

# Securing Linux Systems with AppArmor

May, 2007

**Novell.**<sup>®</sup>

# AppArmor



*AppArmor is an open source application security tool that helps protect Linux systems from unknown security flaws*

Designed for  
ease of use

Deploy security  
policy in hours  
not days

Allow programs  
to do only what  
they are supposed  
to do and  
nothing else

- AppArmor 2.0 integrated with SLES/SLED for “out of the box” protection
- AppArmor support included with SUSE Linux Enterprise support contracts
- Consulting available to assist customer with deployment and custom policy development if needed

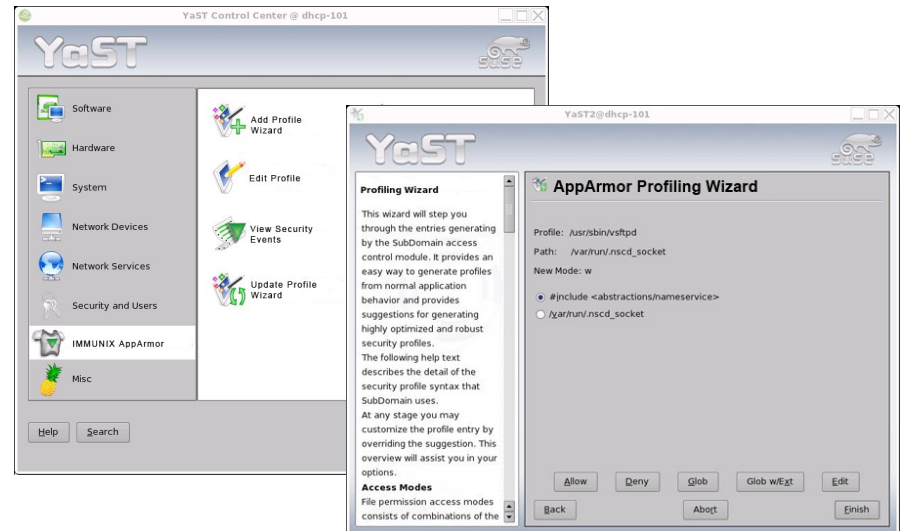
# Best Targets for AppArmor

- 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 application that matters and is exposed to attack



# Novell® AppArmor Linux Application Security

- Creates firewall around any Linux program (custom, open source, third party)
- Prevents the exploitation of application vulnerabilities
- Protects against unknown or undiscovered flaws
- Prevents unauthorized access to all system resources
- Doesn't rely on attack signature database



# Benefits of Novell® AppArmor

- Increased IT productivity
  - Empowers IT professionals to plan system updates, not just react
- Software Reliability
  - Much easier to specify what your application should do than to make bug-free software
- Peace of mind
  - Protects against unknown threats and “zero-day” attacks



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 movement and depth. The stripes are most prominent on the right side and fade towards the left.

