AppArmor Update

2014 Linux Security Summit

Presentation by
Tyler Hicks
tyhicks@canonical.com
www.canonical.com
August 2014

What's driving AppArmor development at Canonical?

- Securing container workloads with the ability to place the container in its own AppArmor policy namespace

- Application isolation for Ubuntu phone and tablet images

  wiki.ubuntu.com/SecurityTeam/Specifications/ApplicationConfinement

Recent improvements

Confining the container and the processes inside

- Rework of AppArmor labeling is underway to gain the ability to cache more than one label on an object
  - Also allows for better caching of permissions to avoid some path lookups

- Compound labels and policy namespaces allow containers to be confined as a whole by one AppArmor profile in the “host” and then an entirely new set of AppArmor profiles can be used to confine the individual processes inside the container
  - Potential for users to load their own AppArmor policy inside their user namespaces

- For more information, see the presentation from LSS 2013

  selinuxproject.org/~jmorris/lss2013_slides/jj_apparmor-labeling-2013.odp

Fine-grained mediation in dbus-daemon

- Sending and receiving of messages can be filtered on bus name, path, interface, member name, peer name, and peer label

```
    dbus (receive, send)
         bus=session
         path=/com/ubuntu/connectivity1/NetworkingStatus,
```

- Bind rules can enforce a specific well-known name and a bus name

```
    dbus bind name=org.gnome.keyring,
```

- Eavesdropping rules can specify the bus name

```
    dbus eavesdrop bus=system,
```

- dbus-daemon patches have been submitted upstream

  https://bugs.freedesktop.org/show_bug.cgi?id=75113

Mediation of signals and ptrace

- Signal mediation allows for rules to specify the signal(s) and the peer

```
# Send SIGHUP and SIGINT to any process
signal (send) set=(hup, int),
# Allow libvirtd to send us signals
signal (receive) peer=/usr/sbin/libvirtd,
```

- Ptrace trace and tracedby permissions govern ptrace(2)
- Ptrace read and readby govern certain /proc accesses, kcmp(2), futexes (get_robust_list(2)) and perf trace events

```
# Allow unconfined processes (eg, a debugger) to ptrace us
ptrace (readby, tracedby) peer=unconfined,
```

penguindroppings.wordpress.com/2014/06/06/application-isolation-with-apparmor-part-iv/

Other notable changes

- Userspace utilities were rewritten from Perl to Python3
  - aa-status, aa-enforce, aa-genprof, etc.
  - Google Summer of Code project

- systemd unit config file support for specifying the name of an AppArmor profile to switch to when starting a new process

- Parser improvements
  - Minimization changes provided an average of 40% to 50% improvement in compilation times
  - Differential compression provides a 50% smaller binary policy and a 30% to 40% improvement in compilation times for large profiles
  - Atomic loading of cache files that contain multiple profiles decreases load times

Looking forward

Network mediation

- UNIX domain sockets will soon have fine-grained mediation
  - Can specify socket type, path, and socket label

    ```
    # Allow communication with D-Bus session bus
    unix (connect, send, receive) type=stream path=”@/tmp/dbus-*”,
    ```

- Still have course controls available for any protocol family that doesn't yet have fine-grained mediation

- Additional address families will receive fine-grained mediation
  - INET
  - INET6
  - NETLINK

Smarter binary policy caching

- Multiple, versioned binary policy caches
  - Each policy cache will be tied to a unique feature set advertised by AppArmor in securityfs
  - Supports multiple policy versions so that hardware enablement kernels can be used on older releases
- Ubuntu will soon generate the policy cache during kernel install instead of doing it at boot
- Some cached policies for the Ubuntu phone images are already being generated server side to avoid having to compile them on the phone

Additional important pieces

- Provide library interface for policy compiler and loader
  - Needed for full systemd support

- More policy compiler performance enhancements

- Expose a wider permission set to the policy language
  - For example, the write permission currently expands to setattr, create, delete, chmod, chown, open, and delete but it may be useful to expose more of these permissions

- Finish labeling and profile stacking work to provide full container confinement

Questions please

Thank you

Tyler Hicks

tyhicks@canonical.com

www.canonical.com