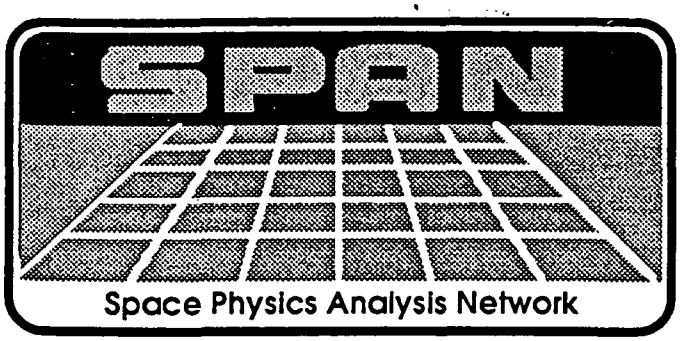


NSSDC/WDC-A-R&S 89-21

October 1989

SPAN SECURITY POLICIES AND GUIDELINES

by
Patricia L. Sisson
Science Applications Research
Lanham, Maryland 20771

and

James L. Green
National Space Science Data Center
Greenbelt, Maryland 20771

for the
National Space Science Data Center (NSSDC)
National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

NSSDC/WDC-A-R&S 89-21

SPAN SECURITY POLICIES AND GUIDELINES

Patricia L. Sisson
Science Applications Research
Lanham, MD 20777

James L. Green
National Space Science Data Center
Greenbelt, MD 20771

Revised: October 24, 1989

** P R E F A C E **

This document refers to computer networking security issues in general, as well as to those specific to the VAX computer environment. It assumes that the reader is acquainted with the VAX computer software architecture, networking terminology, and DCL, as well as other operating system software interfaces. Most references are to the VMS environment; future editions will address these same issues on the ULTRIX and UNIX computers.

The intended audience for this publication is the system or network manager of a VAX computer system that is currently connected to the Space Physics Analysis Network (SPAN) or is planning on connecting in the near future. Privileges are required to implement most of the changes and suggestions identified in this document.

** A C K N O W L E D G M E N T S **

The following individuals are recognized for their time and effort in making outstanding contributions to the content of this manual.

> Dan Anderson - Johnson Space Flight Center, Houston, TX

> Mike Bartman - Goddard Space Flight Center, Greenbelt, MD

> Linda Porter - Marshall Space Flight Center, Huntsville, AL

> Ron Tencati - Goddard Space Flight Center, Greenbelt, MD

We would also like to thank members of the Data Systems Users Working Group (DSUWG) security subcommittee for their contributions to this manual. Special thanks to the SPAN Management Team at the Goddard Space Flight Center for their helpful suggestions for the layout of this document.

CONTENTS

1	INTRODUCTION	1
1.1	Definition Clarification	1
1.2	System Manager Responsibilities	1
1.3	Computer Security Threats	2
1.4	Inappropriate Use Of A System	3
1.5	Computer Security Laws	4
2	SPAN SECURITY POLICIES	7
2.1	Password Policy	7
2.2	Captive Accounts	8
3	SPAN SECURITY REQUIREMENTS AND GUIDELINES	17
3.1	System Level Requirements And Guidelines	17
3.1.1	Authorization (UAF) Security	18
3.1.2	VMS Accounting/Security Alarms	19
3.2	DECnet - How To Protect It	20
3.2.1	DECnet Default Account	20
3.2.2	NETSERVER.LOGs	21
3.2.3	DECnet Objects	21
3.2.3.1	Task Object 0	21
4	IMPLEMENTATION TECHNIQUES	23
4.1	System-Level Guidelines	23
4.1.1	Authorization (UAF) Security	29
4.1.2	VMS Accounting/Security Alarms	31
4.2	DECnet - How To Protect It	36
4.2.1	DECnet Default Account	37
4.2.2	NETSERVER.LOGs	39
4.2.3	DECnet Objects	41
4.2.3.1	TASK Object 0	41
5	DETECTING/REPORTING SECURITY INCIDENTS	43
5.1	Detecting Security Incidents	43
5.2	Reporting Security Incidents	45
6	SPAN SECURITY TOOLKIT	47
6.1	Introduction	47
6.2	Why Was It Created?	48
6.3	Contents Of The Toolkit	48
6.4	License Agreement	55

1 INTRODUCTION

This document is a guide to system security with emphasis on requirements and guidelines that are necessary to maintain an acceptable level of security on the network. To have security for the network, each node on the network must be secure. Therefore, each system manager, must strictly adhere to the the requirements and must consider implementing the guidelines discussed throughout this document.

There are areas of vulnerability within the operating system that may not be addressed in this manual. However, when a requirement or guideline is discussed, implementation techniques will be included (in section 4) for completeness. Information related to computer and data security is discussed to provide information on implementation options. Although some items may seem obvious, they are included here for completeness. References will be made to other sources of information where appropriate. This information is presented as it relates to a VAX computer environment. Although current emphasis is on the VMS aspect, future editions of this manual will cover ULTRIX, UNIX, and other systems as they become connected to SPAN.

1.1 Definition Clarification

The document includes some information related to requirements and other information that is considered as guidelines.

The requirements MUST be implemented. They are mandatory for continued connectivity to SPAN. Throughout this document, requirements are indicated by the verb "WILL."

The guidelines SHOULD be implemented at the the discretion of the system manager or the organization. Throughout this document, guidelines are indicated by the verb "SHOULD."

1.2 System Manager Responsibilities

Network security begins with each user on a computer system. The system manager is usually the person who takes care of adding, modifying, or removing user accounts from a computer system and also is responsible, in most cases, for system security. The existing system manager position description may need further refinement to include the following security-related responsibilities:

- Contribute to the development of a security policy for the system (if what is presented in this document is not sufficient)

- Track personnel changes in the organization that might affect the vulnerability of the computer system
- Monitor all software updates, additions, or deletions to the system in order to detect any potential occurrence of a "Trojan horse," "virus," or "worm"
- Motivate management to give high priority to computer security awareness training for the organization
- Handle all computer security incidents to completion
  - Follow standard reporting and documentation procedures as described in this manual
  - Consult with the SPAN security manager to reach final resolution of all security incidents that involve other networks

Lax security practices on networked computers, especially in a research environment, can present security problems for other hosts on the network. Potential damage resulting from a successful break-in is easily unrecognized, not always fully understood, and, therefore, easily ignored. System managers are often not concerned with security and often do not know that the systems are easy targets for unauthorized access attempts or conduits for break-in attempts on other SPAN host computers. The following sections highlight information that affects the security of a computer system.

1.3 Computer Security Threats

There are several ways to identify security threats to a computer system. They can be categorized as follows:

- User Irresponsibility
  - Poor password choice
  - Giving out account/password to other users' files
  - "Browsing" on the system; reading other users' files
  - Sophisticated user trying to create a virus, worm, or Trojan horse

  - Inappropriate setting of file protections

- o System Penetrations
  - Use of holes in operating system
  - Use of poor security management
  - Use of dial-in modems or X.25 connections
  - Abuse of network objects

- o Potential Threats
  - Unauthorized users (commonly known as hackers)
  - Unhappy employees
  - Competitors
  - Foreign governments
  - Students

1.4 Inappropriate Use Of A System

Use of a computer and network is inappropriate if it:

- o Compromises the privacy of users and their personal data.
- o Destroys the integrity of data or programs stored on a computer system.
- o Disrupts the intended use of system or network resources.
- o Wastes resources (people, network bandwidth, or CPU cycles).
- o Uses or copies proprietary software when unauthorized to do so.
- o Uses a computer system as a conduit for unauthorized access attempts on other computer systems.

- Uses a government-, corporation-, or university-owned system for private purposes or for purposes not in the direct interest of the government, corporation, or university.

1.5 Computer Security Laws

SPAN network security guidelines are based on the following key United States laws:

- Title 18 of United States Code 1030, entitled The Computer Fraud and Abuse Act of 1986, which protects against unauthorized access. It is directed to

> "Whoever knowingly accesses a computer without authorization or exceeds authorization and ... obtains national defense information ... obtains information in a financial record ... affects use of the computer by the government ... alters, damages, or destroys information in a Federal interest computer ... modifies or impairs medical examinations, medical treatment or care ... traffics in any password through which a computer can be accessed without authorization ..."

  A Federal interest computer is in brief " ... a computer used exclusively by a financial institution or the United States government...or is one of two or more computers used in committing the offense, not all of which are located in the same State."

- Title 18 of United States Code 2701, entitled The Electronic Privacy Act of 1986, which protects against unauthorized interception of electronic communications. It is directed to "Whoever intentionally accesses without authorization a facility through which an electronic communication service is provided or intentionally exceeds an authorization to access that facility and obtains, alters, or prevents authorized access."

Other computer related fraud statutes include:

- Title 15 of United States Code 1693n, which deals with electronic fund transfer fraud.

- Title 17 of United States Code 506b, which deals with copyright violations.

- Title 18 of United States Code 1029, which deals with access code, password, logon sequence, credit card, and account number abuse.

- Title 18 of United States Code 1341, which deals with wire fraud.

Additionally, with regard to computer security, the Office of Management and Budget (OMB) has issued Circular No. A-71, Transmittal Memorandum No. 1, July 27, 1978, "Security of Federal Automated Information Systems," which requires that each Federal Government department and agency implement a computer security program.

In accordance with the OMB Circular A-71, NASA Management Instruction (NMI) 2410.7, "Assuring Security and Integrity of NASA Computer Processing," was issued to establish the NASA Computer Security Program to implement the policy contained in NMI 2410.7.

There are many existing state and local laws that pertain to computer security fraud. There is also the Computer Virus Eradication Act of '89, which has been introduced to the Senate. This is not yet a law, but it might become one during the next year or two.

2 SPAN SECURITY POLICIES

Computer and data security have always been important issues, but recent advances in remote computer access methods have highlighted the need for further computer security awareness. This document establishes a general SPAN policy position addressing computer and information security for SPAN.

2.1 Password Policy

User passwords on a computer system can be the system's biggest exposure to outside threats. The actual password, for a VMS system, is not stored in a normal text string. Passwords are one-way encrypted and are stored in the SYSUAF.DAT file located in the SYS$SYSTEM directory. This section will give the system manager a good idea for a password policy that can be implemented on the system.

