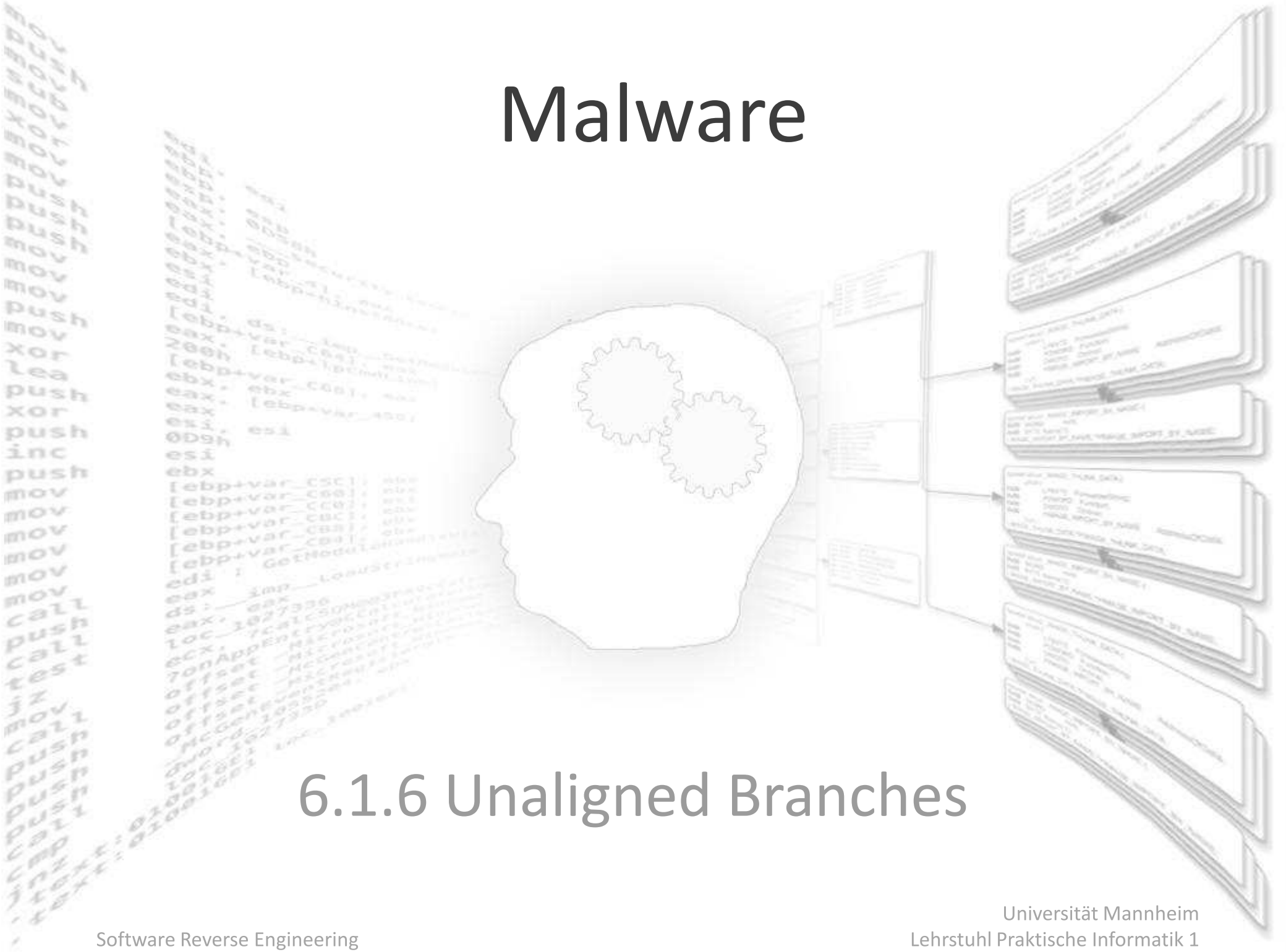
Malware



6.1.6 Unaligned Branches

Unaligned Branches

- Was macht folgender Codeausschnitt?

```
:004000F0                jmp     short near ptr loc_4000F2+2
:004000F2 ; ---------------------------------------------------------------------------
:004000F2
:004000F2 loc_4000F2:                            ; CODE XREF: start↑j
:004000F2                imul    eax, [edx+ebp*2+11C6840h], 2EB0040h
:004000FD                imul    eax, dword ptr loc_400122[eax+ebp*2], 2EB9090h
:00400108                imul    eax, [edx+ebp*2+1EE800h], 2EB0000h
:00400108 start          endp ; sp-analysis failed
:00400108
:00400113                imul    eax, [edx+ebp*2+19E800h], 61540000h
:0040011E                db      64h
:0040011E                popa
:00400120                and     [eax], eax
:00400122
```

- Erster Jump springt mitten in Maschinencode der folgenden Instruktion
- Wird Code der Reihe nach disassembliert, so wird völlig anderer Kontrollfluss dargestellt

Unaligned Branches

- „Korrektes“ Disassembly:

```
004000F0                jmp     short loc_4000F4
004000F0 ; ---------------------------------------------------------------------------
004000F2                db  69h ; i
004000F3                db  84h ; ä
004000F4 ; ---------------------------------------------------------------------------
004000F4
004000F4 loc_4000F4:                            ; CODE XREF: start↑j
004000F4                push    40h             ; uType
004000F6                push    offset Caption  ; "Tada!"
004000FB                jmp     short loc_4000FF
004000FB ; ---------------------------------------------------------------------------
004000FD                db  69h ; i
004000FE                db  84h ; ä
004000FF ; ---------------------------------------------------------------------------
004000FF
004000FF loc_4000FF:                            ; CODE XREF: start+B↑j
