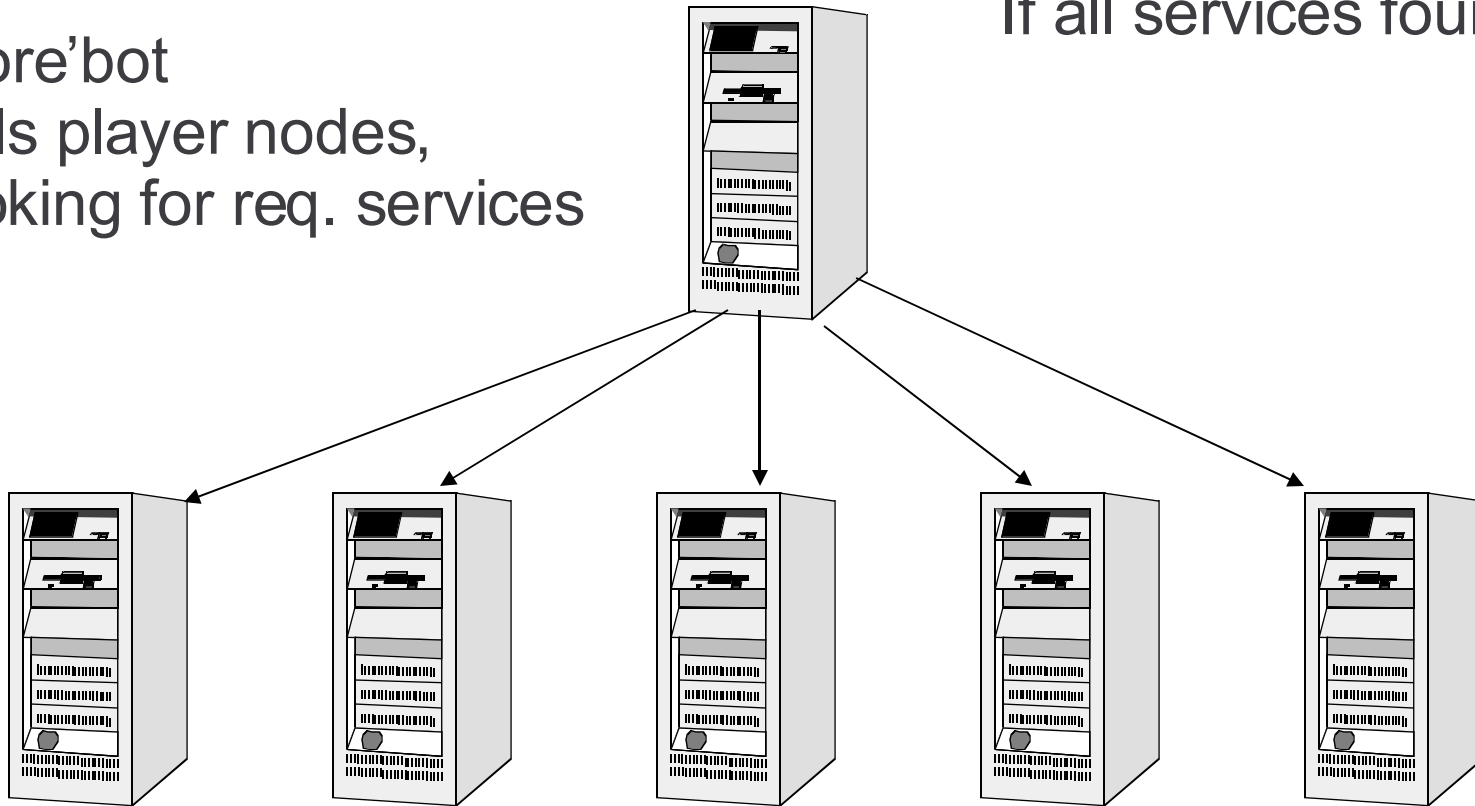


Player Nodes

Basic Defcon CtF Rules

Score'bot
Polls player nodes,
Looking for req. services

If all services found ...

