RBAC Tutorial

Brad Spengler
Open Source Security, Inc.

Locaweb - 2012

Overview

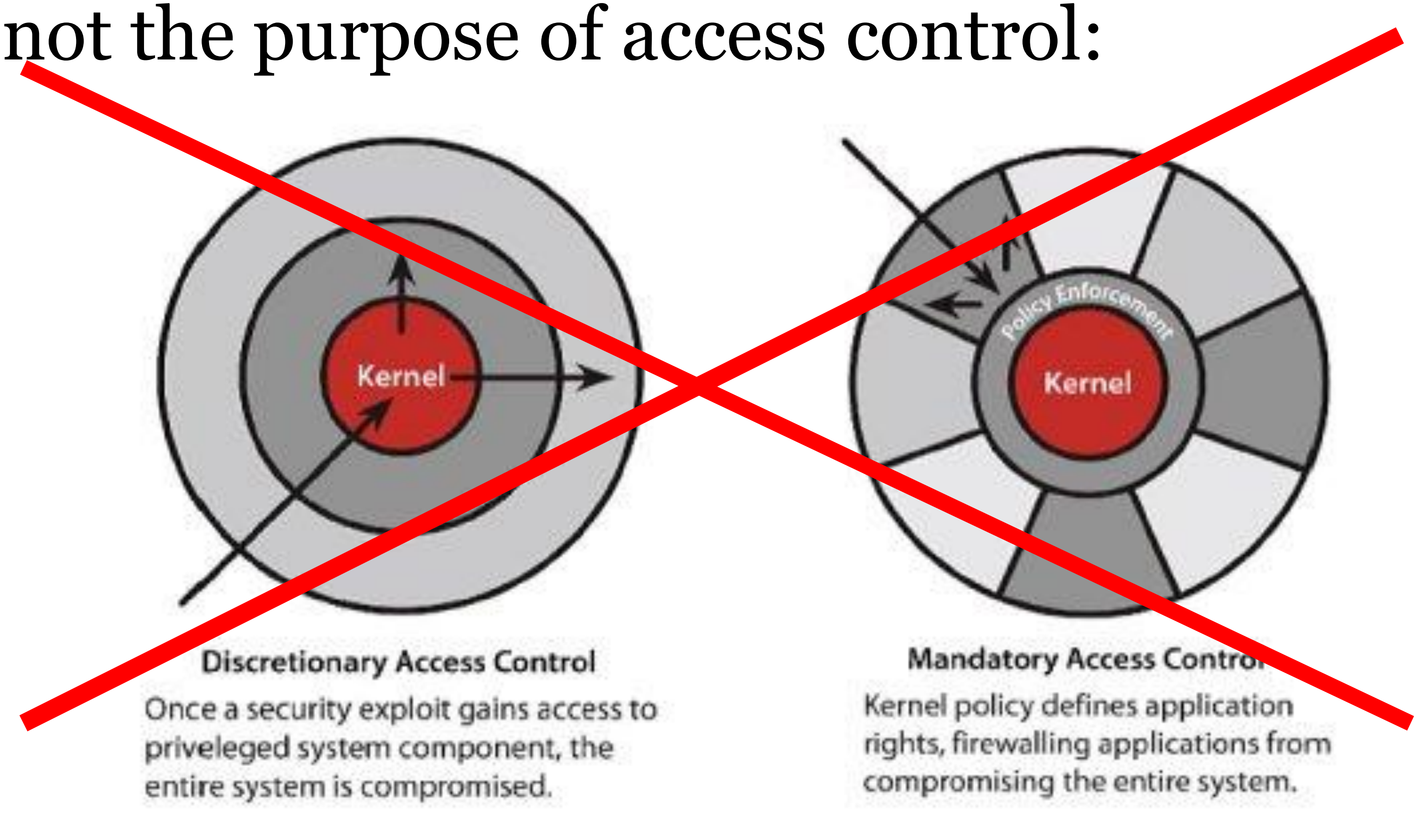
- Why Access Control?
- Goals
- Architecture
- Implementation
- Lookup example
- Subject example
- Questions/Requests

Why Access Control?

- Access Control is just one part of system security
- Useful tool, not a cure-all
- “Modern” mandatory access control uses decades-old technology and retains its antiquated assumptions
  - See Labeled Security Protection Profile (LSPP)
  - Not Internet-connected or even heterogenous Intranet-connected (3.3.4)
  - No active attacker or careless admin (3.3.0, 3.3.2)
  - Basically only accidental downgrade of sensitive info (4.1)

Why Access Control? (cont.)

