The Inner Moat - Security when Users have Access to your System

Presentation Number 4055

Steen Hansen Hviid

University Technology Services
The Ohio State University
2650 Kenny Rd
Columbus OH 43210

Phone: 614/292-2501
E-mail: hansen+@osu.edu

Introduction

Attacks from the Internet catch the headlines, but many security problems are caused by company insiders. Insiders often know more about the systems and may already have some access, so they have an advantage over intruders coming from the outside. The vast majority of incidents used to be created by insiders, but now it is more evenly distributed between insiders and outsiders. The FBI and the Computer Security Institute conduct an annual Security Survey, which shows that about half of all security incidents are caused by company insiders.

	Insider	Outsider
1997	43%	47%
1996	46%	53%

The reader survey in the July 1996 issue of SunWorld On-line (www.eu.sun.com/ sunworldonline) came up with a similar result (45% outsiders, 50% insiders).

Insiders are not becoming more honest, it is the overall problem that is growing, with more attacks coming from the Internet. The GAO (Government Accounting Office) reported in 1996 to the US Congress that US military computers were subjected to an estimated 250,000 attacks in 1995, 65% of these were considered successful. They also remark that the number of attacks doubles every year.

This paper deals with simple means to strengthen security when users already have access to your system (some insiders may not have access, but most probably do). These methods can also be your last line of defense against intruders from the outside, who have been able to breach your outer fortifications, and gained some access to your system.

The only computer system that is absolutely safe, is one that is turned off and placed in a windowless room with the door welded shut. It is possible to break into everything else, though it can be made so hard to do that most intruders will choose an easier target. Security is always a compromise.

Who and what?

There are many ways security can be breached, some more serious than others. Identifying the different kinds can help you determine what type of measures to consider.

◊ Reading secret information: Sensitive medical data, salary information, trade secrets, etc.
◊ Modifying information: Change account balances for personal gain, sabotaging the system, or simply "having fun."
◊ Unintended data loss: Data is unintentionally modified or deleted, by no ill will.
◊ Denial of service: System is tied up so the users cannot use it.

The persons that break into systems, or use their access improperly, do it for a variety of reasons, such as:

◊ Curious, "just looking around"
◊ Get a thrill out of doing it
◊ Industrial spies from other companies or organizations
◊ Persons seeking personal advantage in accessing/modifying the data
◊ Disgruntled employees, seeking a way to get back at the company

Physical Security

Access to the computer room should be limited. It is much easier to break in, or do damage, with physical access. An experienced person would be able to boot in single-user mode, and gain complete access.

The backup tapes should be well protected, such as by a fireproof safe. A safe will make it harder for a person to steal the tape, and copy files from it, besides protecting the tapes better in case of fire. We purchased a data-grade safe a couple of years ago for about $600.

A recent backup set should also be kept in another building, or maybe at a bank. Relying on a person to store a set of tapes at home is not as safe, and inconvenient when that person goes on vacation - or attend HP World.

Divide and conquer

No matter how good the security is, some users will have to be given access to the system. If possible, do not give anyone complete access to everything. This can be difficult, especially at a small site, but consider splitting parts of sensitive functions between multiple people. If you receive payments, let one person receive the payments, and another balances out the system and deposits the money.

The primary line of defense is the password

I am sure you have heard this a million times, but just one more time: Make sure the users change their passwords, and do not share them or write them down! Breach of passwords are the primary cause of problems.

HP-UX has the ability to age the passwords, forcing the users to change it. We expire the password every 90 days at our site. If we do it more often, the users will be more likely to write it down and tape it under the keyboard or in their desk drawer.

The largest problem we see now is when a user does not complete the password change (i.e., do not offer an acceptable password) and then does not understand that the password has not changed. It would be nice if the system would display a message that states the password has not changed, as many users become confused and think they are locked out of the system.

Another problem is that some users simply alternate between the same two passwords. The ability to store a password history would be another welcome enhancement.

