AppArmor

Easy-to-use Security for Ubuntu Servers
or
What is this “AppArmor” Thing and why should I care?

Thomas Merkel
tm@core.io

2015-10-05

Agenda

- Software Security Problem
- AppArmor Solution
- Architecture
- Profile
- Workflow
- Target
- SELinux vs. AppArmor
- Questions

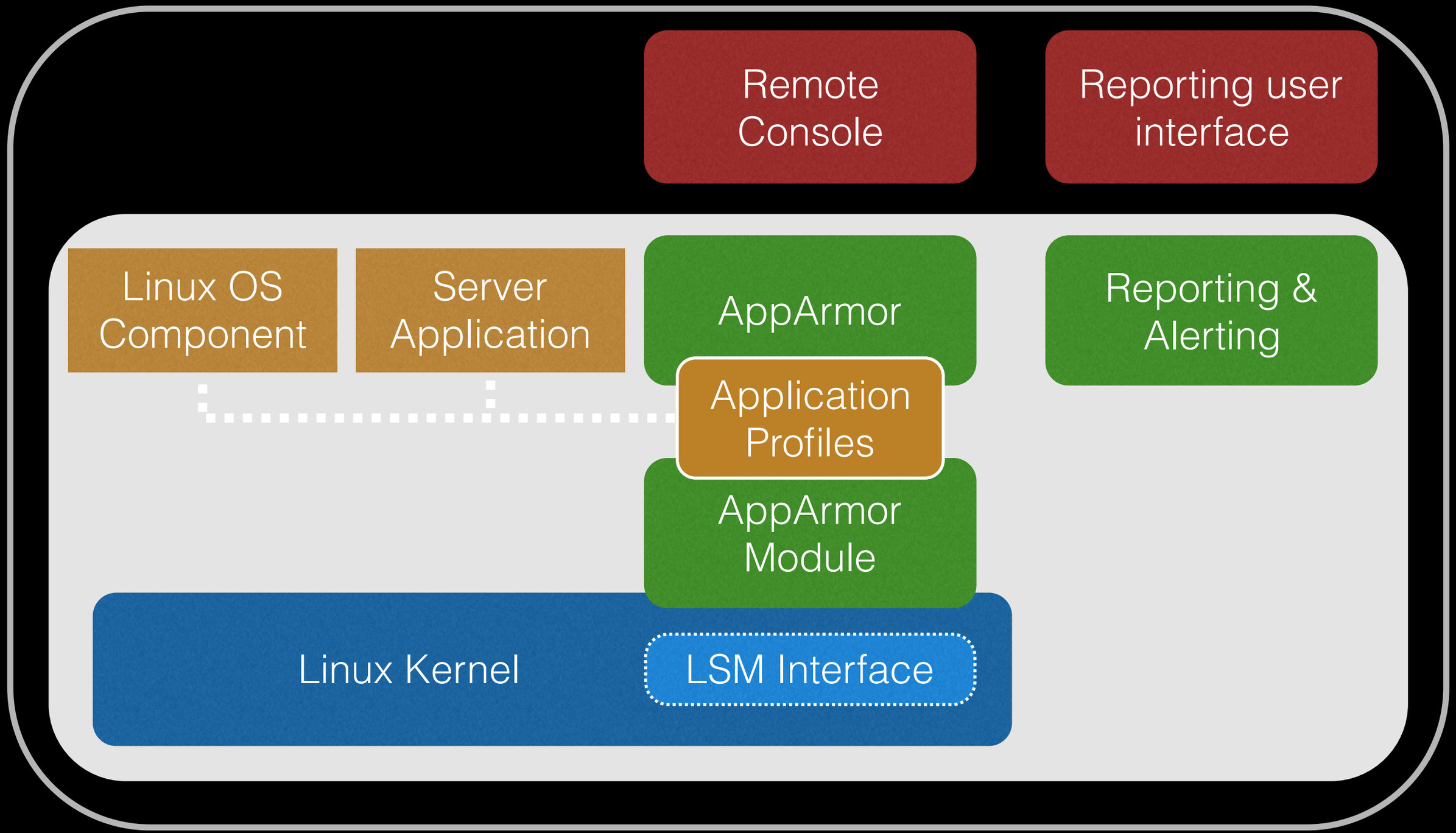
Software Security Problem

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

AppArmor Architecture



AppArmor Security Profile

- Whenever a protected program runs regardless of UID, AppArmor controls:
  - The POSIX capabilities it can have
  - The directory/files it can read, write and execute

```
#include <tunables/global>
#include <tunables/ntpd>
/usr/sbin/ntpd {

  capability ipc_lock,
  capability net_bind_service,
  capability setgid,
  capability setuid,

  network inet dgram,
  network inet6 dgram,
  network inet stream,
  network inet6 stream,

  @{PROC}/net/if_inet6 r,
  @{PROC}/*/net/if_inet6 r,
  @{NTPD_DEVICE} rw,

  /{,s}bin/      r,
  /usr/{,s}bin/  r,

  /etc/ntp.conf r,

  /etc/ntp.drift rwl,
```

Automated Workflow

Server Analyser

- Auto Scan server for open network ports
- Finds programs listening to the network ports
- Detect programs *without* AppArmor profiles
- Idenitifies applications to be confined with AppArmor

Policy Template Generator

- Statically analyses application
- Auto-generates profile template

Auto Learn

- Runs the application through normal operation
- Profile rule violations are *reported* but *not enforced*
- Logged events are accumulated into the profile

Automated Workflow

Interactive Optimiser

- Suggests replacement with regular expressions
- Synthesises log events into a profile
- Suggests foundation classes

Visual Edit

- Colorised highlighting of profile
- Highlights regular expressions and foundation classes
- Excellent for quick visual validation of profile

Automated Workflow

```
#include <tunables/global>
#include <tunables/ntpd>
/usr/sbin/ntpd {
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/user-tmp>

  capability ipc_lock,
  capability net_bind_service,
  capability setgid,
  capability setuid,