- Despite what Red Hat wants you to think, this is not the purpose of access control:



Discretionary Access Control

Once a security exploit gains access to priveleged system component, the entire system is compromised.

Mandatory Access Control

Kernel policy defines application rights, firewalling applications from compromising the entire system.

Why Access Control? (cont.)

- Often used as a last line of defense (memory corruption post-exploitation)
- Front line defense for certain bug classes (arbitrary file disclosure, ../../../../etc/shadow)
- Typically not involved in reducing TCB attack surface
  - Proper sandboxes help here, but sufficiently complex/efficient code will touch rare paths
  - perf_counter()

Why Access Control? (cont.)

- Particularly useful in combination with a hostile attack environment
  - NX, ASLR, other userland hardening
- PaX can provide removal of arbitrary code execution in memory
- Access Control can provide the same at the filesystem level

Goals

- Design around Access Control strengths in combination with anti-exploitation measures
- Protect entire system, not just specific first-party apps
- Don’t create a “framework”, create a system with specific intent
  - Allows detection of stupid/wrong usage and enables user education
- Human readable, intuitive policy with understandable error messages and suggestions

Goals (cont.)

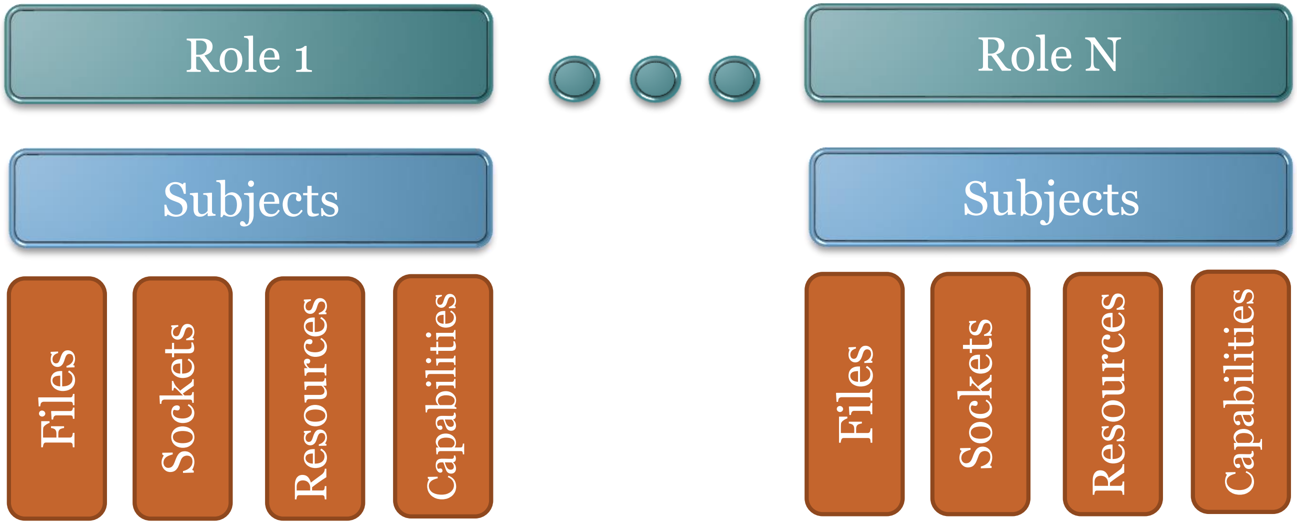
- Force users toward policies where base ambient permission is restrictive and unprivileged
- Provide full-system learning to automatically produce secure policies
  - Generally better than those a distro or user could create
  - Tailored to how software is used, not how it could be used in all configurations (inflation of ambient permission)

Goals (cont.)

