Using HP-UX Role-Based Access Control

Document Information

This White Paper from the Hewlett-Packard Company introduces the HP-UX Role-based Access Control (RBAC) product and describes the fundamentals of access control and its use within HP-UX. This paper also explains how to configure and use HP-UX RBAC.

Intended Audience

This White Paper is intended for IT professionals—specifically, network officers, architects, and administrators who perform security-related administrative functions.

Terms and Definitions

Every technical field has a unique and specific vocabulary. The following table lists and briefly defines fundamental terminology related to HP-UX RBAC and this document.

Term	Definition
Authentication	To verify the identity of a user, device, or other entity in a computer system, often as a prerequisite for allowing access to resources in a system.
Authorization	The process of determining whether a subject is allowed to perform an operation on a particular resource by evaluating applicable access control information. Usually, authorization is in the context of authentication. Once a subject is authenticated, it may be authorized to perform different types of access. In the context of this document, an authorization, commonly referred to as permission, specifically refers to the pairing of an operation with an object.
Object	Any system resource subject to access control, for example: a file, printer, terminal, database record, etc.
Operation	A specific mode of access to one or more protected object. In the context of this document, an operation refers to a general action, such as backup or restore, represented as a string such as `hpux.backup`. Operations are paired with objects to form authorizations.
PAM	Pluggable Authentication Module. PAM gives system administrators the flexibility of choosing any authentication service available on the system to perform authentication. The framework also provides the ability to plug-in new authentication service modules without modifying existing PAM applications.
Privilege	A user's right to perform a specific task, usually one which affects an entire computer system rather than a particular object. Privileges are assigned by administrators to individual users or groups of users as part of the security settings for the computer. In the context of this document, privilege refers to some combination of [effective | real] user id and [effective or real] group id.
Principle of Least Privilege	The concept that users should be granted the least privilege required to accomplish their tasks.
RBAC	Role-based Access Control. A mechanism to map users to their permitted authorizations (operation, object). Essentially an umbrella term that includes the definitions of roles and authorizations.
Role	A job function, within the context of an organization, with associated semantics regarding the authority and responsibility given to users assigned to the role.
Subject	The originator of an action requiring an authorization decision. In the context of this document, the term subject is synonymous with user.

Introduction

Security—especially platform security—has always been an important issue for enterprise infrastructure. Even so, concepts such as “individual accountability” and “least privilege” are not the most stimulating or popular conversation topics, and therefore are often neglected or over-looked by many organizations. Recently introduced legislation—including HIPAA and Sarbanes-Oxley—has helped to highlight the importance of these security concepts. In order to help address these growing demands, HP is releasing a series of products aimed at further strengthening the security of the HP-UX Operating Environments. This paper discusses the first of these products: HP-UX RBAC.

Problem Statement

In virtually all enterprise environments there are systems administered by multiple users. Typically, as is the case for most systems running UNIX®, this is accomplished by providing the administrators with the password to a common, shared account, i.e. root.

On one hand, the root account simplifies access control management because every administrator with the root password has the ability to perform all operations. However, the root account also presents several inherent obstacles for access control management, for example:

- After providing administrative users with the root password, there is no easy way to further constrain those users.
- In the best case, revoking access for a single administrator requires changing the common password and notifying other administrators. More realistically, simply changing the password is probably not sufficient to effectively revoke access, as alternative access mechanisms might have already been implemented.
- Individual accountability with a shared account is virtually impossible. Consequently, proper analysis after a security-significant event becomes difficult—and in some cases impossible.

The HP-UX RBAC Solution

Ideally, each user should be assigned a set of tasks they are permitted to perform. Administrative tasks could then be performed by ordinary—but appropriately configured—user accounts. This would help preserve the “Principle of Least Privilege” while also retaining individual accountability.

The HP-UX RBAC product achieves these goals while helping to mitigate the management overhead associated with assigning and revoking individual authorizations. Additionally, the HP-UX RBAC product offers several advantages over other role-based access control solutions available today, including:

- Pre-defined configuration files specific to HP-UX for a quick and easy deployment
- Flexible re-authentication ability via PAM to allow restrictions on a per command basis
- Integration with HP-UX (C2) audit system to produce a single, unified audit trail
- Fully supported HP product
- Plug-able architecture for customizing access control decisions

HP-UX RBAC Overview

The information in this section reviews access control fundamentals and explains how HP-UX RBAC simplifies access control management using roles. This section also describes the HP-UX RBAC product architecture, components, and operation.

Access Control Basics

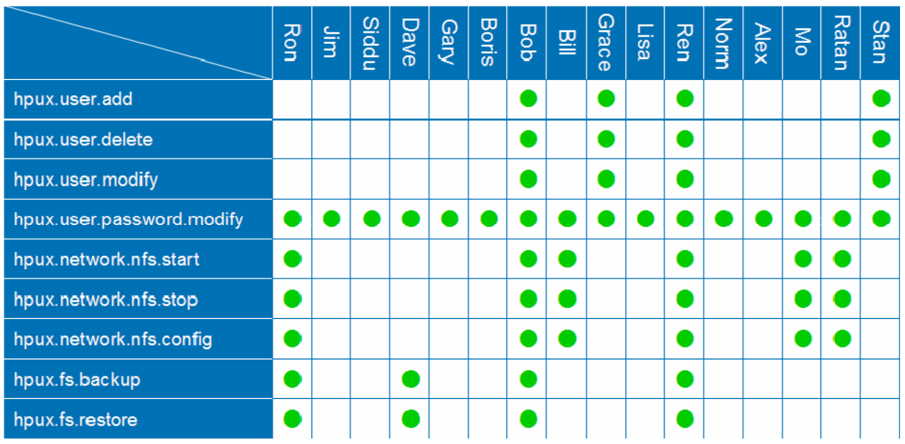
The goal of an access control system is to limit access to a resource based on some set of constraints. Typically, these constraints and their associated attributes fit into the following three categories:

- **Subject:** The entity attempting to access the resource. In the context of an operating system, the subject is commonly a user or a process associated with a user.
- **Operation:** An action performed on a resource. An operation can correspond directly to an application or command. In the case of HP-UX RBAC, the operation is a dot-separated, hierarchical string such as `hpux.user.add`.
- **Object:** The target of the operation, which may be the same as the end resource, but is sometimes different.

An access control request can be thought of as a question combining the above elements, where the response to the question, usually allow or deny, determines whether access to the resource is granted. For example:

Is the user **`Ron`** *authorized to perform the operation* **`hpux.fs.mount`** *on the object* **`/dev/c0t1d0`***?*

Often, the term authorization is used as a synonym for access control. In the HP-UX RBAC product, an authorization is a noun that refers to the ability to perform an operation on an object. A user then, as shown in the following graphic, is said to have a set of authorizations, each of which allow access to a resource.



	Ron	Jim	Siddu	Dave	Gary	Boris	Bob	Bill	Grace	Lisa	Ren	Norm	Alex	Mo	Ratan	Stan
hpux.user.add							●		●		●					●
hpux.user.delete							●		●		●					●
hpux.user.modify							●		●		●					●
hpux.user.password.modify	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
hpux.network.nfs.start	●						●	●			●			●	●	
hpux.network.nfs.stop	●						●	●			●			●	●	
hpux.network.nfs.config	●						●	●			●			●	●	
hpux.fs.backup	●			●			●				●					
hpux.fs.restore	●			●			●				●					

Note: The graphic above shows only the operation element of the authorizations—the object element of the authorization is not shown.

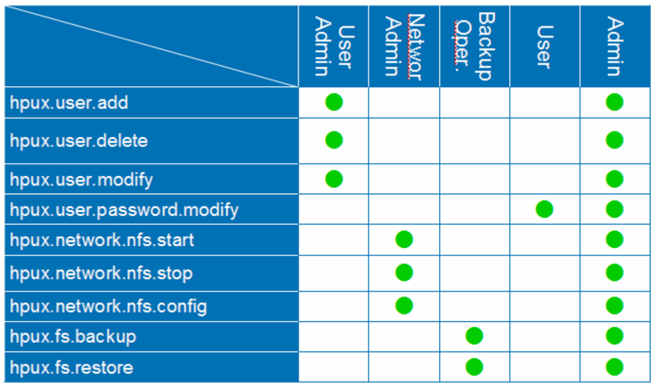
Simplifying Access Control with Roles

The preceding overview of access control presented does not address how policy is represented and how decisions are made. One obvious approach is to simply maintain a list of users and the authorizations (operation, object pairs) assigned to each of them. This approach has the advantage of being very flexible, as each user's set of authorizations can be completely different from that of the other users.

Unfortunately, this approach is also difficult to manage because as you add users, it is necessary to determine exactly which authorizations each user requires. Also, when performing audits, each user must be individually examined to determine their associated authorizations.

RBAC addresses these issues by grouping users with common authorization needs into roles. Roles serve as a grouping mechanism to simplify authorization assignment and auditing. Rather than assigning an authorization directly to a user, you assign authorizations to roles. As you add users to the system, you assign them a set of roles which determine the actions they may perform and the resources they may access.

Compare the following graphic to the first graphic, which lists the authorizations assigned to each user. By comparing these two graphics, you can see how roles simplify authorization assignment.



	User Admin	Networ Admin	Backup Oper.	User	Admin
hpux.user.add	●				●
hpux.user.delete	●				●
hpux.user.modify	●				●
hpux.user.password.modify				●	●
hpux.network.nfs.start		●			●
hpux.network.nfs.stop		●			●
hpux.network.nfs.config		●			●
hpux.fs.backup			●		●
hpux.fs.restore			●		●

Note: The graphic above shows only the operation element of the authorizations—the object element of the authorization is not shown.