  capability sys_chroot,
  capability sys_resource,
  capability sys_time,
  capability sys_nice,

  network inet dgram,
  network inet6 dgram,
  network inet stream,
  network inet6 stream,

  @{PROC}/net/if_inet6 r,
  @{PROC}/*/net/if_inet6 r,
  @{NTPD_DEVICE} rw,

  /{,s}bin/      r,
  /usr/{,s}bin/  r,
  /usr/sbin/ntpd rmix,

  /etc/ntp.conf r,
  /etc/ntp.conf.dhcp r,
  /etc/ntpd.conf r,
  /etc/ntpd.conf.tmp r,
  /var/lib/ntp/ntp.conf.dhcp r,

  /etc/ntp.keys r,
  /etc/ntp/** r,
```

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

Generate Profile

- Create the profile template
  - `cd /opt/bin/`
  - `aa-genprof myApp`
- Execute
  - start, run, stop the application
- Create profile entries
  - `[S]can log for profile entries`
  - `[F]inish (myApp profile is loaded)`
- View profile
  - Check your profile “opt.bin.myApp”

Best Targets for AppArmor

- Any Company whose networked servers are running mission critical applications
- Any organisation with a high cost associated with compromised data
- Any organisation faced with regulatory compliance
- **Any Linux application is exposed to attack and that matters :-)**

Best Targets for AppArmor

- **Networked servers**
  - Isolate all program interacting with the outside world
  - Auto-scan tool finds applications that should be profiled
  - Profiles represent your total exposure - audit-able policy
- **Business applications**
  - Complex, not easily audit-able for security
  - May be closed source
  - Prevents attacking on one component from spreading to other components of systems

SELinux vs. AppArmor

SELinux

Type Enforcement

- Assign users or programs to domains
- Label files with types
- Write policy in terms of which domains ca access which types

AppArmor

Pathnames

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

SELinux

- Think of SELinux as Post-It Note security
  - Label files & programs with coloured stickers
  - Policy decides which colours 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

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

Questions?

So long thanks for all the fish :-)