# Global Information Assurance Certification Paper

# TOO MUCH, TOO LITTLE

## eTrust Access Control for HPUX



Derrick Chacko
September 22, 2004
GIAC Security Essentials Certification (GSEC)
Practical Assignment Version 1.4b – Option 2

## 1.0 ABSTRACT

This paper will seek a solution to the problem that I faced and many security administrators also are confronted with. The dilemma is that many times there are too many servers to secure but too little resources to do it with. The risk involved is that because security administrators have other responsibilities, servers cannot be properly secured. In order to deal with this issue, the company I worked for purchased a product that assisted us in securing our servers.

The product from Computer Associates was called eTrust Access Control. This security application enabled us to control access to resources and files listed within the database. In addition, the security application increased the efficiency of the security staff because it assisted in user administration and user issues. This paper will delve into detail on how install and configure Access Control, and will also cover the benefits offered by this product.

## 2.0 DOCUMENT CONVENTIONS

According to the GIAC certification Administrivia, a paper's font format should be 12 point Arial font. However, this paper has utilized different sizes and typefaces in certain areas in order to distinguish these parts and as well as in assisting with the ease of reading. The format of words that are used in this way include the following:

| | |
|---|---|
| *Computer output*<br>*Commands*<br>*Configuration file* | The output of software installation, commands and elements in configuration files are portrayed in this manner. |



## 3.0 BEFORE

### *3.1 Defining the problem*

It is widely known that UNIX is an open source operating system that was not designed with security in mind. Different vulnerabilities are constantly being found on the UNIX operating system and as soon as you think you are up to date on your security patches a new one is released. The company that I worked for understood this threat. They realized that the threat was not only external, but

also from within the organization itself. If any of our UNIX systems were compromised then confidentiality, integrity and availability all would be compromised.

The problem that we faced is that we had a team of three individuals attempting to secure over 200 HPUX servers. We were understaffed and did not have the manpower to properly secure this many servers. We were becoming extremely backlogged in our work. Not only were we responsible for securing the servers, but we also had other responsibilities such as account administration, fixing user problems, etc. Due to the fact that we had other responsibilities, little time could be spent securing the servers. The problem with this setup was that servers were not being secured and a solution was paramount.

Therefore, the group formed two goals that if met would secure our servers to an acceptable level. The first goal was that we wanted a product that would secure our servers and at the same time increase the efficiency of the security staff. Second, we wanted a product that would provide a layered defense to our HPUX servers.

## 4.0 DURING

### *4.1 Production Choice - Access Control*

In order to meet the goal of having access control as a means of security, the product that the company purchased that fit our requirements was a product from Computer Associates called, eTrust Access Control. This security application was purchased to mitigate many of the issues that we faced and had many functions that greatly increased the security of our servers.

The security application, eTrust Access Control, is a software solution that provides security for Open Systems, specifically an active, comprehensive security software solution tied dynamically to the operating system. Every time a user requests a security-sensitive operation (such as opening a file, switching user or obtaining a network service), Access Control intercepts the event in real time and evaluates its validity before passing control to the standard UNIX functions. In essence the UNIX operating system and Access Control must both agree before access is granted to a user to access a resource. (eTrust Access Control For UNIX 1-1). If any of the two denies the request, the action is denied.

The security of the 200 HPUX servers could be performed from a centralized server thus increasing the speed and efficiency at which we could perform our duties. Not only could the security be done from the centralized server, but the user administration and password resets could be performed as well. The benefits of this product were numerous and the following paragraphs will describe in detail how to install and configure the eTrust Access Control software.

## 4.2 INSTALLING ACCESS CONTROL

### *4.2.1 Memory and Disk Space*

The first step in installing Access Control is to ensure that there is sufficient memory and disk space on the partition where the product will be installed. The partition will need to have enough space for the program itself, the Access Control database, binaries, logs, etc. The OS installed at the company I work for was HPUX11. An OS of this type usually requires at least 90 Megabytes (MB) of disk space and15 MB of physical RAM. The product must be installed using the root account. I installed and configured the security application on many of the HPUX servers at my company.

The following paragraphs will cover the installation of Access Control, modifying the configuration file, converting to fail mode (and finally the security and benefits that access control provides), which will explain why this product was selected.

### *4.2.2 Installation Procedure*

**1) Log in as the root account.**

The security application must be installed with root level privileges.

**2) Create /usr/seos directory.**

It is recommended to create a separate partition in which the software will reside. The system administrator, depending on the setup of the company, will probably perform this aspect. All of the application's binaries, logs and database will be kept in this partition. In our example, we will create a directory called /usr/seos in which all the files will be kept.

**3) Grab necessary files from CD**

- The "install_base" file will need to be grabbed from the CD. This file will install Access Control from the tar file.

- The second file is the compressed tar file, which contains files necessary for the installation of the security application. The name of the file is _HPUX11_510.tar.Z

**4) Run the install base script, by using the install_base command.**

The installation script will find the appropriate compressed tar files that you have copied over from CD. The instructions below are screen shots from installing on a HPUX11 server.

```
# ./install_base
```

The section below basically provides information concerning the install. It informs the administrator of the following: the target directory of the installation, the location of the compressed tar file, the interactive setup option and finally which eTrust packages will be installed such as the client, server or both packages.

*Target Directory : /usr/seos*
*Source installation file(s) : /usr/seos/./_HPUX11_510.tar.Z*
*Group owner of files : 0*
*Interactive setup : yes*

*Installing the following eTrust package(s):*

- *Client package*
- *Server package*

**5) Answer Y for yes.**

Answering yes will begin the installation process, which will begin extracting the compressed tar file. The following will create a link to the library files located in /usr/seos/lib. This is done so if any of the links need to be changed then a new link can be created without changing the program itself.

