Aplicații practice ale limbajului de asamblare

Arhitectura Sistemelor de Calcul

Ciprian Oprișa, PhD

Bitdefender

21 decembrie 2016

Cuprins

Cuprins

De la cod la binar

```c
#include <stdio.h>

const char *msg = "Hello world!";

int main(void){
    printf("%s\n", msg);
    return 0;
}
```

```
55 8B EC A1 20 30 40 00    Uï‹í 0@
50 68 E0 20 40 00 FF 15    Phα @  §
A0 20 40 00 83 C4 08 33    á @ â─▬3
C0 5D C3 68 E8 13 40 00    └]├hè‼@
```

De la cod la binar

```c
#include <stdio.h>

const char *msg = "Hello world!";

int main(void){
    printf("%s\n", msg);
    return 0;
}
```

← cod C

cod binar →

```
55 8B EC A1 20 30 40 00   Uï¤í 0@
50 68 E0 20 40 00 FF 15   Phα @  §
A0 20 40 00 83 C4 08 33   á @ â─▬3
C0 5D C3 68 E8 13 40 00   └]├hᵹ‼@
```

De la cod la binar

```c
#include <stdio.h>

const char *msg = "Hello world!";

int main(void){
    printf("%s\n", msg);
    return 0;
}
```

← cod C

↘

```
55              U       push    ebp
8BEC            ï∞      mov     ebp, esp
A1 20304000     í 0@    mov     eax, [0x403020]
50              P       push    eax
68 E0204000     hα @    push    0x4020e0
FF15 A0204000    §á @  call  (1) [MSVCR100.dll:printf]
83C4 08         â─■      add     esp, 0x8
33C0            3└      xor     eax, eax
5D              ]       pop     ebp
C3              ├       ret
```

↘

cod binar →

```
55 8B EC A1 20 30 40 00    Uï∞í 0@
50 68 E0 20 40 00 FF 15    Phα @  §
A0 20 40 00 83 C4 08 33    á @ â─■3
C0 5D C3 68 E8 13 40 00    └]├hè‼@
```

Limbajul de asamblare și codul binar

Important

Calculatoarele înțeleg doar codul binar.

- Limbajul de asamblare este o formă inteligibilă de cod binar.
- Limbajul de asamblare poate fi translatat direct în binar **și invers**.

Bitdefender

Limbajul de asamblare și codul binar

Important

Calculatoarele înțeleg doar codul binar.

- Limbajul de asamblare este o formă inteligibilă de cod binar.
- Limbajul de asamblare poate fi translatat direct în binar **și invers**.

Limbaje compilate vs. interpretate:

- Un limbaj compilat (C, C++, Pascal, Rust) este translatat în Assembly / cod binar.
- Un limbaj interpretat (Java, Python, C#) necesită un interpretor care este de asemenea binar.

Cuprins

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

```
push      ebp
mov       ebp, esp
push      ecx
mov       byte [ebp-0x1], 0x1b
mov       byte [ebp-0x2], 0x4
movzx     eax, byte [ebp-0x1]
push      eax
movzx     ecx, byte [ebp-0x1]
push      ecx
push      0x4020d0 ;-> 'before: %d (%02x) '
call  (1) [MSVCR100.dll:printf]
add       esp, 0xc
mov       cl, [ebp-0x2]
ror       byte [ebp-0x1], cl
movzx     edx, byte [ebp-0x1]
push      edx
movzx     eax, byte [ebp-0x1]
push      eax
push      0x4020e4 ;-> 'after: %d (%02x) '
call  (2) [MSVCR100.dll:printf]
add       esp, 0xc
xor       eax, eax
mov       esp, ebp
pop       ebp
ret
```

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

```
push      ebp
mov       ebp, esp
push      ecx
mov       byte [ebp-0x1], 0x1b
mov       byte [ebp-0x2], 0x4
movzx     eax, byte [ebp-0x1]
push      eax
movzx     ecx, byte [ebp-0x1]
push      ecx
push      0x4020d0 ;-> 'before: %d (%02x) '
call  (1) [MSVCR100.dll:printf]
add       esp, 0xc
mov       cl, [ebp-0x2]
ror       byte [ebp-0x1], cl
movzx     edx, byte [ebp-0x1]
push      edx
movzx     eax, byte [ebp-0x1]
push      eax
push      0x4020e4 ;-> 'after: %d (%02x) '
call  (2) [MSVCR100.dll:printf]
add       esp, 0xc
xor       eax, eax
mov       esp, ebp
pop       ebp
ret
```

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

```
push    ebp
mov     ebp, esp
push    ecx
mov     byte [ebp-0x1], 0x1b
mov     byte [ebp-0x2], 0x4
movzx   eax, byte [ebp-0x1]
push    eax
movzx   ecx, byte [ebp-0x1]
push    ecx
push    0x4020d0 ;-> 'before: %d (%02x) '
call    (1) [MSVCR100.dll:printf]
add     esp, 0xc
mov     cl, [ebp-0x2]
ror     byte [ebp-0x1], cl
movzx   edx, byte [ebp-0x1]
push    edx
movzx   eax, byte [ebp-0x1]
push    eax
push    0x4020e4 ;-> 'after: %d (%02x) '
call    (2) [MSVCR100.dll:printf]
add     esp, 0xc
xor     eax, eax
mov     esp, ebp
pop     ebp
ret
```

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

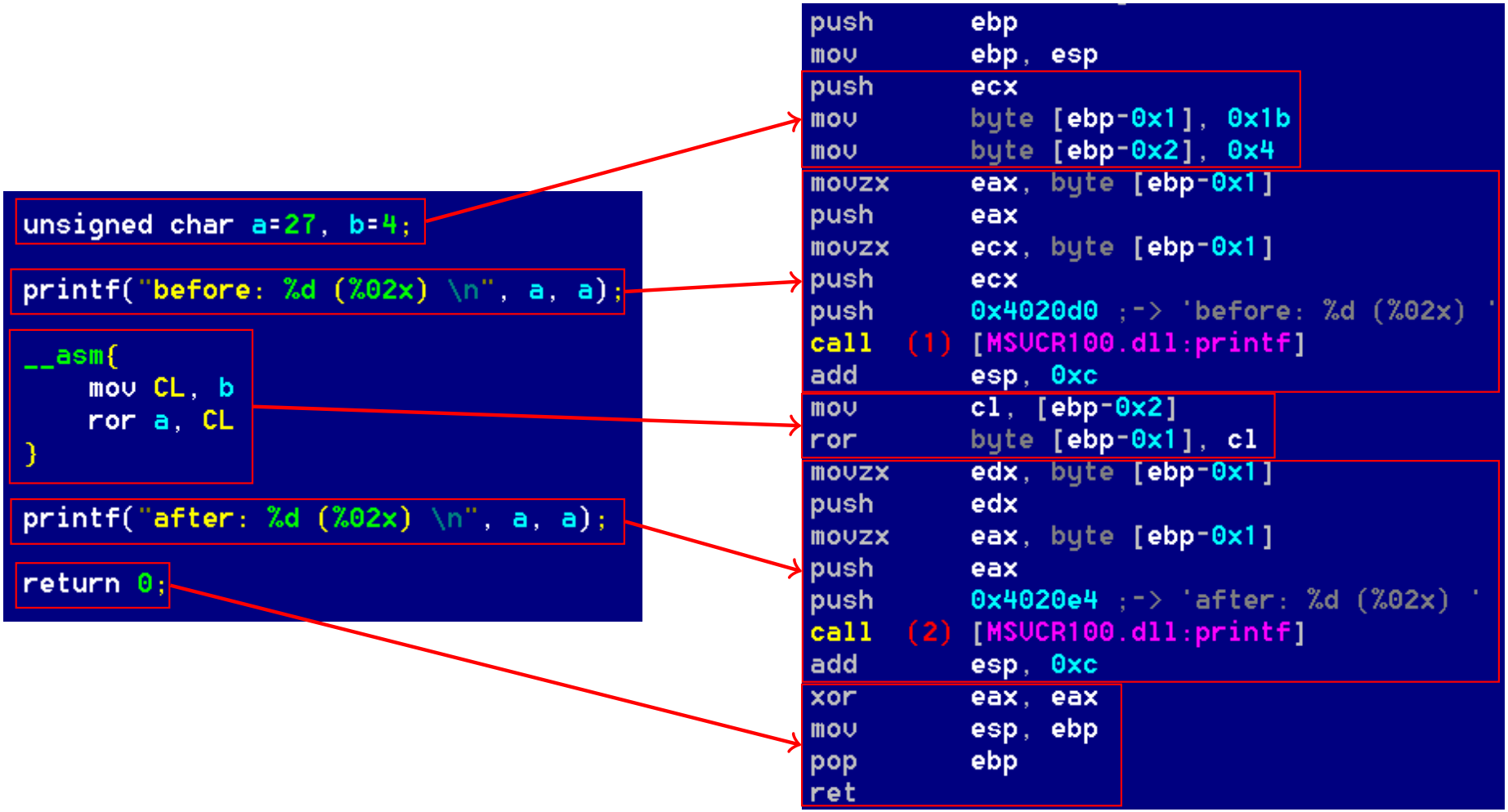
```
push    ebp
mov     ebp, esp
push    ecx
mov     byte [ebp-0x1], 0x1b
mov     byte [ebp-0x2], 0x4
movzx   eax, byte [ebp-0x1]
push    eax
movzx   ecx, byte [ebp-0x1]
push    ecx
push    0x4020d0 ;-> 'before: %d (%02x) '
call    (1) [MSVCR100.dll:printf]
add     esp, 0xc
mov     cl, [ebp-0x2]
ror     byte [ebp-0x1], cl
movzx   edx, byte [ebp-0x1]
push    edx
movzx   eax, byte [ebp-0x1]
push    eax
push    0x4020e4 ;-> 'after: %d (%02x) '
call    (2) [MSVCR100.dll:printf]
add     esp, 0xc
xor     eax, eax
mov     esp, ebp
pop     ebp
ret
```

