F.2.27

# Toehold -- A Workstation Login Facility

Preliminary Design Draft

Jim Aspnes


## Abstract

Toehold is intended to be a comprehensive system for managing user login sessions on Athena workstations. This will include management of Kerberos authentication tickets, management of system and personal Remote Virtual Disks, and creation of temporary accounts for workstation users. The system will also provide information to the User Locator server, and will provide a hook for an automatic workstation software update facility.

# Table of Contents

## 1 Purpose

The workstation environment differs significantly from the current Athena timesharing environment. Differences important to Toehold tend to fall into three classes:

- Software management. Unlike timesharing machines, workstations will need to support some sort of automatic software distribution system. Thus it will be necessary for workstations to spin down their /srvd and /urvd RVD's periodically, allowing their contents to be updated from a central location. It will also be necessary for workstations to periodically compare their software installation with a centralized distribution and update locally resident software if needed.

- User account management. It is unfeasible for each of many thousands of workstations to maintain /etc/passwd and /etc/rvdtab entries for all Athena users. Similarly, the existing Unix tools for user location and notification, designed for environments with one or only a few closely-connected machines, are unsuited for the Athena workstation environment. It will thus be necessary for the Toehold system to interact with centralized database and user locator services to dynamically create accounts at login and to notify the user locator server of this fact.

- Security. Since workstations will not be physically secure, it will be necessary to have a secure software-based authentication protocol. This facility is provided by Kerberos; Toehold will need to support it.

## 2 Design Issues

Several other issues come up in designing a login/logout system for the Athena workstation environment. Most have come up as questions regarding the ability to carry over certain desirable features of the timesharing user environment to the workstation environment.

### 2.1 Multi-User Workstations

There is some controversy as to whether more than one user should be logged into a workstation at one time. On the one hand, we would like to guarantee that the user on the workstation console should be able to treat the workstation as a private machine; but on the other hand, we should not preclude multiple users if it is reasonable to have them. Our design provides mechanisms for providing additional privileges to the primary user, and does not assume that only a single user will be present. This allows restrictions on multi-user use to be enforced optionally by the rlogin software, and for other programs (for example, su, o_kill, or o_renice) to be made usable only by the

primary user.

## 2.2 Kerberos and /etc/passwd

In the long run, all but a few system-specific users should be authenticated by Kerberos. Even when this is the case, though, there will be some need for users to be authenticated by the local /etc/passwd file even though they are not recognized by Kerberos. Thus our design allows authentication through /etc/passwd, although this will only be tried if Kerberos fails and will prevent automatic RVD spinup or account creation.

## 2.3 Background Processes

Several users have pointed out the utility of being able to leave batch-mode jobs running as background processes after logout. Unfortunately, most background processes left behind will not have much justfication for their existence, since neither the user's Kerberos tickets, RVD locker, nor /etc/passwd entry will survive his or her logout, and thus most background processes should be terminated on logout. However, this is not the case for a user who is authenticated by /etc/passwd rather than Kerberos, so batch-mode processes for such users could reasonably be left running.

# 3 Proposed Design

Changes to the current system will be made by altering /bin/login and by implementing a new program, /etc/toehold, which will run under init on the console. /bin/login will be responsible for those parts of the toehold system which are user-dependent; /etc/toehold will be, for the most part, responsible for system-wide operations.

## 3.1 Modifications to /bin/login

/bin/login will be modified to perform the following additional actions on login:

1. Spinup of system RVDs. This allows, for example, a user to be logged in remotely even if no one is on the console.
2. Kerberos Authentication.
3. Notification of User Locator.
4. Temporary account creation (from information in the user database).
5. Spinup of personal RVD locker, which is mounted on the temporary home directory.

Each login process will remain running during its login session, in contrast to the present system under which the user's shell is exec'd in place of /bin/login. This will allow it to send periodic updates to the User Locator service; it will also allow it to perform the following actions on logout, if this is the last login session remaining for this user. (This cleanup will not be necessary for passwd-authenticated users).