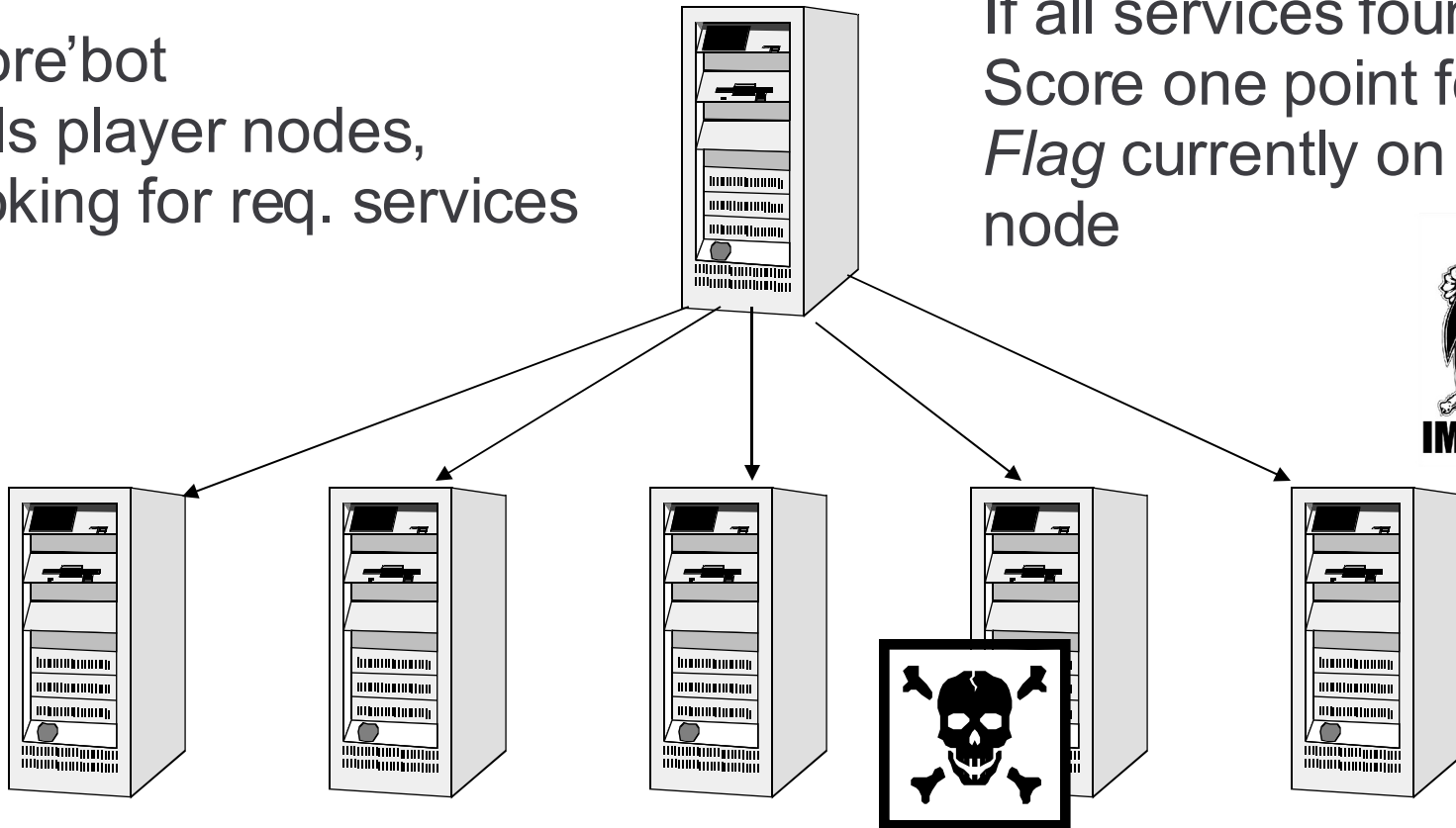


Player Nodes

Basic Defcon CtF Rules

Score'bot
Polls player nodes,
Looking for req. services

If all services found,
Score one point for the
Flag currently on that
node

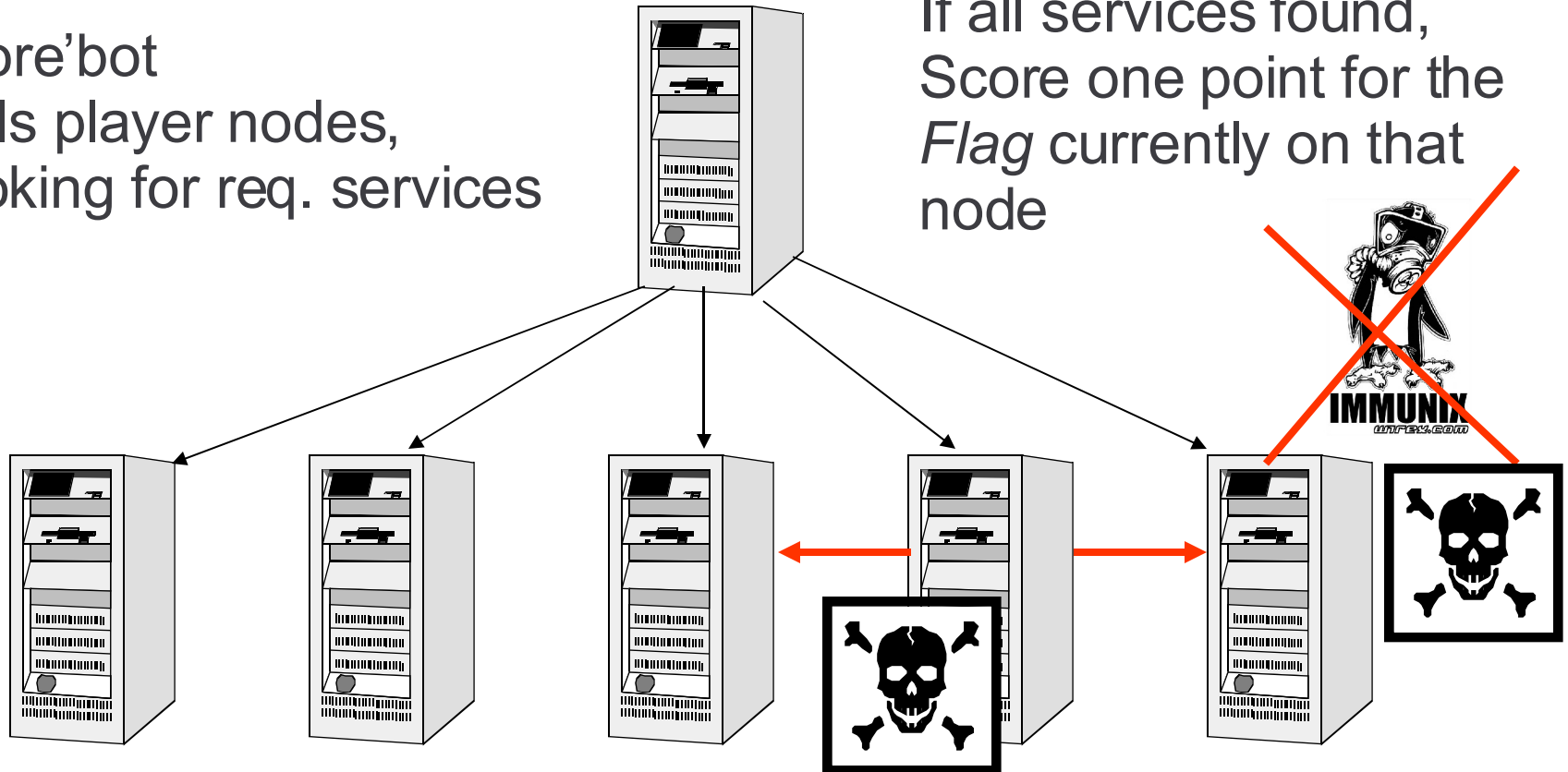


Player Nodes

Basic Defcon CtF Rules

Score'bot
Polls player nodes,
Looking for req. services

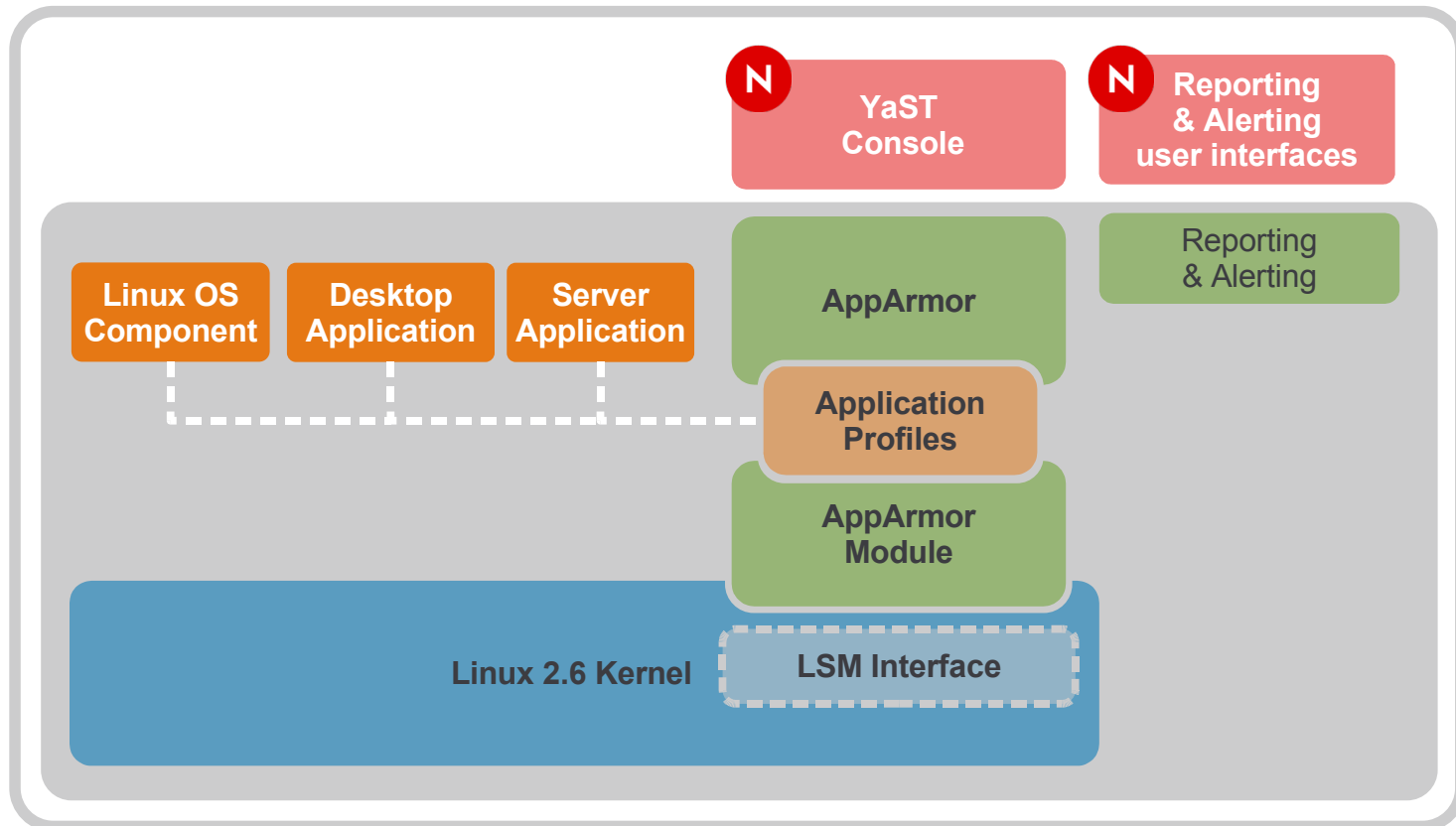
If all services found,
Score one point for the
Flag currently on that
node