Product Architecture and Components

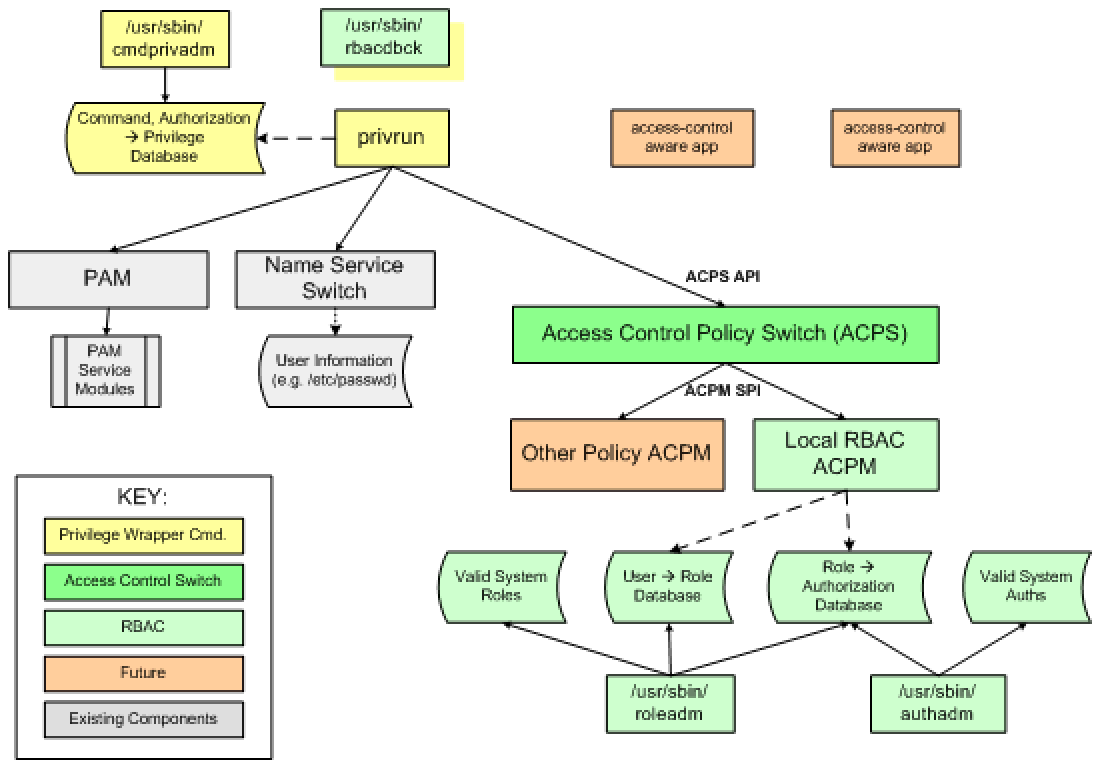
HP-UX RBAC brings the benefits of role-based access control to the HP-UX Operating Environment. The following is a list of the primary HP-UX RBAC components:

- **privrun Wrapper Command** to run existing legacy applications without modification and with varying privileges based on user authorizations
- **Access Control Policy Switch** to determine whether a subject is authorized to perform an operation on an object
- **Access Control Policy Module** to evaluate RBAC databases and apply mapping policies to service access control requests
- **Management Commands** to edit and validate RBAC database files
    - `roleadm`: Edits role information in RBAC database files
    - `authadm`: Edits authorization information in RBAC database files
    - `cmdprivadm`: Edits command authorizations and privileges in `privrun` database
    - `rbacdbchk`: Verifies syntax of RBAC and `privrun` database files

The primary component of HP-UX RBAC is the privrun command, which is used to invoke existing administrative commands, applications, and scripts. The privrun command uses a new subsystem, called the Access Control Policy Switch, to make access control requests based on a configuration file. An access request may be granted or denied based on a set of configuration files that define user→role and role→authorization mappings.

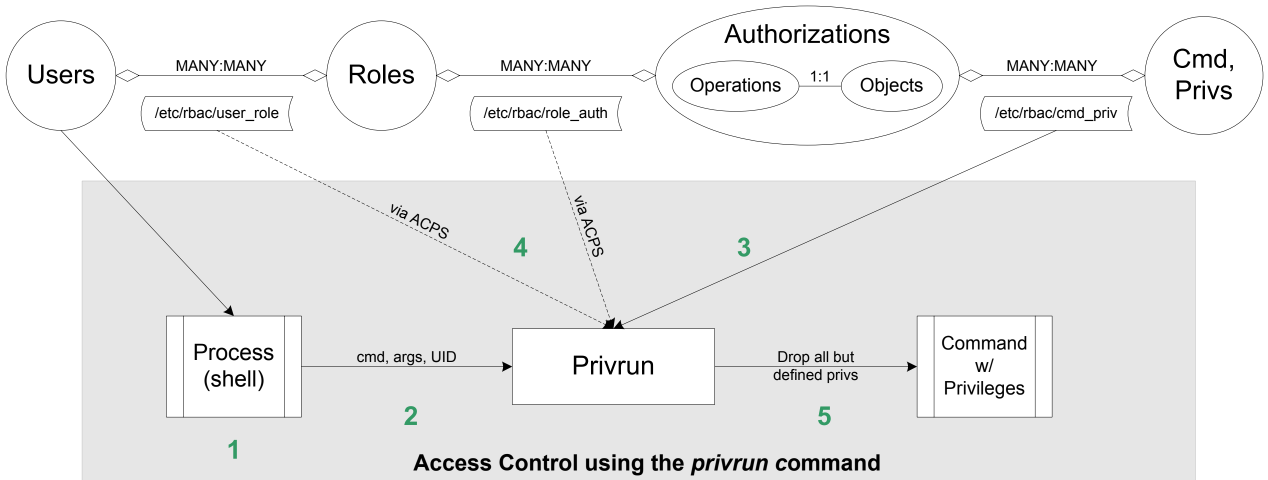
If the access request is granted, `privrun` invokes the target command with additional privileges. These privileges—specifically, a new uid and or gid—are configured to allow the command to run successfully.