*For installation options, please use 'install_base -h'.*
*Select 'Y' to install, 'N' to exit the script: Y*

*Extracting tar file .....*
*Creating symbolic link for shared cipher: /usr/seos/lib/adcipher*
*Creating symbolic link for shared library: /usr/seos/lib/liblangapi.sl*
*Creating symbolic link for shared library: /usr/seos/lib/libscramble.sl*
*Creating symbolic link for shared library: /usr/seos/lib/libcrypt*
*Creating symbolic link for shared library: /usr/seos/lib/libseadmapi.sl*
*Creating symbolic link for shared library: /usr/seos/lib/libselang.sl*
*Creating symbolic link for shared library: /usr/seos/lib/libsepass.sl*
*Creating symbolic link for shared library: /usr/seos/lib/libsnmp.sl*

The following changes the permissions on the binary files to 555. The files in which the permissions are being changed are Access Control commands that can be executed by any user.

*Changing permissions of binaries*
*Setting permission 555 for files:*
**dbdump *selang* dbutil seaudit secompas secons secredb sedb2scr seerrlog segrace seini semigrate semsgtool senone sepurgdb seretrust serevu sereport sesu seversion sewhoami winsetup UxImport sebuildla seuidpgm senable dbmgr*
*se_loadtest*

There are some binary files that need to have root level access in order to be executed and therefore the suid bit is set. The /usr/seos/seosdb directory is also created. This is where the security application's database will be kept.

*Setting the setuid bit on the following binaries:*
*sepass sesudo secompas issec*
*A database has now been created in /usr/seos/seosdb*
*-----*

The Access Control configuration file, seos.ini, is being created as can be seen below. Also, the Access Control PAM settings are added to the system PAM configuration file. The /etc/passwd file is added to the security application database so that it may be protected.

*seos.ini created in /usr/seos.*
*Installing eTrust PAM support...*
*PAM configuration file had change....*
*Defining /etc/passwd in eTrust database.*
*Setting up default eTrust SURROGATE for sesudo rules.*
*Successfully updated eTrust options*
*Successfully updated eTrust options*
*Setting the owner, group and permissions of eTrust files.*
*Setting the setuid bit on the following binaries:*
*selock selockcom selogo segracex*
*Installing GUI X-resource files in /usr/openwin/lib/app-defaults.*
*Creating eTrust configuration files in /usr/seos/etc.*
*Creating initialization files for seosd for _HPUX.*
*Creating I18n convertion tables at /etc/ca/localx*
*Starting interactive setup.*

### 6) Press Enter to continue installing.

*Press ENTER to continue*

### 7) Enter infosec as the audit group name.

This will specify the group that will have access to the security application and related files. In our case, the name of the security group is called infosec.

*----------[ eTrust audit group setup ]----------*
*eTrust needs to know the audit group at your site.*
*The name of the group should be a valid UNIX group. The eTrust*
*audit files will be created so that the specified group can read these*
*files. The root user is always allowed to read the audit files,*
*unless you deny root read access in eTrust access rules.*
*If you don't have an audit group, hit ENTER.*

*Specify the audit group name [none] : infosec*

### 8) Answer Y for yes.

This will import and create all the users and groups from the password and group file into Access Control's database.

*----------[ Import of users, groups and hosts ]----------*
*eTrust can import users, groups and hosts at your site*
*into its database. The import process uses local host files*
*or the NIS maps as the source of the information.*
*The task of importing users, groups and hosts may take some time,*
*depending on the number of users, groups and hosts defined.*
*You are also able to import this data into the eTrust database*
*after the installation process, by using the UxImport utility.*

*Import users, groups and hosts now? [y/N] : y*

*eTrust UxImport v5.10b (5.10) - Imports UNIX information to database*
*Portions of this product Copyright (c) 1999-2001 Computer Associates International, Inc.*
*Portions of this product Copyright (c) 1995-2001 Memco Software Ltd., a CA company. All rights reserved.*

*Successfully created USER nobody*
*Successfully created GROUP root*
*Successfully created GROUP other*
*Successfully created GROUP bin*
*Successfully created GROUP sys*
*Successfully created GROUP adm*
*Successfully created GROUP daemon*
*Successfully created GROUP mail*
*Successfully created GROUP lp*
*Successfully created GROUP tty*
*Successfully created GROUP nuucp*
*Successfully created GROUP users*
*Successfully created GROUP nogroup*
*Successfully created GROUP yeller*
*Successfully created GROUP sshd*
*Successfully updated USER root*
*Successfully created USER daemon*
*Successfully created USER bin*
*Successfully created USER sys*
*Successfully created USER adm*
*Successfully created USER uucp*
*Successfully created USER lp*
*Successfully created USER nuucp*
*Successfully created USER hpdb*
*Successfully created USER flipper*
*Successfully created USER sshd*
*Successfully created USER shamu*
*Successfully joined USER root to group sys*
*Successfully joined USER daemon to group daemon*
*Successfully joined USER bin to group bin*
*Successfully created HOST guess.boo.com*
*Successfully created HOST localhost.boo.com*

**9) Press Enter to continue with the installation**

*Press ENTER to continue*

**10) Answer none for parent policy model name.**

The parent policy model database is a hierarchal structure in which rules are written from the parent database and pushed down to all other servers or child servers registered to the parent. I did not use this feature because of problems encountered during its use. When rules were pushed down from the parent to agent databases, error messages would not be written to the screen concerning which hosts did not receive the update. Therefore, alternatives were used that solved this problem.

