

CTRL + F intrebarile (daca aveti ceva ce nu e pe docs, adaugati aici TEXT)

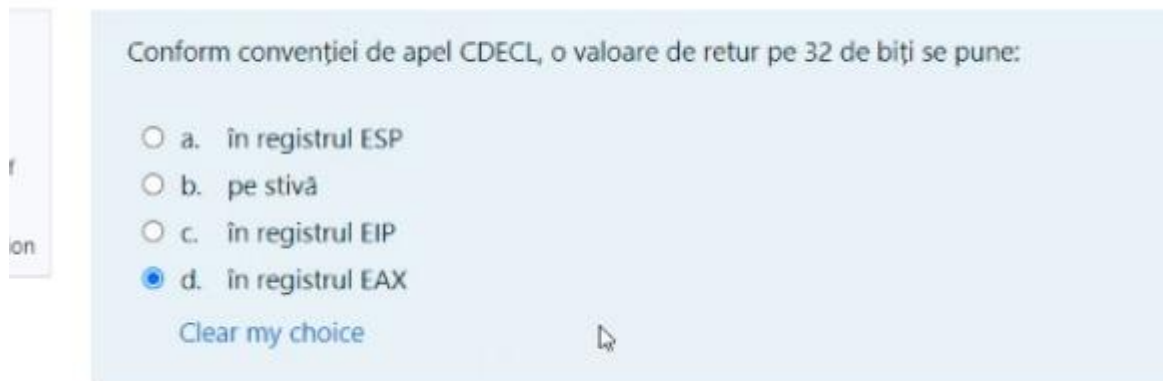
GRESIT

CORECT

POATE

Dați si voi un search înainte sa repetati întrebarea.

Conform convenției de apel CDECL, o valoare de retur pe 32 de biți se pune:



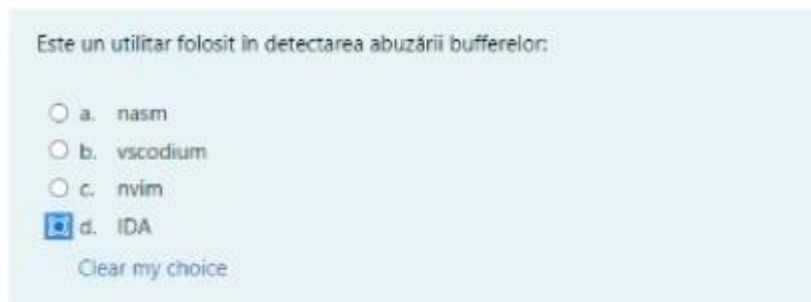
Conform convenției de apel CDECL, o valoare de retur pe 32 de biți se pune:

- ☐ a. în registrul ESP
- ☐ b. pe stivă
- ☐ c. în registrul EIP
- ☒ d. în registrul EAX

[Clear my choice](#)

d) în registrul EAX

Este un utilitar folosit în detectarea abuzării bufferelor:



Este un utilitar folosit în detectarea abuzării bufferelor:

- ☐ a. nasm
- ☐ b. vscodeium
- ☐ c. nvim
- ☒ d. IDA

[Clear my choice](#)

d

Folosim payloadul `64 * "A" + "\x23\x57\x04\x04"` pentru a exploata o vulnerabilitate de tipul buffer overflow și pentru a suprascrie adresa de retur a funcției cu valoarea `0x04045723`. La ce adresă (relativ față de `ebp`) se găsește bufferul având în vedere că avem un sistem de 32 de biți?

Folosim payloadul `64 * "A" + "\x23\x57\x04\x04"` pentru a exploata o vulnerabilitate de tipul buffer overflow și pentru a suprascrie adresa de retur a funcției cu valoarea `0x04045723`. La ce adresă (relativ față de `ebp`) se găsește bufferul, având în vedere că avem un sistem pe 32 de biți?