The following graphic illustrates the HP-UX RBAC product architecture.



Example Usage and Operation

The following diagram and footnotes illustrate a sample invocation of privrun and the configuration files that privrun uses for determining whether a user is allowed to invoke a command. The details and procedures for creating and updating HP-UX RBAC files, as well as the details of the privrun invocation, are explained in greater detail later in this paper.



Access Control using the *privrun* command

1) A process (specifically a shell) associated with the user execs privrun with the goal of executing a target command with elevated privilege.

2) The target command line (cmd, args) is explicitly passed to privrun, the UID of the invoking user is implicitly passed via the process context.

3) privrun attempts to find a match (or set of matches) within cmd_priv database for the specified command line. Each matching entry will also specify a required authorization (operation, object pair) as well as the resulting privileges if the user has the specified authorization.

4) privrun makes a call (for each matching cmd_priv entry) to the Access Control Policy Switch. The RBAC backend of the ACPS will consult the user_role and role_auth databases to determine whether the user has the specified authorization, and will pass this result back to privrun.

5) Assuming that the user associated with the process has the required authorization specified in the cmd_priv database for the requested command, privrun will drop all privileges except those specified in the cmd_priv entry and exec the requested command. (privrun is setuid 0 and starts with all the necessary privileges.)

Acquiring and Installing HP-UX RBAC

The information in this section explains how to acquire and install HP-UX RBAC.

Acquiring HP-UX RBAC

HP-UX RBAC is available free-of-charge from HP's Software Depot, http://www.software.hp.com. Use the following steps to acquire HP-UX RBAC:

1. Go to HP's Software Depot: http://www.software.hp.com
2. Search for HP-UX RBAC (keyword RBAC)
3. Read the information on the RBAC release page
4. Select `Receive for Free >>` at the bottom of the page
5. Enter your registration information and accept the Terms and Conditions
6. Select `Download` at the bottom of the page
7. Save the RBAC depot as a local file on your system, for example in:
   `/tmp/<RBAC-depotname>.depot`
8. Use the following command to verify the depot file is on your system:
   ```
   # swlist -d @ /tmp/<RBAC-depotname>.depot
   ```

Installing HP-UX RBAC

Use the following steps to install HP-UX RBAC:

1. Review the HP-UX RBAC Release Notes for compatibility and installation requirements.
2. Logon to your system as the root user
3. Install HP-UX RBAC using the following command:
   ```
   # swinstall -s /tmp/<RBAC-depotname>.depot AccessControl
   ```
   Note: You do not need to reboot the system to install HP-UX RBAC.
4. Verify the installation using the `swlist` command. If HP-UX RBAC is installed on the system, the `swlist` command will report approximately the following:

   ```
   AccessControl  B.11.23.01    HP-UX Role-based Access Control Infrastructure
   ```

HP-UX RBAC installs in `/usr/bin/` and `/usr/sbin/`. The `swinstall` tool displays an error message if the installation fails. If you receive an error message or if the installation failed, check `/var/opt/adm/sw/swagent.log` for information.

Planning Your HP-UX RBAC Deployment

The information in this section explains the planning steps you should take before deploying HP-UX RBAC.

Step 1: Planning Roles for Users

The first, and possibly the most crucial step in deploying the HP-UX RBAC product is to plan an appropriate set of roles for the users of the target system. In some enterprises, this set of roles already exists and can be re-used when configuring the HP-UX RBAC product. More often, the roles must be designed based on the existing tasks associated with administrative users on the system.

The following are a few guidelines to follow when designing roles:

- There should be considerably fewer roles than the total number of users of the system. If each user requires a special role, then all of the simplified management associated with the use of roles is no longer in place.
- The roles should have some relation to the actual business roles of the users.
- Users may have multiple roles, and therefore some roles may be designed simply to group authorizations common to multiple (business) roles

Step 2: Planning Authorizations

Once the roles have been defined, the next step is to plan the authorizations associated with the roles. If the roles align nicely with the pre-existing operation hierarchy, then the authorization assignment may be trivial. Use the following command to list the system defined authorizations:

```
# roleadm list sys
```

If the existing authorization hierarchy does not align with your roles, defining the authorizations associated with each role may be a bit more tedious. A good approach is to list the system commands commonly used by each role. The target commands can then be compared against the supplied sample `cmd_priv` database in `/etc/rbac/cmd_priv`. If a corresponding entry is found, then the authorization from that entry may be used as a guide for assigning authorizations. For example, assume that one of the desired roles is UserOperator. This role would commonly run commands such as `useradd`, `usermod`, `userdel`, etc. To determine what authorization might be appropriate for this role, use the following command:

```
# grep useradd /etc/rbac/cmd_priv
/usr/sbin/useradd:dflt:(hpux.user.add,*):0/0//:dflt:dflt:dflt:
```

In this case, we see that the `/usr/sbin/useradd` command requires the `hpux.user.add` authorization. We might choose to assign this authorization directly, or in this case it might be more appropriate to assign `hpux.user.*` as the authorization.