The background of the slide is a solid green color with a pattern of diagonal stripes in varying shades of green, creating a sense of motion and depth. The stripes are most prominent on the right side and fade towards the left.

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

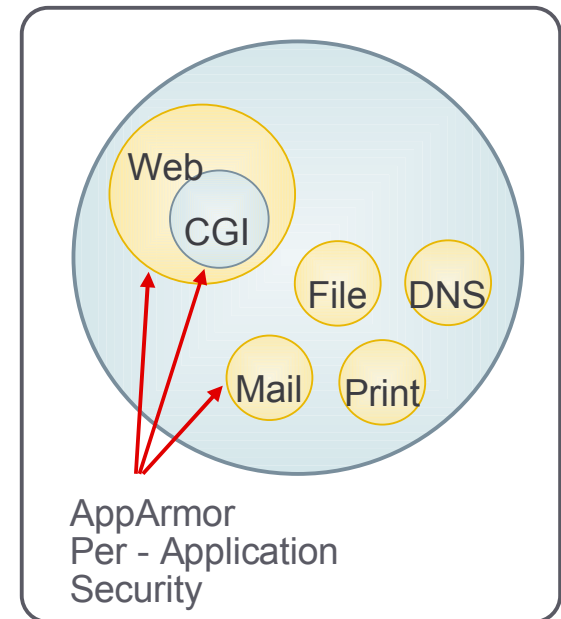
Critical Issue #2: Security Model

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

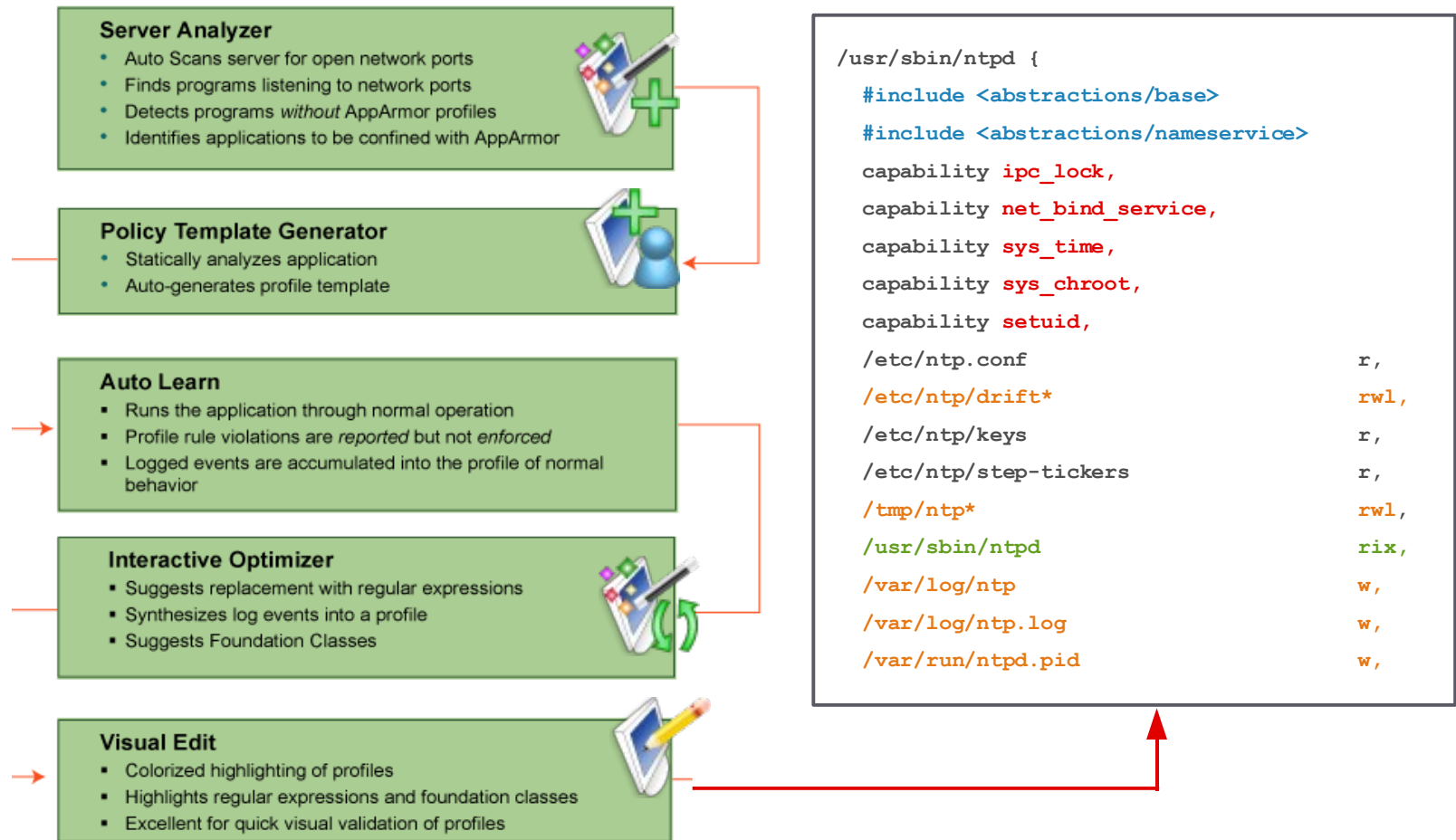
Whenever a protected program runs regardless of UID, AppArmor controls:

- 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...

The background of the slide is a solid green color with a pattern of diagonal stripes in varying shades of green, creating a sense of motion and depth. The stripes are most prominent on the right side and fade towards the left.