- ☐ a. `ebp-60`
- ☐ b. `ebp-68`
- ☐ c. `ebp-56`
- ☐ d. `ebp-64`

În urma unui out of bounds(OOB) pe stivă:

În urma unui out of bounds (OOB) pe stivă:

- ☐ a. Întotdeauna programul generează Segmentation Fault
- ☐ b. Execuția programului nu este afectată
- ☒ c. Se poate face disclose sau overwrite la adresa de retur
- ☐ d. Stiva se redimensionează

[Clear my choice](#)

Se dă funcția cu număr variabil de parametri `"int magic(int n, ...)"` și apelul `"magic(3, 20, 14, 6)"`. Știind că adresa de retur se află la adresa `Y`, la ce adresă se află parametrul 14 pe o arhitectură de x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

Se dă funcția cu număr variabil de parametri `"int magic(int n, ...)"` și apelul `"magic(3, 20, 14, 6)"`. Știind că adresa de retur se află la adresa `Y`, la ce adresă se află parametrul 14 pe o arhitectură x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

- ☒ a. `Y + 12`
- ☐ b. `Y + 4`
- ☐ c. `Y + 8`
- ☐ d. `Y + 16`

[Clear my choice](#)

Ce payload poate fi folosit pentru a rescrie adresa de retur a unei funcții cu valoarea `0x01234567` (exploatând o vulnerabilitate de tipul buffer overflow), știind că un buffer începe la adresa `ebp-40`?

Ce payload poate fi folosit pentru a rescrie adresa de retur a unei funcții cu valoarea `0x01234567` (exploatând o vulnerabilitate de tipul buffer overflow), știind că un buffer începe la adresa `ebp-40`?

- ☐ a. `"Z"*48 + "\x67\x45\x23\x01"`
- ☒ b. `"Z"*40 + "\x01\x45\x23\x67"`
- ☐ c. `"Z"*44 + "\x67\x45\x23\x01"`
- ☐ d. `"Z"*40 + "\x01\x23\x45\x67"`

[Clear my choice](#)

c) z*44

În cazul în care avem două fișiere: a.c și b.asm și dorim să obținem un executabil din cele două surse, ce utilitare vor fi folosite în acest proces?

În cazul în care avem două fișiere: a.c și b.asm și dorim să obținem un executabil din cele două surse, ce utilitare vor fi folosite în acest proces?

- ☐ a. Asamblor și compilator.
- ☐ b. Nu se poate obține executabil mixt.
- ☐ c. Compilator și linker.
- ☒ d. Compilator, asamblor și linker.

[Clear my choice](#)

Fie următorul cod: `char buffer[100]; fgets(buffer, len, stdin);` Ce proprietate trebuie să aibă len pentru a NU se realiza buffer overflow?

Fie următorul cod: `char buffer[100]; fgets(buffer, len, stdin);` Ce proprietate trebuie să aibă len pentru a NU se realiza buffer overflow?

- ☐ a. `len > 100`
- ☐ b. `len = 100`
- ☐ c. `len <= 100`
- ☐ d. Nu există o condiție pentru valoarea lui len

a) `len > 100`

Un buffer începe la adresa ebp-92. Cât este offsetul necesar pentru a suprascrie adresa de retur, pe un sistem pe 32 de biți?

Un buffer începe la adresa ebp-92. Cât este offsetul necesar pentru a suprascrie adresa de retur, pe un sistem pe 32 de biți?

- ☐ a. 88
- ☐ b. 104
- ☐ c. 100
- ☒ d. 96

[Clear my choice](#)

96

Se da următoarea declarație a unui buffer: `char buf[64];`. Care dintre următoarele variante NU poate genera un buffer overflow?

Se dă următoarea declarație a unui buffer: `char buf[64]`. Care dintre următoarele variante NU poate genera un buffer overflow?