There are simple rules that a user can follow in choosing a password. They are as follows:

- o Password Rules:
  - SHOULD be a minimum of 6 characters
  - SHOULD be a "pass-phrase" or include special characters
  - SHOULD NOT be any part of the user's name or userid
  - SHOULD avoid company, department, and project group names
  - SHOULD avoid hobbies, nicknames, and nonwork activities
  - SHOULD avoid names of family, friends, animals, or associates
  - SHOULD NOT be all the same characters
  - SHOULD NOT be found in the dictionary
  - SHOULD NOT be written on paper, saved in files, or transmitted electronically in mail messages or files
  - SHOULD NEVER be put into terminal function keys
- o Unprivileged Account Authorization Rules

- WILL be at least six characters in length and SHOULD contain at least one special character (ex: $, _, etc.). To set the password length in authorization, do the following:

  AUTH> MOD username/PWDMINIMUM=6

- Account owner SHOULD be required to change the password every 90 days (3 months). To set the password lifetime in authorization, do the following:

  AUTH> MOD username/PWDLIFETIME="90-"

o Privileged Account Authorization Rules

- WILL be at least eight characters in length and SHOULD contain at least one special character (ex: $, -, etc.). To set the password length in authorization, do the following:

  AUTH> MOD username/PWDMINIMUM=8

- Account owner SHOULD be required to change the password every 30 days (1 month). To set the password lifetime in authorization, do the following:

  AUTH> MOD username/PWDLIFETIME="30-"

- SHOULD be required to have dual passwords. To set this in authorization, do the following:

  AUTH> MOD username/PASSWORD=(PRIMARY,SECONDARY)

  However, beware that use of dual passwords will prevent explicit network access (e.g., NODEname"username password"::) to an account on the system.

2.2 Captive Accounts

Captive accounts are used on many systems for a variety of reasons. They are sometimes used to allow people who have no account on a system to perform some specific task, such as SET HOST to another machine, PHONE other users, get HELP on how to apply for an account, or TYPE specific files, such as system descriptions, contact lists, or a list of SPAN hosts.

Captive accounts can be very useful, but they are obvious security threats, since they allow nearly anyone to have some level of access to the system. Once some access is granted, it is always possible to increase the access through system bugs, clever tricks, and system configuration errors. DEC has made statements to the effect that it is impossible to make a captive account 100% secure, and that one should be very cautious in allowing them on the system. If one does decide to implement captive accounts, there are a number of things that should be avoided and still others that should be done. This section will attempt to cover some of the major dos and don'ts of captive account procedure construction.

A captive account is an account that, simply put, is forced to run a specific command procedure upon login. Being flagged as captive does not make an account secure. Implementation of adequate security is up to the login command procedure writer and whoever designs the account. To allow future writers of such procedures to avoid the mistakes of the past, the following list of things to do and things to avoid is presented. Many of these guidelines were discovered the hard way, by having someone break out of a captive procedure that violated the rules.

- o In System Authorization File

  - SHOULD ALWAYS set the CAPTIVE and LOCKPWD flag in AUTHORIZE

    ```
    UAF> MOD captive_account/FLAG=CAPTIVE

    UAF> MOD captive_account/LOCKPWD
    ```

    It seems obvious, but more often than you might think an account is created and used as a captive account, without setting the captive flag. Setting the captive flag on an account does three things:

    1. It identifies the account as captive for future manager reference.

    2. It disables Control_Y interrupts.

    3. It prohibits the specification of all login qualifiers, such as /COMMAND, /NOCOMMAND, /DISK, and /CLI, when a user logs in.

    It is also a good idea to consider the use of the DISWELCOME, DISMAIL, and DISNEWMAIL flags. DISRECONNECT and DISREPORT might be good ideas as well.

    This will prevent anyone who escapes the command procedure from disabling the account for other users.

This assumes that you have set a password on the account in the first place. It is also a good idea (though not for security reasons) to turn the password expiration feature off for the captive account.

- SHOULD NOT have privileges beyond TMPMBX and NETMBX

  Since it is not possible to make a captive procedure 100% escape proof, the best protection is to make sure that the account is harmless even if someone does get control of it. If any privileges are granted to a captive account, then they can be used to access the other parts of the system (files belonging to other users, system owned files, etc.). Some privileges are safer than others. Things like BYPASS, SYSPRV, SETPRV, CMKRNL, WORLD, and SECURITY are obviously dangerous and should never be granted to a captive account. Others, like TMPMBX and NETMBX, are so common that most system managers grant them to all accounts, including the captive ones. This is not always wise. NETMBX allows the account to SET HOST to other machines on the network and attempt break-ins, but the record shows the attempt as coming from your machine. TMPMBX is not as bad, but it will allow the account to be used to interfere with your own users (mostly limited to annoyance levels, but possibly serious interference with work). What would happen, for instance, if a job started sending large mail messages to everyone on the system until their disk quotas were exhausted or the disks were filled?

  In general, a privilege should never be given to a captive account unless it is vital to the functioning of the account. Even then, alternatives should be considered. One alternative is writing a program to perform the privileged function and then installing the program with the required privilege(s). This limits the use of the privilege to the required function and removes the need for the account itself to hold the privilege.

- SHOULD NEVER give a captive account a system UIC (will not fall within the MAXSYSGROUP group)

  Never set the UIC group number of a captive account to one of the system group of UICs (less than or equal to 10 octal by default, but check the value of the SYSGEN parameter MAXSYSGROUP to determine the value on your system). Giving the captive account one of these group numbers will automatically grant it SYSPRV privilege and allow access to sensitive parts of the system.

- SHOULD ALWAYS try to make the UIC group unique

  If you set up the captive account in the same group as other accounts, it may have access to the other account's files through the group protection. This is considerably worse if the member number is not unique as well! You can check the group number for uniqueness with the AUTHORIZE command:

  ```
  UAF> SHOW [groupuic,*]
  ```

- SHOULD consider setting the PRCLM limit to 1

  Unless the captive account has a need to spawn subprocesses, this limit should be one. This prevents the creation of subprocesses through the use of the DCL SPAWN command or the VMS MAIL SPAWN command, or through a SPAWN command on an X.25 public pad (see the discussion above for reasons not to allow the user access to MAIL in the first place).

- SHOULD ALWAYS limit access to the account as much as possible

  AUTHORIZE will allow for specification of limitations on access to accounts. Access to the account across the network or through dialup lines, for example, can be prevented. Also there can be a limit on the hours and the days that an account is available. Captive accounts should never be accessible through methods or at times when it is not necessary. If the account is only needed between 9 am and 5 pm on Monday through Friday, set it up that way. If it does not need to be used except by local users, restrict dialup and network access. There is no sense in being vulnerable more than absolutely necessary.

- SHOULD disable access via BATCH mode

  ```
  UAF> MOD captive_account/NOBATCH <cr>
  ```

o Captive Account Login Command Procedure

The following section will cover things that should be avoided when creating a Captive Account Login Command Procedure.

- SHOULD NEVER use the EXIT or (especially) the STOP statements

  There should never be any way for the procedure to EXIT.

The only way the captive procedure should ever terminate is with a LOGOUT command. Having an EXIT or STOP command accessible through the use of a password will never remain secure. If anyone figures out how to get VERIFY turned on (see the INQUIRE command, below), the password will not be protected. If anyone manages to get read access to the procedure file, the password is worthless. It is also possible that they will guess the password as well. There is no need to have an EXIT or STOP in a captive account command procedure. It is supposed to be captive, and it should stay that way.

In the event that one needs to have a nonprivileged person maintain a captive account (for updating data files or altering the captive login procedure itself), any of several things can be done:

1. Set up an ACL entry on the files that need to be maintained and let the person use the personal account to do the maintenance. This may not be practical if disk quotas are enabled and the two accounts are on different disks.

2. Set up a second account with the same UIC and home directory as the captive account, and let the person use this "captive manager" account to do the maintenance.

3. Let the files and quota be owned by the maintainer's account (or a special maintenance account), and set up ACL entries to allow the captive account to access them. This has the added advantage that modification of the files by the captive account itself may be easily prevented through the use of the correct protections, so that even if someone breaks out of the procedure they will not be able to do much damage or create "trap doors" to make access easier in future. See the discussion of version numbers on page 12 for further explanation of this subject.

- SHOULD NEVER use the INQUIRE command (always use the READ command)

  Use the READ/PROMPT=<prompt>/END=<end_label>/ERROR=<error_label> command instead. The INQUIRE command automatically evaluates any symbols included in the input. It is possible that one of these symbols might be something like, "''f$verify(1)'." This would set VERIFY "ON" and result in the user's being able to see how your procedure works, possibly allowing location of a weak

spot or a password (which should not be there in the first place, as is noted below and in the EXIT topic above). The inclusion of the /END option prevents the typing of a Control-Z from causing the procedure to branch into unintended code and possibly exiting. The /ERROR option allows device errors and other problems to be caught and handled (even if the method of handling the problem is just to log off!). Check the DCL Dictionary manual for more information on READ and its options.

- SHOULD always specify version numbers on all file specifications

  This includes the specification of the captive account procedure itself in the SYSUAF entry maintained by AUTHORIZE. If RENAME is used to make all of the sensitive files (those that would cause problems if they were altered by a trespasser) version one, then specify version one in all references. The modification of a file will result in a version two file, which will then be ignored. This is especially powerful in conjunction with the use of ACL entries or alternate ownership of the files; it prevents the captive account from deleting the original version one file and renaming the modified one back to version one. It helps reduce the damage in the event that the captive account procedure is not foolproof.

- SHOULD NEVER use "ON CONTROL_Y"

  If the account is set up with the DISCTLY authorization flag (as it should be!) there is no need for this command. Using it opens up the possibility that the user will type a CONTROL_Y, cause a trap to the handler routine, and then type one again before the trap can be reset. This will result in the procedure exiting, and a break-in would be sustained.

- SHOULD NEVER use an embedded password (in a command procedure)

  Embedded passwords are a definite "no-no"; they never stay secret. See the EXIT and INQUIRE commands for alternatives and reasons not to use them.

- SHOULD NEVER allow the user access to VMS MAIL

  The use of the /EDIT qualifier on the SEND or FORWARD commands allows full file access. Once one is in the editor, it is possible to modify any file to which the account has access, not just the message in progress.

The editor's INCLUDE and WRITE commands are very dangerous! This is especially true if the captive account has the ability to alter its own login command procedure. The procedure can be edited to allow an exit; the account is then reentered, the trap door used, and the account security violated.

- SHOULD NEVER allow user access to EDT or any other editor

  Never allow user access to EDT or any other editor that allows full file access. See the MAIL restriction above for details.

