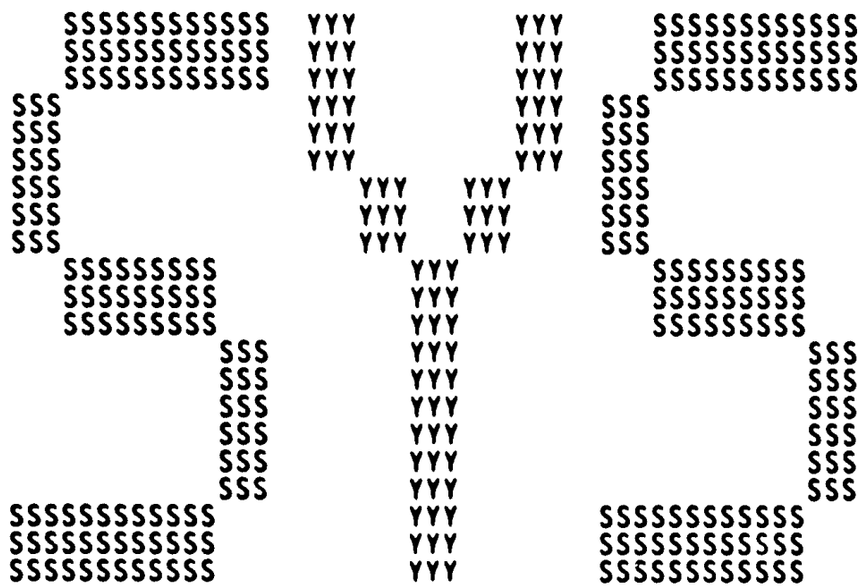

FILEID**SECAUDIT

```
  SSSSSSSS  EEEEEEEEEE    CCCCCCCC    AAAAAA    UU      UU  DDDDDDDD      IIIIII    TTTTTTTTTT
  SSSSSSSS  EEEEEEEEEE    CCCCCCCC    AAAAAA    UU      UU  DDDDDDDD      IIIIII    TTTTTTTTTT
SS          EE          CC          AA      AA  UU      UU  DD      DD      II          TT
SS          EE          CC          AA      AA  UU      UU  DD      DD      II          TT
SS          EE          CC          AA      AA  UU      UU  DD      DD      II          TT
SS          EE          CC          AA      AA  UU      UU  DD      DD      II          TT
  SSSSSS    EEEEEEEE    CC          AA      AA  UU      UU  DD      DD      II          TT
  SSSSSS    EEEEEEEE    CC          AA      AA  UU      UU  DD      DD      II          TT
        SS  EE          CC          AAAAAAAAAA  UU      UU  DD      DD      II          TT
        SS  EE          CC          AAAAAAAAAA  UU      UU  DD      DD      II          TT
        SS  EE          CC          AA      AA  UU      UU  DD      DD      II          TT          ....
        SS  EE          CC          AA      AA  UU      UU  DD      DD      II          TT          ....
SSSSSSSS    EEEEEEEEEE    CCCCCCCC  AA      AA  UUUUUUUUUU  DDDDDDDD      IIIIII      TT          ....
SSSSSSSS    EEEEEEEEEE    CCCCCCCC  AA      AA  UUUUUUUUUU  DDDDDDDD      IIIIII      TT          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

N 6

Table of contents

```
0000     1          .TITLE  SECAUDIT - SECURITY AUDITING ROUTINES
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;++
0000    29 ; Facility: Executive
0000    30 ;
0000    31 ; Abstract:
0000    32 ;       This module contains the centralized Security Auditing routines.
0000    33 ;
0000    34 ; Enviornment:
0000    35 ;       VMS Paged Exec - Kernel mode
0000    36 ;--
0000    37 ;
0000    38 ; Author:
0000    39 ;
0000    40 ;       R. Scott Hanna, CREATION DATE: 04-Mar-1983
0000    41 ;
0000    42 ; Modified by:
0000    43 ;
0000    44 ;       V03-009 ACG0425         Andrew C. Goldstein,    27-Apr-1984  11:06
0000    45 ;               Move code to a paged psect (and fix up the one leg
0000    46 ;               at elevated IPL).
0000    47 ;
0000    48 ;       V03-008 MHB0119         Mark Bramhall           26-Mar-1984
0000    49 ;               Support mandatory auditing flag NSA$x_ARG_FLAG_MANDY.
0000    50 ;               Have mandatory auditing imply alarm for now.
0000    51 ;
0000    52 ;       V03-007 RSH0108         R. Scott Hanna          28-Feb-1984
0000    53 ;               Change time field in security auditing record header
0000    54 ;               from a longword to a quadword.
0000    55 ;
0000    56 ;       V03-006 RSH0094         R. Scott Hanna          01-Feb-1984
0000    57 ;               Changes for field test 2.
```