- ☒ a. `read(STDIN_FILENO, buf, 30)`
- ☐ b. `fgets(buf, 128, stdin)`
- ☐ c. `memcpy(buf, info, 256)`
- ☐ d. `write(STDOUT_FILENO, buf, 72)`

[Clear my choice](#)

a

Ce date despre procedura curentă NU sunt puse pe stivă pe o arhitectură x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

Question 4
Not yet answered
Marked out of 1.00
Flag question

Ce date despre procedura curentă NU sunt puse pe stivă pe o arhitectură x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

- ☐ a. variabilele locale
- ☐ b. parametrii
- ☐ c. vechiul esp
- ☐ d. adresa de retur

C. vechiul esp

Pe arhitectura x86 (32 de biți), în urma apelării unei funcții scrise în limbaj de asamblare, din cadrul unui program C, parametrii de pe stivă sunt eliberați de:

Time left 0:09:22

1

Pe arhitectura x86 (32 de biți), în urma apelării unei funcții scrise în limbaj de asamblare, din cadrul unui program C, parametrii de pe stivă sunt eliberați de:

- ☐ a. în mod implicit de către codul generat de compilator
- ☒ b. programatorul trebuie să specifice
- ☐ c. se eliberează toți la finalul programului
- ☐ d. funcția chemată

[Clear my choice](#)

a) In mod implicit de catre codul generat de compilator

Cu ajutorul payloadului `16 * "A" + "\x56\x34\x12\x07"` este suprascrisă adresa de retur a unei funcții (exploatare a unei vulnerabilități de tipul buffer overflow). La ce adresă relativă față de registrul `ebp` se afla bufferul? Am auzit că **ebp-12**

Cu ajutorul payloadului `16 * "A" + "\x56\x34\x12\x07"` este suprascrisă adresa de retur a unei funcții (exploatând o vulnerabilitate de tipul buffer overflow). La ce adresă relativă față de registrul `ebp` se află bufferul?

- ☐ a. `ebp-20`
- ☐ b. `ebp-8`
- ☐ c. `ebp-12`
- ☐ d. `ebp+4`

Care dintre următoarele poate fi considerat un DEZAVANTAJ al analizei dinamice?

Question 4

Not yet answered

Marked out of 1.00

Flag question

Care dintre următoarele poate fi considerat un DEZAVANTAJ al analizei dinamice?

- ☐ a. nu poate fi realizată pentru orice aplicație (inflexibilitate)
- ☐ b. nu se poate realiza analiză dacă n-avem acces la codul sursă
- ☐ c. nu poate detecta vulnerabilități apărute la runtime (de exemplu segmentation fault)
- ☐ d. în general, acoperirea (coverage) este mai redusă decât în cazul analizei statice

d) în general, acoperirea

Care va fi rezultatul în urma compilării, linkării și rulării următoarei secvențe de cod, în cazul în care nu mai există niciun alt modul?

Care va fi rezultatul în urma compilării, linkării și rulării următoarei secvențe de cod, în cazul în care nu mai există niciun alt modul?

```
#include <stdio.h>
extern int x;
int main(void) {
    x++;
    printf("%d", x);
    return 0;
}
```

- ☐ a. Programul va compila și va afișa o valoare aleatoare în funcție de ce se afla la adresa variabilei `x`.
- ☐ b. Programul nu va compila deoarece variabila `x` este doar declarată, nu și definită.
- ☐ c. Programul va compila și va afișa valoarea 1.
- ☐ d. Programul nu va compila deoarece variabila `x` este doar definită, nu și declarată.

b?

Fie semnătura de funcție `"void func(int a, int b, int c)"`. La ce offset față de `ebp` se afla parametru `a`, respectiv `c` pe un sistem pe 32 de biți?

Question 3

Not yet answered

Marked out of 1.00

Flag question

Fie semnătura de funcție `"void func(int a, int b, int c)"`. La ce offset față de `ebp` se află parametru `a`, respectiv `c` pe un sistem pe 32 de biți?

- ☐ a. -8, -16
- ☐ b. +4, +12
- ☐ c. +8, +16
- ☐ d. -4, -12

+8, +16

Care dintre următoarele unelte este potrivită pentru a vizualiza codul binar și cel de asamblare al unui executabil?

Question 2

Not yet answered

Marked out of 1.00

Flag question

Care dintre următoarele unelte este potrivită pentru a vizualiza codul binar și cel de asamblare al unui executabil?

- ☐ a. nm
- ☐ b. readelf
- ☒ c. objdump
- ☐ d. strings

[Clear my choice](#)

c

Care din urmatoarele este considerata o metoda de protectie impotriva atacurilor de tipul buffer overflow?