- SHOULD NEVER allow the user access to any program sending output to a disk file

  For the same reasons given under the MAIL restriction above, never allow the user to write files with user-selected file names.

- SHOULD NEVER evaluate a user-entered string without checking it first

  For example, the following two commands do the same thing, but the first is NOT secure. It allows the string "USERENTERED" to be evaluated, resulting in the execution of any lexical functions (such as f$verify...) that the user happened to type in.

```
IF "''USERENTERED'" .EQS. "FILES" THEN GOTO SHOWFILES

IF USERENTERED .EQS. "FILES" THEN GOTO SHOWFILES
```

  If the ability to evaluate a user-entered string is necessary, always check it for things like single quotes, "f$"s, double quotes, etc., before allowing evaluation to proceed.

- SHOULD always try to eliminate multilevel command procedures

  Procedures that call other procedures are harder to secure than ones with everything in one command file. They increase the number of sensitive files and make it harder to see all the code at one time. If you use multilevel procedures, remember that error traps for each level must be specified. Also note that all parameters sent to the new procedure are automatically evaluated (see the item on evaluating parameters, above).

- SHOULD always be sure to set up handling for all possible errors

  If any error occurs for which a trap has not been set up, the procedure will exit, leaving the user at the DCL prompt. There must be some sort of handling for ALL errors, even if the action taken is just to log out. Never use an OPEN, READ, etc., without specifying /ERROR=. Always use the ON ERROR command to set up general trapping (unless the SET NOON command is used, which will cause errors to be ignored, except those specified with the /ERROR= qualifiers).

- SHOULD never allow a captive account to use the TECO editor

  TECO has enough capability to break out of ANY captive command procedure.

- SHOULD consider placing an alarm ACL on the most sensitive files

  Setting an alarm ACL on the most sensitive files in the captive account can be a good way to catch people altering them. It will at least let you know that the file has been tampered with so that the damage can be repaired. The command might be:

```
$ SET ACL CLOGIN.COM /ACL= -
$_ (ALARM=SECURITY,ACCESS=WRITE+DELETE+CONTROL+FAIL+SUCCESS)
```

  And, on the security terminal:

```
$ REPLY/ENABLE=SECURITY
$ SET AUDIT/ENABLE=ACL/ALARM
```

- SHOULD consider putting the CLOGIN.COM file in a different directory and have it owned by SYSTEM. Have the authorization record for that account point to the correct file and directory path. The file must be "executable" by the captive account, however.

  In authorization:

```
UAF> MOD captive_account/LGICMD=dir_path:[dir]CLOGIN.COM;1
```

o Captive Account System Considerations

This section will cover additional things to consider while setting up a captive account.

- SHOULD always limit the disk quota of the account

  Never allow the captive account more quota than is needed for the functions of the account. This is especially true on disks where there are "overloaded" quotas (more quota assigned than there is disk space).

- SHOULD always check the SYLOGIN.COM (or equivalent) for violations of the above rules

  If the system has a SYS$SYLOGIN.COM procedure, check to be sure that it follows all of the above rules or has a line in it similar to the following near the top of the procedure (before any statements that violate any of the above rules):

  ```
  $ IF F$ENVIRONMENT("CAPTIVE") THEN $EXIT
  ```

  The system login procedure is as much a part of captive account security as the captive procedure itself.

- SHOULD have READ only protection mask on the captive accounts files

  ```
  $ SET FILE/PROT=(O:R) [...]*.*;* <cr>
  ```

  This will change the file protection to Owner:READ only for all files owned by the captive account (in all directories).

3 SPAN SECURITY REQUIREMENTS AND GUIDELINES

This section will list, in brief form, all of the "Requirements" and "Guidelines" for systems connected to SPAN. Section 3 ("Implementation Techniques") will give explicit details on how to check certain system parameters, along with how to implement the requirements or guideline listed in this section.

3.1 System Level Requirements And Guidelines

The following requirements and guidelines pertain to overall system security.

- o SYS$ANNOUNCE/SYS$WELCOME Message
  - WILL change the system announcement message or system welcome message to:
    - set value to "null" (blank)
    - set value to predescribed statement (see Section 4.1 for examples of statements to use)

- o WILL delete USERLIST.LIS and NODEINFO.LIS files out of the DECnet default directory
- o SHOULD consider setting a password on the dial-up lines (this would work for systems that have dial-up modems attached to them)
- o SHOULD consider using the "secure server" feature of VMS for terminals (ports) on your system
- o Authorizing New Accounts or Modification of Existing Accounts
  - SHOULD create an authorization form to request changes to the System Authorization file and keep these forms on file (for future reference)

- o Directory and File Protection
  - UIC Based Protection
    - Default protection should be, at minimum, (S:RWED,O:RWED,G:RE,W)

- Access Control Lists (ACLs)
  - Can be used to grant or deny access to a system object. Can be placed on the following objects:
    - Devices
    - Files (including directory files)
    - Group global sections
    - Logical name tables
    - System global sections
    - Queues (with V5.0 VMS)

  This manual will not go into detail on how to set up any ACL identifiers. For complete information on how to set these up, please refer "Guide to VAX/VMS System Security."

3.1.1 Authorization (UAF) Security

- o USER and USERP accounts WILL have a password associated with them or WILL be deleted or WILL be DISUSERed (these are for MicroVMS systems only)
- o Accounts FIELD, SYSTEST, and SYSTEST_CLIG (if they exist) SHOULD be DISUSERed when not in use
- o The flag DISREPORT should not be used on any account
- o Shared accounts SHOULD NOT be used
  - SHOULD set DISRECONNECT flag for shared accounts (if possible).
- o SHOULD limit hours of use for accounts (if possible)
  - DIALUP and NETWORK access are more anonymous than directly connected terminals

  - DIALUP and NETWORK are easier for intruders to use to try to break into systems
  - Certain data only need to be accessible onsite
  - Certain users need to be supervised and do not need access when their supervisor is not around

- User accounts SHOULD NOT fall within MAXSYSGROUP value (usually <= 10 octal). Only system accounts should fall within this group.
- User accounts should have no privilege beyond TMPMBX and NETMBX (unless formerly approved by the system security officer or by the user's supervisor).

3.1.2 VMS Accounting/Security Alarms

VMS accounting should be turned on. For systems with an X25 pad, turn on PSI accounting. Also turn on auditing and security alarms on the VMS system.

- VMS accounting - at a minimum log:
  - DETACHED (detached job termination)
  - INTERACTIVE (interactive job termination)
  - LOGIN_FAILURE (login failures)
  - NETWORK (network job termination)
  - PROCESS (any process termination)
  - SUBPROCESS (subprocess termination)
- PSI accounting should be turned on
- Check/modify LGI SYSGEN parameters on the system
- Auditing and security alarms
  - Security alarms should be enabled for the following:

- ACL (any access to a file with an ACL protection)
- AUDIT (any change to the current AUDIT)
- AUTHORIZATION (any modification to the current System Authorization file SYSUAF.DAT)
- BREAKIN (any break-in from any source)
- INSTALL (any access to the INSTALL utility)
- LOGFAILURE (DIALUP,REMOTE,NETWORK)

3.2 DECnet - How To Protect It

3.2.1 DECnet Default Account

The following requirements and guidelines pertain to the DECnet default account on a VAX/VMS system:

- o In system authorization file
  - WILL change DECnet default password (remember to change default password on the executor in NCP)
  - WILL have no privilege beyond TMPMBX and NETMBX
  - UIC WILL not fall within the MAXSYSGROUP group value
  - UIC should be in a unique group
  - SHOULD only allow NETWORK access
  - SHOULD consider setting LOCKPWD, CAPTIVE, DISMAIL, DISCTLY flags
  - SHOULD consider setting CPU time limit
  - WILL set LOGIN command to null
  - SHOULD consider not having a DECnet default account at all

3.2.2 NETSERVER.LOGs

- SHOULD edit NETSERVER.COM (NOTE: Not supported by DEC) to stop automatic purging of NETSERVER.LOGs
- SHOULD protect NETSERVER.LOG files (S:RWD,O,G,W)
- SHOULD modify NETSERVER$TIMEOUT (length of time "server" processes live after object termination)
- SHOULD use FAL$LOG to enhance audits

3.2.3 DECnet Objects

3.2.3.1 Task Object 0 -

- WILL protect or remove Task Object 0 from the DECnet data base by one of the following methods:
  - Remove it completely from the DECnet data base
  - Specify an invalid username for the Task Object (allow you to control access to the Task Object) in the permanent data base.
  - Move it to a protected directory and require system staff help to access and use it. This does not prevent use of Task Object 0 once it is set up, but at least system staff will know who or what is using Task Object 0.

4 IMPLEMENTATION TECHNIQUES

The following sections will give the additional detail needed to implement the SPAN Security Requirements and Guidelines. Where applicable, detailed steps will be given.

4.1 System-Level Guidelines

- SYS$ANNOUNCE/SYS$WELCOME Message

  All node managers on SPAN WILL change their system announcement message (message that is displayed before the username prompt) to one of the following:

  - Set value of SYS$ANNOUNCE to null (blank)

    ```
    $ ASSIGN/SYSTEM SYS$ANNOUNCE " "
    ```

    Be sure to have this line in your system startup file SYSTARTUP.COM (or SYSTARTUP_V5.COM on V5.x systems) so that each time the system reboots this message is displayed.

  - To set value of SYS$ANNOUNCE, do the following:

    ```
    $ ASSIGN/SYSTEM SYS$ANNOUNCE ANNOUNCE.TXT
    ```

    In the file ANNOUNCE.TXT, have it say:

    ```
    ANY UNAUTHORIZED ATTEMPT TO ACCESS THIS SYSTEM IS A FEDERAL OFFENSE
    ```

    or

    ```
    *WARNING**WARNING**WARNING**WARNING**WARNING**WARNING**WARNING
    *                                                             *
    * This computer is operated by/for the U.S. Government.       *
    * Unauthorized access to and/or use of this computer system   *
    * is a violation of law and punishable under the provisions   *
    * of 18 USC 1029, 18 USC 1030, and other applicable statutes. *
    *                                                             *
    *WARNING**WARNING**WARNING**WARNING**WARNING**WARNING**WARNING*
    ```

    or

  You are connected to a U.S. Government computer system. Any unauthorized ATTEMPT to gain access to this system may be punishable by fine and/or imprisonment.