The password and group files

The two central files to keep track of the users are the /etc/passwd and /etc/group files. The /etc/passwd file stores such information as the name of each user, what group they are in and their password in encrypted form. /etc/group contains a list of valid user groups. Everybody must have read-access to these files for Unix to work. This creates a problem, as there are many password cracking programs available that in a few hours could crack some of the weaker passwords. To counter that, HP provided in HP-UX 9 the ability to create a shadow password file, stored in /.secure/etc/passwd and only accessible to the super user. In HP-UX 10, a more elaborate scheme is available in the /tcb/files/auth structure, which also stores the last time the user logged in, password expiration information, etc. In both cases, it is very easy to set it up, using SAM.

There are also facilities for auditing a multitude of events. Enabling them would generate a lot of logging information, that only very secure sites would need. It is not necessary to enable the other security features to use the shadow password file.

There is one caveat, though. At rev 9, SAM is not able to reverse the procedure, and remove the shadow passwd file, which is necessary to do before upgrading to rev 10. A simple script can be made that builds a new passwd file based on the /etc/passwd and /.secure/etc/passwd file (available from my web-site, see later). At rev 10, it is easy to convert back from the shadow password structure.

Before using these features in HP-UX 10, make sure to first install the patches for “trusted systems” and the “passwd” command (available from HP’s web-site).
At rev 10, HP added new library routines to access the password information for trusted systems. This means some programs that need password information will not work without modification. I had to modify a POP-mailer to get it to work (available from my web-site).

To create the shadow password file, enter “SAM” and select “Auditing and Security” and then “System Security Policies.” You will then get this screen:

```
+--------------------------------------------------------------------+
|You need to convert to a Trusted System before proceeding.  The      |
|conversion process does the following things:                        |
|                                                                     |
|   1. Creates a protected database on the system for storing security |
|      information.                                                   |
|   2. Moves user passwords in "/etc/passwd" to this database.        |
|   3. Replaces all password fields in "/etc/passwd" with "*".        |
|                                                                     |
| For more details, refer to the "System Security" chapter of the     |
| "System Administration Tasks" manual.                               |
|                                                                     |
| Do you want to convert to a Trusted System now?                     |
+--------------------------------------------------------------------+
| [ Yes  ]                                                 [[No  ]] |
+--------------------------------------------------------------------+
```

Select “Yes,” and the conversion will be done in a couple of minutes. This conversion can be done with people logged in on the system, though people may not be able to login during the few minutes it takes to build the shadow password structure in the /tcb/auth directory.

Security Policies

When you have converted to the shadow password file, other facilities become available and you will be asked to specify security policies for the system. Please go through the four options carefully, though you can always change the settings later.

- Password Format Policies
- Password Aging Policies
- General User Account Policies
- Terminal Security Policies

Password Format Policies allows you to specify restrictions on the password, such as whether the user may select it, or the system should generate one. If more than one option is chosen, the user is allowed to choose how the password is changed. To retain the same method as before engaging “secure system,” choose “User Specifies” only.

```
[ ] System Generates Pronounceable
[ ] System Generates Character
[ ] System Generates Letters Only
[X] User Specifies
```

Use Password Aging Policies to enable the aging of the passwords and set how long they can stay unchanged before the system will force the user to change it. The total lifetime of the password can also be set, so if the password has not been changed within a period, the account is blocked. Make sure to set this one much higher than the password expiration date, or infrequent users will often become locked out. Password aging is disabled by default.

General User Account Policies specifies how many times a user can make an unsuccessful attempt at logging in, before the account is disabled. This feature prevents brute-force attacks. This feature is disabled by default. If you choose to activate this feature, consider setting the level higher, the default only allow three attempts - I have users who sometimes take ten attempts to get in. Deactivated users can be reactivated using SAM, under the “Accounts for Users and Groups” -> “Users” selection.

Another feature allows you to limit the lifetime of the account, practical for temporary employees, etc.

Terminal Security Policies set the delay between unsuccessful logins, and also allows for blocking ports that have high numbers of unsuccessful logins.

