Software
Product
Description

Safety V1.2 - Comprehensive Data Safety for your VMS systems.

from General Cybernetic Engineering

Executive Summary:

There are many perils your data faces, and loss of data can cost time, money, and jobs. Intruders, disgruntled insiders, or hidden flaws in installed software can destroy records. What is more, mistaken losses occur constantly.

Safety protects your system and your critical data in three ways:

1. A comprehensive security system adds extra checks for access to VMS files so that access by intruders or by people in non-job-required ways can be regulated or prevented. This allows your business - critical data to finally be protected against misuse, tampering, or abuse. Access from programs doing background dirty work (viruses, Trojans, worms, and the like, or even programs with security holes which can be exploited remotely (like Java browsers)) can also be blocked without damaging normal use. This active protection works three ways: by *checking integrity* of your files against tampering, by *preventing* of *untrusted images* from *gaining privilege*, and by *regulating* what other parts of the system an image may access.

2. A deletion protection system provides a way to undelete files which were deleted by mistake and to optionally copy deleted files to backup facilities before removal. Unlike all other VMS "undelete" programs on the market, this facility does not rely on finding the disk storage that contained the file and reclaiming it before it is overwritten. Rather, it changes the semantics of the file system delete to use a "wastebasket" system and captures the file intact. Thus, this system works reliably. No others do. This facility is also useful where you have a requirement to keep all files of a certain set of types, since the backup function can be used to capture such files while permitting otherwise normal system function. The shelving or linking functions are also available for moving copies offline if this is desired. The ***Safety*** protection features are fully integrated with the DPS subsystem, so that deletion protection does not involve destroying file security.

3. When space runs out, hasty decisions about what to keep online often must be made, and the risk of accidentally losing something important is high. ***Safety*** protects you from running out of space. Space can be monitored and older items in the wastebasket deleted if it is becoming low, without manual intervention. In addition, ***Safety*** is able to "shelve" files so that they are stored anywhere else desired on your system, and they are brought back automatically when accessed. Thus no manual arrangements need be made for reloading them. ***Safety*** can also keep the files on secondary storage, keeping a "soft link" to the files at their original site so they will be accessed on the secondary storage instead. Also, ***Safety*** can store files compressed, or can store

them on secondary storage so that read access is done on the secondary storage, but write access causes the file to be copied back to its original site. Standard VMS utilities are used for all file movement, and moved files are also directly accessible in their swapped sites with standard VMS utilities. The VMS file system remains completely valid at all times.

Safety gives you a full complement of tools for dealing with space issues automatically according to your site policy. These facilities are safe and easily understood. A comprehensive utility is provided by which you set your site policy to select which files are and are not eligible for automatic shelving. Also you are provided with screen oritented utilities for selecting files to shelve at any time. Access to the shelved files of course causes unshelving if the normal shelving-by-copy mode is used. Also, a simple set of rules permit locating shelved or softlink target files at any time, even without ***Safety*** running. ***Safety*** at no time invalidates your file structures for normal VMS access...not even for an instant.

In addition ***Safety*** contains functions to speed file access and inhibit disk fragmentation.

The major subsystems of ***Safety*** will now be described.

The Security Function System:

Summary:

Managing access to data critical to your business using ACL facilities in native VMS can be cumbersome and still is vulnerable to intruders or people acting in excess of their authority.

Want to be sure your critical records can't be accessed save at authorized places, times, and with the programs that are supposed to access them (instead of, say, COPY.EXE)?

Want to have protection against privileged users bypasssing access controls?

Want to be able to password protect individual files?

Want to be able to invisibly hide selected files from unauthorized intruders?

Have you read that attacks on machines can happen because a Java browser points at a web site that damages the system (as has been reported in the press)? Want to be able to protect **your** systems?

The ***Safety*** security subsystem builds in facilities permitting all of these, and is not vulnerable to intruders who disable the AUDIT facility as all other commercial packages which purport to monitor access are.

Description: When your business depends on critical files, or when you are obliged by law or contract to maintain confidentiality of data on your system, in most cases the options provided by VMS for securing this data can be cumbersome and far too coarse-grained.

The problem is that certain kinds of access to data are often needed by people in a shop, but other access should be prevented and audited. Moreover, the wide system access that can come as a result of having system privileges often does not mean that it should be used to browse or disclose data stored on the system. A system manager will in general not, for example, have any valid reason to browse the customer contact file, the payroll database, or a contract negotiation file, save in a few cases where these files need to be repaired or reloaded from backups. Likewise, a payroll clerk may need read and write access to the payroll file, but not in general with the COPY utility, nor from a modem, nor in most cases at 4AM. Finally, a person who must have privileges to design a driver and test it should ordinarily not have the run of the file system as well.

Given examples like these, it is easy to see that simple authorization of user access to files is inadequate. While it is possible to build systems that grant identifiers to attempt some extra control, these can be circumvented by privilege, and create very long ACLs which become impossible to administer over a long period as users come and go.