If a computer system does not have a SYS$ANNOUNCE message, then the SYS$WELCOME (banner displayed after successfully logging in) should be modified to include the above statement. To do this, do the following:

- Set value of SYS$WELCOME

  $ ASSIGN/SYSTEM SYS$WELCOME WELCOME.TXT

  Have the file WELCOME.TXT include one of the above examples.

- WILL delete USERLIST.LIS and NODEINFO.LIS files

  Several earlier versions of the SPAN documentation set have advocated the use of the files USERLIST.LIS and NODEINFO.LIS. The USERLIST.LIS file was to contain a listing of all users on the system and their usernames. The NODEINFO.LIS file was to contain pertinent information about the computer, such as system manager name, network address and phone number, VMS version running on the system, type of VMS system, etc. These files, as stated in earlier documentation, were to be kept in the DECnet default directory for use by other colleagues in the SPAN community to help them locate other researchers.

  Because of the increase in unauthorized access attempts on systems on SPAN, it has been determined that the files USERLIST.LIS and NODEINFO.LIS WILL be deleted from all SPAN nodes. These files make it easier for an unauthorized user to try valid usernames to break into a computer. NODEINFO.LIS can be used to attack a system running a particular version of VMS, especially one with known holes.

- SHOULD consider setting a password on the dial-up lines (this would work for systems that have dial-up modems attached to them)

  If a computer system has dial-up modems attached to it, it is a good idea to set a system password for these lines. The steps are as follows:

  - Invoke AUTHORIZE and set up the system password

    UAF> ADD/SYSTEM_PASSWORD = xxxxxxxx

  - Modify the terminal set-up file for those lines that coorespond to the dial-up modems

    $ SET TERM xxxx: /SYSPWD/PERMANENT

- SHOULD consider using the "secure server" feature of VMS for terminals (ports) on the system

  The "secure server" feature of VMS ensures that users can only log into terminals (ports) that are already logged out. It requires the user to press the BREAK key on a terminal, and the secure server, if enabled, responds by first disconnecting any logged in process. The purpose of the secure server is to ensure that the VAX/VMS login program is the ONLY program able to receive the login. This prevents just about all possibilities of a user logging into a program that mimics the login sequence. These programs, typically, steal passwords. To use the secure server, do the following:

  - Modify the terminal set-up file for those lines that correspond to your dial-up modems

    ```
    $ SET TERM xxxx: /SECURE/DISCONNECT/PERMANENT
    ```

    Be sure to modify the appropriate file that is invoked by SYSTARTUP.COM [SYSTARTUP_V5.COM under VMS V5.x] so that the terminal characteristics will remain the same after reboot.

  This feature is documented in greater detail in the "Guide to VAX/VMS System Security" written by Digital Equipment Corporation.

- Authorizing New Accounts

  An accounting authorization form should be used to request additions, changes, and deletion of user accounts on a computer system. This form, if not already in existence, should contain the following information (at a minimum):

  1. Full name, address, and phone number for the account owner

  2. Disk and quota specifications

  3. Privileges, besides the normal TMPMBX,NETMBX, that are required

  4. Expiration date for the account

  There should be in effect an account authorization mechanism whereby the system manager and whomever that person is accountable to has signed off. The form should have a spot that clearly states that the user has received a security briefing from the system security officer. It also should state that the user understands the responsibilty for

computer security.

o Directory and File Protection

- UIC based protection

  The following SYSGEN parameter will show the default VMS protection.

```
$ MCR SYSGEN <cr>
SYSGEN>  SHO RMS_FILEPROT <cr>
Parameter Name    Current   Default   Minimum   Maximum Unit  Dynamic
--------------    -------   -------   -------   ------- ----  -------
RMS_FILEPROT      64000     64000         0     65535 Prot-mask
SYSGEN>  EXIT <cr>
```

The protection is expressed as a mask. By default, the mask is 64000 (decimal) or FA00 (hexidecimal), which represents the protection (S:RWED,O:RWED,G:RE,W).

To change the VMS default protection on the system, the system manager should add the following line to the SYLOGIN.COM (every user account invokes this system-wide login command procedure upon login to your system).

```
$ SET PROTECTION=(S:RWED,O:RWED,G,W)/DEFAULT
```

If users on the system want to change the normal file protection for files that they create, they will have to change the protection after creation of the file or add the appropriate SET PROTECTION line to their personal login command procedure.

- Normal sequence of events VMS does to determine accessibility of a file

  When someone is trying to access a file on the system, VMS goes through the following sequence to validate access to that file:

  - If the file has an ACL identifier:

    - If the ACL grants access, then access is given

    - If the ACL denies access, then use SYSTEM and OWNER fields of the UIC-based protection

    - If the ACL does not explicitly grant or deny access, then use the UIC-based protection

- If the file has no ACL identifier:
  - Use the UIC-based protection mask

- If the user has BYPASS, READALL, SYSPRV, or GRPPRV privileges, then VMS allows the appropriate access to the file

- UIC-Based Protection

  Files have an ownership UIC and a protection mask. The default protection mask for most systems is (S:RWED,O:RWED,G:RE,W). This means:

  - SYSTEM - S:RWED
    - System - READ, WRITE, EXECUTE, and DELETE access to the file. System (S) constitutes:
      - User's UIC group number <= MAXSYSGROUP
      - User has SYSPRIV privilege
      - User has GRPPRV privilege and the user's UIC group number is equal to the file's UIC group number
      - User's UIC is equal to the file's UIC group

  - OWNER - O:RWED
    - Owner - READ, WRITE, EXECUTE, and DELETE access to the file. Owner (O) constitutes:

      User's UIC group and member number matches the file's UIC group and member number

  - GROUP - G:RE
    - Group - READ and EXECUTE access to the file. Group (G) constitutes:

      User's UIC group number matches the file's group number

- WORLD - W

  - World - no access

    Any user not in the system, owner, or group category on the system. Includes all users on the network.

To show the current protection mask of a file, do the following:

```
$ DIR/PROT filename.ext

Directory:  directory_path:[username]

filename.ext;version_#          (S:RWED,O:RWED,G:R,W)
```

or

```
$ DIR/SEC filename.ext

Directory:  directory_path:[username]

filename.ext;version_#     [Username] (S:RWED,O:RWED,G:R,W)
```

To change the current protection mask of a file, the user would do the following:

```
$ SET FILE/PROT=(W:RWED) filename.ext
```

We discourage all users of a computer system from allowing world access to their files. With the protection set to W:RWED, anyone on the system or on the network is allowed to READ, WRITE, EXECUTE, and DELETE these files.

- ACL-Based Protection

Please refer to the Digital Equipment Corporation's "Guide to VAX/VMS Security" for details on how to use ACL identifiers. NOTE: There will be an increase in the system overhead when accessing files. ACLs are not easy to maintain and you might cause problems when using BACKUP and ACL recovery. It may be preferred to set these up only on system-critical files.

4.1.1 Authorization (UAF) Security

The system manager should also consider the following for account security:

- o USER and USERP accounts WILL have a password associated with them or WILL be deleted or WILL be DISUSERed (these are for MicroVMS systems only)

  To check the status of the USER and USERP accounts on the system, do the following in AUTHORIZE,

  ```
  UAF> SHO USER <cr>
  ```

  which will result in the display of either the UAF record or the error message %UAF-W-BADSPC, no user-matched specification (which means that this account does not exist on the system).

  To set a password on this account, do the following:

  ```
  UAF> MOD user/PASSWORD=xxxxxxxx <cr>
  ```

  (where xxxxxxxx is the desired password).

  To delete this account from the system, do the following:

  ```
  UAF> REMOVE user <cr>
  ```

  To DISUSER this account, do the following:

  ```
  UAF> MOD user/FLAG=DISUSER <cr>
  ```

  The account USERP can be substituted for USER; the commands are the same. Do not forget to delete the directory and files that are owned by these accounts.

- o The accounts FIELD, SYSTEST, and SYSTEST_CLIG (if in existence) should be set to DISUSER when not in use

  ```
  UAF> MOD field/FLAG=DISUSER <cr>
  ```

  - SHOULD consider modifying the system's midnight or daily system command procedure (if in existence) to automatically DISUSER the accounts FIELD, SYSTEST, and SYSTEST_CLIG

- The flag DISREPORT SHOULD NOT be used on any account
- Shared accounts SHOULD NOT be used, but if necessary:
  - Shared accounts should have the DISRECONNECT flag set

    UAF> MOD shared_account/FLAG=DISCONNECT <cr>

- If an account has no legitimate uses outside certain hours or requires only specific restricting access methods and/or hours
  - Consider setting the following in AUTHORZE
    - /NONETWORK - the user cannot log in from the network (could be appropriate for the OPERATOR account or FIELD account)
    - /NOREMOTE - the user cannot log in remotely
    - /NODIALUP - the user cannot use the dial-in modems to access the system
    - /NOBATCH - the user cannot run batch jobs on your system (good for student accounts that do not need BATCH capability)
  - /PRIMEDAYS - only allow access to the system on particular days

    An example would be a user account called PENNY (whose owner Copper Penny never works on weekends). Set the account to only allow access Monday-Friday. Command would be

    UAF> MOD penny/PRIMEDAYS=(NOSAT,NOSUN) <cr>

    where the account is not allowed access on Saturdays or Sundays

    or

    UAF> MOD penny/PRIMEDAYS=(MON,TUE,WED,THU,FRI) <cr>

    where the account is only allowed access Monday through Friday.

- User accounts should not fall within the MAXSYSGROUP group value

```
$ MCR SYSGEN <cr>
SYSGEN>  SHO MAXSYSGROUP <cr>

Parameter Name     Current   Default   Minimum   Maximum Unit  Dynamic
--------------     -------   -------   -------   ------- ----  -------
MAXSYSGROUP            8         8         1     32768 UIC Group   D
SYSGEN>  EXIT <cr>
```

This will show the current settings for the system.

The default parameter value says that any account that has the UIC group between 0 and 10 will automatically get full system privileges. As an example:

```
$ SD SYS$SYSTEM <cr>
$ RUN AUTHORIZE <cr>
UAF> SHOW/BRIEF [1,*] <cr>

       Owner    Username  UIC     Account  Privs Pri Directory

System Manager  SYSTEM    [1,4]   SYSTEM   All     4 SYS$SYSROOT:[SYSMGR]
Field Service   FIELD     [1,3]   SYSTEM   All     4 SYS$SYSROOT:[SYSMAINT]
UAF> EXIT
```

Make sure that only system-type accounts (SYSTEM, FIELD) fall within this group.

