Profile and System Shells 1

1.1 General Description

This document discusses the design for four commands related to execution of commands in restricted environments defined by the Profile mechanism:

- Profile Shell (`pfsh`),
- System Shell (`sysh`),

1.1.1 Background

1.1.1.1 Profile Shell

The Profile mechanism provides a way to control the capabilities of users and roles in Trusted Solaris 2.5. Each user and role has one or more profiles assigned by an administrator. A profile defines commands, CDE actions and authorizations that are permitted to the user. As described in ts2_408, CDE limits a user or role's CDE actions based on the assigned profiles. The Profile Shell (`pfsh`) provides equivalent restrictions for commands. The Profile Shell is a modified version of the Bourne Shell.

By assigning it an action that invokes the Profile Shell, an administrative role can be provided with a command-line interface for performing administrative tasks. This gives the administrator a way to use commands that are needed to accomplish a task for which no appropriate GUI-based tool is available, but leaves the administrator restricted in the commands that he can execute. This helps satisfy the principle of least privilege, in that the administrator is granted only the set of commands needed for his tasks.

The Profile Shell can also be used for nonadministrative roles.

The Profile Shell can provide a restricted shell environment for normal users, too. By specifying the Profile Shell in a user's `passwd` entry, an administrator can restrict the user to those commands specified in his assigned profiles.

The principle of least privilege requires that commands be executed with only the privileges they need to accomplish the authorized task. For each command defined in a profile, the profile specifies the privileges with which the command is to be executed. The Profile Shell is responsible for executing each command with the specified privileges.

Additionally, profiles can specify for each command ~~a range of sensitivity labels at which the Profile Shell is permitted to run the command, and~~ the UID/GID with which the Profile Shell runs the command.

1.1.1.2 System Shell

During execution of the `rc` scripts at system start-up, a number of commands need privilege in order to perform their tasks. For example, `mount` and `chown` are run from the `rc` scripts; `mount` needs privileges that include `priv_sys_mount`, and chown requires `priv_file_chown`.

The Profile mechanism provides a way to specify the privileges needed by these commands. The list of commands and their required privileges are stored in a profile, and `init` invokes the rc scripts using the System Shell (`sysh`). The System Shell runs commands, setting their privileges as specified in their entries in the profile. Like the Profile Shell, the System Shell is a modified version of the Bourne Shell that executes commands with the privileges specified in the profile. Unlike the Profile Shell, the System Shell does not restrict the set of commands that may be executed. Commands not listed in the profile are run without privileges.

Additionally, profiles can specify for each command ~~the sensitivity label and~~ the UID/GID with which the System Shell runs the command.

1.1.2 Overview

1.1.2.1 Profile Shell

The Profile Shell (`pfsh`) is a command-line interface. It is typically invoked in a `dtterm` window. ~~It runs at a specified sensitivity label within the user or role's label range, which may include the administrative labels ADMIN_LOW or ADMIN_HIGH.~~ It is based on the Bourne Shell (`sh`) and provides essentially the same capabilities, with one important distinction: it only executes commands that are defined for the process's real UID, which in the case of a role is the UID of the role.

The commands that can be executed by a specific user or role are defined via profiles. An administrator uses a profile administration tool to store the profile definitions in the profiles database `tsolprof`(4TSOL), and to associate one or more profiles with a UID in the `tsoluser`(4TSOL) database. Profiles are described in ts2_408 and the `tsolprof` and `tsoluser man` pages. Each entry in a profile defines a list of commands and the security attributes with which `pfsh` should run the commands.

The Profile Shell generates an audit record for each command that it executes. The audit record includes the path of the command, command parameters, the privileges inherited by the command, the current directory, and, if the AUDIT_ARGE audit policy is on, the environment variables.

Different audit event numbers are used to distinguish the security relevance of commands, based on whether the command is run by a process with the trusted path attribute, and whether the command is executed with privileges or a UID/GID change. The four audit events indicate:

- Trusted path process, command run with special security attributes
- Trusted path process, command run with no special security attributes
- Not trusted path process, command run with special security attributes
- Not trusted path process, command run with no special security attributes

These four audit events can be mapped into different audit classes to reflect their differing levels of interest.

Because the environment may be large, `pfsh` optionally excludes the environment variables, based on the setting of the AUDIT_ARGE auditing policy.

~~The Profile Shell does not manipulate the process information label in any way.~~

1.1.2.2 System Shell

The System Shell (`sysh`) is used to control the use of privileges in commands run from the `rc` scripts. It allows any command to be executed, but consults a profile for the privileges, UID, GID and SL with which the command is to be run. It can also be used for shell scripts run from roles.

The System Shell is a forced privilege program. It can only be run from a process with the Trusted Path attribute.

The System Shell generates an audit record for each command that it executes with privileges or with a change in UID, GID~~ or sensitivity label~~. The audit record includes the command pathname, the command parameters, the privileges inherited by the command, the current directory, and, optionally, the environment variables. Because the environment may be large, `sysh` only includes the environment variables if the AUDIT_ARGE audit policy is on.

The System Shell starts each command with an IL of ADMIN_LOW.

In order to pass any possible privilege on to commands, the System Shell executable file has all privileges on in its forced privilege set. At initialization, System Shell turns off all effective privileges. It turns on the `proc_setid`, `proc_audit_tcb`, `proc_setclr`, `proc_setsl`, and `proc_setil` effective privileges at those points in the program where they are required.

1.2 Rationale

The Profile Shell and System Shell are each separate executables built from conditional compilations of modified Solaris 2.5 source for `sh`. They are separate executables to better separate the Trusted Solaris modifications from the standard shell, and because the System Shell uses forced privileges.

The Profile Shell has all forced and allowed privileges. It enforces privilege bracketing by passing any privileges specified in a command's profile entry via the inheritable privilege set. When executing in an administrative role context, it checks for the trusted path attribute to make sure it has been invoked by aTCB process.

In order to keep the number of modifications to init to a minimum, `sysh` is launched with forced privileges. Since forced privileges are used, `sysh` checks for the trusted path attribute to make sure it has been invoked by a TCB process. When it is invoked by `init` to run the `rc` scripts, it inherits the trusted path attribute from `init`.

The profiles database can be stored in local files or NIS+ tables. To isolate the handling of the actual location of the profiles, the shells use `getuserent`(3TSOL) and `getprofent`(3TSOL) to retrieve profile data. Those functions have a mechanism for handling profiles from files or NIS+ tables based on `nsswitch.conf`.

The Profile Shell only needs to audit security-relevant commands, such as commands executed with privileges in an administrative role. Rather than hard coding the decision about which commands are security-relevant, four different audit event numbers are used to indicate that the command is executed with or without privileges by a trusted or untrusted process. Auditing will typically be configured to filter out unprivileged commands executed by a normal user, but `pfsh` provides the means to audit all of a user's commands, which is a useful feature in some environments.

1.3 Future Work

A Profile Korn Shell should be implemented at a later date.