# Technical Details

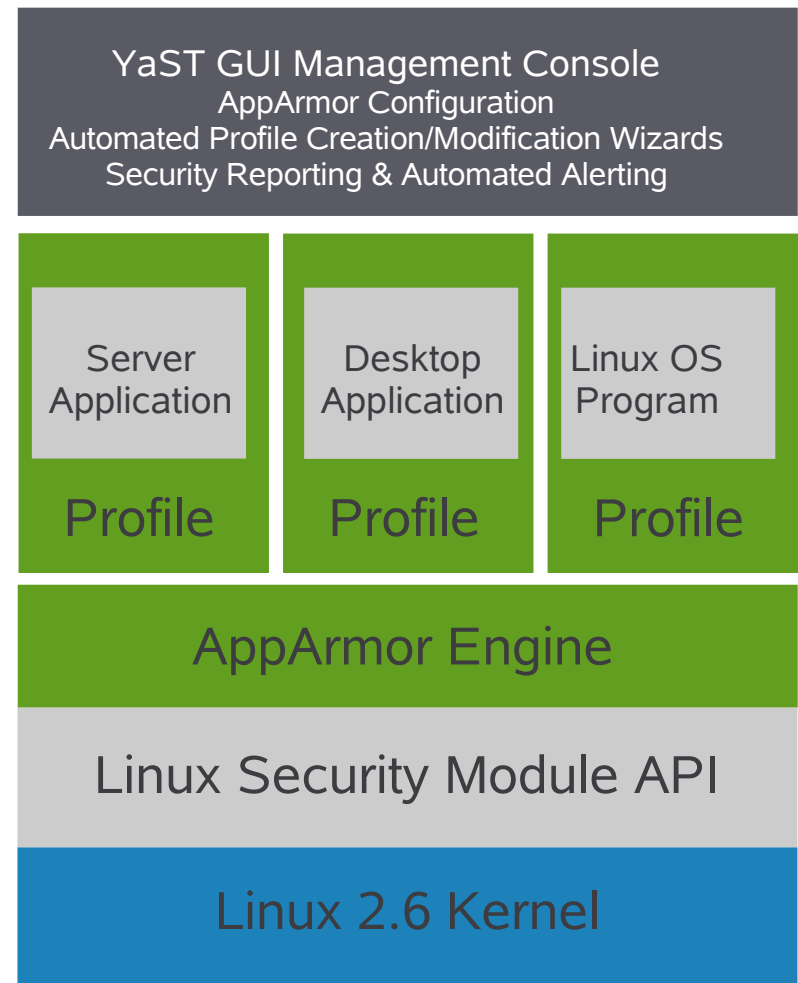
# A Closer Look at AppArmor

## 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

- Auto-scan: finds applications listening to open network ports and checks for existing profile
- Auto-generate: create profile template based on static analysis
- Auto-learn mode: automatically expands profile while running the application through normal operation
- Interactive optimizer: suggests best rules, assists in simplifying profiles