004000FF                push    offset Text     ; "Hello World!"
00400104                nop
00400105                nop
00400106                jmp     short loc_40010A
00400106 ; ---------------------------------------------------------------------------
00400108                db  69h ; i
00400109                db  84h ; ä
0040010A ; ---------------------------------------------------------------------------
0040010A
0040010A loc_40010A:                            ; CODE XREF: start+16↑j
0040010A                push    0               ; hWnd
0040010C                call    MessageBoxA
00400111                jmp     short loc_400115
```

Unaligned Branches

- Maschinencode ist bei **x86** generell **unaligned**
  - Maschinencode muss **nicht** an einer durch 32bit teilbaren Adresse beginnen
- *Idee*: Maschinencode von Instruktion A in Maschinencode von Instruktion B **verstecken** oder beide **überlappen** lassen
- *Effekt*:
  - Statische Analysesoftware wird verwirrt, Disassembly spiegelt nicht mehr den tatsächlichen Kontrollfluss wider
  - Entstehender Kontrollfluss ist nur **mühselig** zu **debuggen**
- *Ziel*: Möglichst **verwirrenden** Kontrollfluss erzeugen

Unaligned Branches

- Guter Disassembler löst Kontrollfluss richtig auf
- *Problem*: Statisch ist dies oft **schwer**, wenn nicht gar **unmöglich**
- Einfaches Beispiel:

```
.text:00401020                  push    (offset loc_401028+1)
.text:00401025                  pop     ecx
.text:00401026                  jmp     ecx
.text:00401028 ; ---------------------------------------------------------------------------
.text:00401028
.text:00401028 loc_401028:                              ; DATA XREF: .text:00401020↑o
.text:00401028                  imul    edi, [eax+12345678h], 0C7B9h
.text:00401032                  add     [edx+0E813h], dh
```

- **Sprungziel** wird erst zur **Laufzeit** berechnet
- In Realität oft sehr viel komplexer…
  - Siehe Einführungsvorlesung Unterscheidung Code/Daten