Step 3: Planning Command Mappings

The final step in planning your HP-UX RBAC deployment is to define any commands that are commonly used by any of the defined roles, but that do not exist in the pre-defined `cmd_priv` file that is provided. The `cmd_priv` file defines the mapping between authorizations and commands. For each command, it is necessary to determine the following:

- The full path of the command
- The necessary authorization to check before running the command
- Any special privileges needed by the command, for example, `euid = 0`

The strings of text that comprise the operation and object entries in the `cmd_priv` file are arbitrary, but should correspond logically to a command or set of commands. Consider the following guidelines when planning your authorization to command mappings in `cmd_priv`:

- Define operations into logical groups to easily assign the operations to roles.
- Operation branches should not contain too many (~more than 10) or too few (~1) child elements. The overall tree should not be overly wide, making it difficult to assign groups of operations, or overly tall, with individual operation names that are long and hard to use.
- The last element of an operation name should end with an action (verb).
- Define operations so that new commands can be clearly placed when added.

Configuring HP-UX RBAC

Once your planning is complete, physically configuring the roles, authorizations and commands is relatively straightforward. In order to demonstrate the HP-UX RBAC administrative commands, we'll assume that the planning session resulted in the following mappings and that these fictitious users already exist on the target system.

Users	Roles	Authorizations (Note: Objects assumed to be *)	Typical Commands
chandrika, rwang	UserOperator	hpux.user.*	/usr/sbin/useradd
		hpux.security.*	/usr/sbin/usermod
bdurant, prajeesh	NetworkOperator	hpux.network.*	/sbin/init.d/inetd
luman	Administrator	hpux.*	
		company.customauth	/opt/customcmd

Step 1: Configuring Roles for Users

Configuring roles is a two-step process:

1. Create the roles
2. Assigning roles to users

Creating Roles

Use the `roleadm` command to create roles and associate them with users. The following shows the `roleadm` syntax:

```
syntax: roleadm add role [comments]
            | delete role
            | modify oldrolename newrolename
            | assign user role
            | revoke user [role]
            | list    [user=username] [role=rolename] [sys]
```

- The `add` and `delete` arguments add and remove the role from the system defined list of roles in `/etc/rbac/roles`.
- The `assign` and `revoke` arguments modify the list of roles associated with the user in `/etc/rbac/user_role`.
- The `modify` argument updates role names in all three role-related database files: `roles`, `user_role`, and `role_auth`.
- The `list` argument provides a listing of either the valid system roles (sys), or the user→role mappings.

For the purposes of the sample configuration defined in this paper, you must first add the roles that do not already exist and then assign users to those roles. Note that the default configuration files delivered with the HP-UX RBAC product contain a single pre-configured role: Administrator. By default, the Administrator role is assigned all HP-UX system authorizations (`hpux.*, *`) and is associated with the root user.

Use the following command to add new roles:

```
# roleadm add UserOperator
roleadm: added role UserOperator

# roleadm add NetworkOperator
roleadm: added role NetworkOperator
```

This defines the roles as valid, at which point they may be assigned to one or more users. If this step is not performed, attempting to assign the role to a user will display an error message indicating that the role does not exist.

Assigning Roles to Users

There are two advantages of separating role creation from role assignment:

- By requiring that the role be added before assignment, any typographical errors will be caught when specifying the role at assignment time.
- More importantly, separating the operations allows them to be granted to different users.

The HP-UX RBAC administration commands check to ensure that the user running them has the required authorizations. For example, a user must have the authorization `hpux.security.access.role.add`, to run the `roleadm add` command. This allows the role of SecurityOfficer—which defines roles and authorizations—to be distinct from that of UserAdministrator— which might only be able to assign existing roles. More generally, the fact that the HP-UX RBAC administration commands check the authorizations directly means that the HP-UX RBAC administration commands do not need to be wrapped with the `privrun` command[1].

Once the roles are created, use the following command to assign them to the appropriate users:

```
# roleadm assign luman Administrator
roleadm assign done in /etc/rbac/user_role

# roleadm assign chandrika UserOperator
roleadm assign done in /etc/rbac/user_role

# roleadm assign rwang UserOperator
roleadm assign done in /etc/rbac/user_role

# roleadm assign prajeesh NetworkOperator
roleadm assign done in /etc/rbac/user_role

# roleadm assign bdurant NetworkOperator
roleadm assign done in /etc/rbac/user_role
```

Use the `list` argument to verify the roles were assigned correctly:

```
# roleadm list
root: Administrator
luman: Administrator
chandrika: UserOperator
rwang: UserOperator
prajeesh: NetworkOperator
bdurant: NetworkOperator
```

Step 2: Configuring Authorizations

Configuring authorizations follows a very similar process to role creation and assignment. The only significant difference with authorization creation is that an authorization contains two elements—an operation and an object.

In many cases, the object is left unspecified, meaning that the operation applies to all objects. This is often used for authorizations that apply to wrapped commands, as it can be difficult to determine the target of an action from the command name. An example of this object ambiguity is the `/usr/sbin/passwd` command. The `passwd` command may be operating on a number of repositories, for example the `/etc/passwd` file, an NIS table, and an LDAP entry. The actual object is not available by looking at the command line[2] and so it is typically easiest to require that the user have the operation on all objects, for example: (`hpux.security.passwd.change, *`).

[1] The reason the HP-UX RBAC administration commands do not need to be wrapped by privrun is because they are setuid 0. The HP-UX RBAC administration commands run with root privileges regardless of who invokes them. The access control checks limit who may effectively use the HP-UX RBAC administration commands.
[2] In the case of a password change, the actual target is determined by the contents of /etc/pam.conf and /etc/nsswitch.conf files.