Fragmente de limbaj de asamblare în C

```
unsigned char a=27, b=4;

printf("before: %d (%02x) \n", a, a);

__asm{
    mov CL, b
    ror a, CL
}

printf("after: %d (%02x) \n", a, a);

return 0;
```

```
push    ebp
mov     ebp, esp
push    ecx
mov     byte [ebp-0x1], 0x1b
mov     byte [ebp-0x2], 0x4
movzx   eax, byte [ebp-0x1]
push    eax
movzx   ecx, byte [ebp-0x1]
push    ecx
push    0x4020d0 ;-> 'before: %d (%02x) '
call    (1) [MSVCR100.dll:printf]
add     esp, 0xc
mov     cl, [ebp-0x2]
ror     byte [ebp-0x1], cl
movzx   edx, byte [ebp-0x1]
push    edx
movzx   eax, byte [ebp-0x1]
push    eax
push    0x4020e4 ;-> 'after: %d (%02x) '
call    (2) [MSVCR100.dll:printf]
add     esp, 0xc
xor     eax, eax
mov     esp, ebp
pop     ebp
ret
```

Fragmente de limbaj de asamblare în C



Cuprins

Bitdefender

Inginerie inversă

Definiție

Analiza unui program pentru a se înțelege ce face și cum funcționează.

- analiză statică - analiza codului
- analiză dinamică - analiza comportamentului programului

Bitdefender

Analiza statică

Problemă: Programele binare nu au un cod sursă de analizat.

Bitdefender

Analiza statică

Problemă: Programele binare nu au un cod sursă de analizat.

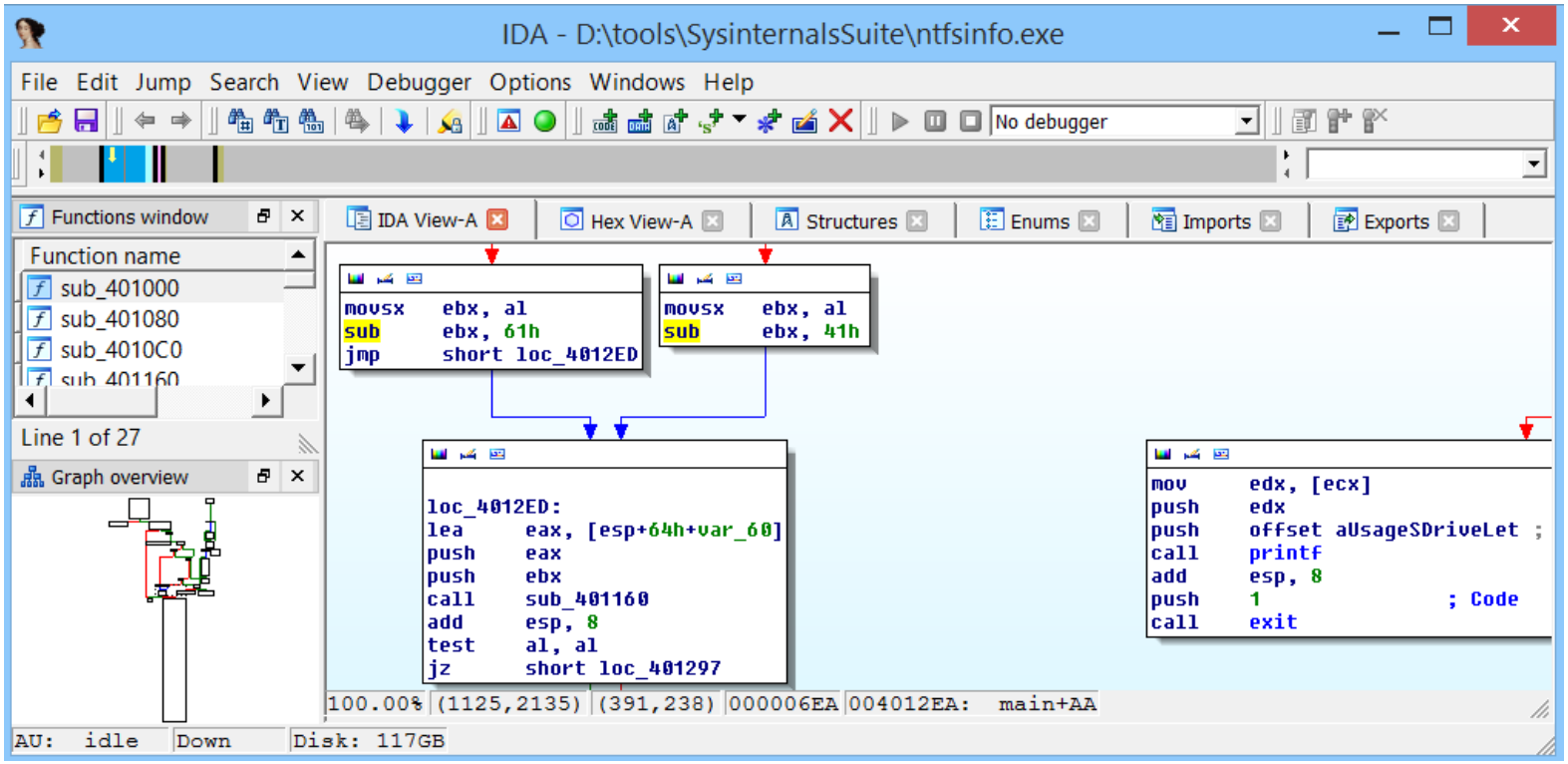
Să ne amintim: *"Assembly poate fi translatat direct în binar* **și invers***"*.

Analiza statică

Problemă: Programele binare nu au un cod sursă de analizat.

Să ne amintim: *"Assembly poate fi translatat direct în binar* **și invers***"*.