You might want to enable the features one at a time, to see the effect. There are many more features available that are not mentioned here.

With the secure system enabled, you can also specify time limits for when a user can login. For instance, they can only login on weekdays, and during certain times of the day. This check is only done upon login, it does not prevent the user from staying logged in after hours.

Multiple user groups

A user is usually only a member of one user group at a time. It is possible for a user to be in several user groups at the same time, by registering them in the /etc/logingroup file. This file has the same format as the /etc/group file, so one can simply make them the same by linking them.
If a user is only a member of one group, there is no need to list them in the /etc/group file, the entry in /etc/passwd takes care of that.

Unless the user uses the "newgrp" command, the users' primary group will continue to be the same as the one listed in the /etc/passwd file. When a file is created, the primary group will be the group-owner. The difference is when accessing a file. Then any of the groups the user is registered for in the /etc/logingroup file is considered for access.

For the secondary groups, it is simply to list the user, as shown below:

```
/home/steen> ll /etc/logingroup
lrwxr-x---  1 root  sys  5 Apr 30 16:11 /etc/logingroup ->
/etc/group

/home/steen> grep steen /etc/group
view:*:31:steen,jdoe,metoo
dcadm:*:34:steen,nfuller,kcoyan
net:*:36:steen,khimmelb
```

In the abbreviated sample above, we can see the secondary groups I am a member of: "view," "dcadm" and "net."

Inactive Users

People leaving the company, or who simply no longer need access, should have their accounts removed. In reality, it can be hard to get notification when employees leave, but try to cooperate with Human Resources to be included on their termination check-list.

As an aid to help us find inactive users, I built a program that goes through the login records (in the /etc/wtmp file) and flag users that have not logged in for at least a hundred days. This program is available from my web-site.

Basic protection of files and directories

When you list the content of a directory with the “ll” or “ls -l” commands, you will see the permissions for each file or directory, such as:

```
drwxrwxr-x  steen users       my-dir
-rwxr-xr-x  steen users       my-file
```

The example above shows a file and a directory. The directory can be recognized by the initial “d” in front of the permission string. The rest of the permission string indicates the access permissions for the owner “steen,” the group “users” and everybody else.

Here is an overview of the meaning of the permission letters for files and for directories:

r	File:	Permission to read the file.
	Dir:	Permission to list the contents of the directory. Not needed to access a file/dir in the directory, if the name is known.
w	File:	Permission to modify or delete a file.
	Dir:	Permission to create or delete files in the directory.
x	File:	Execute a program.
	Dir:	Access the directory, such as “cd” to it. Needed for accessing any object inside the directory.

Example: Set file to be writable by owner (dbman), readable by group (users), and no access to anyone else.

```
chmod u=rw,g=r,o-rw myfile
ll myfile
-rw-r------   dbman   users    myfile
```

Example: Set the file to be readable by everyone, except the group (users):

```
chmod g-rw,o=rw myfile
ll myfile
-rw----r---   dbman   users    myfile
```

Access Control Lists

An ambitious project took place at MIT in the sixties to develop a modern operating system. It became Multics, a forerunner of Unix and the lesser known Primos. One feature of Multics was the Access Control List, or ACL (pronounced “ack-l”), which is a superset of the basic Unix permissions. ACL’s are used today by HP-UX and NT.

ACL’s allow us to specify a list of access permissions, instead of just the three traditional Unix permissions. It is also possible to specify permissions for specific users, without making the user the owner of the file. It is even possible to specify different permissions for the same user, depending on which group the user currently is in. It is implemented in HP-UX as a superset of the regular permission bits, with a limit of 16 permission entries per file or directory.