1. Termination of all process owned by the user.
2. Destruction of Kerberos authentication tickets.
3. Spindown of personal RVD(s) if possible.
4. Elimination of temporary account.
5. Notification of User Locator.

This has the effect of restoring the workstation to its state before login was run for the current user, except that system RVDs are not spun down. This is to allow other users to continue to use the system; in the event that there are no other users, system RVDs will be spun down by /etc/toehold.

## 3.2 /etc/toehold

/etc/toehold is principally responsible for system-wide operations. These include managing the X server and the system RVDs, and performing cleanup operations when there are no users on the system. Under normal conditions, /etc/toehold will wait for a keystroke on the console, which will tell it to bring up the system RVDs and start X; during this time it will periodically check to see if cleanup is necessary.

Cleanup operations will include:

- Stopping X.
- Forcible spindown of system RVDs, terminating all processes which depend on same.
- Rebuilding the /etc/passwd dbm database from the /etc/passwd file, thus eliminating any temporary entries which were accidentally not removed.
- Checking the workstation for software consistency and updating if necessary. (The method by which updates will be performed is at present somewhat vague.)

The cleanup procedure will be separated from the rest of

/etc/toehold as much as possible; it will be run when the console user logs out, if no other user is logged in. It will also be run periodically if no user is logged in, so that login windows will time out if they are not used.

## 4 User-visible differences

Most of the functions provided by the Toehold system can be performed invisibly, without requiring changes to the user interface presently provided by /etc/getty and /bin/login. Certain differences, however, are unavoidable.

### 4.1 X Startup

In order to meet the goal of minimizing the amount of software stored locally to the workstation, X will not be running on an unused workstation. Instead, the user will be prompted on the console to hit a key to start X; only when this is done will X be started and a login window created. A quick test of this feature determined that this would take about 30 seconds, not including additional time to spin up system RVDs if the workstation is in a totally dormant state.

### 4.2 RVD selection

If a user is associated with more than one RVD locker, he may need to specify which RVD lockers, if any, the login program should spin up. I have been assuming that each user will have at least one designated RVD locker which will replace his or her home directory and into which other lockers will be mounted. The exact behavior of this feature is dependent on the interface available to the user database, which has not yet been fully specified.

### 4.3 Background process termination

Some users might find it annoying that their background processes are terminated on logout, although it is my hope that most such processes are unnecessary and will not be missed. It might be worthwhile to establish a standard, no-password guest account which is not Kerberos-verified to allow users to run background processes which survive between login sessions if this is truly a desirable feature.

## 5 Implementation Issues

The design is set up in such a way that a framework for it could be constructed quickly, with most of the features missing; this would include:

- /etc/toehold with X startup, system RVD spinup, and cleanup facilities.

- /bin/login, with Kerberos authentication and temporary account creation/deletion, but without automatic RVD spinup.

Missing would be features dependent on being able to obtain RVD information from the user database, and notification of the User Locator Service, since neither of these facilities have been documented at present. These additional features could be added without much effort once the initial framework was in place.

Also need a floppy interface

→ .cshrc
+ .login } easier late.

M. I. T. PROJECT ATHENA

# Toehold in Release 5.0

by John K. Bartholomew and William J. Bryant

## 1. Introduction

The *Toehold* system manages login sessions on the Athena Workstation. It also provides a means by which Project Athena can update a workstation's software without having to disrupt its service. *Toehold* was designed to increase the functionality and manageability of the Athena Workstation environment:

- Users must be able to login to any public workstation, without having an account on each workstation. Because the workstation operating environment is not secure, network services must be able authenticate users reliably.
- A workstation should return to a very quiescent state when not in use. Because users don't have permanant accounts on public workstations, a remote filesystem must be attached or a temporary home directory must be created when a user logs in, and detached/removed when he or she logs out. Additionally, a workstation should not rely on network services while it is not in use.
- Workstation software needs to be distributed and updated automatically. One thousand workstations cannot be updated manually. The workstation must not become an obstacle to new software releases.