```
0000     58 ;
0000     59 ;       V03-005 RSH0072         R. Scott Hanna          06-Oct-1983
0000     60 ;               Change journal name and access mode. Eliminate PROT and
0000     61 ;               FLAGS keywords from $ASSJNL_S.
0000     62 ;
0000     63 ;       V03-004 RSH0066         R. Scott Hanna          15-Sep-1983
0000     64 ;               Fix problem in ALARM_FORMAT_FILACC.
0000     65 ;
0000     66 ;       V03-003 RSH0061         R. Scott Hanna          06-Sep-1983
0000     67 ;               Modify PRV$KEYWORD routine to not return keyword length.
0000     68 ;               Remove trailing CRLF from alarm messages. Change
0000     69 ;               BUILD_RECORD_DTPKT to reference field size as a word.
0000     70 ;               Change ALARM_RECORD to set SECURITY privilege in OPCOM
0000     71 ;               message header. Fix problem in ALARM_FORMAT_PRVMSK related
0000     72 ;               to bogus privilege bits.
0000     73 ;
0000     74 ;       V03-002 RSH0045         R. Scott Hanna          24-Jul-1983
0000     75 ;               Replace temporary OPC$M_NM_OPER1 and OPC$_RQ_RQST with
0000     76 ;               OPC$M_NM_SECURITY and OPC$_RQ_SECURITY
0000     77 ;
0000     78 ;       V03-001 RSH0038         R. Scott Hanna          17-Jun-1983
0000     79 ;               Make changes due to the addition of NSA$L_ARG_COUNT in
0000     80 ;               $NSAARGDEF.
0000     81 ;--
```

```
                     0000     83          .SBTTL  Declarations and Assumptions
                     0000     84 ;
                     0000     85 ;        Define Symbols
                     0000     86 ;
                     0000     87
                     0000     88          $DSCDEF                                 ; Argument descriptor defs
                     0000     89          $IPLDEF                                 ; Priority defs
                     0000     90          $JIBDEF                                 ; Job Information Block defs
                     0000     91          $MSGDEF                                 ; Operator message defs
                     0000     92          $NSAARGDEF                              ; Auditing argument list defs
                     0000     93          $NSAIDTDEF                              ; Auditing impure data table defs
                     0000     94          $NSARECDEF                              ; Auditing record defs
                     0000     95          $OPCDEF                                 ; Operator class defs
                     0000     96          $PCBDEF                                 ; Process control block defs
                     0000     97          $PRDEF                                  ; Processor register defs
                     0000     98          $PRVDEF                                 ; Privilege bits defs
                     0000     99          $PSLDEF                                 ; Processor status longword defs
                     0000    100          $RSNDEF                                 ; Resource wait defs
                     0000    101          $SBDEF                                  ; SCS System Block de's
                     0000    102          $SSDEF                                  ; System service defs
                     0000    103
                     0000    104 ; Minimum size in record to allow insertion of another packet
                     0000    105
 00000010            0000    106          REC_MIN_SIZE_LEFT = NSA$K_PKTHDR_LENGTH*4
                     0000    107
                     0000    108 ; Make sure that the largest record will fit in the buffer
                     0000    109
                     0000    110          ASSUME  NSA$K_REC_MAXLENGTH LE NSA$S_IDT_RECORD_BUF
                     0000    111
                     0000    112 ; All code is pageable.
                     0000    113
                 00000000    114          .PSECT  Y$EXEPAGED
```