The items on the permission list is stored as sets of “user and group.” It is possible to specify the permission for the same user, depending on which group the user currently uses as the default group (using the “newgrp” command). In most cases, it is enough to use wildcards, which here is the percent-character. Examples:

john.users	Permission for user john in group “users”

jack.%	Permission for user jack, regardless of group
%.users	Permission for all members of the group “users”
%.%	Everyone

The “ls -l” command shows the presence of an ACL, by putting a plus-sign at the end of the permission string. The “lsacl” command reveals the complete set of permissions:

```
ll myfile
-rw-r-----+  1 steen    dc           774 Dec 24 15:59 myfile

lsacl myfile
(dbman.%,rw-)(steen.%,rw-)(%.dc,r--)(%.osudds,rw-)(%.%,---) myfile
```

We can see that the users “dbman” and “steen” both have read-write access to the file. Members of the “osudds” group also have read-write access, while the “dc” members only have read access. The rest of the users have no access.

If we were to add another set of permissions, to add read-write rights to user nancy, we simply enter:

```
chacl nancy.%=rw myfile
```

And the result is now

```
(dbman.%,rw-)(steen.%,rw-)(nancy.%,rw-)(%.dc,r--)(%.osudds,rw-)(%.%,---) myfile
```

If a user is a member of multiple listed groups, the access permissions are cumulative so the highest level of access is granted.

```
ll myfile
-rw-r-----+  1 dbman    grp1       0 Apr 19 14:17 myfile

lsacl myfile
(dbman.%,rw-)(%.grp1,r--)(%.grp2,---)(%.%,---) myfile
```

In the above example, the user is a member of both “grp1" and “grp2" and thus will gain read access to the file.

Using the @-sign, it is possible to specify permissions for the owner and group-owner of the file. This command sets the permissions for the existing owner of the file to “rw”:

```
chacl @.%=rw myfile
```

There are a couple of caveats when using ACL’s: They can unintentionally be removed when using the “chmod” command. To make HP-UX backward compatible with “standard Unix,” use of the “chmod” command on an ACL-protected file or directory will remove the ACL. It is therefor a good idea to use ACL’s sparingly. Another problem is that only “fbackup” is able to backup ACL’s to tape. The other tape tools (tar, cpio, pax, dump, etc.) ignores it, though they do display a warning message.

ACL’s are also not supported by the Veritas file system on HP-UX 10, though it is said to be in later revisions.

With these limitations, it is good to only use them sparingly. This also limits the overhead associated with a more detailed access specification.

In one case, we would like one group of users (“osudds”) to have both read and write access to all files in a directory, while another group (“devel”) has only read rights. Users outside these groups should not have any access. What we did was to use an ACL to protect access to the directory, so we know that anyone in the directory is a member of one of the two groups. We can then use the regular Unix permissions scheme to give read-write access to the “osudds” group, while we only allow read-only access to the rest - which only can be the “devel” group.

```
  lsacl /prod
  (dbman.%,rwx)(%.devel,r-x)(%.osudds,rwx)(%.view,r-x)(%.%,---)
/prod

  lsacl /prod/*
  (dbman.%,rw-)(%.osudds,rw-)(%.%,r--) /prod/file1
  ...
```

Avoid using specific user-id’s in ACL’s, it can be a nightmare to maintain, and there is a limit of sixteen ACL entries per object. Use groups where possible.

Setting default permissions

Files created by users have no protection by default, but that can be changed. The “umask” command sets the default protection of newly created files and directories. It can be set in /etc/profile to effect all users (except VUE-users). Users can change it again in their own “.profile,” if another mask is desired. A typical setting is “umask 027.” Here is a list of common masks:

umask 000	All rights to everybody (default)
umask 007	All rights to user and group, none to everybody
umask 027	All rights to user, read-only to group, none to everybody (typical)
umask 077	All rights to user, none to everybody else (safest)

Sticky bit

The “sticky bit” is an extra protection feature that can be put on a directory to prevent users from deleting or renaming files owned by other users, even when they have write-access to the directory. This can be used to prevent some problems in shared directories, such as /tmp and /var/tmp, /usr/share/man/cat*.Z and directories with log files created by the user processes.