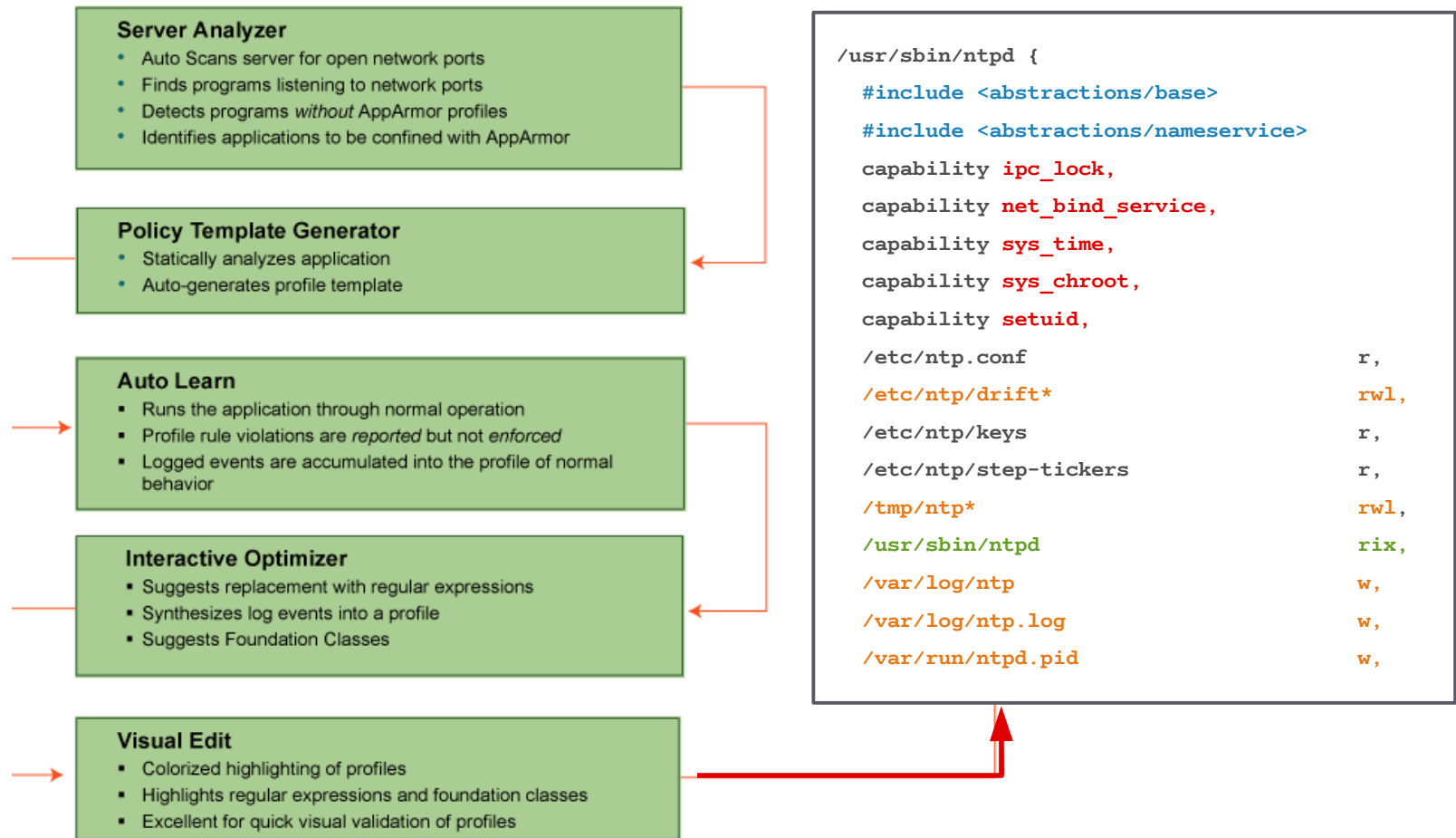
# Program-based Access Control

- 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



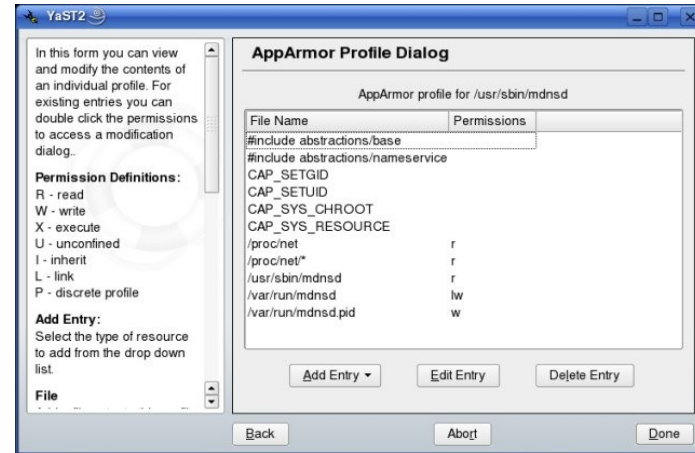
# Includes Standard Set of Profiles

- Component library includes fully-configured profiles for common operating system services and applications:
  - Apache Web server
  - Postfix mail server
  - Sendmail mail server
  - OpenSSH
  - Squid
  - ntpd
  - nscd
  - Others

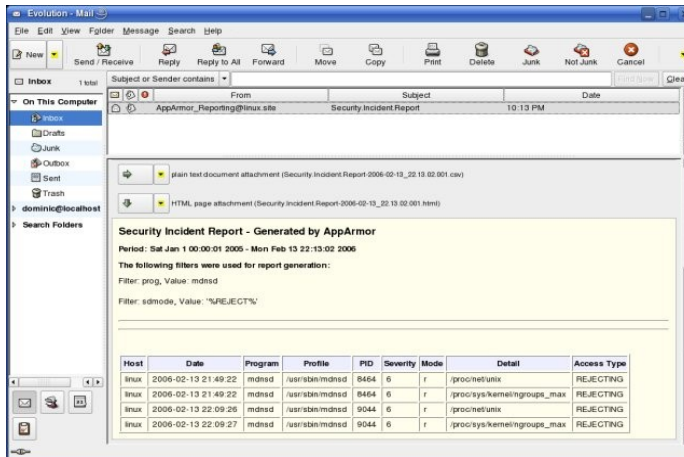
# Interacting with AppArmor via YaST



Configuration



Automated Policy Development



Reporting / Alerting



Security Event Notification



# Command-line Interface

- There is also a command-line interface



# Deployment Scenarios

# 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

The background of the slide is a solid green color with a pattern of diagonal stripes in a lighter shade of green, creating a sense of movement and depth.