```
                     0000   116 .SBTTL  NSA$EVENT_AUDIT - Event Auditing Routine
                     0000   117 ;++
                     0000   118 ; NSA$EVENT_AUDIT - Event Auditing Routine
                     0000   119 ;
                     0000   120 ; FUNCTIONAL DESCRIPTION:
                     0000   121 ;
                     0000   122 ;       This routine is called when an auditable system event occurs. It
                     0000   123 ;       takes the data in the argument list and builds the journal record.
                     0000   124 ;       The record is then written to the security audit journal and/or
                     0000   125 ;       sent to OPCOM to print a security alarm message.
                     0000   126 ;
                     0000   127 ;       This routine must be called in KERNEL mode at IPL 0 in the context
                     0000   128 ;       of the process which caused the event being audited.
                     0000   129 ;
                     0000   130 ; CALLING SEQUENCE:
                     0000   131 ;
                     0000   132 ;       CALLS or CALLG to NSA$EVENT_AUDIT
                     0000   133 ;
                     0000   134 ; INPUTS:
                     0000   135 ;
                     0000   136 ;       AP = Pointer to the event argument list. Event argument lists are
                     0000   137 ;            defined by the $NSAARGDEF macro.
                     0000   138 ;
                     0000   139 ; OUTPUT:
                     0000   140 ;
                     0000   141 ;       R0 = status of operation
                     0000   142 ;       R1 = Destroyed
                     0000   143 ;--
                     0000   144
                OFFC 0000   145         .ENTRY  NSA$EVENT_AUDIT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
     56  00000000'9F   DE   0002   146         MOVAL   @#NSA$T_IDT,R6                  ; Get impure data block pointer
     57  00000000'GF   D0   0009   147         MOVL    G^SCH$GL_CURPCB,R7              ; Get PCB
           7E    01   D0   0010   148         MOVL    #1@PSL$C_KERNEL,-(SP)           ; Assume kernel ast's enabled
        02 0D A7   00   E4   0013   149         BBSC    #PSL$C_KERNEL,PCB$B_ASTEN(R7),1$ ; Br if kernel ast's enabled
                   6E   D4   0018   150         CLRL    (SP)                            ; Kernel ast's were not enabled
                   32   10   001A   151 1$:     BSBB    BUILD_RECORD_DT                 ; Build the record descriptor table
              58   01   D0   001C   152         MOVL    #1,R8                           ; Init record number
         59   0437 C6   DE   001F   153         MOVAL   NSA$T_IDT_RECORD_DT+1(R6),R9    ; Get record descr table pointer
                00B1   30   0024   154 2$:     BSBW    BUILD_RECORD                    ; Build the record
                   01   E1   0027   155         BBC     #NSA$V_ARG_FLAG_JOURN,-         ; Br if this record is
            06 08 AC        0029   156                 NSA$B_ARG_FLAG(AP),3$           ; ...not to be journaled
                 011B   30   002C   157         BSBW    JOURNAL_RECORD                  ; Journal the record
              17 50    E9   002F   158         BLBC    R0,5$                           ; Br if error
                   93   0032   159 3$:     BITB    #<NSA$M_ARG_FLAG_ALARM!-        ; Alarm and/or
                        0033   160                   NSA$M_ARG_FLAG_MANDY>,-         ;  mandatory audit
           08 AC   05        0033   161                 NSA$B_ARG_FLAG(AP)              ;   for this record?
                   06   13   0036   162         BEQL    4$                              ; Br if neither
                 0113   30   0038   163         BSBW    ALARM_RECORD                    ; Alarm the record
              0B 50    E9   003B   164         BLBC    R0,5$                           ; Br if error
FFDE 58   01   0436 C6   9D   003E   165 4$:     ACCB    NSA$T_IDT_RECORD_DT(R6),#1,R8,2$ ; Br if more records
              50   01   D0   0046   166         MOVL    #SS$_NORMAL,R0                  ; Return success
           0D A7   6E   88   0049   167 5$:     BISB    (SP),PCB$B_ASTEN(R7)            ; Restore kernel ast enable bit
                   04   004D   168         RET
```