To set the sticky bit for /var/tmp, enter:

```
chmod o+t /var/tmp
```

Setuid programs

Unix has the feature that a user can become another user while executing a program. This is a way to give people access to specific files through programs, and not by any other means, so you can control how the files can be manipulated. A lot of the system functions, such as printing ("lp"), E-mail ("mail", "mailx", "elm") and change of password ("passwd") use this feature. If you have applications written in a compiled language, this is an excellent method to protect the files. To be on the safe side, create a special user to own your application files, do not use the superuser account. If your programs have a security problem, a user can only attain the rights of the application owner, the entire system is not compromised. Many security holes have been found, and are still being discovered, in setuid programs.

- Never let superuser be owner of your own setuid programs.
- Always use absolute paths in filenames.
- Always use absolute paths if executing Unix commands, and do so sparingly.
- Never let a setuid program execute a script of any kind
- Always make sure input cannot overrun buffers, and thus modify the executable code. This is where many Unix security holes have been found.
- Make the setuid program as small and simple as possible.

The back side of setuid facility is that it can also be used to put in back doors to the system. If a person once broke in, and you have changed the password, the intruder may have left such a program behind to gain superuser rights. This command lists all setuid files on the system:

```
find / -perm -u+s -print
```

Be very suspicious of any setuid program that resides outside the system directories.

Special files

Peripherals, such as terminals, network card, tape drive and disks are controlled by device files in the /dev directory. In most cases, users should not have access to these devices, as they access them through the operating system. Hewlett-Packard notes in their manuals that their devices have built-in security checks, but that does not help in all cases, and third party products may not be as well behaved. If in doubt what a device file is for, use the "lssf" command to display some information about it, and ask the vendor.

Here are some you may want to remove any world-access (chmod o-rwx):

♦	/dev/console	Device for console
♦	/dev/systty	Device for console
♦	/dev/syscon	Device for console
♦	/dev/vg*	The directories used by LVM
♦	/dev/klog	kernel logging info
♦	/dev/kmem	kernel memory access
♦	/dev/dsk	Buffered disk access
♦	/dev/rdsk	Raw disk access

Some devices must allow everyone write-access, such as those used for logins over direct wires (/dev/tty#*) and over the network (/dev/ttyp*, /dev/ttyq* ...).

At our site, we plug in the next backup tape during the day, and a nightly cron-job automatically does the backup. If someone had access to the device-file for the tape drive, it would be possible to restore files from the old backup on the tape once it's been plugged in awaiting the nightly backup. To protect against that:

- Remove all world-access to the tape device files (/dev/rmt/*)
- Change the group ownership to a separate group, such as "tape"
- Add users who need access to the "tape" group in the /etc/logingroup file

If only the superuser uses the tape, omit the last two steps.

Looking over the system

So where to strengthen security on the system? The COPS security-tool has been around for years, and is still quite usable. It is a collection of programs that looks over the system and reports potential trouble-spots. It is created by Dan Farmer, the father of the infamous SATAN network scanning program, while at the Computer Emergency Response Team at Carnegie Mellon University. The tool is available for free from CERT's FTP site.

Searching for writable files and directories

Check for world-writable files - files writable by anyone on the system. Also check for files writable by generic user groups, such as "users."

The "find" command is very handy for listing files and directories with potential problems. Here are some common uses:

List files writable by all users:
```
find / -type f -perm -o+w -exec ls -l {}\;
```

List directories writable by all users:
```
find / -type d -perm -o+w -exec ls -dl {}\;
```

List files writable by all in "users" group:
```
find / -type f  -group users -perm -g+w -exec ls -l {}\;
```

To avoid searching through CD-ROM's and NFS-mounted filesystems, add the "-fsonly hfs" option to the find command.