- User accounts should have no privilege beyond TMPMBX and NETMBX

4.1.2 VMS Accounting/Security Alarms

- VMS Accounting

  The following line should appear somewhere in the system startup file (SYSTARTUP.COM):

  ```
  $ SET ACCOUNTING/ENABLE=(DETAC,INTER,LOGIN,NET,PROC,SUBP)
  ```

- PSI accounting SHOULD be turned on

  If the PSI software is being used, its accounting should be enabled to track X.25/X.29 calls. Somewhere in the system startup file, the following line should appear:

```
$ @SYS$MANAGER:PSIACCOUNTING ON
```

- Check/modify SYSGEN LGI parameters on the system

  To check the SYSGEN LGI parameters on the system, do the following:

```
$ MCR SYSGEN <cr>
SYSGEN>  SHO/LGI <cr>

Parameters in use: Active

Parameter Name    Current  Default  Minimum  Maximum Unit     Dynamic
--------------    -------  -------  -------  ------- ----     -------
LGI_BRK_TERM        0        1        0        1 Boolean      D
LGI_BRK_DISUSER     0        0        0        1 Boolean      D
LGI_PWD_TMO        30       30        0      255 Seconds      D
LGI_RETRY_LIM       3        3        0      255 Tries        D
LGI_RETRY_TMO      20       20        0      255 Seconds      D
LGI_BRK_LIM         5        5        0      255 Failures     D
LGI_BRK_TMO       960      300        0       -1 Seconds      D
LGI_HID_TIM       300      300        0       -1 Seconds      D

SYSGEN>  EXIT
```

  Several of these parameters determine WHEN break-in and evasion take place on the system. It is a good idea to change the default settings so that break-in and evasion kick off earlier than the default. SYSGEN LGI parameters are dynamic, so any change will take immediate effect on the system. Remember, though, to add the modified LGI parameters to the MODPARAMS.DAT file (located in the SYS$SYSTEM directory) so that the next time that AUTOGEN and reboot are run, the tightened LGI parameters will still be in effect.

  This document will only cover those LGI parameters we feel should be changed from their default settings. For additional information on LGI parameters, please see the "VAX/VMS Utilities Reference Volume" and the "Guide to VAX/VMS System Security" manuals.

- LGI_BRK_TERM - default setting for this parameter is 1.

  If the system parameter is 1, a user can attempt to log onto the system and fail several times until it trips break-in/evasion, but only from THAT terminal. A user can simply go to another terminal (or start another LAT session) and try all over again. If this parameter is turned off (set value to 0), log failures are tracked by username across ALL terminal types. To change the default setting, do the following:

  ```
  $ MCR SYSGEN <cr>
  SYSGEN>  SET LGI_BRK_TERM 0 <cr>
  SYSGEN>  WRITE ACTIVE <cr>
  SYSGEN>  WRITE CURRENT <cr>
  SYSGEN>  EXIT <cr>
  ```

- LGI_BRK_TMO - this is the average number of minutes allowed between login failures (e.g., default setting is 300 seconds or 5 minutes). This means that the system will not notice login failures that occur to accounts every 5 minutes. To change this default setting, do the following:

  ```
  $ MCR SYSGEN <cr>
  SYSGEN>  SET LGI_BRK_TMO 960 <cr>
  SYSGEN>  WRITE ACTIVE <cr>
  SYSGEN>  WRITE CURRENT <cr>
  SYSGEN>  EXIT <cr>
  ```

- LGI_BRK_LIM - number of tries before break-in is detected. This parameter, along with LGI_HID_TMO, determines when the system decides that there is a break-in attempt in progress. If you have noisey dialup lines or thumb-fingered users, a LGI_BRK_LIM less than 3 could inadvertently lock out legitimate users. To change the default setting, do the following:

  ```
  $  MCR SYSGEN <cr>
  SYSGEN>  SET LGI_BRK_LIM 3 <cr>
  SYSGEN>  WRITE ACTIVE <cr>
  SYSGEN>  WRITE CURRENT <cr>
  SYSGEN>  EXIT <cr>
  ```

- LGI_HID_TIM (random number between 1 and 1.5 seconds) - the duration for the evasive action is influenced by the LGI_HID_TIM parameter value. This helps to obscure the true evasion time.

o Auditing and Security Alarms

Security alarms are messages that are sent to the security operator's terminal when specific events take place. For the system security manager, these alarms assist in maintaining an acceptable level of security. They can help in detecting break-in attempts to the system and users doing undesirable things to the system.

The VAX/VMS system allows several types of alarms to be turned on. The following events can be audited:

- ACL (any access to a file with an ACL protection)

  One can audit successful or unsuccessful access to a file. To do so, add an ALARM_JOURNAL ACE to a file's ACL table. See the ACL section in this manual

  Then enable the ACL alarms. The syntax is

  ```
  $ SET AUDIT/ALARM/ENABLE=ACL
  ```

- AUDIT (any change to the current AUDIT)

  One can enable auditing for execution of the SET AUDIT command. The syntax is:

  ```
  $ SET AUDIT/ALARM/ENABLE=AUDIT
  ```

- AUTHORIZATION (any modification to the current system authorization file SYSUAF.DAT)

  - Changes to system authorization data base

    One can audit changes made to the system or network UAF data base. The types of changes that can be audited are:

    - adding a system UAF record (a new user)
    - deleting a system UAF record (delete a user)
    - changing a system UAF record (change a user record)
    - copying a system UAF record (a new user)
    - renaming a system UAF record
    - adding a network UAF record
    - deleting a network UAF record
    - modifying a network UAF record

    The syntax is:

    ```
    $ SET AUDIT/ALARM/ENABLE=AUTHORIZATION
    ```

Example of an alarm:

```
%%%%%%%%%%  OPCOM  10-MAY-1989 09:48:04.43  %%%%%%%%%%
Security alarm on NSSDCA / Identifier added
Time:          10-MAY-1989 09:48:04.42
PID:           00000000
User Name:     SYSTEM
Image:         HSC$DUA0:[SYS0.SYSCOMMON.][SYSEXE]AUTHORIZE.EXE
ID name:       new_user_id
ID value:      [999,99]
Id attributes:     (None)
```

- Changes to rights data base

  One can audit changes made to the rights data base. The types of change that can be audited are:

  - creation of a new rights data base
  - addition of an identifier
  - removal of an identifier
  - granting of an identifier to a holder
  - revoking of an identifier from a holder
  - modification of a holder

  The syntax is:

  ```
  $ SET AUDIT/ALARM/ENABLE=AUTHORIZATION
  ```

- BREAKIN (any break-in from any source)

- INSTALL (any access to the INSTALL utility)

- LOGFAILURE (DIALUP,REMOTE,NETWORK)

- File_Access - selected types of access to files and global sections. An example would be:

  Turn on the alarm:

  ```
  $ SET AUDIT/ALARM/ENABLE=(FILE_ACCESS=(BYPASS:DELETE,-
              SYSPRV:DELETE))
  ```

  Example of output in OPERATOR.LOG:

  ```
  %%%%%%%%%%  OPCOM  11-MAY-1989 00:00:28.08  %%%%%%%%%%
  Security alarm on NSSDCA / Successful file access
    Time:          11-MAY-1989 00:00:28.07
    PID:       00000000
    User Name:     username
    Image:         system_disk:[SYSEXE]DELETE.EXE
  ```

```
File:           diskname:[user_directory]filename.ext
Mode:           DELETE
Privs Used:     SYSPRV
```

One might want to set up a specific terminal or port on the system to have all alarms sent to it. One could add something like the following to the SYSTARTUP.COM:

```
$ DEFINE/USER SYS$COMMAND Txxx:
$ REPLY/ENABLE=SECURITY
$ SET AUDIT/ALARM/ENABLE=(ACL,AUDIT,AUTHORIZATION,INSTALL,-
    BREAKIN=(DIALUP,NETWORK,REMOTE),-
    LOGFAIL=(DIALUP,NETWORK,REMOTE)
```

Or, one could turn on all components of the BREAKIN and LOGFAIL detection (no matter what the source) by doing the following:

```
$ DEFINE/USER SYS$COMMAND Txxx:
$ REPLY/ENABLE=SECURITY
$ SET AUDIT/ALARM/ENABLE=(ACL,AUDIT,AUTHORIZATION,INSTALL,-
    BREAKIN=(ALL),LOGFAIL=(ALL)
```

One could also have this type of information going to the system console port. To do this, add the following to SYSTARTUP.COM:

```
$ DEFINE/USER SYS$COMMAND OPA0:
$ REPLY/DISABLE=SECURITY
```

Remember that once a user trips the break-in alarm, the incorrect passwords are printed in the security log file. It may be relatively easy to some unauthorized person to "guess" at a user's correct password from seeing the incorrect password displayed on the console. For this reason, it is a good idea NOT to print security log messages to the system console port (OPA0:).

- ACL - event request by an ACL on a file or global section

4.2 DECnet - How To Protect It

4.2.1 DECnet Default Account

- o In system authorization file

  - WILL change DECnet default password

    The DECnet default account permits certain types of access to your system from remote nodes without requiring them to specify the account and password information (i.e., access control string). Instead, this information is supplied in the DECnet executor and object data bases. When installing DECnet on a VAX/VMS system, the default settings for the DECnet account and network objects are:

    ```
    Username:  DECnet
    Password: DECnet
    ```

    This allows the following:

    - Any remote user can access files on the system based on the world protection mask (W:RWED) of that file

    - Any remote user can, with a little ingenuity, run programs or command procedures on the system via the DECnet account

    If the default password is being used for the DECnet account (password is DECnet), change it. Someone can use the explicit access control string "DECnet DECnet" to run a remote job on the system. To change the password, do the following:

  - Run authorize and change the default account password

    ```
    UAF> MOD DECNET/PASSWORD=xxxxxxxx
    ```

    where DECNET=default account name. This may have been changed already.

  - Change password that is referenced by the DECnet object data base (Note: BYPASS and CMKRNL privileges required)

    ```
    $ RUN SYS$SYSTEM:NCP
    NCP> SHO EXEC CHAR
    NCP> MOD EXEC NONPRIV PASSWORD xxxxxxxx
    NCP> SHO KNOW OBJ
    (any object that has a DECnet password associated with it
     must show the new password)
    NCP> SET OBJECT FAL USER DECNET PASSWORD xxxxxxxx
    ```

If the password in the DECnet object data base does not match the password in the SYSUAF record, then a "login information invalid" error will be generated. Any network access requests for the DECnet object will fail.

