

Curs

→ Curs 1 - 24.02.2023

↳ detalii administrative

↳ materiale pe Teams

→ $N = 60\% \text{ lab} + 40\% \text{ proiect final}$

taskuri de reverse engineering complexe
poate fișiere binare primite

↳ pt. trecere >5 la ambele părți + fără plagiat

→ laboratoarele sunt obligatorii 7/9 (max. 2 absențe)

→ ca setup:

↳ mesurimi virtuale

→ python, C

→ pt. proiectul 0

↳ cerințe pe site

→ pt. a trimite cheia → form la sf. săptămânii pe Teams

→ cerințe + archive pe site

→ pași care trebuie urmăriți:

↳ scrie cod pt. algoritmi

→ testează cu datele date ca test

→ note cheia și verifica de JSON-ul este corect

→ Curs 2 - 03.03.2023

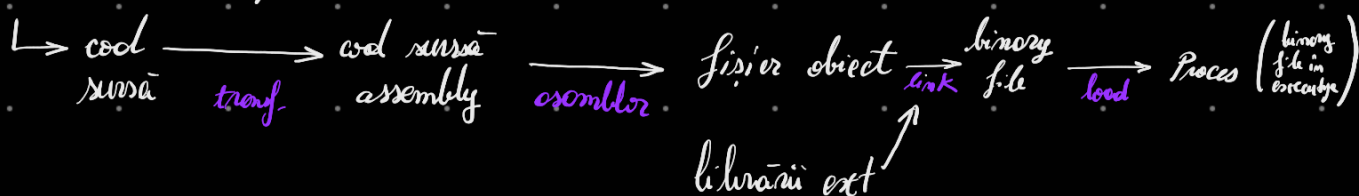
↳ Tehnici de analiză a fișierelor binare

↳ black box ^{→ need to create this one for dissection} → nu se poate "desface", poți doar obs. interacțiunile cu SO, librării, alte binare etc.

→ white box → analiză pe assembly + cu ce interacționează

→ gray box → white box + black box

→ proces de compilare



→ objdump -d fișier_compilat - cu cod maxim compact

↳ odr. la care se află codul

→ codul mașină

→ codul assembly

→ un output al com. objdump nu se poate colita cu un alt.

olărmuit (nu sunt toate chs. reprezentabile)

Task: editează fișierul binar simplu (cu printf) pt. a afișa
altcun

→ fișiere binare

↳ conțin cod mașină, nu assembly

→ codul mașină este înconjurat de headers + alte info
pt. a forme fișier binare:

↳ headers $\left\{ \begin{array}{l} \rightarrow \text{ELF} \rightarrow \text{linux} \\ \rightarrow \text{PE} \rightarrow \text{windows} \\ \rightarrow \text{WASM} \rightarrow \text{web (netir)} \end{array} \right.$

DOOM3 a fost portat
pe WASM = web assembly

// cum ştie procesorul să se oprească într-un frânt
 // limor? Teoria de lg. variabilă a lui Shannon
 // codare cu lg. var. \rightarrow pt. e scurta codării şerilor
 // în fet. de frecvenţa apariţiei + fiecare are decodare
 // unică (codarea scurtă \rightarrow în fet. de modelul arhitecturii pt. că
 // aceste coduri sunt înţelese de hardware)

\rightarrow cod assembly

↳ cod mai rapid

\rightarrow de la cod mizeră $\xrightarrow{\text{imposibil}}$ cod

\rightarrow ACC: registers

↳ SSE / SSE2 \rightarrow SIMD (?) \rightarrow to much SSE, SSE2, SIMD
 single instruction, multiple data

\rightarrow GPR, RIP, RSP, RBP
 \uparrow \uparrow
 stack pointer Base pointer

\rightarrow stack = memoria unui program

↳ are dim. fixă, heap-ul are memorie dinamică

\rightarrow ACC: arithmetic and logic

MOV RAX, 0 : $\text{rax} = 0$

XOR RAX, RAX : $\text{rax} = \text{rax} \wedge \text{rax}$



↳ $rax \rightarrow 64b. \Rightarrow$ mov rax funcționază corect de pui^{te}
vel. de 64b \Rightarrow codificarea ei este mult mai lungă
decît $xor\ rax, rax$ deoarece transformă 0 în 64b
(cum ea este inutil), iar $xor\ rax, rax$ nu face
nici o codificare