*----------[ Set up parent policy model database PMDB ]----------*
*eTrust supports distribution of a security policy in a network.*
*In order for the local host to receive updates of the security*
*policy from a policy model database, the local host must be*
*configured to accept updates from a specific policy model database.*
*If you have a policy model already set up at your site, or you*
*intend to set up one in the near future, specify the name*
*of the policy model.*
*A policy model name is composed of a policy name and the*
*at sign '@' followed by the name of the host on which this*
*policy model resides.*
*For example: 'pmdb@superserver'*

*If you want to set up a policy model later, you must manually*
*edit the seos.ini file located in /usr/seos.*
*Specify the policy model name in the token 'parent_pmd' in*
*the section 'seos'.*
*NOTE: On some terminals you need to type ^v before typing @ sign.*

*Enter parent policy model name [none] : none*

**11) Answer none to passwords policy model name.**

The password policy model is password synchronization with mainframes and machines running eTrust Access Control. Since the only other system running eTrust is HPUX in our example, we will answer none to this step.

*----------[ Set up passwords policy model ]----------*
*eTrust supports passwords history checkings..*
*For the passwords history checking to be effective, you need*
*to setup a password policy model.*
*A policy model name is composed of a policy name then the*
*at sign '@' followed by the name of the host on which this*
*policy model resides.*
*For example: 'pmdb@superserver'*

*If you want to setup a policy model later, you will have to*
*manually edit the seos.ini file located in /usr/seos, and*

*specify the policy model name in the token 'passwd_pmd' in*
*the section 'seos'.*
*NOTE: On some terminals you need to type ^v before typing @ sign.*

*Enter passwords policy model name [none] : none*

*----------[ Set up eTrust ]----------*
*eTrust checks whether the local machine serves as an NIS or*
*DNS server. If it is, eTrust must be set up in a special way.*
*If you are using eTrust on Solaris 2.5 and above AIX 4.3 or*
*HP-UX 11 and above, eTrust must also be set up as if it were*
*running on an NIS machine.*

*eTrust is configured for an NIS server - Solaris 2.5 and above.*

The following shows the building of a look aside database in /usr/seos/ladb. A look-aside database increases the efficiency of Access Control. There are four tables that contain resolution information concerning groups, hosts, services and user names. It is quicker for the application to look in the database files than to read the files within /etc or wait for DNS to resolve.

*eTrust can use look-aside databases to resolve*
*user ID to user name, group ID to group name, host IP*
*address to host name, and service port to service name.*
*When using eTrust configured to run on an NIS server,*
*it is recommended that you use look-aside databases.*

*Your system is being configured to use look-aside databases.*

*Creating eTrust look-aside databases.*
*eTrust: Creating users look-aside database.*
*eTrust: Creating hosts look-aside database using DNS.*
*eTrust: Creating services look-aside database.*
*eTrust: Creating groups look-aside database.*
*Creating symbolic links for sebuildla lang exits.*
*Creating a symbolic link for S99MODIFY_u_sebuildla.sh.*
*Creating a symbolic link for S99MODIFY_g_sebuildla.sh.*
*Creating a symbolic link for S99DELETE_u_sebuildla.sh.*
*Creating a symbolic link for S99DELETE_g_sebuildla.sh.*
*Creating a symbolic link for S99CREATE_u_sebuildla.sh.*
*Creating a symbolic link for S99CREATE_g_sebuildla.sh.*

*Rebuild the eTrust lookaside database periodically using cron.*

**12) Press Enter to continue.**

Press Enter to continue with the installation process.

*Press ENTER to continue*

**13) Place security administrator accounts in this field.**

Names placed in this field will be authorized to access the security application database.

*----------[ Set up security administrators ]----------*
*You may define users as security administrators and auditors.*
*Specify user IDs separated by space, other than root.*
*If you do not want to define administrators now, hit ENTER.*

*Please enter administrator names [none] : shamu*

*Successfully updated USER nobody*
*Successfully created USER RSV*
*Successfully added RSV to _default's ACL*
*WARNING: 'guess' is not the TERMINAL's fully-qualified name (it is 'guess.boo.com')*
*Successfully created TERMINAL guess.boo.com*
*WARNING: 'guess' is not the TERMINAL's fully-qualified name (it is 'gues.boo.com')*
*Successfully added root to guess.boo.com's ACL*
*WARNING: 'guess' is not the TERMINAL's fully-qualified name (it is 'guess.boo.com')*
*Successfully added RSV to guess.boo.com's ACL*
*Successfully updated TERMINAL guess.boo.com*
*Successfully added root to guess.boo.com's ACL*
*Successfully added RSV to guess.boo.com's ACL*

### 14) Press Enter to continue with the installation.

*Press ENTER to continue*

### 15) Answer no to replacing the default encryption.

After Access Control is installed, it is possible to change the encryption that is set by default.

*----------[ Change eTrust internal encryption method ]----------*
*eTrust is using a simple encryption method to protect the communication*
*between eTrust programs and eTrust installed hosts.*
*You may want to change the default encryption method to be DES encryption.*
*You can also change the encryption method after the installation process*
*by linking /usr/seos/lib/libcrypt to libscramble or libdes.*
*NOTE: the encryption method must be the same on all eTrust hosts*
*that communicate with each other.*
*The default in previous installations was the simple encryption method.*

*Do you want to replace the default encryption method now? [y/N] :N*

*Verifying that encryption method is : simple.*

### 16) Press Enter to continue with installation.

*Press ENTER to continue*

### 17) Answer no to setting a new encryption key.

*----------[ Change eTrust internal encryption key ]----------*

*eTrust is using an encryption key to protect the communication between eTrust programs.*
*You may want to change the default encryption key by typing in a new key at the prompt below.*
*You can also change the encryption key after the installation process by using the sechkey utility.*
*NOTE: the encryption key must be the same on all the eTrust hosts that communicate with each other.*
*The default in previous installations was the default encryption key.*

*Do you want to set a new encryption key now? [Y/*n] : n

**18) Answer yes to installing a baseline security pack.**

This will install a base security rule set to the eTrust database. It will protect a few key system files such /etc/passwd, /etc/services,etc. You will add on to this initial ruleset.