```
                        004E   170 .SBTTL  Record building routines
                        004E   171 ;++
                        004E   172 ; BUILD_RECORD_DT - Build the record descriptor table
                        004E   173 ;
                        004E   174 ; FUNCTIONAL DESCRIPTION:
                        004E   175 ;
                        004E   176 ;       This routine builds the record descriptor table.
                        004E   177 ;
                        004E   178 ; INPUTS:
                        004E   179 ;
                        004E   180 ;       R6 = Impure data block pointer
                        004E   181 ;       AP = Pointer to the event argument list.
                        004E   182 ;
                        004E   183 ; OUTPUT:
                        004E   184 ;
                        004E   185 ;       NSA$T_IDT_RECORD_DT(R6) = The record descriptor table
                        004E   186 ;
                        004E   187 ;       R0-R4,R9-R11  = Destroyed
                        004E   188 ;--
                        004E   189
                        004E   190 BUILD_RECORD_DT:
51    0437 C6    DE     004E   191         MOVAL   NSA$T_IDT_RECORD_DT+1(R6),R1     ; Init packet pointer
  FF  A1    01    D0     0053   192         MOVL    #1,-1(R1)                       ; Init rec count, flgs, & pkt count
  54    09  AC    9A     0057   193         MOVZBL  NSA$B_ARG_PKTNUM(AP),R4          ; Get total number of packets
            4E    13     005B   194         BEQLU   7$                               ; Br if none (header only)
  53    0C  AC    DE     005D   195         MOVAL   NSA$T_ARG_LIST(AP),R3            ; Init arglist pointer
            49    10     0061   196         BSBB    GET_NEXT_PKT                    ; Get first pkt type, size, addr
            37    11     0063   197         BRB     5$                              ; Init first record
            45    10     0065   198 1$:     BSBB    GET_NEXT_PKT                    ; Get next pkt type, size, addr
      50    04    C2     0067   199 2$:     SUBL    #NSA$K_PRTHDR_LENGTH,R0         ; Subtract out packet header size
      50    5A    D1     006A   200         CMPL    R10,R0                          ; Will packet fit in current rec?
            13    15     006D   201         BLEQ    3$                              ; Br if yes
      10    50    D1     006F   202         CMPL    R0,#REC_MIN_SIZE_LEFT           ; Put part of pkt in this rec?
            24    19     0072   203         BLSS    4$                              ; Br if no
7E    5B    50    C1     0074   204         ADDL3   R0,R11,-(SP)                    ; Save remaining address
7E    5A    50    C3     0078   205         SUBL3   R0,R10,-(SP)                    ; Save remaining size
      5A    50    D0     007C   206         MOVL    R0,R10                          ; Size = size left in record
      62    01    90     007F   207         MOVB    #NSA$M_REC_FLAGS_PKTCON,(R2)    ; Indicate packet continuation
      81    59    B0     0082   208 3$:     MOVW    R9,(R1)+                        ; Insert pkt type in descr table
      81    5A    B0     0085   209         MOVW    R10,(R1)+                       ; Insert size in rec descr table
      81    5B    D0     0088   210         MOVL    R11,(R1)+                       ; Insert addr in rec descr table
        01  A2    96     008B   211         INCB    1(R2)                           ; Increment packet count
      50    5A    C2     008E   212         SUBL    R10,R0                          ; Subtract out packet size
            62    95     0091   213         TSTB    (R2)                            ; Is curr pkt cont in next rec?
            13    13     0093   214         BEQLU   6$                              ; Br if not
      5A    8E    7D     0095   215         MOVQ    (SP)+,R10                       ; Get remaining size and address
      0436  C6    96     0098   216 4$:     INCB    NSA$T_IDT_RECORD_DT(R6)         ; Increment record count
50    03B4  8F    3C     009C   217 5$:     MOVZWL  #NSA$S_IDT_RECORD_BUF- -        ; Init size remaining
                  00A1   218                 NSA$K_RECHDR_LENGTH,R0
      52    51    D0     00A1   219         MOVL    R1,R2                           ; Get new flags byte pointer
            81    B4     00A4   220         CLRW    (R1)+                           ; Init flags byte and pkt count
            BF    11     00A6   221         BRB     2$
        BA  54    F5     00A8   222 6$:     SOBGTR  R4,1$                           ; Br if more packets
                  05     00AB   223 7$:     RSB
```

```
                        00AC   225 ;++
                        00AC   226 ; GET_NEXT_PKT - Get next data packet information
                        00AC   227 ;
                        00AC   228 ; FUNCTIONAL DESCRIPTION:
                        00AC   229 ;
                        00AC   230 ;       This routine returns the type, size, and data address of the next
                        00AC   231 ;       packet to be placed in the record.
                        00AC   232 ;
                        00AC   233 ; INPUTS:
                        00AC   234 ;
                        00AC   235 ;       R3 = arglist pointer
                        00AC   236 ;
                        00AC   237 ; OUTPUT:
                        00AC   238 ;
                        00AC   239 ;       R3 = updated arglist poirter
                        00AC   240 ;       R9  = packet type
                        00AC   241 ;       R10  = packet size
                        00AC   242 ;       R11  = Address of packet data
                        00AC   243 ;--
                        00AC   244
                        00AC   245 GET_NEXT_PKT:
      59    83    3C    00AC   246         MOVZWL  (R3)+,R9                        ; Get packet type
      5A    83    3C    00AF   247         MOVZWL  (R3)+,R10                       ; Get argument passing mechanism
      04    5A    D1    00B2   248         CMPL    R10,#NSA$K_ARG_MECH_DESCR       ; Determine arg passing mechanism
            15    13    00B5   249         BEQLU   1$                              ; Descriptor
            18    1A    00B7   250         BGTRU   2$                              ; Address of descriptor
5A    01    5A    78    00B9   251         ASHL    R10,#1,R10                      ; Value R10 = size
      5B    63    DE    00BD   252         MOVAL   (R3),R11                        ;       R11 = address
7E    5A    03    C1    00C0   253         ADDL3   #3,R10,-(SP)                    ; Round size up to next longword
      6E    03    CA    00C4   254         BICL2   #3,(SP)
      53    8E    C0    00C7   255         ADDL2   (SP)+,R3                        ; Point to next argument
            0B    11    00CA   256         BRB     4$
      5A    83    7D    00CC   257 1$:     MOVQ    (R3)+,R10                       ; Get size and addr from descr
            03    11    00CF   258         BRB     3$
      5A    93    7D    00D1   259 2$:     MOVQ    @(R3)+,R10                      ; Get size and addr from descr
      5A    5A    3C    00D4   260 3$:     MOVZWL  R10,R10                         ; Convert first longword to size
                  05    00D7   261 4$:     RSB
```