Of special interest are those directories that hold executable programs, and are in the search-path of users and the superuser. This script checks those:

```
pathlist=`echo $PATH| tr ":" " "`
for dir in $pathlist
do
  find $dir ! -type d -perm -o+w -exec ls -l {} \;
  find $dir -type d -perm -o+w -exec ls -ld {} \;
  find $dir ! -type d -perm -g+w -exec ls -l {} \;
  find $dir -type d -perm -g+w -exec ls -ld {} \;
done
```

Most cases of world-write access ought to be removed, but some are necessary for the system to function. Examples are the temporary directories (/tmp, /var/tmp), the man-page decompression directories (/usr/share/man/cat*.Z) and possibly some directories used to send print requests, create application log entries, run UUCP, etc., depending on your setup. Be careful when removing permissions, do them one at the time.

Some files and directories are protected by the directory above, so it does not matter what their permissions are, since only authorized users can even access the directory. This can even be used to make more detailed permissions, as mentioned under ACL's.

When checking for files that are writable by everybody, or specific groups, it is not sufficient to prevent write-access to a file. If the user has write-access to the directory it is in, or ANY of the directories higher up in the tree, the user can delete the file.

Watching for changes

Once you have set up the security, it's time to make sure it stays in place. There may be people who have administrative access to your system, without you knowing it, or some have access for specific purposes, and make mistakes. We had a programmer who used his access to export the entire root disk via NFS, so he could mount his home directory on his workstation. A big security risk that a watchdog program soon discovered. About a year later, someone broke in to another computer on campus that exported the root filesystem. The intruders copied the /etc/passwd file, decrypted some of the passwords, and started handing out passwords to other students (the story is told in the July '95 issue of HP-UX/USR).

I had a simple script running nightly, that alerted me to the change the programmer had made to the /etc/exports file. The script was very primitive, as it simply did an "ll" on some of the system directories, and then compared the output with that from the previous run. Any changes to the date and file size would show up when compared with the previous list. That simple script saved the day, but is inadequate for detecting intruders who may try to hide their tracks. It is simple to change the modification date on the file with the "touch" command, and modifications could be made so the file size does not change.

A better system is the script below, which uses a check-sum to verify that the files have not changed. It compares the list with the list from the previous run, and sends you an E-mail if any changes happen. It also checks if any setuid programs are modified, deleted or added. Very simple to implement. Let cron run it weekly or more often.

```
# cklist - check for changes to system setup files and setuid files
# Steen Hansen Hviid
cd /usr/local/etc
rm -f cklist2
if [ -f cklist1 ]
then
  mv cklist1 cklist2
fi

find /etc /sbin /usr/bin /usr/sbin /usr/lbin -type f -print | sort >
cktmp
find / -perm -u+s -print | sort >> cktmp

while read line
do
  if [ -f $line ]
  then
    cksum $line >> cklist1
  fi
done < cktmp
rm cktmp
diff cklist1 cklist2 > cktmp
if [ $0 != 0 ]
then
  mailx -s "changes to setup" root < cktmp
fi
rm -f cktmp
```

A much better system is “tripwire,” created by Gene Spafford and Gene Kim of Purdue University. It keeps a database of information for each file on the system, such as the i-node number, access- and modification dates, size, ownership and multiple digital signatures. A setup file contains rules for what changes to accept. No changes are accepted for the executable files, while it is acceptable that the content of the /tmp directory changes, as long as the permissions on the directory itself does not change.

Tripwire is available for free from the COAST project at Purdue. The setup takes some effort, but a template for HP-UX 10 is available from my web-site. The remaining job is mostly identifying what NOT to check and report changes on. It takes about two hours to run on my G50, but the reward is a good measure of security against unexpected changes to the system.

Restricting database access

The client/server model gets all the press, but many systems are still host based, where the user logs in using terminals and terminal emulation on PC’s.

Some of the databases, such as Oracle and Sybase, place the databases on separate disk partitions, that are only accessible to the database daemons. Here the databases are well protected, as the operating system itself prevents any direct user access. Other databases, such as uniVerse and Progress, require the user to have Unix-access to the database. This makes protection of the databases more complicated. One can rely on the dubious “security by obscurity” where the databases are kept in directories the users cannot list (the user only has “x” permission, not “r” permission), hoping they will not find out the names of the database files. A better method is to lock the users inside the application, never allowing them access to the Unix prompt.