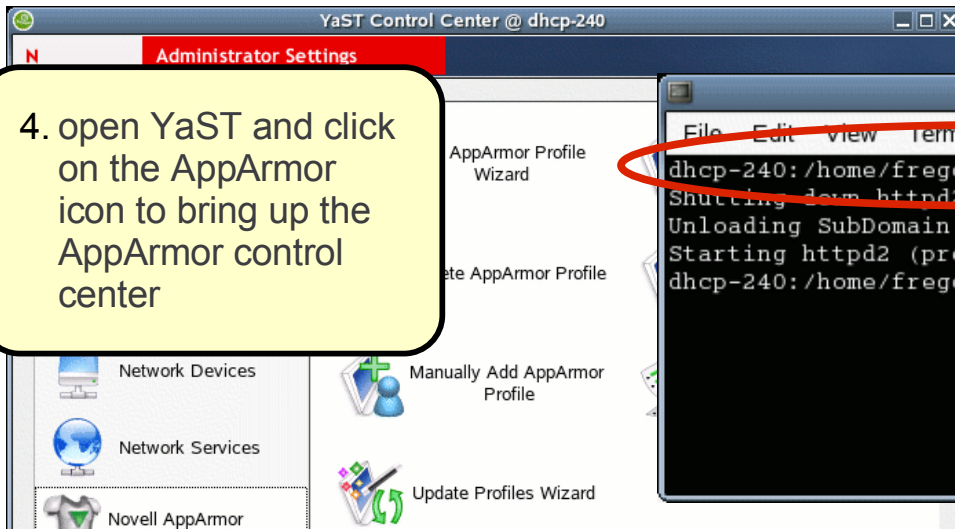
AppArmor Demo

Apache Profiling

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

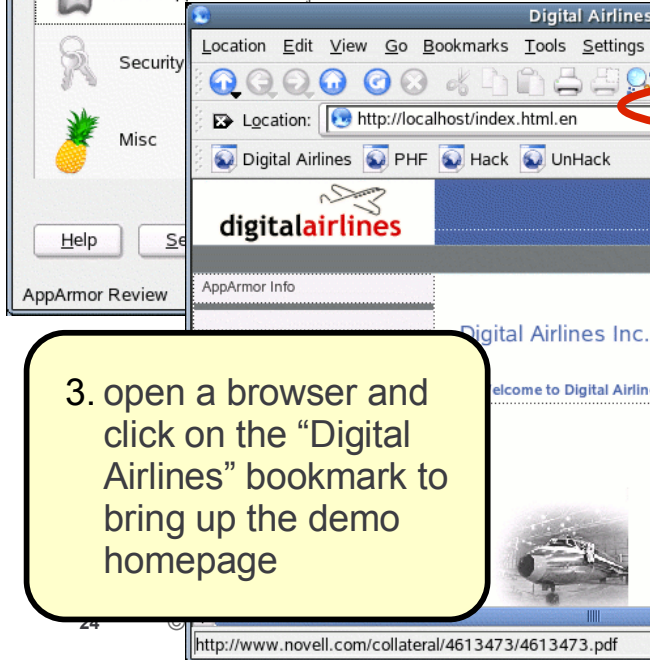
4. open YaST and click on the AppArmor icon to bring up the AppArmor control center



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

```
File Edit View Terminal Tabs Help
dhcp-240:/home/frego # demoreset.sh
Shutting down httpd2 (not running)
Unloading SubDomain profiles
Starting httpd2 (prefork)
dhcp-240:/home/frego #
```

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

```
File Edit View Terminal Tabs Help
dhcp-240:/home/frego # tail -f /var/log/messages
Nov 14 10:46:54 www su: pam_unix2: session started for user root, service
Nov 14 10:47:00 www kernel: SubDomain: LOGPROF-HINT fork pid=9260 child=28
ofile=/opt/novell/groupwise/client/bin/groupwise
ient/bin/groupwise
Nov 14 10:47:31 www kernel: SubDomain: LOGPROF
ofile=/opt/novell/groupwise/client/bin/groupwi
ient/bin/groupwise
Nov 14 10:47:35 www su: (to root) frego on /de
Nov 14 10:47:35 www su: pam_unix2: session sta
Nov 14 10:47:57 www kernel: SubDomain: PERMITT
(7596) profile /usr/sbin/fam active /usr/sbin/
Nov 14 10:47:57 www kernel: SubDomain: PERMITT
(7596) profile /usr/sbin/fam active /usr/sbin/fam,
Nov 14 10:47:57 www kernel: SubDomain: PERMITTING r access to /root/.confi
o (fam(7596) profile /usr/sbin/fam active /usr/sbin/fam)
```


The Hack

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

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

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

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

The screenshot shows a web browser window with the address bar displaying `http://www.novell.com/products/apparmor/`. The page content includes the "digitalairlines" logo, a navigation menu with items like "AppArmor Info", "Product Website", and "Frequently Asked Questions", and a main heading "Digital Airlines Inc. SERVES CRUMMY PRETZELS!!!!". A large red "X" is overlaid on a portion of the page. In the background, another browser window titled "Form for CSO PH query" is visible, showing a form with a "PH Server" field containing `ns.uiuc.edu` and a "Query Results" section displaying the command `/usr/local/bin/ph -m alias=X /bin/cp /srv/www/htdocs/warez /srv/www/htdocs/index.html.en`. A third browser window in the foreground shows the URL `http://localhost/cgi-bin/phf - Konqueror` and the same "Query Results" output.

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 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

AppArmor Review

Help Search

Software

Novell AppArmor

Security and Users

Misc

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

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

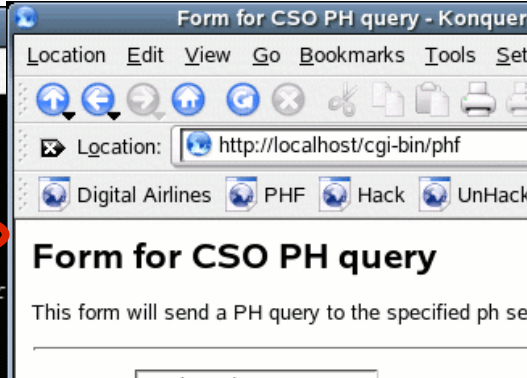
Exercising Apache

```

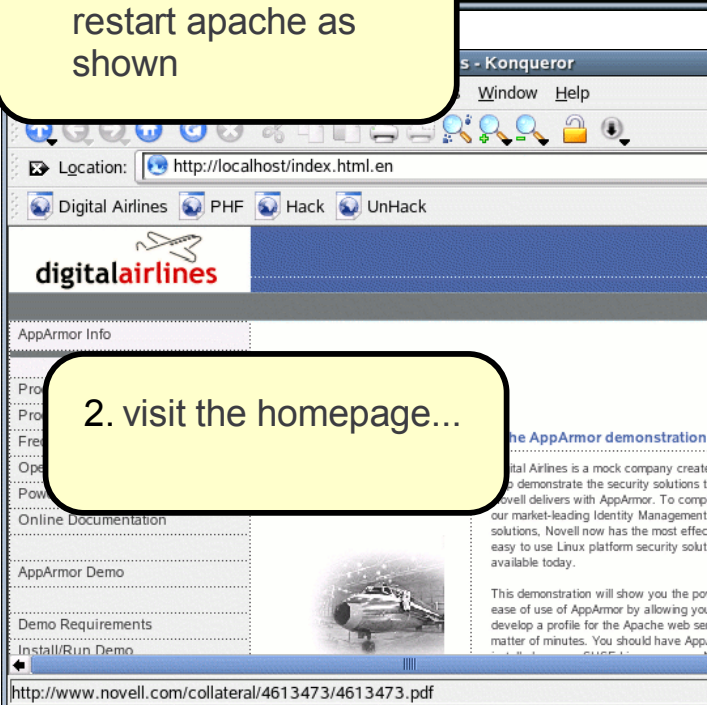
Terminal
File Edit View Terminal Tabs Help
dhcp-240:/home/freg0 # demoreset.sh
Shutting down httpd2 (not running)
Unloading SubDomain profiles
Starting httpd2 (prefork)
dhcp-240:/home/freg0 # /etc/init.d/apache2 restart
Syntax OK
Shutting down httpd2 (waiting for all children to terminate)
Starting httpd2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