- Provide simple configuration for learning based on questions like “what information is sensitive?”
- Performance: < 1% impact
  - SELinux claims 7% average hit, 10% hit on Apache

Architecture

- Kernel modifications perform policy enforcement and generates learning logs
- Userland tool parses and analyzes policy
- Policies have the following basic structure:



Architecture - Roles

- Roles can be applied to a user or group
- Everything without a specific role is given the "default" role
- Arbitrary special roles can be created that can be entered with optional authentication
  - PAM-based authentication is also provided
- Access to a role can be restricted by taint-propagated source IP
- Maximum umask can be enforced per-role

Architecture - Subjects

- Subjects refer to binaries or scripts
- Nested subjects are allowed: a subject whose policy is only applied when executed by another specified subject
- Subjects can "inherit" policy from a more generic subject
    - Allows to have a generic subject for unprivileged apps
    - All other subjects essentially show a "diff" of what makes them privileged

Architecture - Objects

- Objects are files, sockets, resources, capabilities, and PaX markings
- Files support access like read, write, execute, append-only, create, delete, hardlink, set suid/sgid, and hidden
    - Can also create audit logs for any of these accesses
- Sockets can be restricted by family (inet, netlink, etc)
- IPv4 sockets can be restricted by socket type, protocol, bind address, connect destination, and port

Architecture – Objects (cont.)

- Resource policies override those set by setrlimit()
    - CPU time, memory usage, max file size, etc
- Capabilities are subsets of “root” privilege
    - See “False Boundaries and Arbitrary Code Execution” (http://forums.grsecurity.net/viewtopic.php?f=7&t=2522)
- PaX flag support allows mandatory enforcement of PaX flags on user binaries or mandatory removal of flags for problem apps (e.g. PAX_MPROTECT on java)

Implementation

- Does not use LSM
  - History is interesting – initially a “trojan horse” to allow for a commercial security module from Immunix
  - A decade later, still does not support stacking
  - RBAC does much more than the LSM interface allows
- Meanwhile, grsecurity has remained compatible with all other LSMs

Implementation (cont.)

- Grsecurity’s RBAC system uses a combination of pathname and inode-based matching
- File objects support regular expressions, use anchors
  - An anchor is the longest valid path component from fs root not containing a regex
  - E.g.: /home/*/.ssh anchor is /home
- Inode/device pairs are determined for files that exist at enable time

Implementation (cont.)

- Non-existent files at enable time are specially marked internally
- Filenames are kept stored, used when creating a file to find and instantiate the object
- Enables idea of “policy recreation”: an object’s rules across all roles/subjects will persist across deletion/renaming/re-creation
- Filenames are based on the system’s default namespace, not process fs root
  - E.g. In a /srv1 chroot, policy on and logging of a /bin/sh file will appear as /srv1/bin/sh

Implementation (cont.)

- Much talk in the past from other camps about “insecurity” of pathname-based matching
  - Mostly aimed toward AppArmor (with some legitimate concerns there)
- Pitfalls of pathname-only matching:
  - Rename
  - Symlink
  - Hardlink
  - Mount

Implementation (cont.)

- Grsecurity’s RBAC avoids problems via hybrid approach
  - Rename: requires read/write access on both the source and destination name, create on new name (and delete if it exists), and delete on old name
  - Symlink: Not followed by userland tool (e.g. policy on a /tmp/hello.txt symlink to /etc/shadow can’t be tricked to grant access to /etc/shadow)
  - Hardlink: Requires create and link permission in addition to any permission existing on source
  - Mount: requires CAP_SYS_ADMIN, not supported while RBAC is enabled

Implementation (cont.)

- No support yet for filesystem namespaces (used by LXC)
  - Use is somewhat nebulous, in concert with many combinations of namespaces (pid, net, user)
    - Single-application sandbox
    - Entire system in a container
  - Only handle cases where files involved with the namespace are accessible via the main namespace?

Implementation (cont.)