What is needed is a mechanism that is secure, cannot be circumvented by turning on privileges, and which provides a simple to administer and fine grained control that lets you specify who can get at your critical files, with what images, when, from where, and with what privileges. It is also desirable to be able to control what privileges the images ever see, and to be able to check critical command files or images for tampering before use, so that they cannot be used as back doors to your system. It should be possible to demand extra authentication for particular files as well, and to prevent a malicious user from even seeing a particularly critical file unless he can be permitted access.

The ***Safety*** security subsystem is a VMS add-in security package which provides abilities to control security problems due to intruders, to damage or loss by system "insiders" (users exceeding their authority), and to covert code (worms and viruses). It provides a much easier management interface to handle security permissions than bare VMS and provides facilities permitting control over even privileged file accesses, for cases where there are privileged users whose access should be limited. Unlike systems which only intercept the AUDIT output, EACF can and does protect against ANY file accesses, and can protect files against deletion by unauthorized people or programs in real time as well as against access.

The ***Safety*** security subsystem offers the following capabilities:

* ***Files*** can be ***password protected individually.*** If a file open or delete is attempted for such a file and no password has been entered, the open or delete fails.

* Access can be controlled by ***time of day.*** Added protections can be in place only some of the time, access can be denied some times of day, write accesses can be denied at certain times, or various other modalities of access can be allowed.

* You can control ***who*** may access a file, ***where*** they may be (or may not be), ***with what images*** they may or may not access the file, and ***with what privileges*** the file may be accessed. Thus, for instance, it is trivial to allow a clerk access to the payroll file with the payroll programs, but **not** with COPY or BACKUP, not on dialup lines, and not if they have unexpected privileges. The privilege checks can be helpful where there are consultants working on a system who should be denied access to sensitive corporate information but who need privileges to develop programs, or in similar circumstances. You specify what privileges are permitted for opening

the file, and *a process with excess privileges is **prevented** from access*. Vital business data access should not always be implied by someone having privilege. With this system you can be sure your proprietary plans or data stay in house, and are available only to those with business reasons to need them, not to everyone needing system privileges for unrelated reasons. Unlike packages using the VMS Audit facility's output (which can be silently turned off by public domain code), ***Safety cannot** be circumvented by well known means.* Its controls are designed to leave evidence of what was done with them as well.

* You can ***hide files*** from unauthorized access. If someone not authorized to access a file tries to open it, they can be set to open instead some other file anywhere on the system. Meanwhile, ***Safety*** generates alarms and can execute site specific commands to react to the illegal access before it can happen. This can be helpful in gathering evidence of what a saboteur is up to without exposing real sensitive files to danger. Normal access goes through transparently.

* You can arrange that opening a file ***grants identifiers*** to the process that opens it and that closing it revokes these identifiers. Set an interpretive file to do this and set it to be openable only by the interpreter and you have a protected subsystem capability that works for 4GLs which are interpretive. (***Safety*** identifier granting, privilege modification, and base priority alteration is protected by a cryptographic authenticator preventing forging or duplication.)

* You can actively **prevent covert code (*viruses* and *worms*)** from running in two ways. First, ***Safety*** can attach a cryptographic checksum to a file such that the file will not open if it has been tampered with. Second, ***Safety*** can attach a privilege mask to a file which will replace ***all*** privilege masks for the process that opens it. By setting such a mask to minimal privileges, you can ensure that an untrusted image will never see a very privileged environment, and thus will be unable to perform privilege-based intrusions into your system even if run from a privileged user's account.

* You can ***control base priority*** by image. Thus, a particularly CPU intensive image can be made to run at lower than normal base priority even if it is run interactively.

* You can run a site-chosen script to further refine selection criteria. (Some facilities for doing additional checking while an image runs exist also.)

* You can have "suspect" images set a "low-integrity-image" mode in which all file opens are checked with a site script which can report or veto access. This can be used to track or regulate what a Java applet can do, in case someone happens to browse a web site which exploits a Java hole to browse your system or damage it.

Safety allows you to exempt certain images (e.g., disk defragmenters) from access checks, and it is possible to put a process into a temporary override mode also (leaving a record this was done) where this is needed. ***Safety*** facilities are controllable per disk, and impose generally negligible overhead. ***Safety*** will work with any VMS file structure using the normal driver interfaces. Also, ***Safety*** marking information resides sufficiently in kernel space that it cannot be removed from lower access modes, yet it uses a limited amount of memory regardless of volume size.

Best of all, the ***Safety*** protection is provided ***within the file system*** and does not depend on the audit facility. Thus it prevents file access or loss **before** it happens, and does not have to react to it afterwards. ***Safety*** allows all of its security provisions to be managed together in a simple screen-oriented display in which files, or groups of files, can be tagged with the desired security profiles or edited as desired. ***Safety*** protections are in addition to normal VMS file protections, which are left completely intact. Therefore, no existing security is broken or even altered. ***Safety*** simply adds additional checking which finally provides a usable machine encoding of "need to know" for the files where it matters.