```

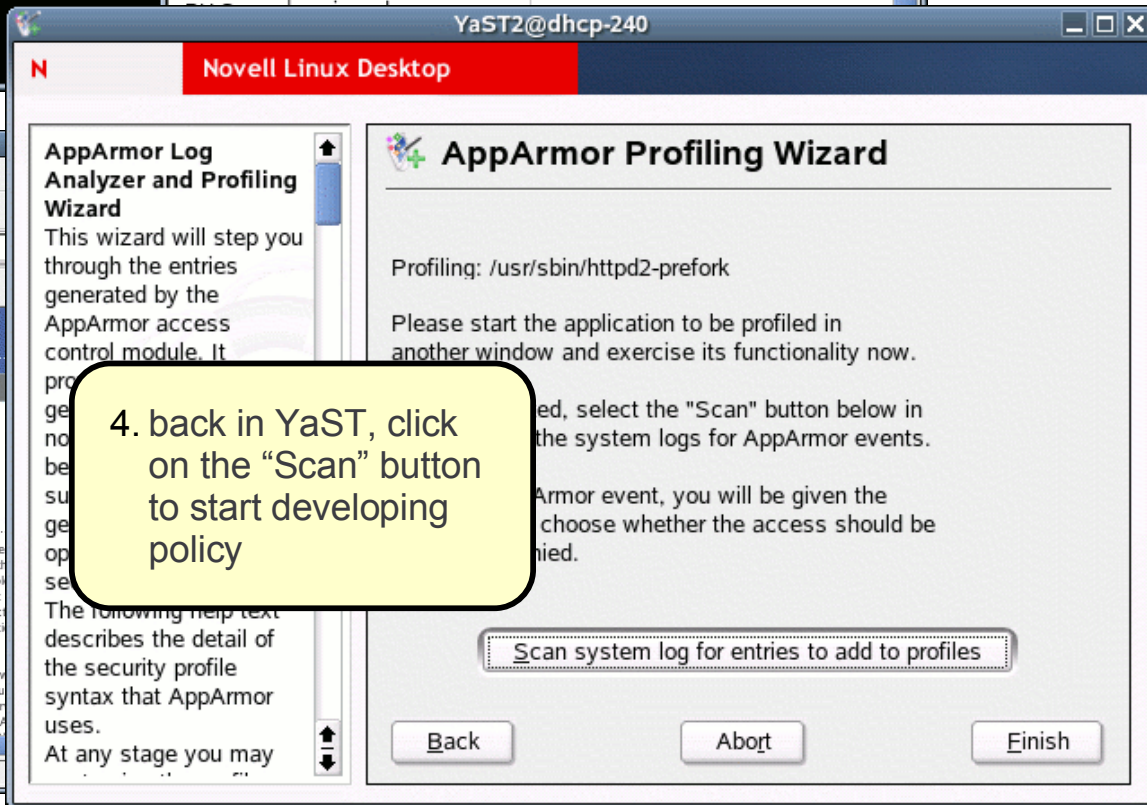
1. at the command line, restart apache as shown



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 background is 'Novell Linux Desktop'.

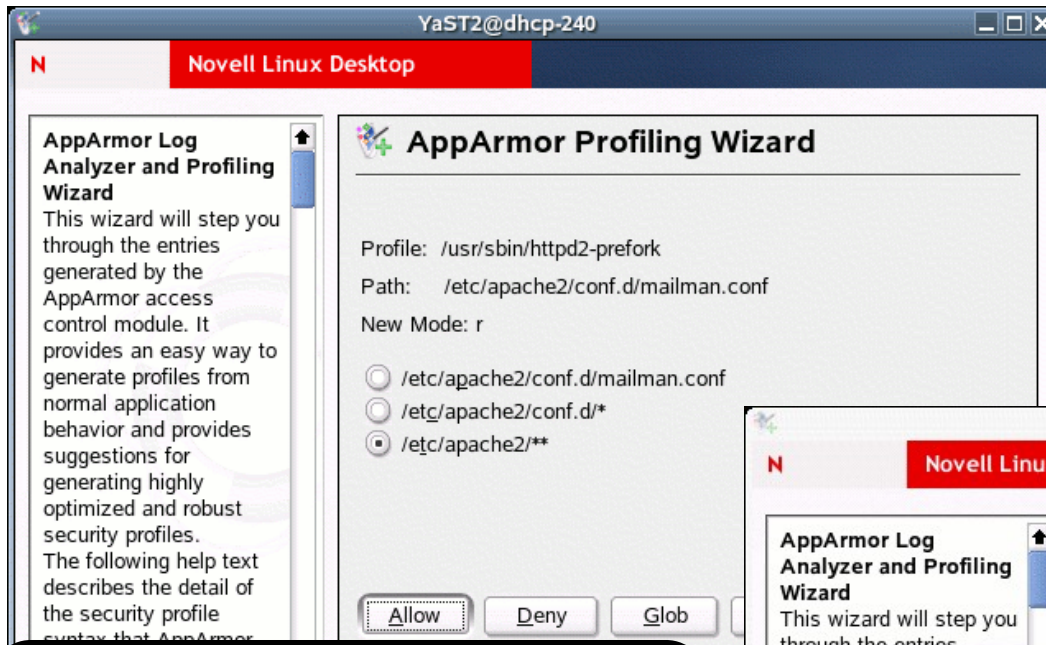
First Screenshot: The wizard is titled 'AppArmor Profiling Wizard'. It shows the profile path as '/usr/sbin/httpd2-prefork' and the application path as '/srv/www/cgi-bin/phf'. There are three radio buttons: 'Inherit', 'Profile' (which is selected), and 'Unconfined'. An 'Allow' button is visible at the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

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

Annotations:

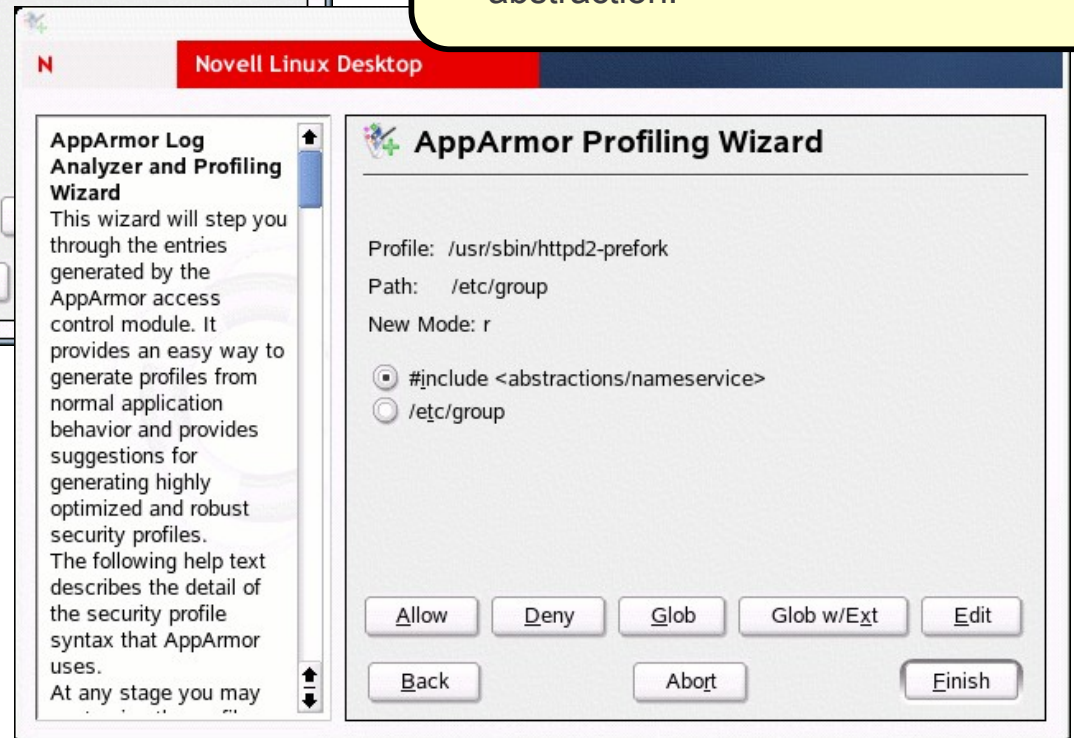
1. the Wizard asks us if the PHF app should have its own profile... we say "yes" by clicking on the "Profile" radio button, then "Allow"
2. now the Wizard notices apache needs a few POSIX capabilities. We "Allow" all of them.

Creating AppArmor Policy 2

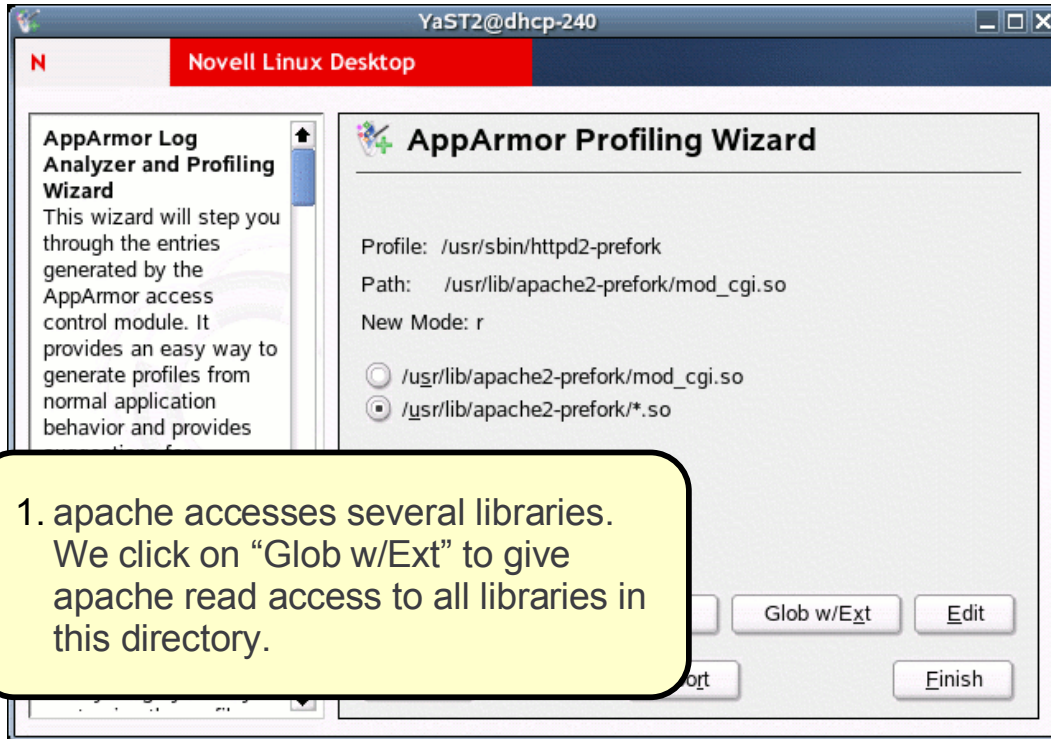


1. the Wizard asks about a file accessed by apache. We click the "Glob" button twice to allow read access to all files in the apache2 directory, then "Allow"

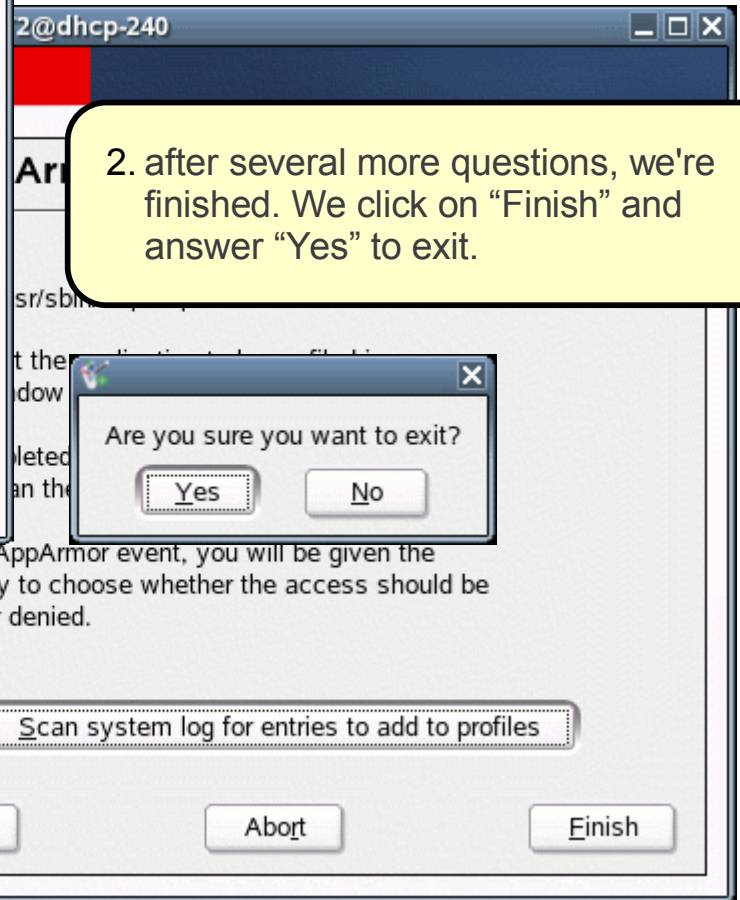
2. the Wizard notices apache needs access to /etc/group and suggests we "include" the nameservice abstraction.