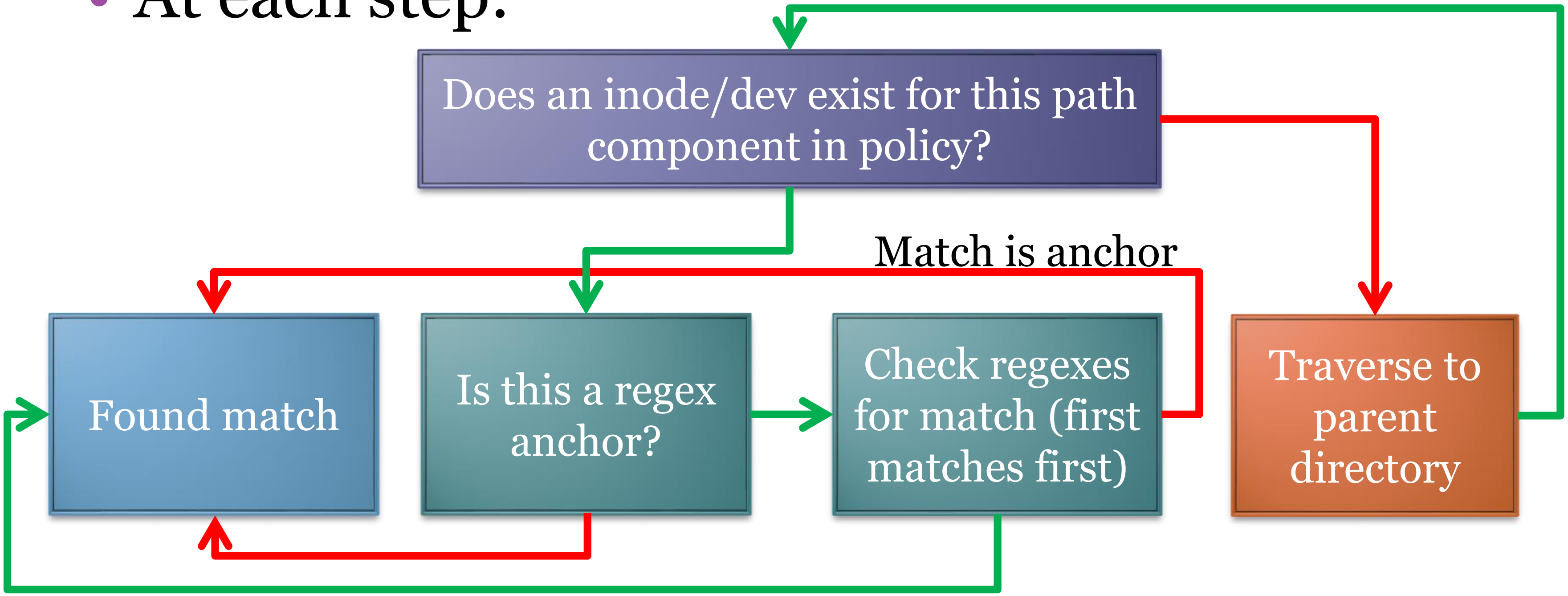
- Full-system learning creates a new subject for a binary when it:
  - Performs network activity
  - Modifies a file in a protected path
  - Reads a sensitive file
  - Uses a capability
- When many files in a given directory are accessed in the same way, access is reduced to the directory
  - Gives learning predictive power
  - ‘many’ determined by configuration

Lookup Example

- Given the following relevant objects:
  - / h
  - /home rwcd
  - /home/*/.bashrc r
- We will perform a lookup on:
  - /home/spender/.bashrc
  - /tmp/exploit

Lookup Example (cont.)

- At each step:


Does an inode/dev exist for this path component in policy?
Match is anchor
Found match
Is this a regex anchor?
Check regexes for match (first matches first)
Traverse to parent directory

Lookup Example (cont.)

- No inode/dev for /home/spender/.bashrc
- No inode/dev for /home/spender
- Inode/dev found for /home
  - It's also an anchor
- Check /home/*/.bashrc against /home/spender/.bashrc
- Match found, read-only access

Lookup Example (cont.)

- No inode/dev for /tmp/exploit
- No inode/dev for /tmp
- Inode/dev found for /
  - Also called the “default” object, as it catches all files without more specific objects
- Match found, not able to create, not able to see file if it already exists

Subject Example

- /usr/bin/cvs
- Interesting binary as it operates both as a server and client, depending on the context
- Policy is for the server context (in pserver mode)
    - ▫ run as user ‘cvs’, straight from grsecurity.net

Subject Example (cont.)

```
role cvs u
subject /
        /               h
        -CAP_ALL
        connect disabled
        bind disabled
```

```
subject /usr/bin/cvs
        /
        /*              h
        /etc/fstab      r
        /etc/ld.so.cache                r
        /etc/localtime r
        /etc/nsswitch.conf              r
        /etc/mtab       r
        /etc/passwd                     r
        /etc/group      r
        /proc/meminfo                   r
        /dev/urandom                    r
        /dev/log        rw
        /dev/null       rw
        /lib            rx
        /usr/lib        rx
        /home/cvs       r
        /home/cvs/CVSROOT/val-tags              rw
        /home/cvs/CVSROOT/history               ra
        /tmp            rwcd
        /var/lock/cvs  rwcd
        /var/run/.nscd_socket           rw
        /proc/sys/kernel/ngroups_max            r
        /proc/sys/kernel/version        r
        /var/run
```