The '*' wildcard—the most commonly used object—is the implicit object used if an object is not specified while invoking the `authadm` command. The `authadm` syntax is similar to the `roleadm` syntax. The following is the `authadm` command syntax:

```
syntax: authadm add operation[object[comments]]
          | delete operation[object]
          | assign role operation[object]
          | revoke [role=name][operation=name[object=name]]
          | list [role=name][operation=name[object=name][sys]
```

Be aware that when assigning an authorization that contains the asterisk '*' character, you must surround the wildcard character with quotes to prevent shell interpretation. For example, the following shows the creation and assignment of the authorizations identified in the planning sections:

```
# authadm add company.customauth
authadm added auth: (company.customauth,*)

# authadm assign Administrator company.customauth
authadm added auth for role Administrator

# authadm assign NetworkOperator 'hpux.network.*'
authadm added auth for role NetworkOperator

# authadm assign UserOperator 'hpux.user.*'
authadm added auth for role UserOperator

# authadm assign UserOperator 'hpux.security.*'
authadm added auth for role UserOperator
```

As with the `roleadm` command, use the `list` argument with the `authadm` command to verify the authorization assignment, for example:

```
# authadm list
NetworkOperator (hpux.network.*, *)
UserOperator (hpux.user.*, *) (hpux.security.*, *)
Administrator: (hpux.*, *) (company.customauth, *)
```

Step 3: Configuring Additional Commands

The final step in configuring the system for use with HP-UX RBAC is to define any additional commands that are not provided in the default configuration. The authorizations needed to run the commands should have already been created and assigned to a role. Without this, the command will be configured, but no user with be appropriately authorized.

The `cmdprivadm` command works in a similar fashion to `roleadm` and `authadm`, but has fewer sub-operations: addition and removal. To modify an existing entry, you must delete the entry and then add the corrected version back in. `cmdprivadm` has several options that may be specified when adding an entry. The following shows the `cmdprivadm` syntax:

```
cmdprivadm add cmd = full path name of a command
                        |[op = operation]|[object = object]
                        |[ruid = ruid]|[euid = euid]
                        |[rgid = rgid]|[egid = egid]
                        |[re-auth = pam_service name]

cmdprivadm delete cmd = full path name of a command
                        |[op = operation]|[object = object]
                        |[ruid = ruid]|[euid = euid]
                        |[rgid = rgid]|[egid = egid]
                        |[re-auth = pam_service name]
```

Refer to the `cmdprivadm(1M)` manpage for details on each of the arguments. When adding an entry, most of the arguments are optional and will be filled in with reasonable defaults if nothing is specified. The most commonly specified arguments are demonstrated in the following example:

```
# cmdprivadm add cmd=\opt\customcmd  \
       op=companyname.customcommand \
       ruid=0 euid=0
/opt/customcmd::(companyname.customcommand,*):0/0/-1/-1::::
cmdprivadm added the entry to /etc/rbac/cmd_priv
```

Arguments act as filters when using `cmdprivadm` to delete entries. For example, specifying # `cmdprivadm delete op=foo` will remove all entries where the operation is `foo`. As a result of this, be careful to insure that sufficient arguments are specified to uniquely identify the entries that are to be removed.

Using HP-UX RBAC

The information in this section explains how to run the `privrun` command and operate HP-UX RBAC.

Running the privrun Command

Once properly configured, actually using HP-UX RBAC is relatively simple. The most basic usage is to invoke privrun with the command as an argument, for example:

```
# privrun ipfstat
```

As long as the user logged in has the necessary authorization defined in `/etc/rbac/cmd_priv`, `privrun` will execute the command with the privileges (e.g. UID, GID) defined in the `cmd_priv` entry. In some cases, there may be multiple entries for the same command, potentially with different required authorizations and different resulting privileges. In this case, privrun will iterate sequentially through the `cmd_priv` database, executing the first command that the user is authorized for. In some cases, this may not be ideal. For example, all users may be allowed to run the `passwd` command to change their own password, but if a user administrator runs it, they need the privileges to change other users' passwords. If the "normal user" entry is first, this might prevent the user administrators from running the more privileged version. For cases like this, privrun has options that allow the user to specify the desired privileges. Only entries matching the specified privileges (e.g. UID) will be used. If no entries match the desired privileges, `privrun` will emit an error message. The following is an example invocation of privrun that will only match entries where the effective uid is set to 0:

```
# privrun –u 0 ipfstat
```

In addition to the standard `privrun` behavior, there are two additional options: `-t` for test mode and `-x` for fall-through mode. Test mode performs all the normal authorization and authentication checks according to the configuration files. The only difference is that rather than executing the command upon success `privrun –t` just returns. This can be useful to preview whether a given `privrun` invocation will succeed. Fall-through mode modifies the behavior of `privrun` only when an authorization or authentication check fails. In this case, rather than exiting with an error, the target command will be run, but without any additional privileges. It will essentially act as though the user ran the command directly.