Creating Apache Policy 3



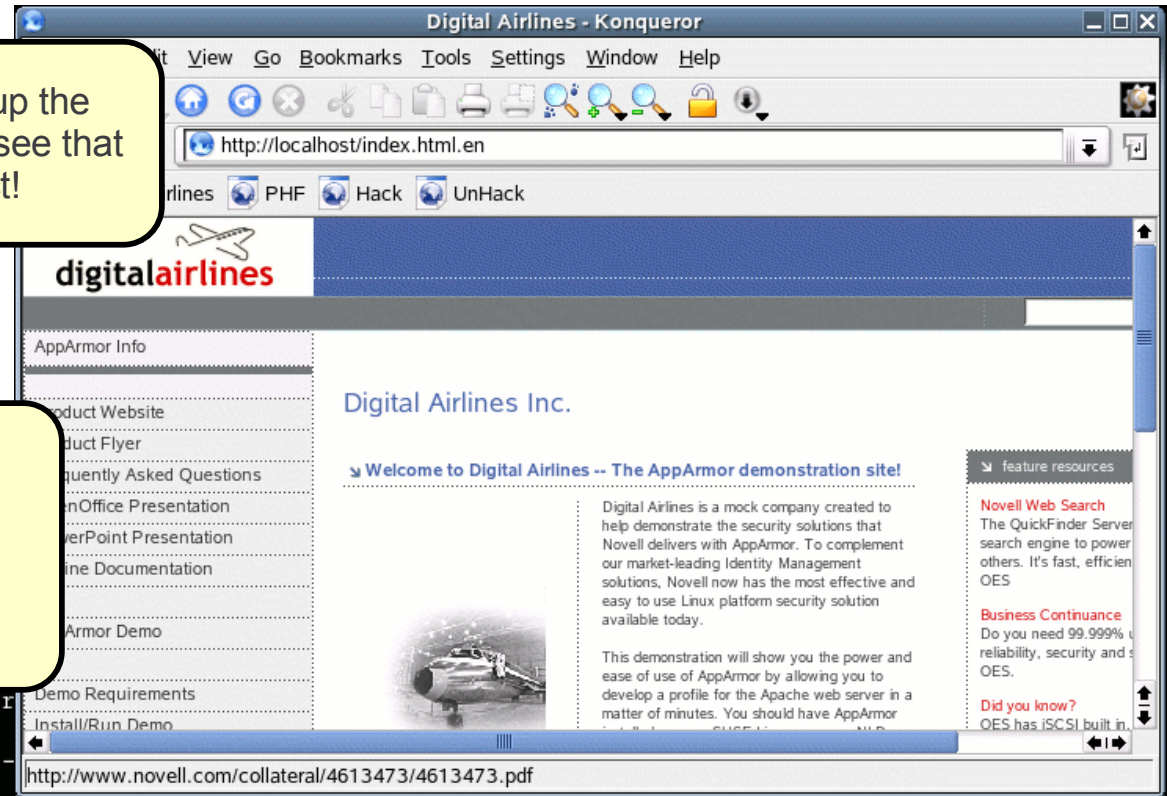
1. apache accesses several libraries. We click on "Glob w/Ext" to give apache read access to all libraries in this directory.



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.

```

cache (phf(465) profile null-complain-pr
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/3b98793f555570
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

The image shows two overlapping windows from the YaST control center. The background window is titled 'Edit Profile - Choose profile to edit' and displays a list of system profiles. The foreground window is titled 'AppArmor Profile Dialog' and shows the configuration for the selected profile, including a list of file paths and their permissions.

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.

Edit Profile - Choose profile to edit

Please make a selection from the listed profiles and press Next to edit the profile.

- /usr/lib/postfix/tismgr
- /usr/lib/postfix/trivial-rewrite
- /usr/sbin/cupsd
- /usr/sbin/fam
- /usr/sbin/httpd2-prefork**
- /usr/sbin/identd
- /usr/sbin/in.identd
- /usr/sbin/nscd
- /usr/sbin/ntpd
- /usr/sbin/postalias

Back Abort Next

AppArmor Profile Dialog

AppArmor profile for /usr/sbin/httpd2-prefork

File Name	Permissions
[+] ^DEFAULT_URI	
[+] ^HANDLING_UNTRUSTED_INPUT	
#include abstractions/base	
#include abstractions/nameservice	
CAP_NET_BIND_SERVICE	
CAP_SETGID	
CAP_SETUID	
/etc/apache2/**	r
/etc/mime.types	r
/proc/sys/kernel/ngroups_max	r
/srv/www/cgi-bin/phf	px
/srv/www/htdocs/*	r
/usr/lib/apache2-prefork/*.so	r
/usr/lib/apache2/*.so	r
/usr/sbin/httpd2-prefork	r
/var/log/apache2/access_log	w
/var/log/apache2/error_log	rw
/var/run/httpd2.pid	w

Add Entry Edit Entry Delete Entry

Back Abort Done

What Else Can I Do?

The screenshot shows the YaST Control Center window titled "YaST Control Center @ dhcp-240" with a sub-header "Administrator Settings". The interface includes a left-hand navigation pane with categories: Software, Hardware, System, Network Devices, Network Services, and Novell AppArmor. The main area displays several AppArmor management icons: "AppArmor Control Panel", "Delete AppArmor Profile", "Edit AppArmor Profile", "Manually Add AppArmor Profile", "Review AppArmor Events", and "Update Profiles Wizard".

Callout boxes provide the following descriptions for these actions:

- Enable/Disable AppArmor and configure reporting and alerting:** Points to the "AppArmor Control Panel" icon.
- Update loaded profiles based on syslogged activity since last update:** Points to the "Update Profiles Wizard" icon.
- View a report showing AppArmor events and filter by program name, date, time, etc.:** Points to the "Review AppArmor Events" icon.

At the bottom of the window, there are "Help", "Search", and "Close" buttons. The status bar at the very bottom reads "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
- Post-authentication, can transition to per-user profile

YaST Integration



Command-line Interface

There is also a command-line interface

- for those of us allergic to mice :-)

GAIM Profile

Console Tools

- Create the profile template
 - cd /opt/gnome/bin
 - genprof gaim
- Exercise GAIM
 - start, chat, stop
- Create profile entries
 - [S]can log for profile entries
 - [F]inish (GAIM profile is loaded)
- View profile
 - vim opt.gnome.bin.gaim
 - syntax on
 - set syntax=subdomain

Makes it safe to
talk to strangers

Network-secure a System