Unealtă de analiză: IDA Pro



Bitdefender

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Bitdefender

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Interceptarea funcțiilor:

```
; HANDLE __stdcall CreateRemoteThread(HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes
                public CreateRemoteThread
CreateRemoteThread proc near            ; DATA XREF: .rdata:off_6B8E85A8↓o

hProcess        = dword ptr  8
lpThreadAttributes= dword ptr  0Ch
dwStackSize     = dword ptr  10h
lpStartAddress  = dword ptr  14h
lpParameter     = dword ptr  18h
dwCreationFlags = dword ptr  1Ch
lpThreadId      = dword ptr  20h

8B FF                   mov     edi, edi
55                      push    ebp
8B EC                   mov     ebp, esp
FF 75 20                push    [ebp+lpThreadId]
8B 45 1C                mov     eax, [ebp+dwCreationFlags]
6A 00                   push    0
25 04 00 01+            and     eax, 10004h
50                      push    eax
FF 75 18                push    [ebp+lpParameter]
FF 75 14                push    [ebp+lpStartAddress]
FF 75 10                push    [ebp+dwStackSize]
FF 75 0C                push    [ebp+lpThreadAttributes]
FF 75 08                push    [ebp+hProcess]
```

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Interceptarea funcțiilor:

```
; HANDLE __stdcall CreateRemoteThread(HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes
                public CreateRemoteThread
CreateRemoteThread proc near               ; DATA XREF: .rdata:off_6B8E85A8↓o

hProcess        = dword ptr  8
lpThreadAttributes= dword ptr  0Ch
dwStackSize     = dword ptr  10h
lpStartAddress  = dword ptr  14h
lpParameter     = dword ptr  18h
dwCreationFlags = dword ptr  1Ch
lpThreadId      = dword ptr  20h

8B FF                   mov     edi, edi
55                      push    ebp
8B EC                   mov     ebp, esp
FF 75 20                push    [ebp+lpThreadId]
8B 45 1C                mov     eax, [ebp+dwCreationFlags]
6A 00                   push    0
25 04 00 01+            and     eax, 10004h
50                      push    eax
FF 75 18                push    [ebp+lpParameter]
FF 75 14                push    [ebp+lpStartAddress]
FF 75 10                push    [ebp+dwStackSize]
FF 75 0C                push    [ebp+lpThreadAttributes]
FF 75 08                push    [ebp+hProcess]
```

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Interceptarea funcțiilor:

```
; HANDLE __stdcall CreateRemoteThread(HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes
                public CreateRemoteThread
CreateRemoteThread proc near            ; DATA XREF: .rdata:off_6B8E85A8↓o

hProcess        = dword ptr  8
lpThreadAttributes= dword ptr  0Ch
dwStackSize     = dword ptr  10h
lpStartAddress  = dword ptr  14h
lpParameter     = dword ptr  18h
dwCreationFlags = dword ptr  1Ch
lpThreadId      = dword ptr  20h

8B FF                   mov     edi, edi
55                      push    ebp
8B EC                   mov     ebp, esp
FF 75 20                push    [ebp+lpThreadId]
8B 45 1C                mov     eax, [ebp+dwCreationFlags]
6A 00                   push    0
25 04 00 01+            and     eax, 10004h
50                      push    eax
FF 75 18                push    [ebp+lpParameter]
FF 75 14                push    [ebp+lpStartAddress]
FF 75 10                push    [ebp+dwStackSize]
FF 75 0C                push    [ebp+lpThreadAttributes]
FF 75 08                push    [ebp+hProcess]
```

se înlocuiește cu:
`jmp interception`

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Interceptarea funcțiilor:

```
; HANDLE __stdcall CreateRemoteThread(HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes
                public CreateRemoteThread
CreateRemoteThread proc near            ; DATA XREF: .rdata:off_6B8E85A8↓o

hProcess        = dword ptr  8
lpThreadAttributes= dword ptr  0Ch
dwStackSize     = dword ptr  10h
lpStartAddress  = dword ptr  14h
lpParameter     = dword ptr  18h
dwCreationFlags = dword ptr  1Ch
lpThreadId      = dword ptr  20h

8B FF                   mov     edi, edi
55                      push    ebp
8B EC                   mov     ebp, esp
FF 75 20                push    [ebp+lpThreadId]
8B 45 1C                mov     eax, [ebp+dwCreationFlags]
6A 00                   push    0
25 04 00 01+            and     eax, 10004h
50                      push    eax
FF 75 18                push    [ebp+lpParameter]
FF 75 14                push    [ebp+lpStartAddress]
FF 75 10                push    [ebp+dwStackSize]
FF 75 0C                push    [ebp+lpThreadAttributes]
FF 75 08                push    [ebp+hProcess]
```

se înlocuiește cu:

```
jmp interception
```

```
interception:
    push ebp
    mov ebp, esp

    ... cod de logare ...
```

Analiza dinamică

Depanare:

- rularea instrucțiunilor Assembly pas cu pas *(demo)*

Interceptarea funcțiilor:

```
; HANDLE __stdcall CreateRemoteThread(HANDLE hProcess, LPSECURITY_ATTRIBUTES lpThreadAttributes
                public CreateRemoteThread
CreateRemoteThread proc near            ; DATA XREF: .rdata:off_6B8E85A8↓o

hProcess        = dword ptr  8
lpThreadAttributes= dword ptr  0Ch
dwStackSize     = dword ptr  10h
lpStartAddress  = dword ptr  14h
lpParameter     = dword ptr  18h
dwCreationFlags = dword ptr  1Ch
lpThreadId      = dword ptr  20h

8B FF                   mov     edi, edi
55                      push    ebp
8B EC                   mov     ebp, esp
FF 75 20                push    [ebp+lpThreadId]
8B 45 1C                mov     eax, [ebp+dwCreationFlags]
6A 00                   push    0
25 04 00 01+            and     eax, 10004h
50                      push    eax
FF 75 18                push    [ebp+lpParameter]
FF 75 14                push    [ebp+lpStartAddress]
FF 75 10                push    [ebp+dwStackSize]
FF 75 0C                push    [ebp+lpThreadAttributes]
FF 75 08                push    [ebp+hProcess]
```

se înlocuiește cu:

```
jmp interception
```

```
interception:
    push ebp
    mov ebp, esp

    ... cod de logare ...

    jmp fn+5
```

Cuprins

Bitdefender

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

O privire mai atentă la cadrele de stivă

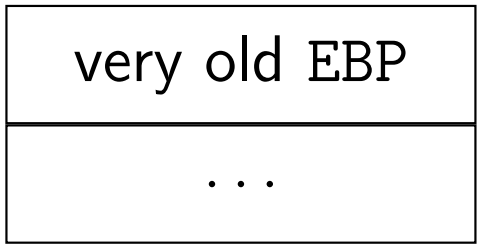
```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```



O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

O privire mai atentă la cadrele de stivă

```
void func(const char *s){
    char myCopy[8];
    strcpy(myCopy, s);
}

int main(void){
    func("12345");
    return 0;
}
```

```
sub_401000
push    ebp
mov     ebp, esp
sub     esp, 0x8
mov     eax, [ebp+0x8]
push    eax
lea     ecx, [ebp-0x8]
push    ecx
call    j_MSVCR100.dll:strcpy
add     esp, 0x8
mov     esp, ebp
pop     ebp
ret