The *Toehold* system has two parts: a system management program, */etc/toehold*; and a user management program, the modified */bin/login*.

### 1.1. The /etc/toehold Part

The */etc/toehold* program "activates" a workstation when someone wants to use it, and "deactivates" it when it sits idle. An activated workstation is one that has its remote software libraries mounted; a deactivated workstation is one that does not. When a user begins a session on a workstation, */etc/toehold* imports the most up-to-date system libraries. System software updates are not performed on a per workstation basis; when a machine is activated, it receives the latest system software from the imported system libraries. */etc/toehold* also starts the X window system server and displays an *xterm* login window. After the user logs out, */etc/toehold* waits for 120 seconds, kills the X server, and deactivates the machine by running the */etc/athena/deactivate* script.

### 1.2. The /bin/login Part

The modified */bin/login* program manages a user's workstation session. As part of the login process, the program requests an initial ticket-granting-ticket on behalf of the user from the *Kerberos* database. (Please see the *Kerberos Technical Plan* for more a more detailed description of the workings of the *Kerberos* database facility.) If the user is registered, *Kerberos* will issue the ticket-granting-ticket, thereby authorizing the user to access network services (e.g., *PostOffice*, *NFS*, etc.). This ticket is returned in a message

encrypted with the user's password as a key. */bin/login* authenticates the user by trying to decrypt the ticket-bearing message with the user-supplied password. A valid message result from this decryption means the user supplied the proper password. It also yields his or her Kerberos ticket-granting-ticket.

*/bin/login* also attaches the user's NFS home directory, if one exists. If the user has no NFS directory and no local directory on the workstation, */bin/login* builds a temporary home directory. */bin/login* informs the Zephyr Notification Service of the login. When a user logs out, */bin/login* detaches the NFS home directory or removes the temporary home directory, if either exist, and destroys any active *Kerberos* tickets. It also informs Zephyr that the user has logged out.

## 2. *toehold* at Login Time

### *2.1. toehold's First 120 Seconds after Reboot*

*toehold* takes control of the console device, clears it, and displays the following message at a random location on the screen:

```
Hit any key to start.
```

*toehold* then waits for a keystroke. If 120 seconds pass and no one has entered a keystroke, *toehold* "deactivates" the workstation by running the */etc/athena/deactivate* script. *toehold* then continues waiting for someone to press a key.

### *2.2. When Someone Hits a Key ...*

When a key is struck, *toehold* uses the */etc/athena/activate* script to "activate" the workstation. This script uses the *attach* command to mount the workstation's */urvd* and */srvd* filesystems. When it has successfully mounted the libraries, */etc/athena/activate* exits and *toehold* resumes control.

### *2.3. toehold and X*

*toehold* then attempts to start the workstation's *X* server. If the *X* server does not respond within sixty seconds, *toehold* kills itself. When this happens, *init* restarts *toehold*. the screen clears, and the whole process begins again.

After the *X* server starts up, *toehold* uses *xterm* to display a login window as specified by the *ttyv0* entry of the */etc/ttys* file. *toehold* waits 120 seconds for someone to login. If no one logs in, *toehold* checks for remote logins, by checking if */etc/utmp* is not empty. If none exist, *toehold* deactivates the workstation with the */etc/athena/deactivate* script.

If someone logs in, */bin/login* takes control, and *toehold* waits for the login *xterm* to die (when the user logs out). If the *X* server dies while *toehold* is waiting for the login session to end, *toehold* unlinks the file containing the user's X process id (*/tmp/X0.pid, X1.pid*, etc. depending on the *X* display server number in use), removes the user's entry in */etc/utmp*, and then kills itself.

### *2.4. toehold after Logout*

When the *xterm* login window exits due to logout, *toehold* resumes control. *toehold* removes the user's entry from */etc/utmp* and uses *xterm* to redisplay a login window. If after 120 seconds, no one logs in and there are no remote logins, *toehold* uses the */etc/athena/deactivate* script to deactivate the workstation. *deactivate* restores */etc/passwd* from */etc/passwd.local*, effectively removing temporary entries, disallows

remote logins via */usr/athena/access_off*, and unmounts all active libraries via the *detach* command.