- The DECnet account WILL have no privileges beyond TMPMBX and NETMBX

  Under NO circumstances should the default account be given more than the above privileges. DECnet only requires the above privileges to work; giving this account extra privileges is of little use and could be a large security problem.

- UIC WILL NOT fall within the MAXSYSGROUP group value

  See previous instructions under Authorization (UAF) Security on how to check this value to make sure this account does not fall within the MAXSYSGROUP group value.

- UIC should be in a unique group

- SHOULD have only NETWORK access in authorization:

  In authorization:

  ```
  UAF> MOD DECNET/NETWORK/NODIALUP/NOREMOTE/NOBATCH/NOINTERACTIVE
  ```

- SHOULD consider setting LOCKPWD, CAPTIVE, DISMAIL, and DISCTLY flags in AUTHORIZE

  In authorization:

  ```
  UAF> MOD DECNET/FLAGS=(LOCKPWD,CAPTIVE,DISMAIL,DISCTLY)
  ```

  LOCKPWD locks the password and prevents network users from changing it. Setting CAPTIVE and DISCTLY prevents network users from misusing the account. DISMAIL stops people from sending mail using this account.

- SHOULD consider setting a CPU time limit on this account

  Prevents long network jobs from executing. Usually network jobs should take no longer than a few seconds. In authorization:

  ```
  UAF> MOD DECNET/CPU=00:01:00
  ```

  Putting a CPU limit on the DECnet default account may cause problems if a lot of people copy large files to and from the

system. Please keep this in mind when choosing to use this option.

- o WILL set the LOGIN command to the null device

  In authorization:

  ```
  UAF> MOD DECNET/LGICMD=NL: <cr>
  ```

  Prevents network users from copying a LOGIN.COM into the default account and having it execute.

- o WILL set the PRCLM value in the UAF record to 1.

  Prevents remote users from executing the SPAWN command.

- o SHOULD consider not having a DECnet default account at all

  In authorization:

  ```
  UAF> REMOVE DECNET <cr>
  ```

4.2.2 NETSERVER.LOGs

- o SHOULD edit NETSERVER.COM (NOTE: Not supported by DEC) to stop the automatic purging of NETSERVER.LOGs. To do this:

  ```
  $ SET DEF SYS$SYSTEM <cr>
  $ EDIT NETSERVER.COM
  ```

  Comment out the following lines:

  ```
  $! IF PERMANENT_NETSERVER_COUNTER .LT. 25 THEN GOTO LOOP
  ```

  and

  ```
  $! IF F$SEARCH ("SYS$LOGIN:NETSERVER.LOG;-10") .NES. "" THEN -
  $!   PURGE/KEEP:3 SYS$LOGIN:NETSERVER.LOG
  ```

  This will stop the automatic purge of the NETSERVER.LOG files. The command procedure NETSERVER.COM is invoked each time a new server process starts up (e.g., incoming network connection for a DECnet object). Remember: The next time the system software is upgraded, it will be necessary to modify the command procedure NETSERVER.COM for this to stay in effect.

- SHOULD protect NETSERVER.LOG files

  This will protect the NETSERVER.LOG files contained in the DECnet default directory from unauthorized browsing or deleting. It also makes it impossible for a hacker to completely cover the tracks.

  - Suggest (S:RWED,O,G,W)

- SHOULD modify NETSERVER$TIMEOUT (length of time server processes live after object termination)

  This makes each network connection create its own NETSERVER.LOG instead of having multiple object accesses in one NETSERVER.LOG file. It also forces a VMS accounting record for EACH network access.

  - To set this value to 10 seconds, do the following:

    ```
    $ DEFINE/SYSTEM/EXEC NETSERVER$TIMEOUT "000 00:00:10"
    ```

    NOTE: This will cause some additional processes to be created on the system and might cause system degradations.

  - SHOULD use FAL$LOG to enhance audits

    ```
    $ DEFINE/SYSTEM/EXEC /FAL$LOG "1"
    ```

    This is an UNDOCUMENTED feature in VMS. It inserts extended information into the NETSERVER.LOG file. The possible bit-mask definitions are:

    - Bit-mask (best defined as a character string):

      - Bit 0 - enables logging of filename(s)

      - Bit 1 - enables generation of throughput statistics

      - Bit 2 - enables logging of DAP messages

      - Bit 3 - enables logging of XMIT and RECV AST completions

      - Bit 4 - enables logging of XMIT and RECV QIO requests

- Bit 5 - reserved
- Bit 6 - disables DAP message blocking
- Bit 7 - disables DAP CRC error checking
- Bit 8-31 - reserved

NOTE: By appending the string "/DISABLE=8" to the end of the definition you can disable "poor-man's routing" from being used on your node.

```
$ DEFINE/SYSTEM/EXEC FAL$LOG "1/DISABLE=8"
```

4.2.3 DECnet Objects

4.2.3.1 TASK Object 0 -

The TASK object (i.e., Object 0) is created during system startup by the STARTNET.COM command procedure. The TASK object may be used by unauthorized users to run a program under the DECnet default account on a given system remotely and interactively. Unauthorized users have access to the system's facilities without knowing a username and password and with no accountability. There are many methods to avoid a possible security problem associated with the use of the TASK Object. Two are presented below.

o Remove the TASK object

Remove the TASK object from the volatile DECnet data base by doing the following:

```
$ RUN SYS$SYSTEM:NCP <cr>
NCP> CLEAR OBJECT TASK ALL <cr>
NCP> EXIT <cr>
```

In addition, because the TASK object gets defined every time the system is rebooted, add the following commands to the system-specific startup command procedure (also known as SYSTARTUP.COM or SYSTARTUP_V5.COM under VMS V5.x) after the execution of the STARTNET command procedure:

```
$ @SYS$MANAGER:STARTNET.COM
$ RUN SYS$SYSTEM:NCP <cr>
```

```
CLEAR OBJET TASK ALL <cr>
EXIT <cr>
```

o Specify an invalid username for the TASK object (control access to the TASK object)

One can set an invalid default account for the TASK object in the network object data base. The following are used to do this:

```
$ RUN SYS$SYSTEM:NCP <cr>
NCP> DEFINE OBJECT TASK NUMBER 0 USER invalid_user -
     PASSWORD invalid_password <cr>
NCP> SET OBJECT TASK NUMBER 0 USER invalid_user -
     PASSWORD invalid_password <cr>
NCP> EXIT <cr>
```

An invalid username and password are specified so that the default access on inbound connects to this object (e.g., unauthorized users) will be unsuccessful.

NOTE: By specifying a valid userid but an invalid password, a login failure record will be generated which will alert you to the unauthorized attempt to use the TASK object.

Authorized users can connect to the TASK object; however, an access control string is required:

```
$ @NODE"valid_username valid_password"::"0=procedure
```

NOTE: Both of these methods will search for the specified procedure in the home directory of the user specified in the access string or proxy.

- One can then allow proxy access to the task object

  Add the authorized users to the proxy data base by doing the following:

  ```
  $ RUN SYS$SYSTEM:AUTHORIZE <cr>
  UAF> ADD/PROXY NODE::authorized_username valid_username
  UAF> EXIT <cr>
  ```

  Authorized users connect to the task object via:

  ```
  $ @NODE::"0=procedure"
  ```

5 DETECTING/REPORTING SECURITY INCIDENTS

This section deals with how to determine if security violations have occurred on the system.

5.1 Detecting Security Incidents

Security incidents can be very hard to detect, especially for successful attempts at accessing the system. Systems managers, however, can gain insight by close daily monitoring of their systems. In particular, they should pay close attention to the following:

- User accounts
  - Someone logged in during a time that is not normal
  - Several login sessions, at once, to a particular account
- Unexplained change in the system load. To check to see who is using the system resources, do the following:

  ```
  $ MONITOR/PROCESSES/TOPCPU
  ```

- Unfamiliar files in:
  - DECnet default account (there are NO required files in this directory)
  - SYS$SYSTEM
  - SYS$MANAGER
- Access to ACL protected files
- Unexplained alarms in the OPERATOR.LOG file
- Unaccountable changes to the system authorization file (SYSUAF.DAT)
- Excessive login failures and intrusion invasions. To check this, do the following:

  ```
  $ SHO INTRU <cr>

  Intrusion       Type       Count  Expiration   Source
     TERMINAL     SUSPECT        5  15:32:23.58
     NETWORK      SUSPECT        1  14:49:00.47  host::username
  ```

```
NETWORK      INTRUDER      7   14:58:10.70  host::username
```

- Unfamiliar images running on the system. To check this, do the following:

  ```
  $ SHO SYSTEM <cr>
  ```

- Installation of any privileged image

- Unexplained Disk and other system errors

Users of the system can be a good source for detecting security problems. Here are some indications for which they should be on the alert:

- Login messages should be read, especially the following:
  - Last interactive login
  - Last noninteractive login
  - Login failure message (e.g., three failures since last successful login)
  - Disconnect job message for a process the user never started

- File changes, such as:
  - Missing or renamed files
  - Files not owned by the user's account. To check this, a user would do the following:

    ```
    $ DIR/SECURITY

    Directory  DISK:[user_directory]

    FILENAME.EXT          [user_directory]      (RWED,RWED,RE,)
    ANOTHERFILE.EXT       [smith]               (RWED,RWED,RE,)
    ```

    This means the file ANOTHERFILE.EXT is owned by Smith and not by the owner of the directory.

  - Unexplained changes in file protection or ownership of the user files

- Disk quota is exceeded and the user was not responsible

Educating the user community will help the system manager maintain an acceptable level of security on the system.

5.2 Reporting Security Incidents

All suspected security incidents should be reported to the SPAN security manager. When reporting a problem, please be sure to include the following information:

- All related accounting and PSI accounting records
- All related NETSERVER.LOGs
- All related OPERATOR.LOGs

Send the report via electronic mail to NCF::NETMGR. This account is monitored by several individuals in the SPAN Network Office at the Goddard Space Flight Center. The SPAN security manager is alerted to all security incidents reported via this account.

After an investigation is made into the incident, a formal incident report will be filed if the incident is determined to be a security problem. One will find that the majority of the incidents reported end up being legitimate users of the network who make mistakes. These are usually cleared up quickly and no incident report is written.

6 SPAN SECURITY TOOLKIT

6.1 Introduction

With the growth of networks, system security has recently taken on a new importance. The increased exposure of a networked system must be compensated by an increased attention to system integrity. The intention to monitor a system more closely, or to reduce exposure, is good but will prove very difficult without appropriate tools.