On two Unix servers I administrate, we secure access to the databases by locking the users into the application. On one site, where we have multiple programmers, we only allow the programmers read-only access to the databases, which they use to investigate problems and copy test-data from. Otherwise they use a set of development and shadow databases for development and testing.

If a database is used to manage the data, the users often run an application instead of manipulating the data using raw query commands, such as SQL. Besides being much friendlier to use, the application can ensure the integrity of the database files and control the access to the databases by function, granting the users permission to do specific tasks. Another method is to grant access to specific databases and fields, if the DBMS supports it.

It is tempting to add other services to the user's menu, such as e-mail, but be careful! If you allow the user to use some Unix mailers, the user may be able to launch a shell or an editor from it. The elm mail reader has an option ("-b," if yours do not have it, get the patch) that prevents launching a shell, but the user may still select which editor to use, and editors can also be used to edit other files.

We create users with a script that calls the "adduser" command. This command has the feature that it makes the user the owner of the home directory, not very practical when sharing. To avoid having the ownership of the data files changed, we have a separate home-directory for all production users. It only contains a ".profile" file and a ".forward" file.

The ".profile" used for our captive users, contains:

```
# disable break and hangup
trap "" 1 2

# set the file mask
umask 007

# go to production directory
cd /prod/main

# Check that access is allowed
if [ -f LOGIN_BLOCK ]
then
# Display message file when access is blocked
  cat server.down.msg
  sleep 10
else
# Enter DBMS environment
  exec /opt/database/start
fi
echo
echo "Goodbye from OSUDDS"
echo
exit
```

When a user has a login to your system, he may also get unintended access to other services. Some network services are based on the user id, but not affected on the restrictions you put on the regular login.

If FTP is enabled on the system, the user can use this tool to look at files, and possibly modify or delete them. The script we use to create users also updates the /etc/ftpusers file, which stores the users that are *not* allowed to use FTP. If a user attempts to use FTP, the attempt will be denied and the incident logged to syslog. I have seen a couple such attempts, though they did not seem to be with malicious intent. One was simply a user who did not know the difference between telnet and FTP! HP-UX does not have the simpler method of just listing who is allowed to use FTP, as it does for “at” and “cron” (/var/adm/cron/at.allow, /var/adm/cron/cron.allow).

If a user has a login on your system, the system will receive mail and store it in /var/mail. If the user cannot retrieve the messages, they will sit there forever and take up disk space. On our systems, all the “captive users” have the same home-directory, where a *.forward* file will redirect any incoming mail to the database administrator.

If you have pop-mail service enabled on the system, the user will be able to use your system as a mail server. Even if you redirect incoming messages, they can still send messages using your system. Some pop-mail server packages (like “qpop” from Qualcomm) allow the administrator to specify who can use the pop-mail service.

Other network services may be enabled, that relies on a user id. Check the /etc/inetd.conf file.

More information

Chapter 12 in the “HP-UX System Administration Tasks,” has a lot of good information. It is available on the LaserROM distribution.

“Practical Unix Security,” by Simson Garfinkel and Gene Spafford. O’Reilly & Associates.

You can subscribe to the HP Security Bulletins through the HP Electronic Support Center:
http://us-support.external.hp.com

Computer Security Institute: http://www.gocsi.com/csi

My home-page: http://wwws.us.ohio-state.edu/~steen/sw has:
“qpop” pop-mailer for HP-UX 10 with shadow password files
“tripwire” setup file for HP-UX 10
“inactive” software finding inactive users.

Tripwire itself is available from the COAST project at Purdue University: ftp://coast.cs.purdue.edu

COPS is available from CERT: ftp://info.cert.org/pub/cops/

QPOP mailer is available from Qualcomm: ftp://ftp.qualcomm.com/quest/service/unix/popper

The picture on the front page of this paper is of the Kronborg Castle, north of Copenhagen, Denmark. It was built in the 16th century and forms the setting for Shakespeare’s play “Hamlet.”