Troubleshooting HP-UX RBAC

The following is a list of the primary mechanisms used to troubleshoot and debug HP-UX RBAC:

- `rbacdbchk` utility to verify RBAC database syntax
- `privrun -v` to report additional and relevant information

rbacdbchk Database Syntax Tool

The most common bugs are caused by manual editing of the HP-UX RBAC databases, resulting in syntactically invalid configurations, or configurations which are inconsistent between databases (for example, a role in `user_role` that is not defined in `roles`). To assist in diagnosing these common mistakes, the HP-UX RBAC product includes an `rbacdbchk` command. This command will read through the RBAC-related databases and print warnings where incorrect or inconsistent configuration entries are found:

```
# rbacdbchk
[/etc/rbac/user_role] chandrika: UserOperator
        invalid user
        The value 'chandrika' for the Username field is bad.

[/etc/rbac/cmd_priv]
/opt/cmd:dflt:(newop,*):0/0//:dflt:dflt:dflt:
        invalid command: Not found in the system
        The value '/opt/cmd' for the Command field is bad.

[Role in role_auth DB with no assigned user in user_role DB]
Rebooter:(hpux.admin.*, *)

[Invalid Role in user_role DB. Role 'UserOperator' assigned to
user 'chandrika' does not exist in the roles DB]
```

On a correctly configured system, the `rbacdbchk` command should have no output, indicating no errors.

privrun –v Information

The second method for detecting issues is to run the `privrun` command with the `-v` option (verbose mode). This will cause `privrun` to provide additional information on the entries that the input command matched, the status of the authorization checking, as well as other relevant information. In many cases, this output will clarify the issue causing `privrun` to fail. The `-v` option may be specified multiple times for even more verbose output. The following is an example of the `privrun -v` output with the `ipfstat` command:

```
# privrun -v /sbin/ipfstat
privrun: user root intends to execute command /sbin/ipfstat
privrun: input entry: '/sbin/ipfstat:dflt:(,):///:dflt:dflt::'
privrun: found matching entry:
'/sbin/ipfstat:dflt:(hpux.network.filter.readstat,*):0/0//:dflt
:dflt::'
privrun: passed authorization check
privrun: attempting to set ruid/euid/rgid/egid to 0/0/-1/-1
privrun: current settings for ruid/euid/rgid/egid are 0/0/3/3
privrun: executing: /sbin/ipfstat
```

Conclusion

There is great deal of debate within the IT security community as to whether the majority of attacks on an enterprise's infrastructure come from outside or inside of the corporate network. Regardless, one fact generally agreed to is that a significant source of system compromises originate from within the corporate firewall. In these cases, it is critical to be able to determine "who did what"— even after the fact. Whether the goal is to ensure the privacy of health related information or to determine the integrity of corporate financial data—the right tools must be in place to provide access control and accounting. For HP-UX, the HP-UX RBAC product provides a fundamental piece for that overall security solution.

Appendix A: Frequently Asked Questions

Why is real uid used—as opposed to effective uid—for the purposes of identifying the user?

In order for the privrun command to have the ability to elevate privileges for other commands, it is delivered as a setuid binary. Because privrun is delivered as a setuid binary, the effective uid of the process that invokes privrun is not directly available to the privrun command. Therefore, the real uid is used to identify the user.

Does privrun run automatically or do I have to call a command with privrun as the first argument? Is there a warning or error message pointing to privrun if I call the command without using privrun?

You must explicitly invoke privrun to wrap a command that requires additional privileges. The error message received after invoking a command without sufficient privilege will vary depending on the command. Many commands return a message regarding "insufficient privileges."

Is privrun available in single-user mode?

No. The privrun command requires dynamically loading access-control decision providers based on a configuration file and therefore does not function in single-user mode. HP is considering this functionality for future releases.

Is HP-UX RBAC a priced product, a free-of-charge product, or included in core HP-UX with no installation needed?

HP-UX RBAC is currently available at http://software.hp.com for HP-UX 11i v2 (PA-RISC or Itanium) as a free-of-charge product. HP-UX RBAC will be integrated into core HP-UX for future major enterprise releases.

Can I run privrun from within SD scripts?

HP is not aware of any issues from running privrun from within SD scripts. However, SD is currently designed with the assumption that the SD-related commands will be run as root, and therefore the inclusion of privrun within the SD control scripts is unnecessary and may cause unexpected results. HP is considering integrating HP-UX RBAC more closely with SD for future releases.

Are authorizations always assigned to a role instead of an individual user?

Yes. There is currently no way to assign an authorization directly to a user.

What is the connection between the operation string and the command line? Where do the authorization names come from?

The `cmd_priv` file defines the mapping between authorizations and commands. The strings of text that comprise the operation and object entries in the `cmd_priv` file are arbitrary, but should correspond logically to a command or set of commands. Refer to the *"Planning Command Mappings*" section in this paper for guidelines on planning your authorization to command mappings in `cmd_priv`.

Is there a GUI or web interface to managing the roles? Is role management integrated into SAM?

No. Use only the roleadm command to manage roles.

Can a root user still run all commands without using privrun?

Yes. For the foreseeable future, root will still run with an elevated privilege set.

Is HP-UX RBAC pre-configured with some typical roles?

HP-UX RBAC is preconfigured with a single role—Administrator—to which only root is assigned.

If I run a command through a script, can I run privrun on the script?

Yes. HP recommends wrapping commands that have complex access control requirements as arguments in a script, and configuring the script with privrun. Be aware that all elements in the script will run with additional privileges, so if any element is under the user's control, the user will potentially be able to subvert the script and gain unconstrained privilege.