VMS supplies many useful security features, but all of these take time to monitor and most system managers do not have the time available. We have found this to be the case at the National Space Science Data Center (NSSDC), Goddard Space Flight Center. Rather than giving up the idea of a secure system, we have tried to invent some tools to reduce the work involved in tracking the system's vulnerabilities.

There are no claims that these solutions are perfect, or even ideal. Many can, and probably will, be improved in the future. Some were written at NSSDC by the system management staff for use on their own system, and some were acquired from DECUS tapes or from other system managers. All have evolved and been improved with the experience gained in using them, and they will continue to evolve as new ideas are suggested or new threats appear. Based on the many requests for these programs, it was decided that they should be collected into a "toolkit," documented, and modified to improve portability to other systems. In this way, we can answer the requests for help without being swamped in the process.

It is highly recommended that the "Guide to VAX/VMS System Security" that comes with VMS is read. There are a number of features of VMS that can contribute significantly to system security. Many of them are only mentioned in passing in the various .DOC files in toolkit. Reading the VMS documentation, especially the system security the manual, is the only way to learn about everything that is available and that should be avoided or monitored. The toolkit is intended to aid in maintaining a secure system. It cannot take the place of a good understanding of the system or eliminate all of the effort needed.

It is also very likely that other programs, or better variations of the ones we have in the toolkit now, have already been developed for your system. If this is the case, and it is allowable for these programs to be distributed as part of future toolkits, we would appreciate hearing about them. If you have similar programs that are not allowed to be distributed, we would still be interested in hearing about the features these programs have so that we can possibly include similar features in the toolkit.

We expect at least one more release of the toolkit. There may be more in the future, but this will depend on the nature of the new capabilities and how useful they turn out to be.

We hope that this collection will prove useful. If there are any suggestions for improving the toolkit as a whole or any program contained in it, or other useful programs that are not included here, please let us know. Contact us on SPAN at: NCF::SUPPORT or by the phone at (301) 286-9794.

6.2 Why Was It Created?

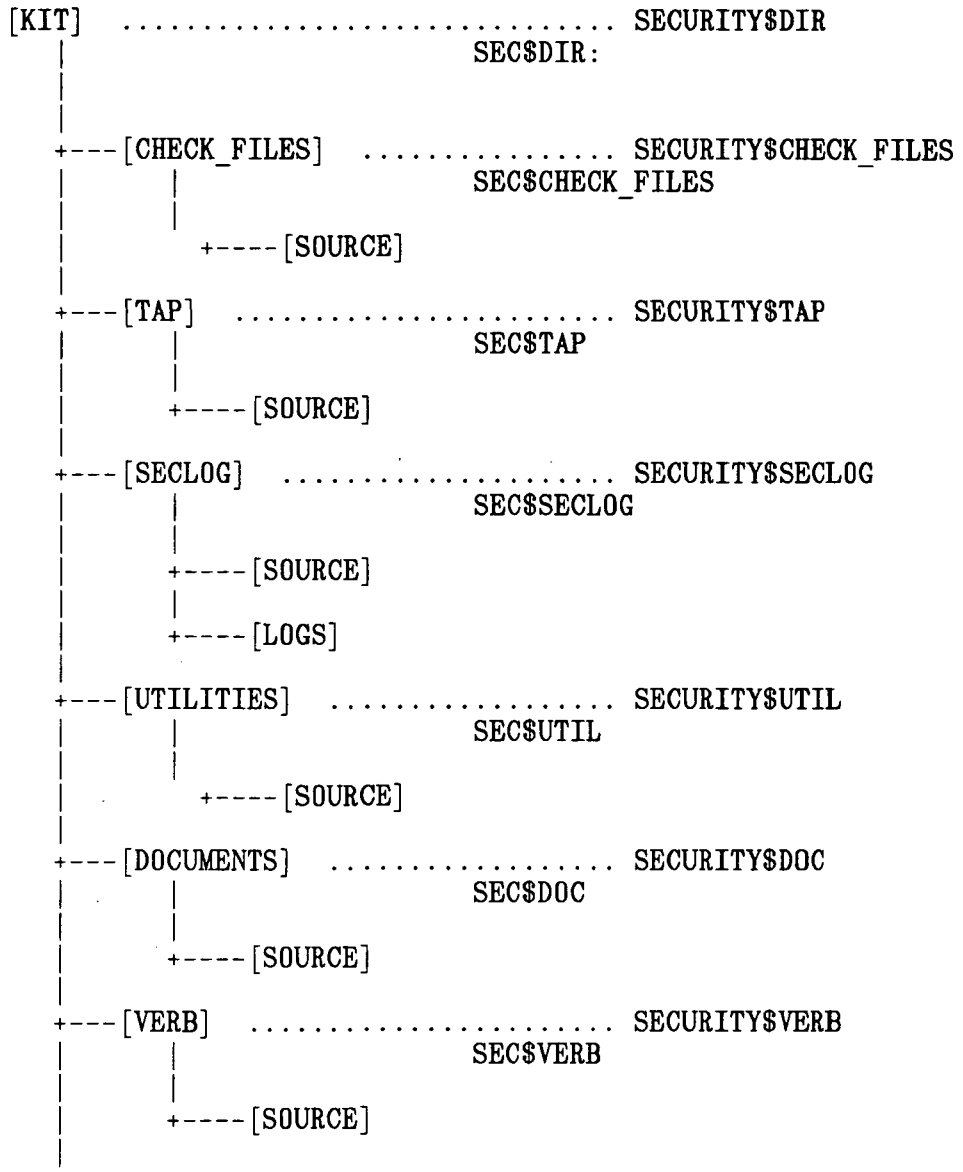
We at SPAN have always stated that the only way to have a secure network is to secure each host attached to the network. During the last Data Systems Users Working Group (DSUWG) meeting, September 1988 in Anaheim, California, several dozen SPAN node managers asked SPAN management to develop a set of system security tools that would help them in their day-to-day system management functions. A presentation on the types of tools being used at the NSSDC system facility helped the group decide on the basic set to use for the SPAN Security toolkit.

Recent unauthorized access incidents have indicated that VAX/VMS systems on SPAN are vulnerable to attacks. We at SPAN want to lessen the possibility of a system being penetrated. Also, the Computer Security Act of 1987 requires computer systems operated by the government or for the government to be protected against unlawful access.

6.3 Contents Of The Toolkit

The Security Toolkit consists of a number of programs, text files, command procedures, and data files. They may be used together, or selected systems may be implemented and the rest ignored, as seems appropriate for the situation.

The kit is organized into a single directory tree. The overall structure of this tree is shown in the following figure. Logical names for the directories (as defined in SECURITY_LOGICALS.COM) appear on the right. The logical names that start with "SECURITY$" are normal logical names used to refer to the given directory. The names beginning with "SEC$" are rooted logical names and are used when referring to subdirectories of the given directory.



The contents of these directories (in terms of programs/systems) are:

- Top level directories, logical name and symbol definition files, security distribution list for use with VMS Mail, an AAAREADME.1ST file that directs the user to this document

- The CHECK_FILES system runtime files: decommented .COM files, data files, documentation, and temporary files. CHECK_FILES is used to check for alterations to a list of specified files. If changes are found, a mail message is sent to notify the appropriate people about it.

- Commented versions of the .COM files (.SRC extensions), *RUNOFF* source for the *DOC* file, and a build procedure

o TAP runtime, documentation, startup, and dictionary files. TAP is used to check passwords of accounts for security. If TAP can guess the password, then any determined person can do the same. It also comes with a sample message to send to the users whose passwords are guessed. The message contains some rules for picking secure passwords.

- C source code for TAP, RUNOFF input for the .DOC and USER.MESSAGE files, and a build procedure

o Runtime procedures and documentation for the security alarm extractor. This makes checking the security alarms that are written to the OPERATOR.LOG file almost painless. It also allows the work to be distributed to several people.

- Commented source code for the command procedures, input to RUNOFF for the .DOC file, and a build procedure.

- SECURITY.LOG files from runs. These contain the extracted alarm messages. This directory is empty until SECLOG_SCAN is run.

o Miscellaneous utility programs and the documentation for them. Many are used by other parts of the kit, but some are intended for use on their own. See the .DOC files in the SECURITY$UTIL directory for more details. The programs are:

- CRC - calculates a CHECKSUM for a file. Used by CHECK_FILES.

- DECOMM - removes comments from DCL command procedures to speed execution. It also allows inclusion of debug lines. This is used by most of the build procedures in the kit.

- CHECK_LOGICALS - verifies that all security kit logical names have been defined and are from the correct version of the kit. This is used by most command procedures in the kit.

- PRIV_CHECK - allows easy identification of accounts with high login failure counts, various combinations of privileges, or accounts that have been DISUSERed. It is used by system managers and not by other parts of the kit.

- SAFETERM - is used to lock a terminal session to prevent access when the terminal must be left unattended. It allows anyone to log the process off to free the terminal for use by another user, but it requires a password to gain access to the original process. Help with commands for the program is built in. This is not used by any part of the kit; it is intended for system managers and users. It may be run from the SECURITY$UTIL directory or moved to wherever public utilities are kept. Note that if users are allowed to run it from the SECURITY$UTIL directory, set the protection to allow access. As shipped, all files in the directory are protected against WORLD and GROUP access.

- Commented source for all utilities. Build procedures and associated files where appropriate. Utilities involving more than one file are each in a separate subdirectory.

o Documentation:

  - Captive account dos and don'ts, documents for the toolkit as a whole

  - RUNOFF source for the .DOC files and a build procedure to run them off and to move and rename the RUNOFF output.

o VERB - program for listing DCLTABLES.EXE definitions; VERB.CLD to define the VERB command, VERB.HLP to build a help library entry for VERB, and VERB.DOC to describe the program and its use. It also includes VERB_CHECK.COM, which allows an automated comparison of DCLTABLES.EXE against previously compiled records of its contents.

o Source code files for VERB, commented version of VERB_CHECK.COM, build procedure to regenerate VERB

There are several things that should be known about the files in this toolkit. We have tried to make things like logical name usage, global symbol usage, temporary file creation, documentation, etc., follow a consistent plan. Most of this will only matter if system managers plan to change any of the programs or alter the overall structure of the kit directories.

This consistent system of naming conventions and common file usage is only partly implemented. The level of consistency is high but not yet perfect. All of the programs in the toolkit were developed independently. Changes have been made in many places to remove differences in style or function, but this process is not complete,

and there are places where consistency is possible but not yet implemented. We will attempt to remedy this for the next release of the toolkit. It makes maintenance and use of the programs much easier.