### 3. The Detailed Workings of */bin/login*

The */bin/login* part of the *toehold* system differs from the standard Unix login program in three ways:

- It authenticates logins with the *Kerberos* authentication system. If this fails, the local */etc/passwd* file is used.
- If the user logging in does not have an NFS home directory to attach, or a local account on the workstation, */bin/login* creates a home directory under */tmp/<username>* and copies in the contents of the */usr/prototype_user* directory so that the user has *.login*, *.cshrc*, etc. files.
- Once the user has logged out, it detaches all imported filesystems and destroys any outstanding *Kerberos* tickets.

When a user logs in to a *toehold*-running workstation, he or she is prompted for a username as in the standard Unix scenario. With the *Kerberos* database facility, */bin/login* must now query the *Kerberos* server for login authentication. User shell and finger information can then be obtained from Hesiod, the Athena nameserver. The authentication procedure can be outlined as follows:

- If there is no entry for the user in */etc/passwd*:

  1. If */etc/nocreate* exists, deny access (this eliminates temporary logins on a private workstation).

  2. Otherwise, get *Kerberos* tickets and verify the password. If a null ticket is returned, register the user with *Kerberos* using */etc/athena/go_register*. Deny access if any other errors occur.

  3. If the password is valid, allocate a user ID, and add a line to */etc/passwd* with a password field of "*". Other fields for the */etc/passwd* entry are provided by Hesiod. The *attach* command is used to mount the user's NFS home directory, if it exists, as */mit/<username>*. The Hesiod nameservice provides the mapping from username to NFS locker location. This arrangement was provided to allow temporary users of a public workstation. (Note that since all remote access are *Kerberos*-authenticated and the user's UID is remapped, the UID is not required to be the same from session to session.)

- If there is an entry for the user in */etc/passwd*, consider three cases:

  1. If there is no password, log the user in (without prompting for a password), but do not obtain *Kerberos* tickets. Do not create a temporary home directory. The user's home directory defaults to the directory specified in his or her */etc/passwd* entry. If this directory does not exist, / becomes the user's home directory. This allows purely local users (i.e. workstation only, not *Kerberos*-registered) to login.

  2. If the password is "*", prompt for a password and attempt to get *Kerberos* tickets. If successful, continue with the login, but do not create a temporary home directory. If the user is not *Kerberos*-registered, attempt to register the user with *Kerberos* using */etc/athena/go_register*. If that fails, deny access

to the user. This case covers users who have a permanent account on a private workstation, but should also be *Kerberos*-registered.

3. If there is an encrypted password, prompt for a password as usual. Attempt to get *Kerberos* tickets. If successful, attempt to mount the user's NFS home directory via the *attach* command. If the *Kerberos* password authentication fails, compare the user-entered password to the entry in */etc/passwd*. If these do not match, deny access to the user. This arrangement accomodates users users who have permanent accounts on a private workstation.

- If a user is logging in and *Kerberos* tickets are not obtained, a warning to this effect is printed.
- When the Zephyr Notification Service becomes available, send it a message about the login.
- At logout time, notify the Zephyr Notification Service of the logout. All imported filesystems are detached and any outstanding *Kerberos* tickets are destroyed.

If *Kerberos* cannot find a full username and password entry for a given user, *Kerberos* will return one of the following error codes:

| | |
|---|---|
| **KDC_NULL_KEY** | *Username* **doesn't have a** *Kerberos* **password.** |
| **KDC_PR_UNKNOWN** | *Username* **doesn't exist in** *Kerberos* **database.** |
| **INTK_BADPW** | *Username* **entered an incorrect password.** |
| **KDC_PR_N_UNIQUE** | *Username* **has multiple entries in** *Kerberos* **database. This usually indicates a database error.** |

If the *Kerberos* server returns any of these error codes, the user cannot be authenticated as a *Kerberos* user. */bin/login* will print a short message concerning the error. If KDC_NULL_KEY is returned, */bin/login* will attempt to register the user with *Kerberos* by running */etc/athena/go_register*.