Allows chdir("/") but no file/directory listing in /

Subject Example (cont.)

```
role cvs u
subject /
        /               h
        -CAP_ALL
        connect disabled
        bind disabled
```

```
subject /usr/bin/cvs
        /
        /*              h
        /etc/fstab      r
        /etc/ld.so.cache                r
        /etc/localtime  r
        /etc/nsswitch.conf              r
        /etc/mtab       r
        /etc/passwd                     r
        /etc/group      r
        /proc/meminfo                   r
        /dev/urandom                    r
        /dev/log        rw
        /dev/null       rw
        /lib            rx
        /usr/lib        rx
        /home/cvs       r
        /home/cvs/CVSROOT/val-tags              rw
        /home/cvs/CVSROOT/history               ra
        /tmp            rwcd
        /var/lock/cvs   rwcd
        /var/run/.nscd_socket           rw
        /proc/sys/kernel/ngroups_max            r
        /proc/sys/kernel/version        r
        /var/run
```

No “o” mode, so inherits file and capability policy from subject /, no capability use permitted

Subject Example (cont.)

```
role cvs u
subject /
        /               h
        -CAP_ALL
        connect disabled
        bind disabled
```

```
subject /usr/bin/cvs
        /
        /*              h
        /etc/fstab      r
        /etc/ld.so.cache                r
        /etc/localtime r
        /etc/nsswitch.conf              r
        /etc/mtab       r
        /etc/passwd                     r
        /etc/group      r
        /proc/meminfo                   r
        /dev/urandom                    r
        /dev/log        rw
        /dev/null       rw
        /lib            rx
        /usr/lib        rx
        /home/cvs       r
        /home/cvs/CVSROOT/val-tags              rw
        /home/cvs/CVSROOT/history               ra
        /tmp            rwcd
        /var/lock/cvs   rwcd
        /var/run/.nscd_socket           rw
        /proc/sys/kernel/ngroups_max            r
        /proc/sys/kernel/version        r
        /var/run
```

No modification of CVS repository

No arbitrary modification of CVS history

Subject Example (cont.)

```
role cvs u
subject /
        /               h
        -CAP_ALL
        connect disabled
        bind disabled
```

```
subject /usr/bin/cvs
        /
        /*              h
        /etc/fstab      r
        /etc/ld.so.cache                r
        /etc/localtime  r
        /etc/nsswitch.conf              r
        /etc/mtab       r
        /etc/passwd                     r
        /etc/group      r
        /proc/meminfo                   r
        /dev/urandom                    r
        /dev/log        rw
        /dev/null       rw
        /lib            rx
        /usr/lib        rx
        /home/cvs       r
        /home/cvs/CVSROOT/val-tags              rw
        /home/cvs/CVSROOT/history               ra
        /tmp            rwcd
        /var/lock/cvs   rwcd
        /var/run/.nscd_socket           rw
        /proc/sys/kernel/ngroups_max            r
        /proc/sys/kernel/version        r
        /var/run
```

No rwx access to filesystem

Subject Example (cont.)

```
role cvs u
subject /
        /               h
        -CAP_ALL
        connect disabled
        bind disabled
```

```
subject /usr/bin/cvs
        /
        /*              h
        /etc/fstab      r
        /etc/ld.so.cache                r
        /etc/localtime r
        /etc/nsswitch.conf              r
        /etc/mtab       r
        /etc/passwd                     r
        /etc/group      r
        /proc/meminfo                   r
        /dev/urandom                    r
        /dev/log        rw
        /dev/null       rw
        /lib            rx
        /usr/lib        rx
        /home/cvs       r
        /home/cvs/CVSROOT/val-tags              rw
        /home/cvs/CVSROOT/history               ra
        /tmp            rwcd
        /var/lock/cvs  rwcd
        /var/run/.nscd_socket           rw
        /proc/sys/kernel/ngroups_max            r
        /proc/sys/kernel/version        r
        /var/run
```

Warning! No network policy specified, allows any normally-permitted network activity! Gradm will alert you to this

Questions/Requests?

- Tried RBAC before and had a policy question?
- Features you would like to see?

- Thank you for supporting the research and development of grsecurity