```
crispin@groo:/tmp
File Edit View Terminal Tabs Help
groo:/tmp # unconfined
2569 /usr/sbin/hpiod not confined
2869 /usr/sbin/mdnsd confined by '/usr/sbin/mdnsd (enforce)
2869 /usr/sbin/mdnsd confined by '/usr/sbin/mdnsd (enforce)
2898 /sbin/portmap not confined
2898 /sbin/portmap not confined
3192 /usr/lib/postfix/master not confined
3192 /usr/lib/postfix/master not confined
3279 /usr/sbin/sshd not confined
3296 /usr/bin/python2.5 not confined
3309 /usr/sbin/cupsd not confined
3309 /usr/sbin/cupsd not confined
3309 /usr/sbin/cupsd not confined
3536 /usr/bin/Xorg not confined
3536 /usr/bin/Xorg not confined
3900 /sbin/dhclient not confined
groo:/tmp #
```

Network-secure a System

1. Pick an unconfined service from the list
 2. Confine it the way we confined Apache and GAIM
 3. Continue until all open ports lead to AppArmor profiles
-

Result:

- There is no way onto the machine except through an AppArmor profile
- AppArmor policy completely controls network access to the machine
- Nowhere *near* having profiled all software on the machine

The background of the slide is a vibrant green color with a pattern of diagonal stripes in varying shades of green, creating a sense of motion and depth. The stripes radiate from the right side towards the left.

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

So What Happened at CtF?

2002

- Target was Red Hat, easy to port to Immunix
- Too focused on Immunix, not enough on the game
- Delayed launching any server until we had it running on Immunix
- Placed 2nd *not bad for first try*

2003: Target OpenBSD

- Target was OpenBSD, took longer to port
- SQL injection attacks, AppArmor does not stop them
- Placed 3rd *hmmm ...*

So What Happened at CtF?

2004:

- Target Windows
- A weekend is not enough time to port 5 applications from Windows to Linux under fire :-)
- Placed 4th *this trend does not look good*

2005:

- Kenshoto takes over game from Ghettohackers
- Target is now under Kenshoto's control, so no more OS defensive techniques
- CtF game now focused on binary code reverse engineering

... 2007 0tB/OtB brings focus back to OS

The background of the slide is a solid green color with a pattern of diagonal stripes in varying shades of green, creating a sense of movement and depth. The stripes are most prominent on the right side and fade towards the left.

Comparisons

Application Least Privilege for Linux

SELinux

Type Enforcement

- Assign users or programs to Domains
- Label files with Types
- Write policy in terms of which Domains can access which Types

AppArmor

Pathnames

- Name a program by path
- When it runs, it can only access the files specified by pathname
- Generalize pathnames with shell syntax wild cards

Label Splitting: SELinux

Think of SELinux as Post-it NoteTM security

- Label files & programs with colored stickers
- Policy decides which colors can play together

A single label in SELinux is an equivalence class

- All files with that label are treated identically by security policy

A **human** has to decide which files should have the same label, and which files need a different label

When you get it wrong, must split the label

- Relabel all affected files
- Revise all policies that reference that label

AppArmor

AppArmor uses explicit pathnames and regular expressions to achieve the same thing

A profile rule of `'/srv/www/htdocs/**/*.html r'` is an equivalence class, with 2 differences

- The class is evaluated at access time: new files are checked against policy
- The class is local to a single profile: don't need to re-label the world to be able to distinguish 2 files that some *other* profile treats as the same

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: 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

AppArmor vs. SELinux:

Compare Resulting Policy

■ SELinux

```
#####
#
# Rules for the ftpd 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, sysadmfile;

can_network(ftp_d)
can_ybinder(ftp_d)
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 {qotcop setcap};
allow ftpd_t self:file rw_file_perms;

allow ftpd_t bin_t:dir search;
can_exec(ftp_d, 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('cond.te',
system_cron_entry(ftp_exec_t, ftp_d)
can_exec(ftp_d, { sbin_t shell_exec_t })
)

allow ftpd_t ftp_data_port_t:tcp_socket name_bind;

ifdef('ftpd_daemon',
define('ftpd_is_daemon', '')
) dn1 end ftpd_daemon

ifdef('ftpd_is_daemon',
rw_dir_create_file(ftp_d, 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, ftp_d)
)

ifdef('inetd.te',
domain_auto_trans(inetd_t, ftp_exec_t, ftp_d)
ifdef('topd.te', 'domain_auto_trans(topd_t, ftp_exec_t, ftp_d)')

# 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;
) dn1 end inetd.te
) dn1 end (alse) ftp_is_daemon

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

# Use capabilities.
allow ftpd_t ftpd_t:capability { net_bind_service setuid setpid fowner fsuid 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(ftp_d, var_log_t, xferlog_t, file)
# Execute /bin/lis (can comment this out for profptd)
# also may need rules to allow tar etc...
can_exec(ftp_d, is_exec_t)

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

)dn1 end if ftp_home_dir
```

■ AppArmor

```
/usr/sbin/in.ftpd {
#include <immunix-standard/base>
#include <immunix-standard/namespace>
#include <immunix-standard/authentication>
#include <user-custom/ftpd>
/
/dev/urandom r,
/etc/fstab r,
/etc/ftpaccess r,
/etc/ftpconversions r,
/etc/ftpghosts 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 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;
daemon_domain(ftp_d, ' auth_chkpwd')
type etc_ftpd_t, file_type, sysadmfile;

can_network(ftp_d)
can_ybind(ftp_d)
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 { gettcp setcp };
allow ftpd_t self:file rw_file_perms;

allow ftpd_t bin_t:dir search;
can_exec(ftp_d, bin_t)
allow ftpd_t { sysext1 sysext1_key_t } :dir search;
allow ftpd_t sysext_kernel_t:file { getattr read };
allow ftpd_t urandom_device_t:file { getattr read };

ifdef('cron.te', `
systemd_cron_entry(ftp_d_exec_t, ftp_d_t)
can_exec(ftp_d_t, { shlib shlib_exec_t })
`)

allow ftpd_t ftp_data_port_t:tcp_socket name_bind;

ifdef('ftp_daemon.te', `
define(`ftp_daemon', `')
) dl end ftp_daemon
ifdef('ftp_is_daemon', `
rw_dir_create_file(ftp_d_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,
ftp_d_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;
`) dn1 end inetd.te
`) dn1 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 };

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

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

`) dn1 end if ftp_home_dir

```

```

.
ifdef(`ftpd_daemon', `
define(`ftpd_is_daemon', `')
`) dn1 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;
`) dn1 end inetd.te
`) dn1 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/filesystems>
#include <immunix-standard/nameservice>
#include <immunix-standard/authentication>
#include <user-custom/ftpd>
/
/dev/urandom
/etc/fstab
/etc/ftpaccess
/etc/ftpconversions
/etc/ftphosts
/etc/ftpusers
/etc/shells
/usr/sbin/in.ftpd
/usr/share/ssl/certs/ca-bundle.crt
/usr/share/ssl/certs/ftpd-rsa.pem
/usr/share/ssl/private/ftpd-rsa-key.pem
/usr/share/ssl/.rnd
/var/log/xferlog
/var/run/ftp.{pids,rips}-all
}

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

```

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

SELinux New GUI Tools

Advanced GUIs for enabling and disabling chunks of pre-written policies

- No help for authoring new policies

Works great for software that someone else has already profiled for you

- Problematic for your in-house and 3rd party software

AppArmor:

- It's not the GUI, it is the fundamental model

The background of the slide is a solid green color with a pattern of diagonal, overlapping stripes in various shades of green, creating a sense of motion and depth.

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

Availability

AppArmor bundled with:

- SLES10
- SLED10
- openSUSE 10.1, 10.2 ...

AppArmor is open source: GPL

- <http://opensuse.org/AppArmor>
- Mailing lists: apparmor-announce, apparmor-general, apparmor-dev
- IRC [irc.oftc.net/#apparmor](irc://irc.oftc.net/#apparmor)

AppArmor for Ubuntu

AppArmor ported to Ubuntu by Magnus Runesson

– <http://www.linuxalert.org/ubuntu/apparmor/>

AppArmor in Universe for Feisty Fawn

AppArmor going into Main for Gutsy Gibbon

User feedback on profiles is **very** helpful

AppArmor for Everyone

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

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 package

Novell.[®]