*----------[ Install Baseline Security Pack ]----------*
*The Baseline Security Pack defines basic rules to protect the operating system. All the rules are defined in WARNING mode.*
*It is recommended to remove the WARNING mode after a test period, thus activating the rules.*
*You can also perform Baseline Security Pack installation after the eTrust installation process, by using the install_baseline.sh utility.*

*Do you want to install Baseline Security Pack now? [y/N] :y*

*---------------------------------------*
*Applying Baseline Security Pack Rules*
*---------------------------------------*

*Installing generic eTrust rules.*
*Successfully created GROUP systemadmin*
*Successfully created GROUP secadmin*
*Successfully updated GROUP datasecurity*
*Successfully updated GROUP sys*
*Successfully updated USER _undefined*
*Successfully created GROUP extusers*
*Successfully updated CONNECT _default*
*Successfully added extusers to _default's ACL*
*Successfully added systemadmin to USER's ACL*
*Successfully added systemadmin to GROUP's ACL*
*Successfully updated TERMINAL _default*
*Successfully added root to _default's ACL*
*Successfully added secadmin to _default's ACL*
*Successfully added sys to _default's ACL*
*Successfully created TERMINAL _CRONJOB_*
*Successfully added sys to _CRONJOB_'s ACL*
*Successfully created GTERMINAL root-not-allowed*

*Successfully added root to root-not-allowed's ACL*
*Successfully added systemadmin to root-not-allowed's ACL*
*Successfully added secadmin to root-not-allowed's ACL*
*Successfully created FILE /usr/seos/etc/**
*Successfully added root to /usr/seos/etc/*'s ACL*
*Successfully added secadmin to /usr/seos/etc/*'s ACL*
*Successfully created FILE /usr/seos/bin/**
*Successfully created FILE /usr/seos/lbin/**
*Successfully added root to /usr/seos/bin/*'s ACL*
*Successfully added secadmin to /usr/seos/bin/*'s ACL*
*Successfully added root to /usr/seos/lbin/*'s ACL*
*Successfully added secadmin to /usr/seos/lbin/*'s ACL*
*Successfully created FILE /usr/seos/log/**
*Successfully added * via PROGRAM /usr/seos/bin/selogrd to /usr/seos/log/*'s ACL*
*Successfully added _undefined via PROGRAM /usr/seos/bin/selogrd to /usr/seos/log/*'s ACL*
*Successfully added root to /usr/seos/log/*'s ACL*
*Successfully added datasecurity to /usr/seos/log/*'s ACL*
*Successfully added sysadmin to /usr/seos/log/*'s ACL*
*Successfully added secadmin to /usr/seos/log/*'s ACL*
*Successfully created FILE /usr/seos/log/serevu_disable.users*
*Successfully added secadmin to /usr/seos/log/serevu_disable.users's ACL*
*Successfully added root to /usr/seos/log/serevu_disable.users's ACL*
*Successfully added datasecurity to /usr/seos/log/serevu_disable.users's ACL*
*Successfully updated eTrust options*
*Successfully updated eTrust options*
*Successfully updated eTrust options*
*Successfully created FILE /.**
*Successfully added root to /.*'s ACL*
*Successfully added systemadmin to /.*'s ACL*
*Successfully created HOST _default*
*Successfully created FILE /etc/**
*Successfully added root to /etc/*'s ACL*
*Successfully added systemadmin to /etc/*'s ACL*
*Successfully updated SURROGATE USER.root*
*Successfully added systemadmin to USER.root's ACL*
*Successfully updated SURROGATE _default*
*Successfully created FILE /var/yp/**
*Successfully added root to /var/yp/*'s ACL*
*Successfully added systemadmin to /var/yp/*'s ACL*
*Successfully updated PROGRAM _default*
*Successfully added * via PROGRAM /usr/seos/bin/selogrcd to /usr/seos/log/*'s ACL*
*Successfully added _undefined via PROGRAM /usr/seos/bin/selogrcd to /usr/seos/log/*'s ACL*

*Installing specific eTrust rules.*

*Successfully updated USER daemon*
*Successfully updated USER bin*
*Successfully updated USER sys*
*Successfully created FILE /bin/**
*Successfully added root to /bin/*'s ACL*
*Successfully added systemadmin to /bin/*'s ACL*
*Successfully created FILE /usr/sbin/**
*Successfully added root to /usr/sbin/*'s ACL*
*Successfully added systemadmin to /usr/sbin/*'s ACL*
*Successfully created FILE /sbin/**
*Successfully added root to /sbin/*'s ACL*

*Successfully added systemadmin to /sbin/*'s ACL*
*Successfully created FILE /sbin/init.d/rc.eTrust.base*
*Successfully added root to /sbin/init.d/rc.eTrust.base's ACL*
*Successfully added systemadmin to /sbin/init.d/rc.eTrust.base's ACL*
*Successfully added secadmin to /sbin/init.d/rc.eTrust.base's ACL*
*Successfully updated FILE /etc/passwd*
*Successfully added root to /etc/passwd's ACL*
*Successfully added systemadmin to /etc/passwd's ACL*
*Successfully added systemadmin to /etc/passwd.tmp's ACL*
*Successfully added * via PROGRAM /usr/bin/passwd to /etc/passwd.tmp's ACL*
*Successfully created FILE /tcb/**
*Successfully added root to /tcb/*'s ACL*
*Successfully added systemadmin to /tcb/*'s ACL*
*Successfully added * via PROGRAM /usr/bin/passwd to /tcb/*'s ACL*
*Successfully added _undefined via PROGRAM /usr/bin/passwd to /tcb/*'s ACL*
*Successfully created FILE /sbin/init*
*Successfully created FILE /usr/bin/autopush*
*Successfully added root to /dev/mem's ACL*
*Successfully added systemadmin to /dev/mem's ACL*
*Successfully created PROCESS /usr/sbin/lpsched*
*Successfully created FILE /usr/sbin/lpsched*
*Successfully created FILE /usr/sbin/syslogd*
*Successfully created GFILE processes*
*Successfully updated GFILE processes*
*Successfully added root to processes's ACL*
*Successfully added systemadmin to processes's ACL*
*Successfully updated PROCESS _default*
*Successfully created TERMINAL /dev/tty0p0*
*Successfully created TERMINAL /dev/console*
*Successfully created TERMINAL 0.0.0.0*
*Successfully created GTERMINAL root-allowed*
*Successfully added root to root-allowed's ACL*
*Successfully added systemadmin to root-allowed's ACL*
*Successfully added secadmin to root-allowed's ACL*