Question 5

Not yet answered

Marked out of 1.00

Flag question

Care din următoarele este considerată o metodă de protecție împotriva atacurilor de tipul buffer overflow?

- ☐ a. Valori de tip canar (canary values)
- ☐ b. Address Space Layout Randomization
- ☐ c. Toate răspunsurile sunt corecte
- ☐ d. Bounds checking

Reprezinta o metoda de protectie impotriva overflow:

Reprezintă o metodă de protecție împotriva overflow:

- ☐ a. Information leak
- ☐ b. Canari
- ☐ c. Memory disclosure
- ☐ d. OOB (out of bounds)

B?

- probabil

https://en.wikipedia.org/wiki/Buffer_overflow_protection

Buffer overflow protection - Wikipedia

A **canary** example — Random XOR **canaries** have the same vulnerabilities as random **canaries**, except that the "read from stack" method of getting the **canary** is a ...

[Overview](#) · [Canaries](#) · [Tagging](#) · [Implementations](#)

Care dintre urmatoarele reprezinta o unealta de analiza dinamica?

Question 2
Not yet answered
Marked out of 1.00
Flag question

Care dintre următoarele reprezintă o unealtă de analiză dinamică?

- ☐ a. strings
- ☐ b. gcc
- ☐ c. gdb
- ☐ d. readelf

Gasit sub alta forma:

Care dintre următoarele utilitare este folosit pentru analiză dinamică? **GDB**

Un buffer incepe la adresa ebp-92. Cat este offsetul necesar pentru a suprascrie adresa de retur, pe un sistem pe 32 de biti? Am auzit ca **96**

Question 4
Not yet answered
Marked out of 1.00
Flag question

Un buffer începe la adresa ebp-92. Cât este offsetul necesar pentru a suprascrie adresa de retur, pe un sistem pe 32 de biti?

- ☐ a. 96
- ☐ b. 100
- ☐ c. 88
- ☐ d. 104

Ce date despre procedura curenta sunt puse pe stiva pe o arhitectura x86 pe 32 de biti, pe un program care foloseste sintaxa de apel cdecl?

Question 1
Not yet answered
Marked out of 1.00
Flag question

Ce date despre procedura curentă sunt puse pe stivă pe o arhitectură x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

- ☐ a. variabilele globale
- ☐ b. parametrii
- ☒ c. vechiul esp
- ☐ d. ebp-ul funcției apelate

[Clear my choice](#)

Cand poate aparea o eroare de tipul OOB (out of bounds)?

Question 4
Not yet answered
Marked out of 1.00
Flag question

Când poate apărea o eroare de tipul OOB (out of bounds)?

- ☐ a. Când nu s-a apelat ret în cadrul unei funcții
- ☒ b. Când se fac operații dincolo de limita bufferului
- ☐ c. Când se face bounds checking
- ☐ d. Când programul nu afișează rezultatul așteptat

[Clear my choice](#)

b) Cand se fac operatii dincolo de limita bufferului

Limbajul C permite folosirea functiilor cu numar variabil de parametri.

Limbajul C permite folosirea funcțiilor cu număr variabil de parametri.

- ☒ a. Adevărat
- ☐ b. Fals

[Clear my choice](#)

a. adevarat

Este un utilitar folosit in detectarea abuzarii bufferelor:

Question 2

Not yet answered

Marked out of 1.00

[Flag question](#)

Este un utilitar folosit in detectarea abuzării bufferelor:

- ☐ a. nasm
- ☐ b. IDA
- ☐ c. nvim
- ☐ d. vscodeium

IDA

Care dintre urmatoarele reprezinta un dezavantaj al folosirii limbajul

Question 1

Not yet answered

Marked out of 1.00

[Flag question](#)

Care dintre următoarele reprezintă un dezavantaj al folosirii limbajului de asamblare, comparat cu folosirea unui limbaj de nivel mai înalt?

