12-6-2016

Role-Based Access Control Filesystem

Scott Linder
Western Michigan University, scott.linder18@gmail.com

Role-Based Access Control Filesystem (RBACFS) Sample Assignment

Scott Linder Ryan DePrekel Justin Lanyon

November 28, 2016

Abstract

UNIX permission bits do not provide fine-grained and flexible control over access, only permitting read (r), write (w), and execute (x), permissions to be set for user (u), group (g), and other(o). Role Based Access Control (RBAC) allows for many roles to have different access to the same object, as opposed to permission bits which allow at most one group and one user to have permissions. You will implement an RBAC policy via a definitions file for a filesystem implementing RBAC (RBACFS).

1 Overview

You have been given a `Vagrantfile` which will create a new virtualbox VM and automatically install RBACFS. From within the VM, change to the `/vagrant` directory, and mount the filesystem with the following line:

```
sudo rbacfs -o allow_other mount root student.defs
```

Test the policy implementation using the following script, which will print nothing when the solution is acceptable:

```
./test.sh
```

In order to change the policy, the filesystem must be unmounted and remounted. The filesystem can be unmounted with the following command:

```
sudo fusermount -u mount
```

2 Requirements

Modify `student.defs` to implement the following policy.

2.1 Policy

The `admin` role must be able to read and write to the printer config file, located at `/etc/lpd.conf`, while the `daemon` role must only be able to read it.

The `admin` role must be able to read the system log file at `/var/syslog`, while the `daemon` role must be able to write to it.

The `user` role must be able to read and write the reports files at and under the `/var/reports` directory, and read and write files at and under the `/home`.

2.2 Submissions

The only file which must be submitted is `student.defs`.