If the user's name and password are located, *Kerberos* returns the user's ticket-granting-ticket in a message encrypted with the user's private password. The user-supplied password is used to decrypt this message. If the message is decrypted, the user is authenticated as a *Kerberos* user, and his or her ticket-granting ticket is deposited in the workstation's (*/tmp*) directory. */bin/login* notes that this user has been authenticated by *Kerberos*.

### *3.1. /bin/login* and the Home Directory

After the user's password has been authenticated, */bin/login* determines whether or not this user has a local directory, by trying to *chdir* to that directory. Even if the user has an NFS home directory (*/mit/<username>*) that has been mounted on his or her private workstation, the local permanent directory (*/site/<username>*, as specified in the */etc/passwd* entry) becomes the user's home directory. In the event that no local directory exists (on a public workstation, for example), and the NFS home directory has been mounted, then the NFS directory becomes the user's home directory.

If the user has neither a local home directory nor an NFS directory, then *make_homedir*() is called to create a temporary home directory (*/tmp/<username>*) for the user. If the attempt is successful, the *make_homedir*() gives the user ownership of the directory, sets the directory's permission bits to 0755 (giving write access only to the user), and sets the working directory to that directory.

### 3.2. Creating Files for Temporary Home Directories

*make_homedir*() copies all the *.login*, *.cshrc*, etc. files from the workstation's */usr/prototype_user* directory to the user's temporary home directory. If it cannot open the prototype-user directory, the function issues an error message to standard error and removes the temporary home directory.

After *make_homedir*() has completed the copying, it changes the new files' ownership to the user and returns control to */bin/login*. If *make_homedir*() is successful, */bin/login* displays the following message:

```
WARNING -- Your home directory is temporary.
It will be deleted when you log out.
```

*/bin/login* then completes the login process, making the newly created directory the user's (temporary) home directory.

### 3.3. The Rest of the Login Procedure

After */bin/login* has set up the user's home directory, it makes entries for the user (in */etc/utmp* and */usr/adm/wtmp*), then uses the *dofork*() function to fork off the rest of the login process. The child process continues the rest of the login process (for example, executing the user's *.login* and *.cshrc* scripts if the *csh* shell is utilized).

### 3.4. If Kerberos Isn't There

If *Kerberos*, or part of the network connecting a workstation to *Kerberos*, is down, the *Kerberos* authentication mechanism will be unavailable. */bin/login* will not be able to verify passwords of *Kerberos*-registered users. On a public workstation, this means that no one will be able to log in, except as the root user. On a private or semi-private workstation, users registered in the workstation's */etc/passwd.local* file will still be able to login using their local password, but will find their actions restricted by the lack of any *Kerberos* tickets for network services.

Multiple servers are employed to make the *Kerberos* facility reliable and to minimize down time.

### 3.5. When the User Logs Out

The parent */bin/login* process waits for the child's shell process to die (when the user logs out), then destroys any outstanding *Kerberos* tickets located in */tmp*. It also recursively removes the user's temporary home directory, if one exists. The local password database (*/etc/passwd*) is also purged of the user's entry. If no one logs in to the machine within the next 120 seconds, and */etc/toehold* fails to detect any remote logins (*/etc/utmp* is empty), the */etc/athena/deactivate* script deactivates the workstation. This involves cleanly terminating the X server, detaching all filesystems, and reinitializing */etc/passwd* from (*/etc/password.local*).

### 3.6. Remote Logins to Workstations

A user can *rlogin* to an activated workstation subject to the status of *access_on*/*access_off*. On a public workstation, *access_off* is set both when the workstation is activated, and when it is deactivated. The owner of a private workstation may place an *access_on* command in the */etc/athena/activate* (or *deactivate*) scripts, in order to allow remote logins while the machine is being used (or when it is not being used). As with a user logging in to the workstation's console, */bin/login* will attempt to authenticate a remote user with the *Kerberos* system.