```
                                00D8   263 ;++
                                00D8   264 ; BUILD_RECORD - Build a record
                                00D8   265 ;
                                00D8   266 ; FUNCTIONAL DESCRIPTION:
                                00D8   267 ;
                                00D8   268 ;       This routine builds a record
                                00D8   269 ;
                                00D8   270 ; INPUTS:
                                00D8   271 ;
                                00D8   272 ;       R6  = Impure data block pointer
                                00D8   273 ;       R7  = PCB pointer
                                00D8   274 ;       R8 = Record sequence number
                                00D8   275 ;       R9  = Record descriptor table pointer
                                00D8   276 ;
                                00D8   277 ; OUTPUT:
                                00D8   278 ;
                                00D8   279 ;       R9 = Updated record descriptor table pointer
                                00D8   280 ;
                                00D8   281 ;       NSA$T_IDT_RECORD_BUF(R6) = Record
                                00D8   282 ;
                                00D8   283 ;       R0-R5,R10-R11 = Destroyed
                                00D8   284 ;--
                                00D8   285
                                00D8   286 BUILD_RECORD:
      5A    042E C6    7E      00D8   287         MOVAQ   NSA$Q_IDT_RECORD_DESCR(R6),R10  ; Get record descr pointer
        53    2E A6    DE      00DD   288         MOVAL   NSA$T_IDT_RECORD_BUF(R6),R3     ; Get record address
   6A   010E004C 8F    D0      00E1   289         MOVL    #NSA$K_RECHDR_LENGTH! -         ; Init the record descriptor
                                00E8   290                 <DSC$K_DTYPE_T@16>! -
                                00E8   291                 <DSC$K_CLASS_S@24>,(R10)
        04 AA    53    D0      00E8   292         MOVL    R3,4(R10)
        83   04  AC    D0      00EC   293         MOVL    NSA$L_ARG_ID(AP),(R3)+          ; Insert record type
           83    58    90      00F0   294         MOVB    R8,(R3)+                        ; Insert record sequence number
           83  0436 C6    90   00F3   295         MOVB    NSA$T_IDT_RECORD_DT(R6),(R3)+   ; Insert last sequence number
           83    89    B0      00F8   296         MOVW    (R9)+,(R3)+                     ; Insert flags byte & pkt count
           83    6A    04      A1   00FB   297    ADDW3   #NSA$K_PKTHDR_LENGTH,(R10),(R3)+ ; insert offset to first packet
           83    04    B0      00FF   298         MOVW    #NSA$K_PKTHDR_LENGTH,(R3)+      ; Insert packet header size
        83    64 A7    D0      0102   299         MOVL    PCB$L_EPID(R7),(R3)+            ; Insert EPID
     83   00000000'GF    7D    0106   300         MOVQ    G^EXE$GQ_SYSTIME,(R3)+          ; Insert event time
     51   00000044'GF    9E    010D   301         MOVAB   G^SCS$GA_LOCALSB+SB$T_NODENAME,R1 ; Get node name ASCIC pointer
           50    81    9A      0114   302         MOVZBL  (R1)+,R0                        ; Get size
63   10   20    61    50      2C   0117   303    MOVC5   R0,(R1),#^X20,#SB$S_NODENAME,(R3) ; Insert node name
        51    70 A7    9E      011D   304         MOVAB   PCB$T_LNAME(R7),R1              ; Get process name ASCIC pointer
           50    81    9A      0121   305         MOVZBL  (R1)+,R0                        ; Get size
63   10   20    61    50      2C   0124   306    MOVC5   R0,(R1),#^X20,#PCB$S_LNAME,(R3) ; Insert process name
           14    28    012A   307         MOVC3   #JIB$S_USERNAME+JIB$S_ACCOUNT,- ; Insert Username and Account name
     63   00000000'9F        012C   308                 @#CTL$T_USERNAME,(R3)
        5B    FF A9    9A      0132   309         MOVZBL  -1(R9),R11                      ; Get number of packets
              11    13      0136   310         BEQLU   2$                              ; Br if none
           83    89    B0      0138   311 1$:     MOVW    (R9)+,(R3)+                     ; Insert packet type
        63   69    04      A1   013B   312    ADDW3   #NSA$K_PKTHDR_LENGTH,(R9),(R3)  ; Insert packet size
           6A    83    A0      013F   313         ADDW2   (R3)+,R10                       ; Add packet size to record size
        63   99    89    28      0142   314    MOVC3   (R9)+,@(R9)+,(R3)               ; Insert field in packet
              EF 5B    F5      0146   315         SOBGTR  R11,1$                          ; Br if more packets
                       05      0149   316 2$:     RSB
```