push    ebp
mov     ebp, esp
push    0x4020c8 ;-> '12345'
call    sub_401000
add     esp, 0x4
xor     eax, eax
pop     ebp
ret
```

Ce se întâmplă dacă trimitem un string de 16 octeți?

Bitdefender

Exploit-uri de tip buffer overflow

- un buffer overflow poate fi folosit ca să se suprascrie adresa de revenire de pe stivă
- o valoare aleatoare probabil va face programul să crape

Bitdefender

Exploit-uri de tip buffer overflow

- un buffer overflow poate fi folosit ca să se suprascrie adresa de revenire de pe stivă
- o valoare aleatoare probabil va face programul să crape
  - ...dar putem face mai mult de atât

Bitdefender

Exploit-uri de tip buffer overflow

- un buffer overflow poate fi folosit ca să se suprascrie adresa de revenire de pe stivă
- o valoare aleatoare probabil va face programul să crape
  - ...dar putem face mai mult de atât
- putem să rulăm cod arbitrar prin schimbarea adresei de revenire ca să pointeze spre el

Bitdefender

Exploit-uri de tip buffer overflow

- un buffer overflow poate fi folosit ca să se suprascrie adresa de revenire de pe stivă
- o valoare aleatoare probabil va face programul să crape
  - ...dar putem face mai mult de atât
- putem să rulăm cod arbitrar prin schimbarea adresei de revenire ca să pointeze spre el

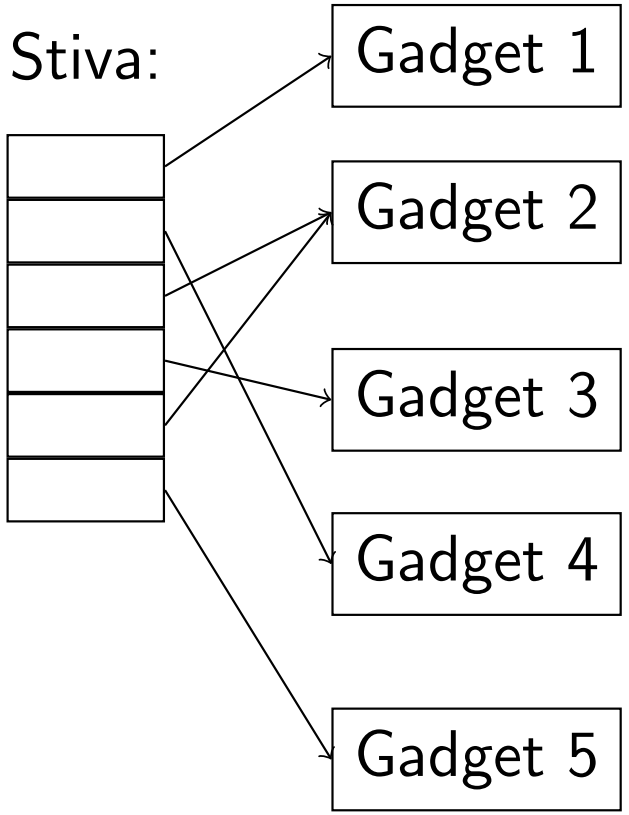
Problemă: Data Execution Prevention

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

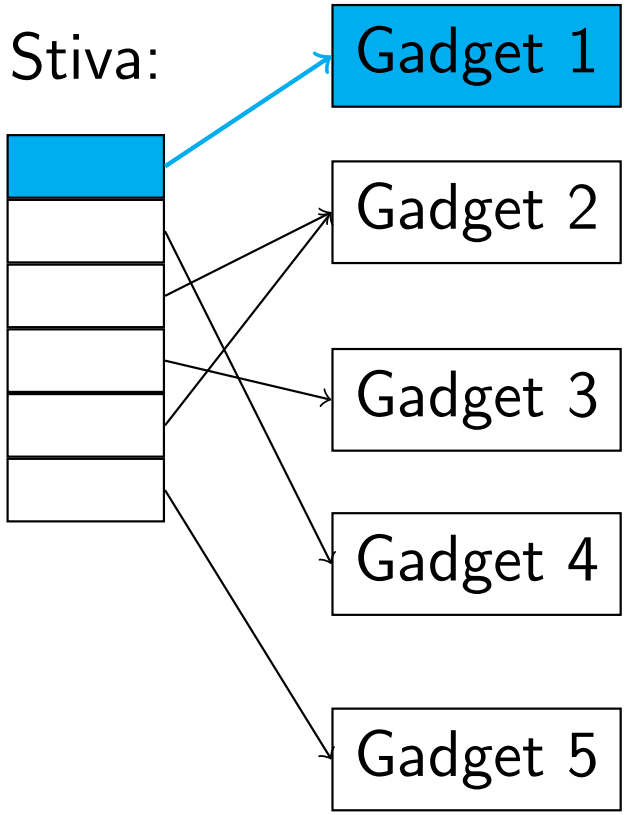


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