- ☐ a. acces la operații low-level
- ☒ b. mentenanța
- ☐ c. slab control asupra memoriei programului
- ☐ d. memoria ocupată de cod

[Clear my choice](#)

Se dau urmatoarele declaratii `char buf[32]`, `char long_string[256]`. Care dintre urmatoarele apeluri de functii poate conduce la un buffer overflow?

Se dau următoarele declarații `char buf[32]`, `char long_string[256]`. Care dintre următoarele apeluri de funcții poate conduce la un buffer overflow?

- ☐ a. `fputs(long_string, sizeof(buf), stdout)`
- ☐ b. `memcpy(buf, long_string, sizeof(long_string))`
- ☐ c. `read(STDIN_FILENO, buf, 31)`
- ☐ d. `memcpy(buf, long_string, sizeof(buf))`

memcpy(buf, long_string, sizeof(long_string))

Se da urmatoarea declaratie a unui buffer: `char buf[64]`. Care dintre urmatoarele variante NU poate genera un buffer overflow? Am auzit ca `read(STDIN_FILENO, buf, 30)`

Question 4

Not yet answered

Marked out of 1.00

[Flag question](#)

Se dă următoarea declarație a unui buffer: `char buf[64]`. Care dintre următoarele variante NU poate genera un buffer overflow?

- ☐ a. `memcpy(buf, info, 256)`
- ☐ b. `read(STDIN_FILENO, buf, 30)`
- ☐ c. `fgets(buf, 128, stdin)`
- ☐ d. `write(STDOUT_FILENO, buf, 72)`

În convenția de apel CDECL pe x86 (32 de biți), care dintre următoarele NU este o sarcină ce ține de apelant (caller)?

În convenția de apel CDECL pe x86 (32 de biți), care dintre următoarele NU este o sarcină ce ține de apelant (caller)?

- ☐ a. conservă EAX, ECX, EDX (dacă este cazul)
- ☐ b. alocă și eliberează variabilele locale
- ☐ c. plasează parametrii pe stivă
- ☐ d. curăță stiva după retur

a) Conserva EAX

Alegeți varianta corectă:

Alegeți varianta corectă:

- ☐ a. Bufferele conțin informații temporare
- ☐ b. Tipul elementelor unui buffer este relevant
- ☐ c. Bufferele sunt simple variabile întregi
- ☐ d. Bufferele nu pot fi alocate dinamic

Bufferele conțin informații temporare

People also ask :

Is buffer a temporary memory?

In computer science, a data buffer (or just buffer) is a region of a physical memory storage used to **temporarily store data** while it is being moved from one place to another.

Conform convenției de apel CDECL, o valoare de retur pe 64 de biți se pune:

Conform convenției de apel CDECL, o valoare de retur pe 64 de biți se pune:

- ☒ a. în EDX:EAX
- ☐ b. în ESP
- ☐ c. nicio variantă nu este corectă
- ☐ d. în EDX:EBX

Clear my choice

a. În EDX:EAX

Reprezintă o metoda de protecție împotriva overflow:

Reprezintă o metodă de protecție împotriva overflow:

- ☐ a. Information leak
- ☐ b. Canari
- ☐ c. Memory disclosure
- ☐ d. OOB (out of bounds)

Care dintre următoarele este potrivita pentru a vizualiza codul binar si cel de asamblare al unui executabil?

Care dintre următoarele unelte este potrivită pentru a vizualiza codul binar și cel de asamblare al unui executabil?

- ☐ a. strings
- ☒ b. objdump
- ☐ c. readelf
- ☐ d. nm

[Clear my choice](#)

Care dintre următoarele este o secvență inline assembly

Care dintre următoarele este o secvență inline assembly ce produce o eroare la compilarea programului C?