```
                     014A   318 .SBTTL  JOURNAL_RECORD - Journal the audit record
                     014A   319 ;++
                     014A   320 ; JOURNAL_RECORD - Journal the audit record
                     014A   321 ;
                     014A   322 ; FUNCTIONAL DESCRIPTION:
                     014A   323 ;
                     014A   324 ;       This routine assigns a channel to the security audit journal, (if it
                     014A   325 ;       has not been assigned already), and writes the event record.
                     014A   326 ;
                     014A   327 ; INPUTS:
                     014A   328 ;
                     014A   329 ;       R6 = Impure data block pointer
                     014A   330 ;
                     014A   331 ; OUTPUT:
                     014A   332 ;
                     014A   333 ;       R0 = Status of operation
                     014A   334 ;       R1 = Destroyed
                     014A   335 ;
                     014A   336 ; NOTE: This routine has been removed. It will be replaced when Journaling is
                     014A   337 ;       supported.
                     014A   338 ;--
                     014A   339
                     014A   340 JOURNAL_RECORD:
50   01   D0         014A   341         MOVL    #SS$_NORMAL,R0
                05   014D   342         RSB
```

```
                                                014E   344 .SBTTL  ALARM_RECORD - Generate Security Alarm
                                                014E   345 ;++
                                                014E   346 ; ALARM_RECORD - Generate Security Alarm
                                                014E   347 ;
                                                014E   348 ; FUNCTIONAL DESCRIPTION:
                                                014E   349 ;
                                                014E   350 ;       This routine sends the security audit record to the OPCOM process
                                                014E   351 ;
                                                014E   352 ; INPUTS:
                                                014E   353 ;
                                                014E   354 ;       R6 = Impure data block pointer
                                                014E   355 ;       R7 = PCB pointer
                                                014E   356 ;
                                                014E   357 ; OUTPUT:
                                                014E   358 ;
                                                014E   359 ;       R0 = status of operation
                                                014E   360 ;       R1-R5 = Destroyed
                                                014E   361 ;--
                                                014E   362
                                                014E   363 ALARM_RECORD:
                53   66   DE                    014E   364         MOVAL   NSA$T_IDT_ALARM_HDR(R6),R3      ; Get alarm header address
                83   08   B0                    0151   365         MOVW    #MSG$_OPRQST,(R3)+              ; Insert message type
                     83   B4                    0154   366         CLRW    (R3)+                           ; No reply mailbox
                                                0156   367         ASSUME  PRV$V_SECURITY GE 32            ; Assume SECURITY in second longword
                     83   D4                    0156   368         CLRL    (R3)+                           ; Zero first longword of mask
              83     40   8F   9A               0158   369         MOVZBL  #<1@<PRV$V_SECURITY-32>>,(R3)+  ; Insert security mask
           83     00BC   C7   D0               015C   370         MOVL    PCB$L_UIC(R7),(R3)+             ; Insert UIC
                     14   28                    0161   371         MOVC3   #JIB$S_USERNAME+JIB$S_ACCOUNT,- ; Insert Username and Account name
        63    00000000'9F                       0163   372                 @#CTL$T_USERNAME,(R3)
                     83   B4                    0169   373         CLRW    (R3)+                           ; Zero base priority and spare bytes
        83    00010007   8F   D0               016B   374         MOVL    #<OPC$M_NM_SECURITY@8+OPC$_RQ_SECURITY>,(R3)+ ; Insert opcom header
                     83   D4                    0172   375         CLRL    (R3)+
           53   042E   C6   3C                  0174   376         MOVZWL  NSA$Q_IDT_RECORD_DESCR(R6),R3   ; Get record size
                53     2E   C0                  0179   377         ADDL2   #NSA$S_IDT_ALARM_HDR,R3         ; Add header size
        55    00000000'GF   DE                  017C   378         MOVAL   G^SYS$GL_OPRMBX,R5              ; Get UCB addr of operator mailbox
                54   66   DE                    0183   379 1$:     MOVAL   NSA$T_IDT_ALARM_HDR(R6),R4      ; Get address of buffer