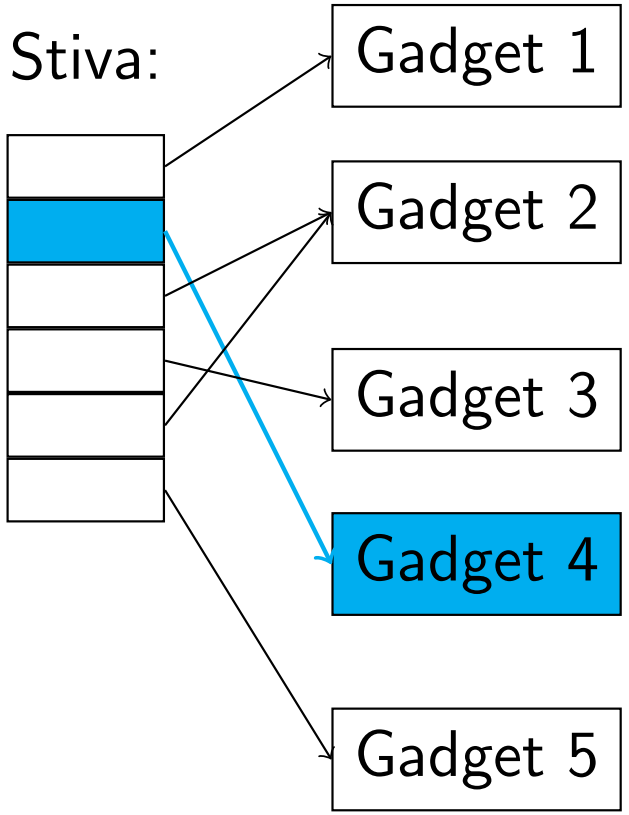


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

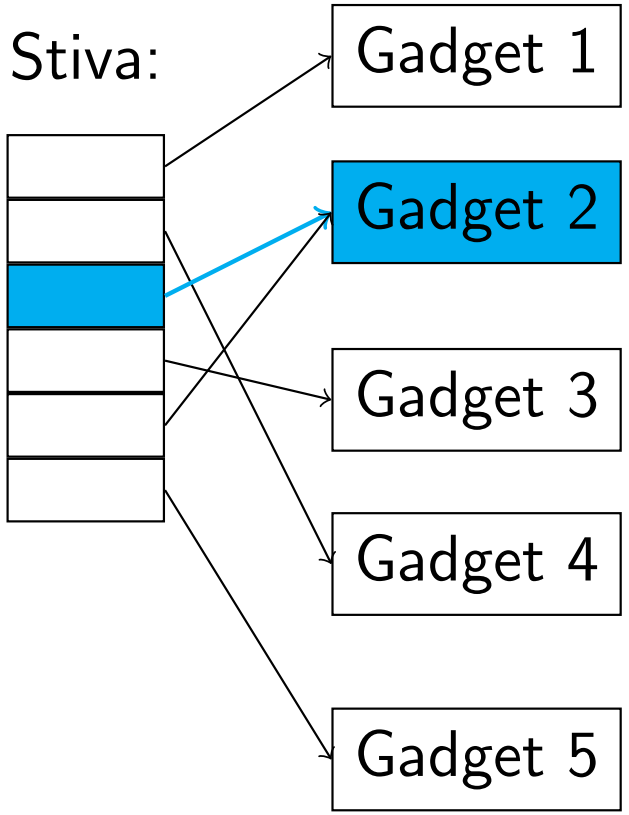


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

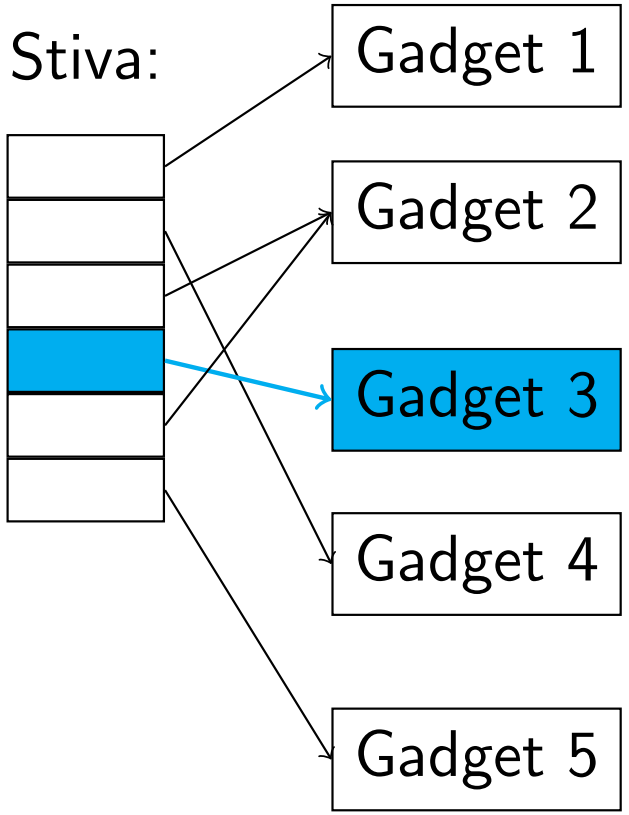


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

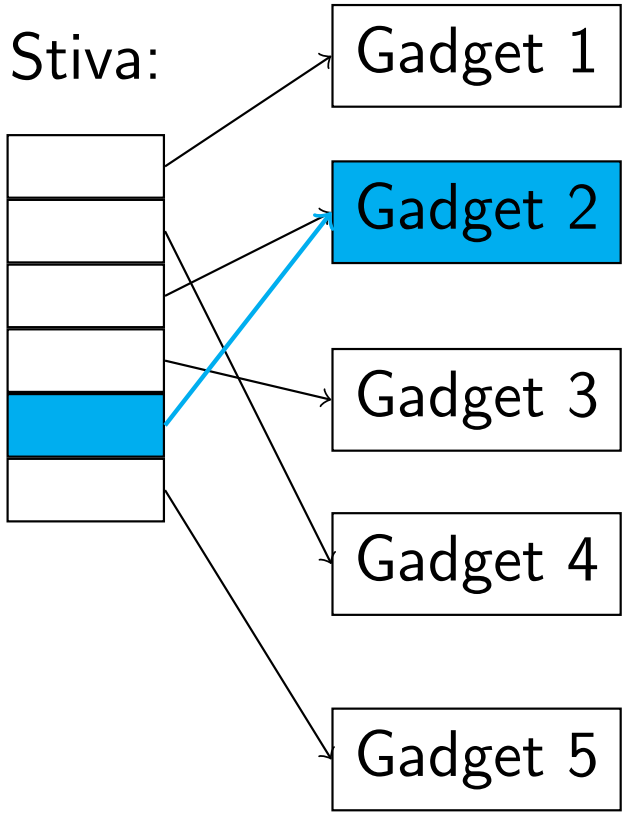


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`

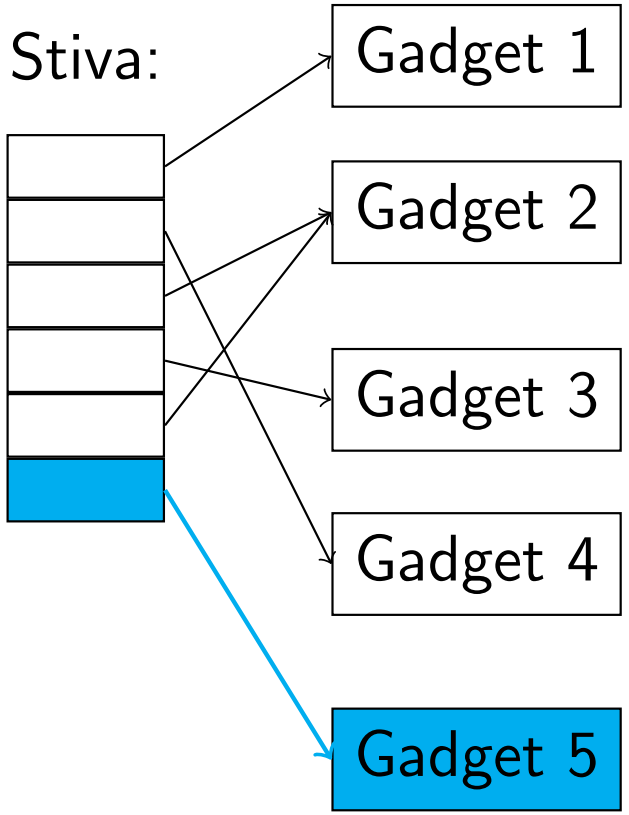


Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

- un program care rulează împreună cu bibliotecile sale conține multe funcții
- fiecare funcție se termină cu o instrucțiune `ret`



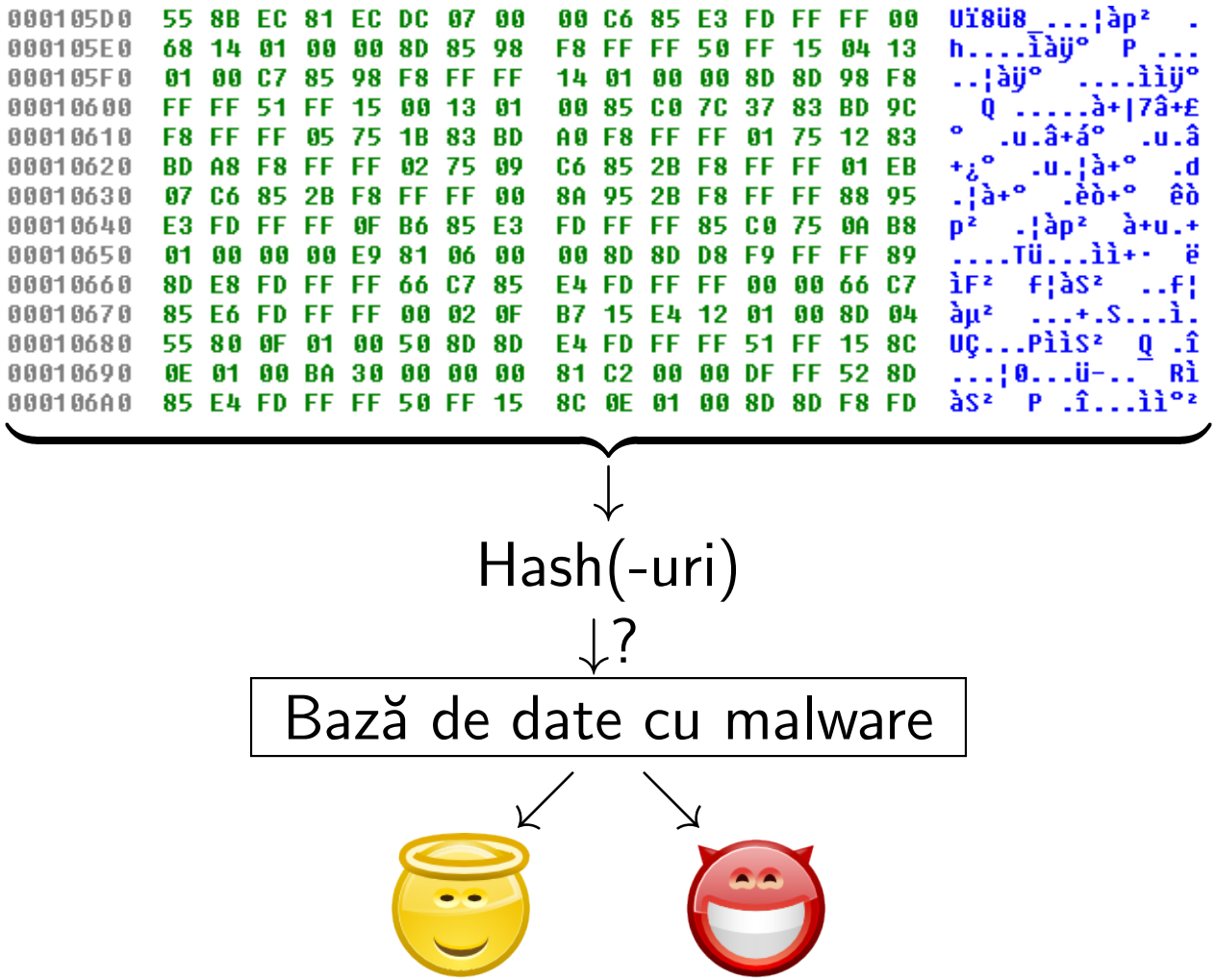
Definiție

Un grup de instrucțiuni terminat cu `ret` se numește un *gadget*.

Putem scrie programe utile prin înlănțuire de *gadget-uri*.

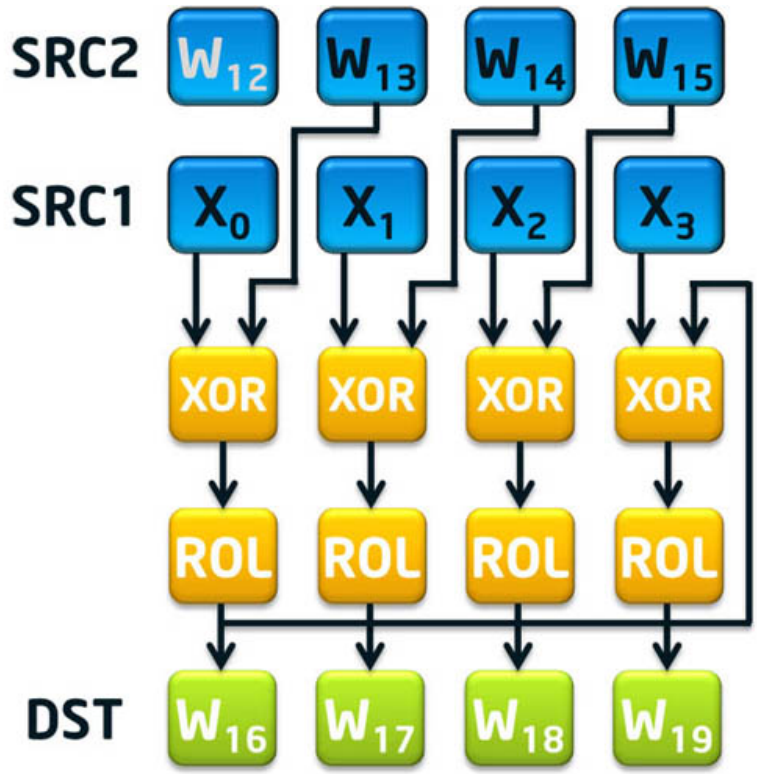
Cuprins

Mecanismul de detecție tradițional



Calculul funcțiilor de hash

- calculul funcțiilor de hash necesită multe operații pe biți
- mai ușor de făcut în hardware decât în software
- limbajul de asamblare de la Intel x86 pune la dispoziție astfel de operații



instrucțiunea `SHA1MSG2`

figură de pe `https://software.intel.com`

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ÌàÿÎ  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿÎ  ....ÌìÿÎ
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ÌÌ+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...Ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PÌìS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ÌìÎ²
```