*----------*
*End of Baseline Security Pack Rules*
*----------*

### 19) Press Enter to Continue with the installation.

*Press ENTER to continue*

### 20) Answer yes to starting eTrust from a remote host.

This is a convenient tool for security administrators to be able to start up the eTrust daemons from a remote host. If the security application is going to be stopped from a remote host, authentication will need to be provided.

*----------[ Install Remote eTrust Loader ]----------*
*eTrust can be started from a remote seload command.*
*You can start eTrust from a remote machine*
*eTrust installation process will now install an inetd*
*service to serve this new option.*

*Do you want to be able to start eTrust from a remote host? [Y/n] : Y*

*Adding eTrust seosload service to /etc/services.*
*Adding eTrust seosload to /etc/inetd.conf.*
*Restarting inetd daemon, in order to activate the changes.*

### 21) Press Enter to continue with installation.

*Press ENTER to continue*

*Interactive setup procedure done.*
*Protecting eTrust look-aside database files.*
*Verifying eTrust seosd.under_NIS_server seos.ini token.*
*Verifying eTrust seosd.resolve_rebind token.*
*Setting eTrust seosd.domain_names token in seos.ini.*
*Setting permissions of bin/sepmd bin/sepmdadm lbin/sepmdd lbin/secrepsw bin/selogrcd.*
*Setting permissions of /usr/seos/man/man8/ files.*
*Successfully updated PROGRAM /usr/seos/bin/sebuildla*
*Successfully updated PROGRAM /usr/seos/bin/selogrcd*
*Successfully updated PROGRAM /usr/seos/bin/selogrd*
*Successfully updated PROGRAM /usr/seos/bin/sesudo*

*Installation complete.*
*Check seos.ini file for the right configuration.*
*Installing eTrust system call.*
*Executing SEOS_HPUX_build.*
*TCP module autopushed*

### 22) Reboot the system for changes to take effect.

After the system is rebooted, there is some cleanup to the /usr/seos directory that can be done to save space on the file system. The tar files can be removed from the directory because they are used only during the installation and will not be needed again.

## 4.3 CONFIGURING SEOS.INI

The /usr/seos/seos.ini file is the main configuration file for eTrust Access Control. The file contains tokens used by the security application. The seos.ini file is protected by eTrust and cannot be updated while eTrust is running. The file is read only and in order to be modified the security application must be taken down. There are some changes to the initial configuration file that can be made that would make the security application more efficient.

### *4.3.1 Cron_program Token*

The cron_program token is set to none by default. If the path of the system's cron binary is placed in this field, the token will improve the ckeck when cron is being executed, which would greatly increase the speed of the security application.

### *4.3.2 UseFileCache Token*

The UseFileCache rule is set to no by default. However, by setting this value to yes will cause Access Control to use a cache when accessing file records, which will greatly increase the efficiency of the security application.

### *4.3.3 Audit_size and Error_size token*

The default size of the audit and error logs can be changed. This value can be increased or decreased depending on the administrator. The default size for the error log file is set to 50 Kilobytes (KB) and the default size of the audit log files by default is 1024 KB. It is advised to increase the size of these log files because it will assist the security administrators in times of investigation and troubleshooting.

### *4.3.4 Error_group Token*

The default setting for the group that will be able to view the error logs is initially set to none (error_group = none). This means that no user except for root will be able to view this file. It would be best to place the group that the security personnel belong to in this field. If an invalid group is placed in this field, no group permissions will be set. The location of the access control error log file is /usr/seos/log/seos.error.

### *4.3.5 Exit_timeout Token*

The exit_timeout field is set to 15 seconds by default. This is the maximum time, in seconds, that eTrust allows the exit program to execute. When user and group records need to updated, Access Control uses exit scripts to issue commands in the UNIX environment. Waiting 15 seconds for an exit script may not be enough time for processes depending on exit scripts to execute. Therefore, it may be necessary to increase this value.

### *4.3.6 UseInvokerPassword Token*

The UseInvokerPassword by default is set to none. This value will need to be set to yes. When set to none, users will not have to provide a password when running the sesu command. The sesu command is similar to the su command in which a user will be able to switch to another user's account. If a user is not

required to enter a password, this offsets the security a password is suppose to provide so obviously you would want to set this token to yes.

### *4.3.7 AllowedUidRange Token*

The AllowedUidRange token is set to 100 to 30000 by default. This token will set the UID integer range in which eTrust can assign when a user is added through eTrust. The possible range of values, which can be set in HPUX, is a minimum of –2 and a maximum of 2,147,483,646. There is no reason to limit the security applications access to the full range of uids. Therefore, the values should be changed to these minimum and maximum values.

### *4.3.8 Selogrd Token*

The selogrd is the log routing emitter daemon used to distribute selected local audit log records to specific hosts or email. The selogrd token is set to no by default. This token should be changed to yes. This will automatically load the selogrd whenever the seload utility is executed.

The other values that are set by default in the seos.ini file can be set as is. The seos.ini file itself is very well documented and will thoroughly explain the use of all the tokens.