The first, and most obvious, standard practice is that all command procedures, and some programs, will print out the version number of the toolkit as soon as they are started. This should help prevent confusion when future releases of this system are made (we anticipate at least one more) and should allow easy identification of programs belonging to this set.

Almost all of the programs and procedures included in the kit make reference to directories through logical names. The names needed are defined in the file SECURITY_LOGICALS.COM in the top level toolkit directory. This file must be executed such that the logical names are defined for any process making use of toolkit programs. One may call this procedure from a personal LOGIN.COM file, call it manually after logging in, put the call in the system LOGIN.COM (SYLOGIN.COM), or call it with "/SYSTEM" as a parameter from the system startup procedure to define the names as system logicals. It does not matter how the logicals are defined as long as any process using the toolkit has them in one of its logical name tables or in the system table. If they are not defined, then when used most of the toolkit programs will return an error message indicating that the logicals are not completely defined or are the wrong version (this is for future release of the toolkit).

Another practice followed throughout the toolkit is the removal of comments from most .COM files. Comments slow the execution of command procedures and are only needed changes are being made to the programs. There is a command procedure in the utilities directory, called DECOMM.COM, that will strip the comments from procedure files. In the process it renames the original .COM file with a .SRC extension and produces a new .COM file, which is identical except for the missing comments. The .SRC files are kept in the [.SOURCE] subdirectories for use when changes are needed or for describing how the procedure works. Some procedures retain their comments and do not have a .SRC equivalent. These are mostly limited to program build procedures, very short procedures, and the logical and symbol definition procedures in the top level directory. DECOMM also aids in debugging command procedures by allowing conditional inclusion of "debug lines." See the DECOMM.SRC file for details.

All of the command procedures in the kit that require modification before use, such as SECURITY$SECLOG:SECLOG_DEFINITIONS.COM, have all of the lines that need changing as close to the beginning of the file as possible (consistent with readability and other standard practices). The section with the site-specific changes will also be marked as such, with instructions on what to change and what sort of values should be used.

Many of the procedures in the kit may be run as part of a daily batch job. To avoid the necessity of reading the batch log files produced by such a method of operation, many of the procedures are designed to send their final results to the appropriate people using VMS Mail. The procedures that work this way will look for a file called SECURITY$DIR:SECURITY.DIS. If the file exists, it will be used as a mail distribution file. If the file does not exist, the message will be sent to SYSTEM.

Exactly which of the tools contained in the kit should be implemented first? That depends on the situation. If one suspects that users are not choosing secure passwords, then TAP is a good first step. If the system is being swamped by security alarm messages every morning, then SECLOG might be best. Not all systems will need everything in the kit, and each system may have a different priority list.

The best first step is probably to read the .DOC files for each of the systems and utilities, or the .SRC files for things such as SAFETERM.COM, which do not have .DOC files, and learn what they do. Then decide on the best order of implementation for the system.

6.4 License Agreement

The SPAN Security toolkit is distributed on a one-by-one basis. All SPAN system managers must sign and agree to the license agreement. To request the toolkit, please do the following:

Send electronic mail to:

```
     NCF::NETMGR
or   NSSDCA::NETMGR
or   NSSDC::NETMGR
```

or call:

(301) 286-7251 or FTS 888-7251

or send postal mail to:

SPAN Network Information Center
SPAN Operations Center
NASA/Goddard Space Flight Center
Code 630.2
Greenbelt, Maryland 20771 USA

After the request is received, the following steps will be taken:

- The license agreement will be mailed to the individual requesting the software (see Attachment A)

- Information requested in the license agreement will be agreed upon by the recipient; and an electronic signature block and host information will be filled in, and the form will be mailed back to NCF::NETMGR

- Someone within the SPAN networking team at GSFC will tell the system manager the directory path to use to get the toolkit on the system.

- A phone call or mail message will be sent letting the remote node manager know that the software was copied to the system and requiring that the software be secured on the host.

The above steps will help the SPAN networking team keep the SPAN Network Information Center On-Line Data Base up to date with host-related information.

ATTACHMENT A

License Agreement for the SPAN Security Toolkit

This Agreement is made by and between the NASA/Goddard Space Flight Center, National Space Science Data Center (NSSDC), Greenbelt, Maryland 20771 (hereinafter "NSSDC"), and the SPAN Remote Node System Manager/Administrator (hereinafter "Recipient") of this Agreement.

Whereas, NSSDC has developed software or modified publicly available software, including programs and supporting documentation, designated as the SPAN Security Toolkit (hereinafter "Toolkit").

Whereas, the Toolkit was designed to enhance the awareness of computer security of the VAX/VMS System Manager/Administrator. (The Toolkit is not designed to be a single comprehensive answer to the problem of maintaining system security.)

Whereas, users of the Toolkit within the SPAN community have determined that this software is useful in enhancing their ability to maintain system security for VAX/VMS V4.x machines, and that further development should be fully supported, including adaptation to VMS V5.x.

Whereas, NSSDC is willing to allow the Recipient access to said Toolkit for the above-stated purposes subject, however, to certain constraints hereinafter stated;

Now, therefore, the parties hereto agree as follows:

1. The Recipient shall not use, disclose, copy, distribute or release the Toolkit for use on any VAX/VMS System other than the system designated in Enclosure 1; or allow it to be used, disclosed, copied or released to any VAX/VMS System other than the system designated in Enclosure 1. These prohibitions apply to members of the Recipients organization as well as others.

2. NSSDC may, but is not required to, provide software updates to the Recipient as they are developed.

3. The Recipient shall be responsible for installation, and reporting the following to NSSDC; within one month after receipt of the Toolkit:

   a. Installation problems;

   b. Errors found in the software with respect to programs and documentation;

   c. A summary explanation and additional detailed documentation covering all enhancements made by the Recipient to the Toolkit;

d. Suggestions for possible enhancements or improvements to the Tookit, supported by reasons why the Recipient believes such to be advantageous.

4. Recipient accepts said toolkit as is. NSSDC makes no representations or warranties regarding the use or performance of said software and expressly makes no warranty or merchantability of fitness for a particular purpose.

5. In no event will NSSDC be liable for any damages resulting from activities related to performance under this Agreement.

6. In no event will NSSDC be liable for any fees or costs for activities related to performance under this Agreement.

7. Recipient shall supply NSSDC with the information requested in Enclosure 1 before distribution of the Toolkit.

8. This agreement is the entire agreement of the parties hereto relating to the Toolkit software. Any changes must be made in writing and agreed to by both NSSDC and the Recipient.

9. Provisions of this Agreement relating to use, disclosure, copying, distribution and release shall be perpetual.

NASA/Goddard Space Flight Center
SPAN Network Information Center
Code 630.2
Greenbelt, Maryland 20771
ATTN: Ron Tencati
SPAN Security Manager
e-mail: NCF::NETMGR (DECNET)

Recipient Electronic Signature:

NAME:
JOB TITLE:
INSTITUTION:
DATE:

SPAN Network Information Center (SPAN_NIC)
On-Line Data Base System

Please supply us with the following information about your SPAN host:

a) NODE NAME:
b) NODE NUMBER:
c) ROUTING TYPE:
   (ex: AREA, ROUTING IV, NONROUTING IV etc...)
d) ROUTING CENTER:
   (ex. ARC, ESOC, GSFC, JPL, JSC, KSC, MSFC)
e) HARDWARE:
   (ex. VAX 11/xxx, MICROVAX xx, IBM xxx, etc...)
f) SOFTWARE VERSION:
   (ex. VMS V4.5, UNIX, etc...)
g) BUILDING/ROOM (MACHINE LOCATION):
h) DEFINED SPAN_WIDE (Y or N):
   (Do you want other NODEs to be able to access this info)
i) CONTACT PERSONS NAME:
   (Systems Manager name)
j) CONTACT PERSONS POSITION:
   (Systems Manager title)
k) CONTACT SPAN ADDRESS:
   (ex. for me it would be NCF::NETMGR)
l) CONTACT PERSONS PHONE NUMBER:
m) SECURITY PERSONS NAME:
   (should be person responsible for security of your system)
n) SECURITY PERSONS TITLE:
o) SECURITY PERSONS SPAN ADDRESS:
   (ex. for me it would be NCF::NETMGR)
p) SECURTIY PERSONS PHONE NUMBER:
q) SCIENCE CONTACTS NAME:
r) SCIENCE CONTACTS TITLE:
s) SCIENCE CONTACTS SPAN ADDRESS:
   (ex. for me it would be NCF::NETMGR)
t) SCIENCE CONTACTS PHONE NUMBER:

   Institution Mailing Address
   ==========================
u) INSTITUTION:
v) DEPARTMENT:
w) STREET:
   (street address for Institution)
x) CITY:
y) STATE:
z) ZIP CODE:
aa) COUNTRY:

bb) DISCIPLINE:
Possible Disciples are as follows:

Astronomy

Gamma Ray Astronomy	X-Ray Astronomy
Ultraviolet Astronomy	Visible Astronomy
Infrared Astronomy	Microwave Astronomy
Radio Astronomy	Cosmic Ray Astronomy

Earth Science

Atmosphere	Land
Interior and Crust	Ocean

Planetary Science

Planetary Geophysics	Planetary Atmospheres
Small Bodies	Fields and Particles

Solar Physics

Radio Observations	Microwave Observations
Infrared Observations	Visible Observations
Ultraviolet Observations	X-Ray Observations
Gamma Observations	

Space Physics

Magnetospheric Science	Ionospheric Science
Interplanetary Studies	

cc) LINKS TO OTHER NETWORKS (NETWORK NAME/NETWORK TYPE):
(ex. ARPAnet, BITnet, NICEnet, SESnet, etc...)

FOR YOUR INFORMATION:
==========

The National Space Science Data Center (NSSDC) located at the Goddard Space Flight Center has an online network information center called SPAN_NIC. The services provided in the online data base system are as follows:

1. General/administrative information on SPAN
2. How to access other network information centers
3. Query SPAN NIC data base for node information
4. SPAN important news briefs
5. SPAN InterMAIL addresses -- proper syntax
6. SPAN documentation library
7. Submit node's information for inclusion in SPAN_NIC data base

If you want to log into the SPAN_NIC data base system, you can do the following:

```
$ SET HOST NSSDCA <cr>
USERNAME:  SPAN_NIC <cr>
```