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  U‹ì8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push    ebp
000105D1 8B EC                mov     ebp, esp
000105D3 81 EC DC 07 00 00    sub     esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov     [ebp+var_21D], 0
000105E0 68 14 01 00 00       push    114h
000105E5 8D 85 98 F8 FF FF    lea     eax, [ebp+var_768]
000105EB 50                   push    eax
000105EC FF 15 04 13 01 00    call    ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov     [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea     ecx, [ebp+var_768]
00010602 51                   push    ecx
00010603 FF 15 00 13 01 00    call    ds:dword_11300
00010609 85 C0                test    eax, eax
0001060B 7C 37                jl      short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp     [ebp+var_764], 5
00010614 75 1B                jnz     short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp     [ebp+var_760], 1
0001061D 75 12                jnz     short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp     [ebp+var_758], 2
00010626 75 09                jnz     short loc_10631
00010628 C6 85 2B F8 FF FF+   mov     [ebp+var_7D5], 1
0001062F EB 07                jmp     short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pmsmplpcmlpctjczczczmJ

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....TÜ...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pmsmplpc mlpctjczczczmJ

<pmsmplpc>

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ p msmplpcm lpctjczczczmJ

<pmsmplpc>, <msmplpcm>

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pm smplpcml pctjczczczmJ

<pmsmplpc>, <msmplpcm>, <smplpcml>

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pms **mplpcmlp** ctjczczczmJ

<pmsmplpc>, <msmplpcm>, <smplpcml>, <mplpcmlp>

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  Uï8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....TÜ...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pmsm plpcmlpc tjczczczmJ

<pmsmplpc>, <msmplpcm>, <smplpcml>, <mplpcmlp>, <plpcmlpc>

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  U‹ì8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....Tü...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PìiS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pmsmplpcmlpctjczczczmJ

<pmsmplpc>, <msmplpcm>, <smplpcml>, <mplpcmlp>, <plpcmlpc>, . . .

Folosirea de *n*-grame din OpCode-uri

```
000105D0  55 8B EC 81 EC DC 07 00  00 C6 85 E3 FD FF FF 00  U‹ì8ü8_...¦àp²  .
000105E0  68 14 01 00 00 8D 85 98  F8 FF FF 50 FF 15 04 13  h....ìàÿ°  P ...
000105F0  01 00 C7 85 98 F8 FF FF  14 01 00 00 8D 8D 98 F8  ..¦àÿ°  ....ììÿ°
00010600  FF FF 51 FF 15 00 13 01  00 85 C0 7C 37 83 BD 9C    Q .....à+|7â+£
00010610  F8 FF FF 05 75 1B 83 BD  A0 F8 FF FF 01 75 12 83  °  .u.â+á°  .u.â
00010620  BD A8 F8 FF FF 02 75 09  C6 85 2B F8 FF FF 01 EB  +¿°  .u.¦à+°  .d
00010630  07 C6 85 2B F8 FF FF 00  8A 95 2B F8 FF FF 88 95  .¦à+°  .èò+°  êò
00010640  E3 FD FF FF 0F B6 85 E3  FD FF FF 85 C0 75 0A B8  p²  .¦àp²  à+u.+
00010650  01 00 00 00 E9 81 06 00  00 8D 8D D8 F9 FF FF 89  ....TÜ...ìì+·  ë
00010660  8D E8 FD FF FF 66 C7 85  E4 FD FF FF 00 00 66 C7  ìF²  f¦àS²  ..f¦
00010670  85 E6 FD FF FF 00 02 0F  B7 15 E4 12 01 00 8D 04  àµ²  ...+.S...ì.
00010680  55 80 0F 01 00 50 8D 8D  E4 FD FF FF 51 FF 15 8C  UÇ...PììS²  Q .î
00010690  0E 01 00 BA 30 00 00 00  81 C2 00 00 DF FF 52 8D  ...¦0...ü-..  Rì
000106A0  85 E4 FD FF FF 50 FF 15  8C 0E 01 00 8D 8D F8 FD  àS²  P .î...ìì°²
```

↓

```
000105D0 55                   push   ebp
000105D1 8B EC                mov    ebp, esp
000105D3 81 EC DC 07 00 00    sub    esp, 7DCh
000105D9 C6 85 E3 FD FF FF+   mov    [ebp+var_21D], 0
000105E0 68 14 01 00 00       push   114h
000105E5 8D 85 98 F8 FF FF    lea    eax, [ebp+var_768]
000105EB 50                   push   eax
000105EC FF 15 04 13 01 00    call   ds:dword_11304
000105F2 C7 85 98 F8 FF FF+   mov    [ebp+var_768], 114h
000105FC 8D 8D 98 F8 FF FF    lea    ecx, [ebp+var_768]
00010602 51                   push   ecx
00010603 FF 15 00 13 01 00    call   ds:dword_11300
00010609 85 C0                test   eax, eax
0001060B 7C 37                jl     short loc_10644
0001060D 83 BD 9C F8 FF FF+   cmp    [ebp+var_764], 5
00010614 75 1B                jnz    short loc_10631
00010616 83 BD A0 F8 FF FF+   cmp    [ebp+var_760], 1
0001061D 75 12                jnz    short loc_10631
0001061F 83 BD A8 F8 FF FF+   cmp    [ebp+var_758], 2
00010626 75 09                jnz    short loc_10631
00010628 C6 85 2B F8 FF FF+   mov    [ebp+var_7D5], 1
0001062F EB 07                jmp    short loc_10638
```

→ push, mov, sub, mov, push, lea, push, call, mov, . . .

→ pmsmplpcmlpctjczczczmJ

<pmsmplpc>, <msmplpcm>, <smplpcml>, <mplpcmlp>, <plpcmlpc>, . . .

↓?

Bază de date cu malware

Bitdefender

Similaritatea codului

Compilatorul nu ia în considerare:

- comentariile și indentarea
- numele de variabile și parantezele redundante

cod C similar $\rightarrow$ cod ASM similar $\rightarrow$ n-grame similare

Similaritatea codului

Compilatorul nu ia în considerare:

- comentariile și indentarea
- numele de variabile și parantezele redundante

cod C similar $\rightarrow$ cod ASM similar $\rightarrow$ n-grame similare

Definiție

Similaritatea Jaccard:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Bitdefender

Emularea codului

Cum se pot deduce în mod automat efectele unui program fără a se rula efectiv?