## 4.4 WARNING AND FAIL MODE

The base rule set that is imported into the database is a stripped down version of what is to be protected on the server. There may be other important operating system files or application files that may need to be protected. The database can be customized according to the application and data on every server. Upon install, the database is set in warn mode. This means that if a user accesses a resource that he is not authorized to then the security application will send a warning message to the security administrator rather than the action getting denied. It is important to view the warning messages constantly because there may be messages that are legitimate and need to be authorized. Sometimes warning messages will appear from actions that users should not be performing. It is important that the security administrator speak with the users who are producing the messages and ensure that the commands they are executing are legitimate. This may require watching the logs for a month until batch processing or end of month processing is finished.

Once the security administrator feels comfortable with the messages being produced, it is necessary to convert the Access Control database to fail mode. This means that when a user access a resources, whether it is legitimate or not, he will be denied. He will need to speak with the security administrator if he needs access. Even after a server is converted to fail mode, it is necessary to

monitor the logs regularly. I was heavily involved in monitoring the logs at our company for a period of six months after which I converted the servers into fail mode. I still monitor all deny messages and contact users concerning suspicious activity.

### 4.5 CONVERTING TO FAIL MODE

The way in which the database is converted to fail mode is first the entire database needs to be dumped to a regular text file. Executing the eTrust command sedb2scr will dump the database.

1) **Sedb2scr –r > db.dump**

Then the file must be opened with an editor such as vi so that changes could be made to the file. The "warning-" parameter must be appended to the end of all lines that begin with chres or editres. There are also editres lines within the dumped database file that are commented out. These are commented out because Access Control normally protects setuid files within its program class. However, there are times when an action will need to be authorized via a specific program that is not a setuid file. For instance, read access can be granted to the systemadmin group to the /usr/lbin/uucp/* directory via the /usr/bin/find command. The /usr/bin/find program will be added to the program class, but this is not a setuid file. Therefore, when the database is dumped to a text file this rule will be commented out. These comments must be removed and the warning- parameter placed at the end of these lines also so they may be added to the database. An example of the lines that should be changed can be seen below.

```
editres SURROGATE ("USER.root") audit(ALL) defaccess(NONE) owner('filownr')

chres  SURROGATE ("_default") audit(FAILURE) defaccess(READ) owner('filownr')

editres TCP ("_default") audit(FAILURE) defaccess(READ WRITE)
```

These lines should be changed so that they look like the following.

2) **Add the warning- parameter at the end of all lines that begin with chres and editres**.

```
editres SURROGATE ("USER.root") audit(ALL) defaccess(NONE) owner('filownr') warning-

chres  SURROGATE ("_default") audit(FAILURE) defaccess(READ) owner('filownr') warning-

editres TCP ("_default") audit(FAILURE) defaccess(READ WRITE) warning-
```

The warning- specifies that, if an accessor's authority is insufficient to access the file or resource, Access Control issues a deny to the user's access to the resource and does not write a warning message. This parameter can only be

used with the chres, editres, chfile and editfile (eTrust Access Control For UNIX 1-27).

Once the changes are made, save and exit the file. This file, with the new changes, must now be imported back into the eTrust database. This is accomplished with the use of the selang command. You can enter Access Control commands from selang, a command shell for entering commands to access and update the Access Control database. In this case, the entire file will need to run through selang. Once the import is done the entire database will be in fail mode.

3) **cat db.dump | selang.**

## 4.6 ACCESS CONTROL BENEFITS

### *4.6.1 Defense-in-Depth*

Access Control provides the flexibility of a layered defense approach to security through the use of its various classes. Through the use of the terminal class, Access Control is able to filter access to the server at the point of entry or login. Access Control has the ability to accept or deny a login based on the source of the login. Access Control also provides access based on login application such as ftp and telnet through the use of the LOGINAPPL class. The HOST class in eTrust can be used to filter TCP/IP connections to the host computer. Therefore, it is possible to reject any remote logins such as rlogin, rcp, etc.

The second layer is that the security application provides protection to resources from those who legitimately login . Any attempt to access a resource that is not authorized will be denied and a message will be sent through email to the security administrators. These two tools provide a two-layered approach to security.

### *4.6.2 Controlling Root*

In the UNIX environment the root account is all-powerful. It can update any file and access any resource without any type of control. This is a major problem with UNIX. Many times absent-minded system administrators can accidentally bring down a server by issuing the wrong command. When users try to access the system, they will be unable to and availability has been compromised. Many times hackers and crackers will try to guess the root account password and if they are successful, malicious actions can occur as a result.

One of the main advantages gained by installing eTrust Access Control is that the root account can be controlled. The root account can be locked down unless specifically authorized by the security application. The root account can be

denied or allowed access to only those resources listed within Access Control's database. As mentioned before, what resources are protected by the security application depends upon the security administrator. The root account can be denied access to something as simple as the find command if not authorized.

### *4.6.3 Logging in from Console*

Most versions of UNIX allows you to configure certain terminals so that users can't login as the superuser from the login: prompt. (Garfinkel and Spafford 117). Therefore, in addition to Access Control, a couple of controls were implemented to better control the root account. First of all, the root account can only login to a server from the console. If an administrator attempted to login to the server from a remote terminal, he would be denied. This was accomplished using the /etc/securetty file. In doing this, we could prevent possible hacks to the root account. In addition, when an administrator did login to the root account from the console, he would have to enter his name and reason for logging in. This information would be sent to the appropriate personnel via email for review.

### *4.6.4 Accountability*

Access Control also assisted in holding the users accountable for their actions. The administrators could still obtain root access via the sesu utility, which is similar to su, provided by the security application. The way this was accomplished is that eTrust will keep track of whom the user is when he first logs on. Therefore, if a user logs in and then surrogates to lets say the oracle or root account, all his actions will be attributed to his personal id.