                                                0186   380         SETIPL  #IPL$_ASTDEL                    ; Block ast delivery
                                                0189   381         ; The buffer must be page aligned and LE 3 pages
                                                0189   382         ASSUME  NSA$T_IDT_ALARM_HDR EQ 0
                                                0189   383         ASSUME  NSA$S_IDT_ALARM_HDR+NSA$S_IDT_RECORD_BUF LE 512*3
      0400 C4   01FE   C4   D1                  0189   384         CMPL    510(R4),1024(R4)                ; Fault in 3 pages
           00000000'GF   16                     0190   385         JSB     G^EXE$WRTMAILBOX                ; Write message
                                                0196   386         SETIPL  #0                              ; Enable ast delivery
                    30   50   E8               0199   387         BLBS    R0,4$                           ; Br if success
        2B 0C A7    00   E0                    019C   388         BBS     #PSL$C_KERNEL,PCB$B_ASTACT(R7),4$ ; Br if we are in a kernel ast
       000008D8 8F   50   D1                    01A1   389         CMPL    R0,#SS$_MBFULL                  ; Mailbox full?
                     05   12                    01A8   390         BNEQU   2$                              ; br if not
                50   02   3C                    01AA   391         MOVZWL  #RSN$_MAILBOX,R0                ; Set resource to mailbox full
                     0C   11                    01AD   392         BRB     3$
       00000124 8F   50   D1                    01AF   393 2$:     CMPL    R0,#SS$_INSFMEM                 ; Not enough pool?
                     14   12                    01B6   394         BNEQU   4$                              ; Br if some other error
                50   03   3C                    01B8   395         MOVZWL  #RSN$_NPDYNMEM,R0               ; Set resource to non-paged pool
                54   57   D0                    01BB   396 3$:     MOVL    R7,R4                           ; R4 = PCB
                     7E   DC                    01BE   397         MOVPSL  -(SP)                           ; Push PSL on stack
                                                01C0   398         SETIPL  #8                              ; Raise IPL to SYNCH
           00000000'GF   16                     01C4   399         JSB     G^SCH$RWAIT                     ; Wait for resource
                     B7   11                    01CA   400         BRB     1$                              ; Try again
```

```
                05  01CC    401 4$:     RSB
                    01CD    402
00000008            01CD    403 5$:     .LONG   IPL$_SYNCH                      ; Lock between the SETIPL and here
```

```
                     01D1    405 .SBTTL Arglist Packet Entry Insertion Routines
                     01D1    406 ;++
                     01D1    407 ; NSA$ARGLST_IMGNAM - Insert The Argument List Entry For The IMGNAM Packet
                     01D1    408 ;
                     01D1    409 ; FUNCTIONAL DESCRIPTION:
                     01D1    410 ;
                     01D1    411 ;       This routine inserts the IMGNAM packet entry in the callers arglist.
                     01D1    412 ;
                     01D1    413 ; CALLING SEQUENCE:
                     01D1    414 ;
                     01D1    415 ;       JSB     NSA$ARGLST_IMGNAM
                     01D1    416 ;
                     01D1    417 ; INPUTS:
                     01D1    418 ;
                     01D1    419 ;       R2 = Pointer to the IMGNAM packet entry in arglist
                     01D1    420 ;
                     01D1    421 ; OUTPUT:
                     01D1    422 ;
                     01D1    423 ;       R2 = Pointer to the next packet entry in arglist
                     01D1    424 ;--
                     01D1    425
                     01D1    426
                     01D1    427 NSA$ARGLST_IMGNAM::
               38    BB  01D1    428         PUSHR   #^M<R3,R4,R5>                   ; Save registers
82  00040001  8F    D0  01D3    429         MOVL    #NSA$K_PKTTYP_IMGNAM+< -        ; Store packet type and passing
                     01DA    430                 NSA$K_ARG_MECH_DESC@16>,(R2)+   ; ...mechanism in arglist
54  00000000'9F      DE  01DA    431         MOVAL   @#CTL$GL_IMGHDRBF,R4            ; Get address of longword that has
                     01E1    432                                                 ; the image header buffer address
    00000000'GF      16  01E1    433         JSB     G^EXE$CHKIMAGNAME               ; Get the image name
        82   53      7D  01E7    434         MOVQ    R3,(R2)+                        ; Put size and address in arglist
               38    BA  01EA    435         POPR    #^M<R3,R4,R5>                   ; Restore registers
                     05  01EC    436         RSB
                     01ED    437
                     01ED    438         .END
```