- crearea unui mediu virtual minimalist
- **dezasamblarea fiecărei instrucțiuni**
- emularea sa în interiorul mediului virtual

Un emulator trebuie să țină evidența la

- regiștri
- memorie
- mediul emulat

Cuprins

Conceptul de bootstrap

- Ce se întâmplă atunci când un calculator pornește?
- Cum poate un sistem de operare să se pornească singur?

Conceptul de bootstrap

- Ce se întâmplă atunci când un calculator pornește?
- Cum poate un sistem de operare să se pornească singur?
- se *“trage singur în sus de curelele cizmelor”*

Conceptul de bootstrap

- Ce se întâmplă atunci când un calculator pornește?
- Cum poate un sistem de operare să se pornească singur?
- se *"trage singur în sus de curelele cizmelor"*

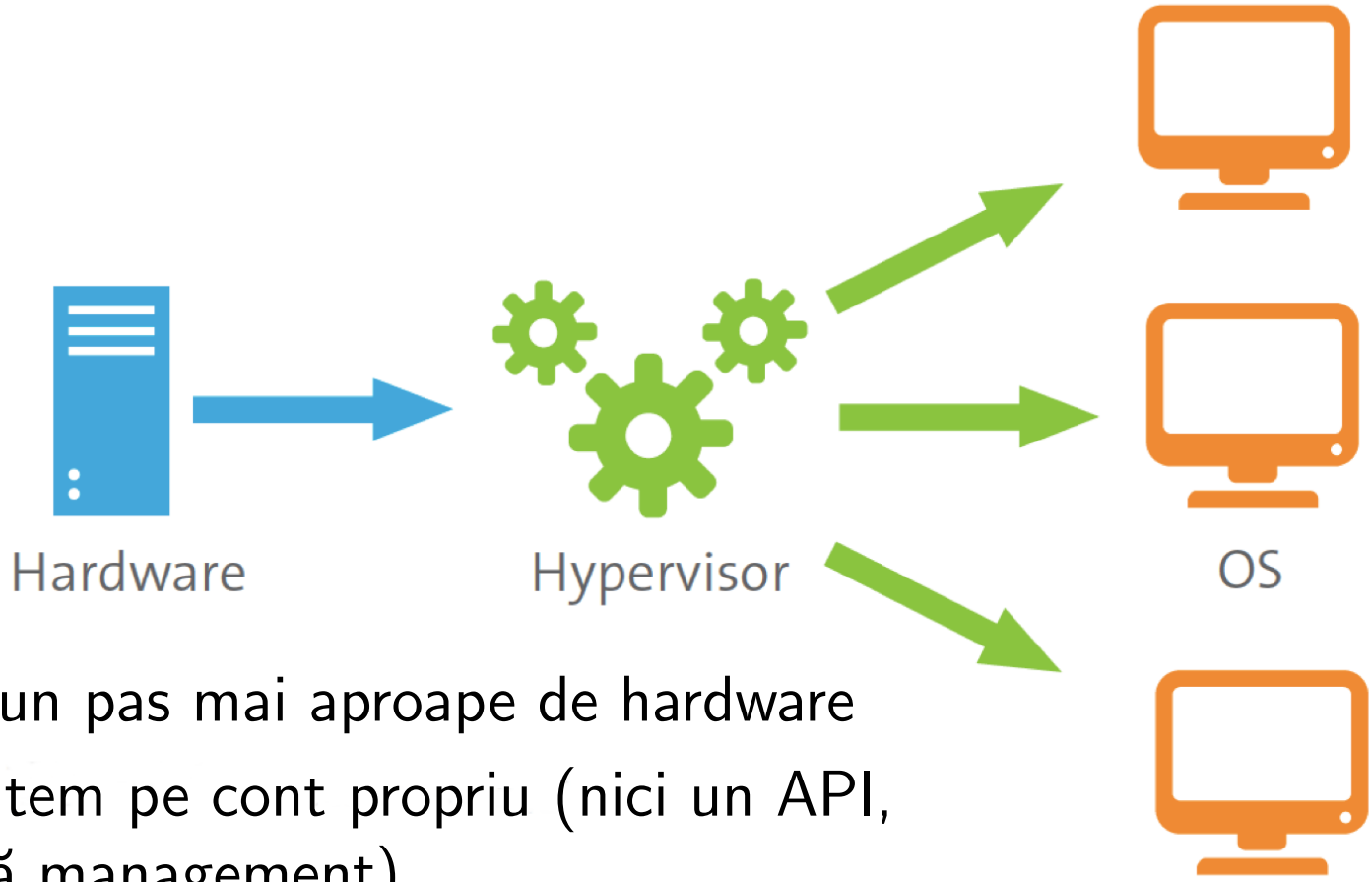
- atunci când e pornit, procesorul începe execuția codului de la adresa `FFFF:0000`, unde e localizat codul de BIOS
- apoi BIOS-ul începe să caute sectoare de boot

Sectorul de boot

- are 512 octeți
- se termină cu `0xAA55`
- se va încărca la adresa 0000 : 7C00
- are acces la întreruperile BIOS


```
[org 0x7c00]                    ; BIOS always loads the boot sector to 00:7c00
EntryPoint:
    jmp     .Code               ; avoid executing our data as code

.Message:
    db "Hello world!", 0        ; embed the message inside the generated binary

.Code:                          ; local label, actually EntryPoint.Code

    ; init data and stack (cs:ip already set, otherwise we wouldn't be here)
    xor     ax,     ax          ; ax <- 0
    mov     ds,     ax          ; data segment starts at 0 (same as cs)
    mov     ss,     ax          ; stack segment starts at 0
    mov     sp,     0x7c00      ; Top-Of-Stack at 7c00 (going backwords)

    ; print the message
    mov     ax,     0xb800
    mov     es,     ax          ; es <- b800 = address of video text matrix
    mov     si,     .Message
    cld                         ; make sure string instructions go forward

.PrintLoop:
    lodsb                       ; al <- ds:[si], si<-si+1 (al = next char)
    test    al,     al          ; check if this is the null terminator
    jz      .PrintDone          ; if so, exit the loop
    stosb                       ; es:[di]<-al, al<-al+1 (write ASCII code to video mem)
    mov     al,     7
    stosb                       ; write the color code (7=white)
    jmp     .PrintLoop

.PrintDone:

    ; stop boot flow
    hlt
    jmp     .PrintDone          ; just a safety measure

times 512 - 2 - ($ - $$) db 0 ; add zeroes until we get to 510
    dw      0x55aa              ; add the boot signature at sector end
```

Hipervizorul



- cu un pas mai aproape de hardware
- suntem pe cont propriu (nici un API, fără management)
- *vedem* totul