# Competitive Information

# AppArmor vs. SELinux:

	AppArmor	SELinux
Type of Security	<ul style="list-style-type: none"> <li>• Pathname-based system does not require labelling or relabelling filesystem</li> <li>• When developing profiles incrementally, there is much less reason to modify other profiles, because all profiles simply refer to the pathnames they use</li> <li>• Pathnames are easy to understand and audit</li> </ul>	<ul style="list-style-type: none"> <li>• Attaches labels to all files, processes</li> <li>• Labels identify the channels of communication, so adding new profiles may require modifying existing profiles to split channels of communication, making incremental policy development difficult</li> <li>• Not all applications preserve labels</li> </ul>
Consequences	<ul style="list-style-type: none"> <li>• Automated tools in place</li> <li>• Easier integration with Novell platforms</li> </ul>	<ul style="list-style-type: none"> <li>• Hard to maintain</li> <li>• Low adoption rate</li> </ul>
Ease of Use	<ul style="list-style-type: none"> <li>• Auditable policies</li> <li>• Integrated GUI/Console toolset</li> <li>• Proficiency with 1-2 days training</li> <li>• Usability is primary goal</li> </ul>	<ul style="list-style-type: none"> <li>• Complex policy language</li> <li>• Hard to manage rules</li> <li>• Lack of integrated tools</li> <li>• Substantial training investment</li> </ul>

# AppArmor vs. SELinux: More Automated

## 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

## More Compact

### 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_t)
can_ybind(ftp_d_t)
allow ftp_d_t self:unix_dgram_socket create_socket_perms;
allow ftp_d_t self:unix_stream_socket create_socket_perms;
allow ftp_d_t self:process {getcap setcap};
allow ftp_d_t self:fifo_file rw_file_perms;

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

ifdef(`crond.te', `
system_crond_entry(ftp_d_exec_t, ftp_d_t)
can_exec(ftp_d_t, { sbin_t shell_exec_t })
')

allow ftp_d_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_t, var_lock_t)
allow ftp_d_t ftp_port_t:tcp_socket name_bind;
allow ftp_d_t self:unix_dgram_socket { sendto };
can_tcp_connect(userdomain, ftp_d_t)
', `
ifdef(`inetd.te', `
domain_auto_trans(inetd_t, ftp_d_exec_t, ftp_d_t)
ifdef(`tcpd.te', `domain_auto_trans(tcpd_t, ftp_d_exec_t,
ftp_d_t)')
# Use sockets inherited from inetd.
allow ftp_d_t inetd_t:fd use;
allow ftp_d_t inetd_t:tcp_socket rw_stream_socket_perms;

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

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

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

# allow access to /home
allow ftp_d_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 proftpd)
# also may need rules to allow tar etc...
can_exec(ftp_d_t, ls_exec_t)

allow { ftp_d_t initrc_t } etc_ftpd_t:file r_file_perms;
allow ftp_d_t { etc_t resolv_conf_t etc_runtime_t }:file
{ getattr read };
allow ftp_d_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/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/ftpd.(pids,rips)-all wr,
}
```

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

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 of the slide.

Availability

# Availability

- AppArmor bundled with all SUSE<sup>®</sup> Linux Enterprise and openSUSE<sup>®</sup> products
- Important new AppArmor features in SUSE Linux Enterprise SP1 are:
  - 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)
- AppArmor is open source: GPL
  - <http://opensuse.org/AppArmor>
  - Mailing lists: apparmor-announce, apparmor-general, apparmor-dev

# For More Information



<http://www.novell.com/apparmor>

<http://www.opensuse.org/Apparmor>

**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. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. 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. The development, release, and timing of features or functionality described for Novell products remains at the sole discretion of Novell. 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.