```
ALARM_RECORD                    0000014E R      02
BUILD_RECORD                    000000D8 R      02
BUILD_RECORD_DT                 0000004E R      02
CTL$GC_IMGHDRBF                 ********   X    02
CTL$T_USERNAME                  ********   X    02
DSC$K_CLASS_S                 = 00000001
DSC$K_DTYPE_T                 = 0000000E
EXE$CRAKIMAGNAME                ********   X    02
EXE$GQ_SYSTIME                  ********   X    02
EXE$WRTMAILBOX                  ********   X    02
GET_NEXT_PKT                    000000AC R      02
IPL$_ASTDEL                   = 00000002
IPL$_SYNCH                    = 00000008
JIB$S_ACCOUNT                 = 00000008
JIB$S_USERNAME                = 0000000C
JOURNAL_RECORD                  0000014A R      02
MSG$_OPRQST                   = 00000008
NSA$ARGLST_IMGNAM               000001D1 RG     02
NSA$B_ARG_FLAG                = 00000008
NSA$B_ARG_PKTNUM              = 00000009
NSA$EVENT_AUDIT                 00000000 RG     02
NSA$K_ARG_MECH_DESCR          = 00000004
NSA$K_PKTHDR_LENGTH           = 00000004
NSA$K_PKTTYP_IMGNAM           = 00000001
NSA$K_RECHDR_LENGTH           = 0000004C
NSA$K_REC_MAXLENGTH           = 00000400
NSA$L_ARG_ID                  = 00000004
NSA$M_ARG_FLAG_ALARM          = 00000001
NSA$M_ARG_FLAG_MANDY          = 00000004
NSA$M_REC_FLAGS_PKTCON        = 00000001
NSA$Q_IDT_RECORD_DESCR        = 0000042E
NSA$S_IDT_ALARM_HDR           = 0000002E
NSA$S_IDT_RECORD_BUF          = 00000400
NSA$T_ARG_LIST                = 0000000C
NSA$T_IDT                       ********   X    02
NSA$T_IDT_ALARM_HDR           = 00000000
NSA$T_IDT_RECORD_BUF          = 0000002E
NSA$T_IDT_RECORD_DT           = 00000436
NSA$V_ARG_FLAG_JOURN          = 00000001
OPC$M_NM_SECURITY             = 00000100
OPC$_RQ_SECURITY              = 00000007
PCB$B_ASTACT                  = 0000000C
PCB$B_ASTEN                   = 0000000D
PCB$L_EPID                    = 00000064
PCB$L_UIC                     = 000000BC
PCB$S_LNAME                   = 00000010
PCB$T_LNAME                   = 00000070
PR$_IPL                       = 00000012
PRV$V_SECURITY                = 00000026
PSL$C_KERNEL                  = 00000000
REC_MIN_SIZE_LEFT             = 00000010
RSN$_MAILBOX                  = 00000002
RSN$_NPDYNMEM                 = 00000003
SB$S_NODENAME                 = 00000010
SB$T_NODENAME                 = 00000044
SCH$GL_CURPCB                   ********   X    02
SCH$RWAIT                       ********   X    02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	000001ED (493.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:01.67
Command processing	108	00:00:00.53	00:00:05.31
Pass 1	385	00:00:13.21	00:00:42.82
Symbol table sort	0	00:00:02.13	00:00:05.09
Pass 2	91	00:00:02.37	00:00:09.06
Symbol table output	8	00:00:00.11	00:00:00.70
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	626	00:00:18.43	00:01:04.67

The working set limit was 1500 pages.
74292 bytes (146 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1404 non-local and 23 local symbols.
438 source lines were read in Pass 1, producing 17 object records in Pass 2.
24 pages of virtual memory were used to define 23 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	8
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	20

1517 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SECAUDIT/OBJ=OBJ$:SECAUDIT MSRC$:SECAUDIT/UPDATE=(ENH$:SECAUDIT)+EXECML$/LIB
```

SHELL
LIS

RSE
LIS

SCBVECTOR
LIS

SDAT
LIS

SHMGSDRTN
LIS

SCSVEC
LIS

SCHED
LIS

SECAUDIT
LIS

SPTSKEL
LIS

RUFSYSVEC
LIS