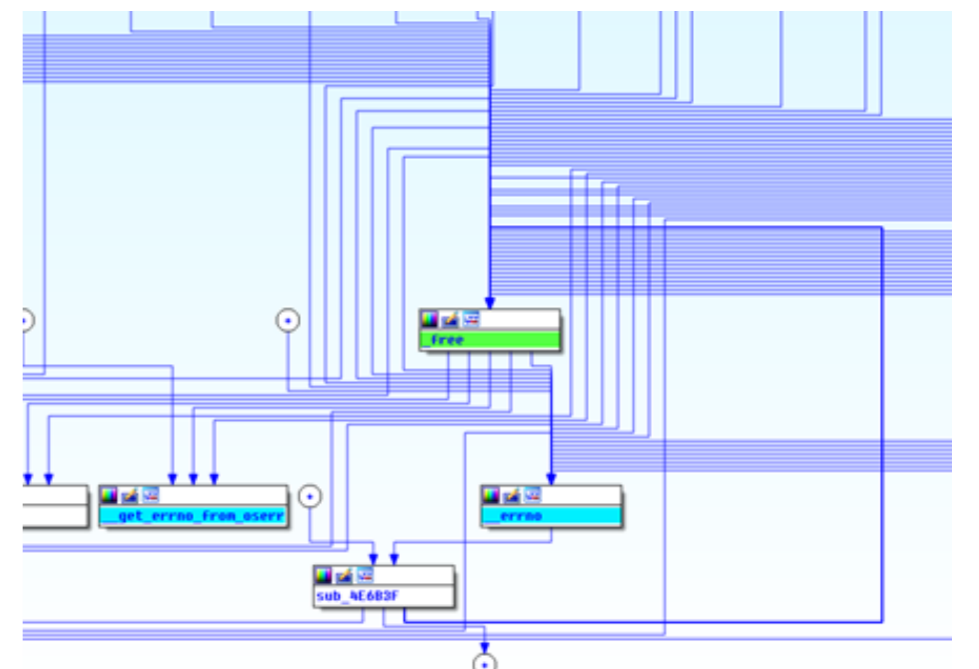


MAŠĪNKODA INSTRUMENTĒŠANA BLOKU IZPILDES STATISTISKAJAI ANALĪZEI UN PAREDZĒŠANAI

Kursa dara pirmajā daļā tika aprakstīti dažādi mašīnkoda apgrieztas inženierijas pielietojumi (autortiesību aizsardzība, komercnoslēpuma aizsardzība, ievainojamību meklēšana programmatūrā, pirmkoda atgūšana). Tālāk tika detalizēti aprakstīti soļi binārās analīzes veikšanai. Soļu apraksta laikā uzsverot un parādot cik svarīga ir programmas bloku izpildes noteikšana – kāda būs nākamā funkcija? Kāds būs nākošais koda lēciens? Ko šāda informācija dod binārās analīzes veicējam katrā no apgrieztās inženierijas pielietojumiem?

Kursa darba otrajā daļā tika aplūkotas, kādas ir šobrīd pieejamas metodes binārās analīzes ātruma uzlabošanai: heuristiski rīki, atklūdotāji, simbolu faili. Pēc apskates tika aprakstīta un piedāvāta jauna metode: bloku izpildes statistiskā analīze un paredzēšana. Veicot bloku analīzi ir iespējams paredzēt programmas darbību, atvieglojot izpēti un ātrāk iegūstot apgrieztās inženierijas pielietojuma rezultātus. Tika apskatīti arī citi šobrīd pieejami risinājumi, un tie salīdzināti ar oriģināli piedāvāto.

```
.def __main: .scl 2: .type 32: .endif
.section .rdata,"dr"
LC0:
.ascii "type a word \0"
LC1:
.ascii "%c \0"
.text
.globl __main
.def __main: .scl 2: .type 32: .endif
__main:
pushl %ebp
movl %esp, %ebp
andl $-16, %esp
addl $-128, %esp
call __main
moub $97, 127(%esp)
movl $0, 120(%esp)
movl $LC0, (%esp)
call _puts
jmp L2
```



The screenshot shows the IDA Pro interface. On the left, there is a control flow graph with nodes and edges. In the center, the assembly code is displayed, showing instructions like 'mov esi, edi', 'cmp [eax], rdi', and 'call cs:_imp_SetConsoleTextAttribute'. On the right, the pseudocode is shown, providing a higher-level view of the code's logic, including conditional statements like 'if (v5 >= 10)' and 'if (v4 == -1)'. The interface also shows various toolbars and windows, typical of a debugger.

