Access Control in OpenXT: XSM/Flask, SELinux, v4v firewall, rpc-proxy, and more!

Stephen Smalley
Trusted Systems Research
National Security Agency

OpenXT Summit

June 7, 2016

Topics

- Unifying access control in OpenXT
- Improving the use of XSM/Flask
- Leveraging SELinux more fully
- Modular policy and policy management
- rpc-proxy and v4v firewall: Current State and Future Directions

Unifying access control in OpenXT

- Makes rich use of a variety of existing access control mechanisms, e.g. XSM/Flask, SELinux, XenStore perms.
- Introduces its own unique access control mechanisms, e.g. v4v firewall, rpc-proxy, service-specific logic.
- Each does something useful on its own, but differ in semantics, policy languages, flexibility / configurability, and trust / assurance implications.
- This poses problems for analysis and validation.
- It also affects TCB size and dom0 decomposition.
- How can we migrate toward a more unified architecture?

D-BUS over v4v: Current Access Checks

- SELinux (SrcDom): Is SrcProc allowed to access the v4v device?
- XSM/Flask: Is SrcDom allowed v4v send to DstDom?
- v4v firewall: Is (SrcDom, SrcPort) allowed v4v send to (DstDom, DstPort)?
- SELinux (DstDom): Is rpc-proxy allowed to access the v4v device?
- rpc-proxy (DstDom): Is SrcDom allowed access to the specific D-BUS destination / interface / method?
- dbusd SELinux (DstDom): Is rpc-proxy allowed D-BUS send_msg to ServiceProc?
- ServiceProc (DstDom): May further limit access based on SrcDom (domid obtained from Xen, conveyed by rpc-proxy to dbusd to ServiceProc).
- (Several other SELinux checks in DstDom omitted for brevity)

Direct v4v: Current Access Checks

- SELinux (SrcDom): Is SrcProc allowed to access the v4v device?
- XSM/Flask: Is SrcDom allowed v4v send to DstDom?
- v4v firewall: Is (SrcDom, SrcPort) allowed v4v send to (DstDom, DstPort)?
- SELinux (DstDom): Is ServiceProc allowed to access the v4v device?
- ServiceProc (DstDom): May further limit access based on SrcDom (domid obtained directly from Xen).

Problems for Analysis/Validation

- How can we tell if such a message is allowed?
  - XSM/Flask and SELinux: Trivial - just run sesearch or other policy analysis tools on the final policy file. Centralized policy, shared security model, composable.
  - v4v and rpc firewalls: Requires manual analysis of per-VM config + any global config, knowledge of userspace program behavior (e.g. xenmgr automatic inverse rules, rpc-proxy logic). No common abstraction/model, not composable with each other or SELinux.
  - Service-specific logic: Requires manual analysis of program logic (e.g. dbd and icbinn restrictions on path).

Problems for Trust / Assurance

- Different abstractions -> possible inconsistencies
  - Labels, domids, uuids, vm type, vm name, …
- Extra trust obligations
  - Relies on rpc-proxy and dbusd for client domid.
  - Services may further rely on XenStore for domid → uuid lookup.
- Hardcoded policy
  - Dom0 assumptions embedded in code and architecture
  - Intermingling of enforcement and policy logic
- Access control for a service depends on IPC mechanism
  - D-BUS over v4v vs direct v4v

Gaps in Current Access Checks

- Almost no process-based control.
  - Could use SELinux within SrcDom and DstDom to control v4v bind/send by individual processes.
  - Could convey SELinux security context of SrcProc to DstDom for use in access checks there.
- No use of Flask label in any DstDom checks.
  - Only domid/UUID-based restrictions.
  - Could leverage it in rpc-proxy, service access controls.

Access Control in SVP

- Discrete components enforce MAC focused at proper abstraction level
  - Components leverage MAC from others
  - Union embodies system MAC policy and helps guarantee important security properties
- XSM/Flask for VM isolation and controlled interaction
- VMs enforce MAC over objects/services they provide
  - Leverage secure IVC
- SELinux MAC for intra-VM protections

Flask Security Architecture

- Flask is not just a security module for Xen (or Linux).
- It is a general access control architecture suitable for a variety of subsystems.
- Already used in various Linux and Android userspace services to provide access control.
- Provides common architecture, abstractions, interfaces, infrastructure support.

Toward a Unified Architecture

- Possible steps, ordered from near term to longer term:
  - Investigate replacing v4v firewall's mandatory uses entirely with XSM/Flask
  - Investigate reducing trust in rpc-proxy through per-client instances and use of dbus SELinux policy
  - Extend v4v access control to process granularity via SELinux.
  - Enable flexible service access control
    - Provide support for obtaining Flask peer labels for v4v (as with event channels).
    - Convey Flask peer labels to service processes (rpc-proxy, dbusd).
    - Convert hardcoded dom0 and other hardcoded logic to Flask access control checks.
  - Integrate flexible MAC into XenStore as well.
  - Investigate eliminating use of D-BUS entirely, removing rpc-proxy and dbusd from TCB.

Improving the use of XSM/Flask

- Current State
  - Enabled and enforcing in OpenXT
  - Separate Flask domain for each VM type
  - Controls all inter-VM comms (IVC), including v4v
- How can OpenXT advance its usage of XSM/Flask to meet platform security goals and better support downstream usage of OpenXT?
  - Leveraging for (further) dom0 decomposition
  - Controlling which domains can use device passthrough and for which devices
  - Creating domains unreadable by dom0
  - Creating fully isolated domains (MCS vs type-self approaches)
  - Enforcing information flow among domains (e.g. making nilfvm unbypassable)

Leveraging SELinux more fully

- Current State
  - Enabled and enforcing in dom0 and network driver VM (ndvm)
  - Separate SELinux domain for each service, confined user logins
  - Confine and isolate qemu instances.
- How can OpenXT leverage SELinux more fully to meet platform goals?
  - Tailoring policy for each VM
  - Confine all processes to least privilege (unconfined removal)
  - Enforcing TCB protection and hardening goals, e.g. W^X, kernel protection, no-ptrace, …
  - Enforcing D-BUS restrictions (to partly obsolete rpc-proxy)
  - Enable in User Interface VM (uivm).
  - Per-process v4v access controls
  - Convey local process security contexts to peer domains for access control.

Modular policy and policy management

- Current State
  - XSM/Flask and SELinux policies each built from single source package.
  - Modular SELinux policy not being leveraged to support per-package or per-VM policies even in source.
  - Measured launch and default policies do not permit policy changes without disabling read-only root fs beforehand and re-sealing afterward.
- What capabilities are needed at build time and runtime for OpenXT?
- How can we achieve them in a way that preserves platform security properties?

v4v firewall and rpc-proxy

- Current State
  - Used in OpenXT to control direct v4v and D-BUS over v4v comms
  - Configurations being audited, tightened to remove legacy/unnecessary rules
- How can these mechanisms be improved in OpenXT?
  - V4v firewall: introduce connection state in hypervisor to eliminate need for inverse rules
  - Rpc-proxy: reduce need for trust in rpc-proxy program code through per-client instances, use of SELinux

Questions?

- Email: sds@tycho.nsa.gov