Bitdefender

Ciclul de viață al unui hipervizor

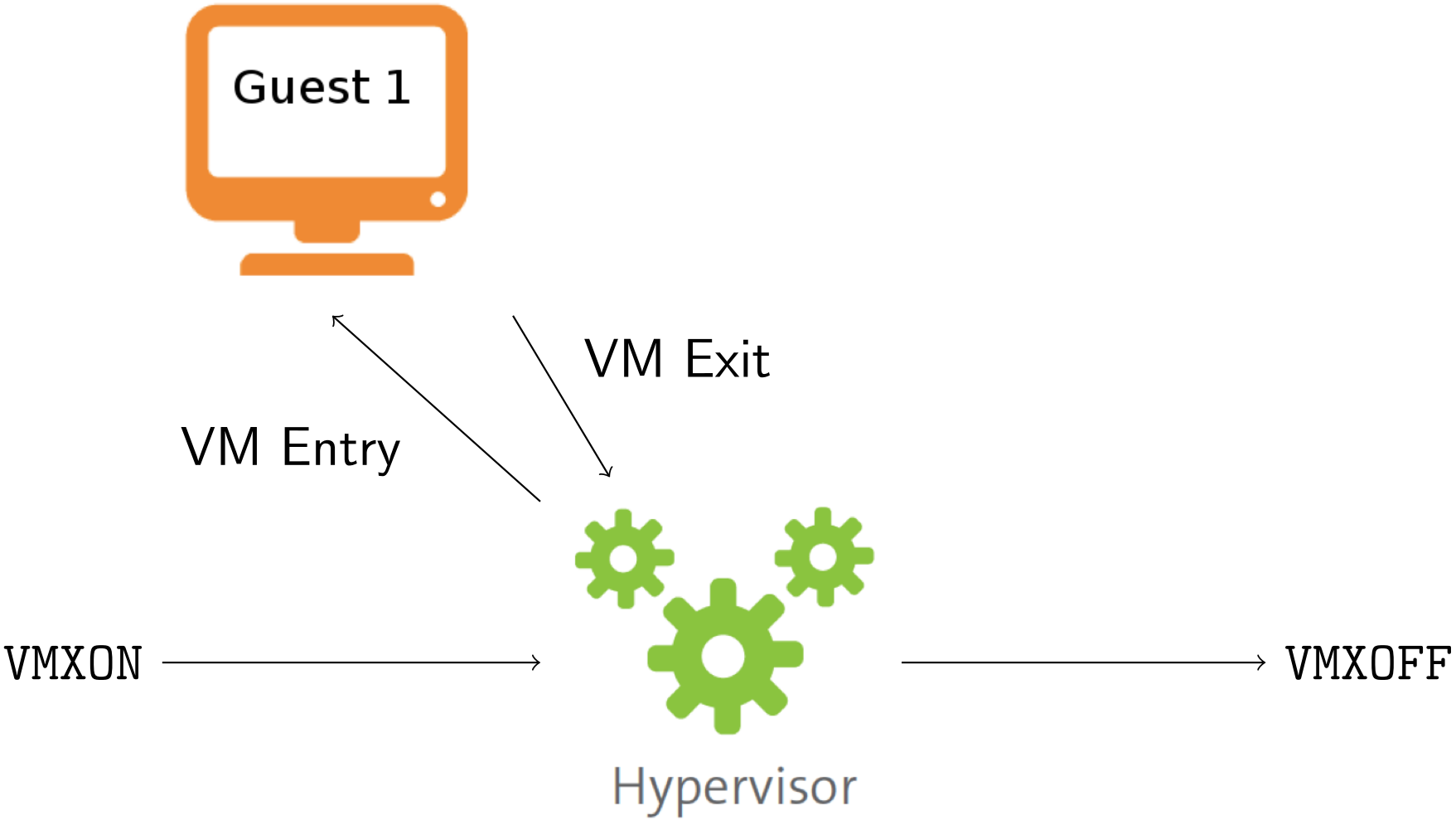
Ciclul de viață al unui hipervizor

VMXON ⟶

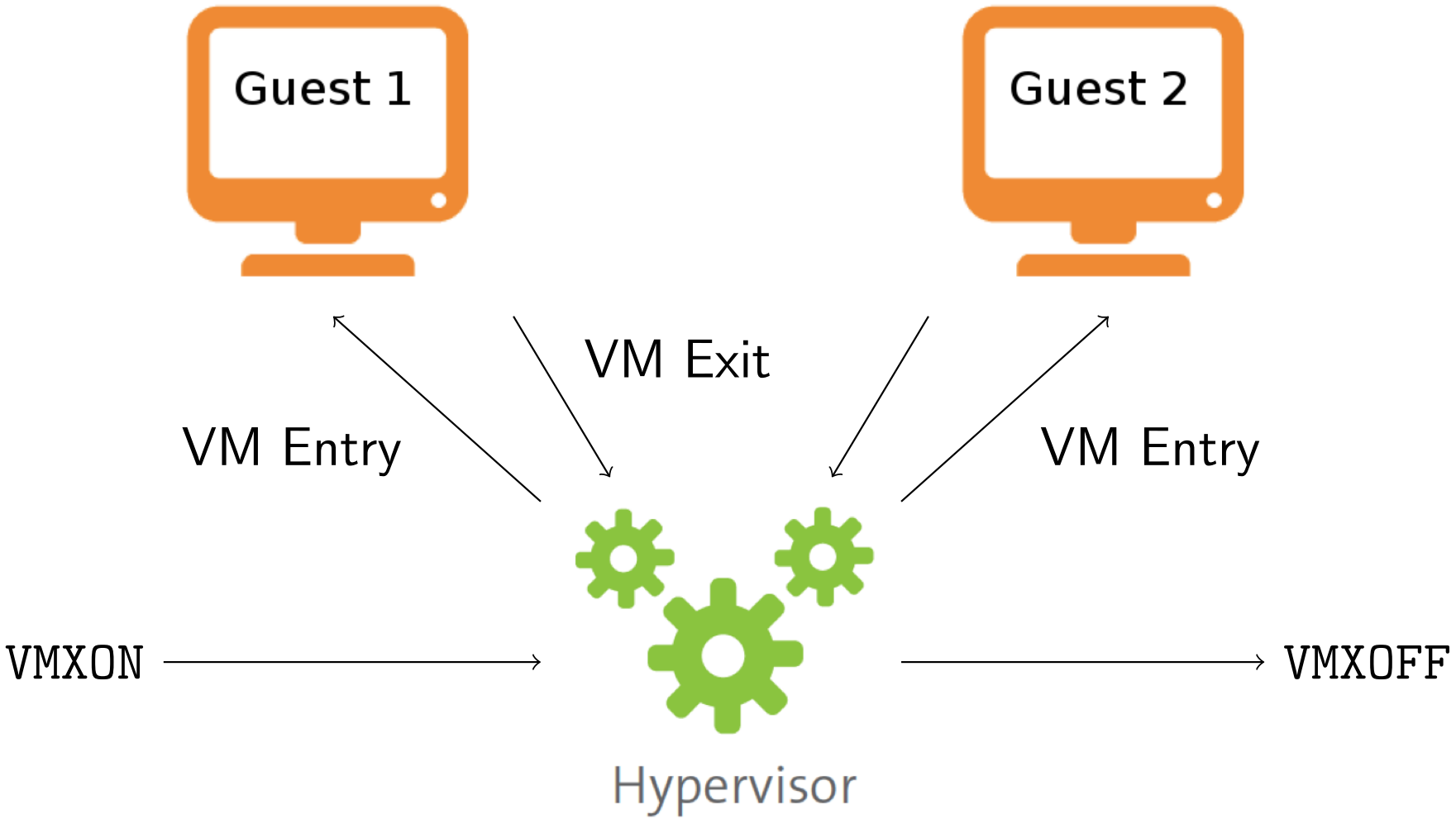
Hypervisor

Ciclul de viață al unui hipervizor

Ciclul de viață al unui hipervizor



Ciclul de viață al unui hipervizor



Bitdefender

Fragmente de cod hipervizor în Assembly

Găsirea adresei curente de memorie:

```
call $+5
pop ebp
```

Fragmente de cod hipervizor în Assembly

Găsirea adresei curente de memorie:

```
call $+5
pop ebp
```

Scriere de cod independent de poziție:

```
mov eax,[my_var] → mov eax, [ebp+my_var]
```

Bitdefender

Fragmente de cod hipervizor în Assembly

Găsirea adresei curente de memorie:

```
call $+5
pop ebp
```

Scriere de cod independent de poziție:

```
mov eax,[my_var] → mov eax, [ebp+my_var]
```

Setarea tabelelor IDT și GDT:

```
lidt    [ebp + bootIdt]

lea     eax, [ebp + bootGdt.tableStart]
mov     [ebp + bootGdt.base], eax
lgdt    [ebp + bootGdt]
```

Fragmente de cod hipervizor în Assembly

Găsirea adresei curente de memorie:

```
call $+5
pop ebp
```

Scriere de cod independent de poziție:

```
mov eax,[my_var] → mov eax, [ebp+my_var]
```

Setarea tabelelor IDT și GDT:

```
lidt    [ebp + bootIdt]

lea     eax, [ebp + bootGdt.tableStart]
mov     [ebp + bootGdt.base], eax
lgdt    [ebp + bootGdt]
```

Trecerea de la 32 la 64 bit:

```
;enable PAE
mov     eax, cr4
or      eax, CR4_PAE
mov     cr4, eax
;set the LME bit in EFER
mov     ecx, IA32_EFER
rdmsr
or      eax, IA32_EFER_LME
wrmsr
;activate paging
mov     eax, [ebp + bootCtx.Cr3]
mov     cr3, eax
mov     eax, cr0
or      eax, 0x80000000 ;PG bit
mov     cr0, eax
```


Vă mulțumesc pentru atenție!
ASM