An advantage of the sesu utility over the su command was that users could use their personal passwords. The problem with the su command is that the user would need to know the password of the account to whom he/she is surrogating to. This can be a problem when many people need root access. If too many people know the root password, you have to depend on that group of people to keep that password a secret. It is always best to keep the number of people with knowledge of the root account password to an absolute minimum usually one or two individual.

The surrogate class in eTrust will provide access for the switching to another user's account. Therefore, a USER.root rule can be added to the surrogate class. Specific users and groups can be authorized to access this rule. Therefore, if the system administrators are in a group called systemadmin, then this entire group can be given access to the USER.root rule, which once again would allow them to surrogate to root.

### *4.6.5 Setuid Files*

Just as most services are a danger because they often run as root, suid root programs always run as root. The danger here is that if someone obtains an account on your computer (legitimately or otherwise), SUID root programs present a potential mans for grabbing root access (Beal). It is possible that many of the system commands that have the setuid bit to root set can be exploited. Once these programs are exploited, the hacker will have in essence gained root access to the server. Most security flaws in UNIX arise from bugs and design errors in programs that run as root or through unanticipated interactions of such programs. If possible, never write SUID shell scripts (AusCERT). Access Control provides a way in which this can be guarded against. Any program that is setuid to root will have to surrogate to root temporarily. This surrogate action will have to be authorized via eTrust. For instance, the /usr/bin/passwd is a setuid program to root. If userX executes this program, this user will need to be authorized to surrogate to root specifically via the /usr/bin/passwd program.

### *4.6.6 File Integrity Check*

The security application also provides its own file integrity check. If any file or program on the server changes for any reason, that file or program will be marked un-trusted. All programs listed within the eTrust database are scanned with an integrity check. If for any reason, such as the size changes, the program will be un-trusted and the security staff will be notified immediately. If the change was authorized and the security staff was aware of the change, then the program can be re-trusted. Otherwise, the reasons for the change would be investigated immediately. However, the only drawback to the integrity check offered by the security application is that it also prevents anyone from executing the program. This can be a huge problem. If, for instance, /usr/bin/login becomes un-trusted because an operating system patch was installed the security application would prevent any user from logging in, and thereby causing a Denial of Service. There is a class in Access Control called SECFILE which can be used and will be described later on, which will prevent this from happening.

However, the advantage of this integrity check is that the security staff will be able to identify quickly if there was an unauthorized change. A Trojan horse program could have been installed in place. Consider the case where an intruder manages to replace a commonly executed file (like login) with a Trojan horse version. Since you can't easily look inside executables, the only way you can flag such a compromise is by an integrity check (Lierley and Light 436). A Trojan is a program that looks like a regular program while it is running. It actually runs a malicious program in the background, unbeknownst to the user (Wong 10).

### *4.6.7 Flags*

As mentioned above, the security application will mark a program un-trusted if the file has changed. The security application performs checks on the flags

within the "pgminfo" property and verifies the information. The following flags are checked within the pgminfo property (eTrust Access Control For UNIX 17-70):

| Flags | Description |
|---|---|
| Ctime | Creation time |
| Mtime | Time last modified |
| Mode | Security Mode (permission bits) |
| Size | File Size |
| Device | Where file resides (logical disk) |
| Inode | Address of file within the file system |
| Crc | Cyclical redundancy check |
| Owner | File owner |
| Group | Group file owner |

### *4.6.8 Remote Administration*

With over 200 HPUX servers to secure, Access Control provided an easy way in which all the servers could be secured from a central server. If one rule needed to be transferred to all 200 servers or maybe about 30 servers, then a small script could be written that would push the new rule to the servers you wished in a more timely fashion. This also increased the flexibility of assisting users with their problems. If a user's password was locked out of a certain server, the password on the server could be reset from the central server. The security application became very useful in account administration. An account could be added through eTrust to any number of servers. The time saved from the product was considerable. No longer did we have to login to every individual server, but we could push out security rules, passwords and user accounts from a central server. Finally, we could reduce the number of helpdesk accounts created on the servers. The helpdesk accounts would only need to be created on one server, the central server. The helpdesk accounts would also need to be added to the security application on all the remote servers. The helpdesk used Access Control to push out password resets from this central server to all of the remote servers on the network.

### *4.6.9 Stack Overflow Protection*

Access Control provides a way in which to protect against hackers who create and exploit stack overflow in order to break into systems. A stack overflow enables hackers to execute commands on a server with the permissions of the original owner. This happens when the stack is overwritten and thus corrupting the stack frame. The stack overflow exploit can then be used to execute arbitrary code.

Stack Overflow Protection (STOP) is installed by default. If the service is to be turned off, the following seini command will modify the seos.ini and deactivate the service.

*seini –s SEOS_syscall.STOP_enabled=0*

In order to start up the STOP service again, simply change the enabled token above to the value of 1.

## 4.7 WATCHDOG DAEMON

The daemon that checks the flags mentioned above and marks a program un-trusted is the watchdog daemon, seoswd. Once a program is marked un-trusted the seoswd prevents this program from being able to be executed. The intervals at which the seoswd will scan the programs can be changed in the seos.ini file. The IgnoreScanInterval specifies whether to scan programs at specific intervals. This value is set to no by default, which means the watchdog will perform interval scanning. If this token is set to yes, it does not scan at intervals. There are many other tokens, which can be set that will customize the scanning of the watchdog. The man pages can be consulted for details on seoswd. The watchdog also checks to ensure the access control daemon (seosd) is up and running. If seosd is not running, it will start seosd.

## 4.8 SECFILE Class

The SECFILE class can be used in conjunction with the program class. The SECFILE class is similar to the program class. The seoswd will scan this class and if a flag on a file changes for any reason, this file will be marked un-trusted. However, rather than preventing anyone from executing this program, the seoswd will mark the file un-trusted and will still permit authorized individuals to execute the program.