Is there a command to "export" the privrun database and "import" the settings so that the database can be restored if the system needs to be reinstalled (or exported to another system)?

No, you cannot export and import the privrun database settings. However, the set of databases that HP-UX RBAC uses is well defined and can be directly copied to the target system.

Does HP-UX RBAC function the same way on trusted, non-trusted, and shadow password systems?

Yes. However, note that when users are required to re-authenticate themselves, the re-authentication step will vary, just as the initial login did. For example, different sets of restrictions are checked in Trusted Mode and in Standard Mode.

Is root the only user who can run roleadm?

No. The roleadm command can be run by any user with the authorization (hpux.security.access.role.*, *). This can be further differentiated into users who can run `roleadm add` versus users who can run `roleadm assign`. Refer to the `/etc/rbac/auths` file for more details.

Can I create new authorizations? For example, if I have my own command "foo", can I allow certain users only to run it?

Yes. This would require defining a new authorization (if it doesn't already exist) using the authadm command. Next, configure the "foo" command using the cmdprivadm command.

Should the full path of the command be given for cmdprivadm? If a user runs their copy of mount, versus /usr/sbin/mount, is it the same?

The full path must be given when configuring the command in the cmd_priv database. At runtime, the user may choose to fully specify that path as an argument to privrun, or they might allow privrun to resolve the path much like a shell would. In either case, the resolved path is checked against the entry in the cmd_priv database. This prevents users from selecting a version of the command under their own control (unless an administrator explicitly allows it).

If you give a user the privilege to run vi and then if while in vi, they escape to the shell, will they be running as root or themselves? Is there a way to block escapes to a shell from vi or other commands that could be root?

Since the target command, vi in this case, is run with additional privileges, anything the command allows—including shell escapes—will also be run with additional privileges. For this reason, HP strongly discourages administrators from configuring commands that directly or indirectly allow users to gain unconstrained privileges. For the common editors, such as nvi or vi, there are usually "secure" versions that do not allow shell escapes and that can be safely configured to edit specific files.

Does HP-UX RBAC support using LDAP to store access control information?

HP-UX RBAC does not currently support LDAP for storing access control information. HP is considering this type of LDAP support for future releases.

Is the privrun –x command the same as running privrun alone, without any arguments (assuming that the user does not have the required authorization)?

It is pretty close, but the following exceptions are worth noting:

- The effective uid associated with the process is masked by the privrun command. In the case where it is necessary to “fall-through”, the real uid is used in place of the effective uid.
- In the cases where privrun encounters an error (for example, failing to open target command), privrun will return an error. The invoking script may need to differentiate between privrun errors and errors returned by the target command.

Are there any backwards compatibility issues with HP-UX RBAC, specifically with respect to custom scripts?

There are no direct backwards compatibility issues because privrun is optional and does not need to be used. If privrun is explicitly used by the script, it will need to differentiate between error codes returned by privrun and error codes returned by the target command. One strategy for this would be examining stdout for errors beginning with the string `privrun:`

Does privrun maintain an audit trail of user-op-target triples?

Yes.

Where can I find the audit trail? (syslog?)

The audit trail is available as part of the standard HP-UX (C2) audit system. The actual location of the audit trail is configurable. See the audsys(1M) man page for more details.

If a command is executed without privrun, what prevents the command from executing?

When an administrative command is run by a non-root user, there are usually three things that may prevent a successful execution:

- The command will attempt to access a file for which the uid/gid associated with the process does not have appropriate permissions. The file open call will fail.
- More generally, the command may make use of a system call that enforces a check (within the kernel). This check is often to ensure that the uid associated with the process is 0 (root).
- In order to avoid unnecessary error handling for the above cases, many commands will perform a check (within userspace) for the uid of the process. If this uid is not 0, the process may exit with an error. Note that this check does not actually provide any security as the user can typically circumvent this check, for example by compiling a custom version of the command.

How should I specify arguments to commands?

To allow users to run commands only with specific arguments, configure the entire command line with `cmdprivadm`. To allow users to run commands with any arguments, configure only the command. If you don’t want to allow any arguments, configure # `cmdprivadm cmd='/path/to/cmd␣`

Note: A space after the command must be included.

Can I allow classes of arguments, for example: foo –a [anything]?

Currently, the only way to accomplish this is to wrap the target command with a script and then configure the script using cmdprivadm. HP is considering this functionality for future releases.

Can I base my roles on user groups? How?

Basing roles on groups is currently not possible. However, you can always define roles in parallel with the same users. HP is considering this functionality for future releases.

For More Information

Use the following resources for more information about HP-UX RBAC:

- HP-UX RBAC Release Notes:
  http://docs.hp.com/hpux/internet/index.html#HP-UX%20Role-based%20Access%20Control
- HP-UX RBAC manpages:
  - rbac(5) — Overview and introduction of the RBAC product.
  - privrun(1m) — Runs legacy applications with varying privileges that are based on authorizations associated with invoking user.
  - rbacdbchk(1m) — Verifies syntax of RBAC and `privrun` database files.
  - roleadm(1m) — Edits role information in RBAC database files.
  - authadm(1m) — Edits authorization information in RBAC database files.
  - cmdprivadm(1m) — Edits command authorizations and privileges in privrun database

Call to Action

Download HP-UX RBAC free-of-charge from HP's Software Depot:
http://www.software.hp.com