The ***Safety*** Deletion Protection Subsystem.

Description: The ***Safety*** Deletion Protection System is designed to provide protection against accidental deletion of file types chosen by the site, and to allow files to be routed by the system to backup media before they are finally removed from the system. This is accomplished by an add-in to the VMS file system so that security holes are not introduced by the system's action.

The user interface is an ***UNDELETE*** command which permits one or more files to be restored to their original locations provided it is issued within the site-chosen time window after the undesired deletion took place. In addition, an ***EXPUNGE*** command is provided which allows files to be deleted at once, irretrievably, where space for such is required. Provision for automatic safe-storing of files prior to final deletion is present also in ***Safety*** DPS.

Safety DPS is implemented as a VMS file system add-in which functions by intercepting the DELETE operation and allowing the file to be deleted to be copied or renamed to a "wastebasket" holding area pending final action, and to be disposed of by a disposal agent. The supplied agent will allow a site script to save the files if this is desired, and then finally deletes any files which have been deleted more than some number N seconds ago. If the UNDELETE command is given, the file(s) undeleted are replaced in their original sites. The supplied system can also be configured to rename files to a wastebasket area or to copy them directly, for undeletion by systems people only. (These options are faster than the site command file option.)

Safety DPS can be configured to omit certain file types from deletion protection (for example, *.LIS* or *.MAP* could be omitted), to include only certain files in the protected sets, or both. This can reduce the overhead of saving files which are likely to be easily recreated, or tailor the system for such actions as saving all mail files (by selecting *.MAI for inclusion).

In addition, ***Safety*** DPS monitors free space on disks, and when a file create or extend would cause space exhaustion, ***Safety*** DPS runs a site script. By setting this script to perform final deletions, ***Safety*** DPS can be run in a purely automatic mode in which deleted files are saved as long as possible, but never less than some minimum period (e.g., 5 or 10 minutes).

Safety DPS files can be stored in any location accessible to VMS. If they are renamed, they must reside on the same disk they came from. Otherwise they can be stored in any desired place.

Safety DPS is installed and configured using a screen oriented configuration utility to set it up, and basically runs unattended once installed.

The ***Safety*** Storage Migration Subsystem

Description:

Safety has the ability to move files to secondary storage and automatically retrieve them when they are accessed. This backing can be similar to what HSM systems call "shelving", though it can be done in multiple levels, or it can be done in a way which permits files moved to secondary storage to be accessed there as though the files remained online. This resembles what are called "soft links" in Unix systems, in that file opens are transparently redirected to a file stored somewhere else reachable on the system, and the channel reset to the original device on close. A "readonly link" mode acts like a soft link for readonly access, and like an unshelve operation where a file is opened read/write, should this be desired. Full control over this shelving and unshelving is provided.

This provides a great deal of flexibility in reclaiming space when the ***Safety*** space monitoring function detects that space is needed. Not only can previously deleted files be finally moved to backup destinations and deleted, but the system can migrate seldom accessed files to nearline storage transparently. The site policy can drive this, or utilities provided can be used instead.

Where it is chosen to run ***Safety*** in a lights-out fashion (with ***Safety*** reacting to low disk situations by emptying older deleted files from the wastebasket and/or file migration to backing store), the policy chosen for controlling such setting is handled by a full-screen, easily used, tool which sets the policy. Should still greater flexibility be needed, the scripts used for a number of operations are supplied together with a full description of the command line interface of the underlying software. This facilitates linking ***Safety*** file management functions with other packages should such be desired.

Safety can be run in a mode where there is essentially no overhead at all imposed (just a few instructions added along some paths and no disk access) for any files except those which need softlinks or possible unshelving. There is no limit to how many files may be so marked on a disk. A fullscreen setup script allows one to select the ***Safety*** run modes. Even if ***Safety*** is forced to examine all files for its markings, the overhead imposes no added disk access and costs only a tiny added time (typically a percent or two) in open intensive applications. In addition, ***Safety*** can be turned off or back on at any convenient point should this be desired. (This must be done using special tools provided for use by those specially authorized to do so.)

Support:

Safety runs on VAX VMS 5.5 or greater or AXP VMS 6.1 or greater. The same facilities exist across all systems. HSM must be installed on each cluster node of a VMScluster where it is to be used but imposes no restrictions on types of disk it works for. ***Safety*** will work with any file structure used by VMS, so long as a disk class device is used to hold it. It is specifically NOT limited to use with ODS-2 disks.

Safety is available for 45 day trial use licenses or can be licensed permanently. ***Safety*** is available for 45 day trial use licenses or can be licensed permanently. ***Safety*** is required on every node of a cluster using it, or its benefits will not be available on nodes not having the software running. Apart from this, there are no problems with having ***Safety*** available on only part of a VMS cluster.

Safety is brought to you by

General Cybernetic Engineering
18 Colburn Lane
Hollis, NH 03049
603 465 9517

For orders, contact the above address or Sales@GCE.COM.
For technical information contact Info@GCE.Com