## 4.9 AUDIT LOGS

There are a couple of ways in which log messages can be sent to the security administrators for review. It is possible to send a message to the administrator as soon as a resource is accessed or it is possible to collect all of the logs for a period of time and send this file at the end of the day for review. If the first option is chosen, the security administrators will receive an email immediately. However, it is possible that the security administrator's inbox could be filled with

hundreds of emails in one day. If the second option is chosen, only one email will be sent at the end of the day, however, you will have to wait until you receive this email before any messages can be reviewed. It is also possible to login to individual servers and view messages from the command line. The following will explain how to perform both options and it is up to the security administrator on which one to choose.

### *4.9.1 Single messages*

Perform the steps below in order to receive messages from every server when an unauthorized access to resources occurs.

1) Add the following lines to the configuration file.

*Rule name*
*mail john@boohost*
*include Code (D).*
*.*

This will cause the log-routing emitter to create a mail message for all deny messages. The configuration file can be configured in many different ways. For instance, it is possible to receive messages only on attempts to surrogate to root. The man pages on selogrd provide detailed information on how to customize the selogrd.cfg file.

### *4.9.2 Compile log messages*

The second option in which all the logs from all the servers are sent to the security administrator at the end of the day is a combination of the log-routing emitter and an in house developed script. In order to collect audit logs from the various servers on the network and send it to a single host the log-routing facility must be used. In order to set up the log-routing facility, perform the following steps:

1) The default log routing configuration file is called /usr/seos/log/selogrd.cfg. Modifying the RouteFile token in the seos.ini file described earlier can change the location of this file.

2) Add the following lines to the configuration file.

*Rule*
*Host destination*
*.*

The destination value should be replaced with the name of the central server to which the logs will be sent. The logs from all servers will be sent to a file called /usr/seos/log/seos.collect.audit by default. A script can be created which will utilize the Access Control command, /usr/seos/bin/seaudit, which will grab all of

the messages for any date specified. The seaudit utility provided by Computer Associates parses the log file according to the options it receives. If you wish to be emailed concerning deny messages for the previous day, a command like this would be executed.

*/usr/seos/bin/seaudit –a –sd 07-OCT-2004 –ed 07-OCT-2004 –fn*
*/usr/seos/log/seos.collect.audit | grep ‘ D ‘*

The –a option will grab all messages such as warning, denies, etc. The –sd stands for start date and the –ed stands for end date. The –fn stands for file name and if this value is not placed here, seaudit will read /usr/seos/log/seos.audit file by default. The /usr/seos/log/seos.audit file is where the local host log files are kept. As mentioned before, the /usr/seos/log/seos.collect.audit file contains the logs for all servers on the network. Finally, the “grep ‘ D ‘ “ option grabs all deny messages from the file. You can set up a script in cron, which will utilize this command to run at whatever time frame the security administrators deems appropriate.

## 5.0 RISKS

The security measures described above did not guarantee that our servers were completely secure. It can never be said that a server is 100 percent secure. The act of securing a server is a continuous process. There were still some risks involved even with the steps performed above.

An assumption is made concerning Access Control that the application will always be up. However, if the application is taken down for any reason, then that layer of security is gone. The two-layered approach to having the OS and the security application approve a request to a resource is no longer there. In addition, any process that depended on the security application will not function. For instance, the SUDO class in eTrust allows a user to perform a certain action with privileges of another user such as root. When Access Control is gone, the user will not be able to run that process.

## 6.0 CONCLUSION

The installation of eTrust was a good start in securing the 200 HPUX servers. Access Control provided a Role Based Access Control type of access control. The security application, eTrust Access Control provided individual access controls for system logons, as well as access to objects, resources, programs and files (CA 1).

Access Control also assisted in access management. By the logs that the security application would produce, the relationship between user accounts and

data could be actively monitored by the security staff (Cole et al. 151). If a user were attempting to access a resource, the access would have to be authorized at the UNIX and Access Control level as well. Access Control would monitor all access by users even that of root. Therefore, the root account or any action that temporarily gain root access did not have free reign.

Configuration management is the discipline of establishing a known baseline condition and then managing that condition (Cole et al. 59). The security application assisted in creating a secure baseline in which to operate. In addition, the centralized administration offered by Access Control increased the efficiency at which we could secure our servers.

The actions performed above did not guarantee a secure system. It is the security administrator's responsibility to stay current with new technologies affecting the security world. However, Access Control did provide a more secure environment than was present before.



## References

AusCERT. Securing UNIX Programming Checklist. 2002. AusCert. 14 Aug. 2004 http://www.auscert.org.au/render.html?it=1975.

Beale, Jay. Shredding Access in the Name of Security: Set UID Audits. 2000. bastille-linux. 13 Aug. 2004 <http://www.bastille-linux.org/jay/suid-audit.html>.

CA. “Data Sheets.” eTrust Access Control Release 5.3 for Linux and UNIX eTrust Access Control Release for Windows. 2004. Computer Associates. 13 Aug. 2004 <http://www3.ca.com/Files/DataSheets/etrust_access_control_data_sheet.pdf>.

Cole, Eric, Fossen, Jason, Northcutt, Stephen and Pomeranz, Hal. SANS Security Essentials & the CISSP 10 Domains – Book 2. SANS Institute, 2004.

eTrust Access Control for UNIX. Reference Guide. New York: Computer Associates International, Inc., 2001.

Garfinkel, Simson, Spafford Gene and Schwartz, Alan. Practical Unix & Internet Security. California: O’Reilly & Associates, Inc.

Lierley, Mark and Light, Light E.H. (Eds.). Security Complete. California: SYBEX Inc., 2001.

Wong, Chris. hp-ux 11i security. New Jersey: Prentice Hall PTR, 2002.