- ☐ a.

```
__asm__ ("mov ebx, %1\n\t"
"mov eax, %0\n\t"
"sub eax, ebx\n\t"
: "r" (val)
: "r" (no)
: "ebx"
);
```
- ☐ b.

```
__asm__ ("mov %0, eax\n\t" : "=r" (val));
```
- ☐ c.

```
__asm__ ( "add eax, ebx\n\t"
: "=a" (res)
: "a" (arg1), "b" (arg2)
);
```
- ☐ d.

```
int no = 100, val ;
__asm__ ("mov ebx, %1\n\t"
"mov %0, ebx\n\t"
: "=r" (val)
: "r" (no) : "ebx"
);
```

b

Care etapa a compilarii permite obtinerea unui executabil mixt din doua sau mai multe fisiere obiect? (C-ASM)

Care etapă a compilării permite obținerea unui executabil mixt din două sau mai multe fișiere obiect? (C-ASM)

- ☐ a. asamblare
- ☐ b. acest lucru nu este posibil
- ☐ c. link-editare
- ☐ d. compilare

link-editare

Cu ce directiva trebuie declarat, in limbaj de asamblare, label-ul unei proceduri ce va fi ulterior chemata intr-un program C?

Cu ce directivă trebuie declarat, in limbaj de asamblare, label-ul unei proceduri ce va fi ulterior chemată într-un program C?

- ☐ a. nu este nevoie de label
- ☒ b. global
- ☐ c. asm
- ☐ d. extern

[Clear my choice](#)

Extern? E mai probabil

Care dintre următoarele variante de raspuns este adevarata cu privire la directiva de asamblare global?

Care dintre următoarele variante de răspuns este adevărată cu privire la directiva de asamblare global?

- ☐ a. Este obligatoriu să specificam tipul etichetei.
- ☐ b. Directiva global marchează o etichetă accesibilă și din alte module ale programului.
- ☐ c. Directiva global se folosește doar pentru variabile.
- ☐ d. Directiva global se folosește doar pentru nume de proceduri.

B, directiva global marcheaza...

Care afirmatie este adevarata in cazul unui buffer overflow?

Care afirmație este adevărată în cazul unui buffer overflow?

- ☐ a. Se poate modifica fluxul normal de execuție al programului
- ☐ b. Se pot citi informații dincolo de limita bufferului
- ☐ c. Toate variantele sunt corecte
- ☐ d. Pot fi afectate informații dintr-un alt stack frame

C, toate variantele sunt corecte

Care dintre următoarele reprezintă un dezavantaj al folosirii limbajului de asamblare, comparat cu folosirea unui limbaj de nivel mai înalt?

Care dintre următoarele reprezintă un dezavantaj al folosirii limbajului de asamblare, comparat cu folosirea unui limbaj de nivel mai înalt?

- ☐ a. memoria ocupată de cod
- ☐ b. slab control asupra memoriei programului
- ☐ c. acces la operații low-level
- ☐ d. mentenanța

d) mententanta

În urma unui out of bounds (OOB) pe stivă:

În urma unui out of bounds (OOB) pe stivă:

- ☐ a. Întotdeauna programul generează Segmentation Fault
- ☐ b. Execuția programului nu este afectată
- ☒ c. Se poate face disclose sau overwrite la adresa de retur
- ☐ d. Stiva se redimensionează

Clear my choice

C. probably

Alegeti varianta corecta:

Alegeți varianta corectă:

- ☒ a. Bufferele conțin informații temporare
- ☐ b. Tipul elementelor unui buffer este relevant
- ☐ c. Bufferele sunt simple variabile întregi
- ☐ d. Bufferele nu pot fi alocate dinamic

a) Bufferele contin informatii temporare

Se da funcția cu număr variabil de parametri ... magic

Se dă funcția cu număr variabil de parametri "int magic(int n, ...)"" și apelul "magic(3, 20, 14, 6)". Știind că adresa de retur se află la adresa Y, la ce adresă se află parametrul 14 pe o arhitectură x86 pe 32 de biți, pe un program care folosește sintaxa de apel cdecl?

- ☒ a. $Y + 12$
- ☐ b. $Y + 4$
- ☐ c. $Y + 8$
- ☐ d. $Y + 16$

[Clear my choice](#)

Potrivit convenției CDECL, pe arhitectura x86 (32 de biți), parametrii unei funcții se plasează:

Potrivit convenției CDECL, pe arhitectura x86 (32 de biți), parametrii unei funcții se plasează:

- ☐ a. în registre consacrate
 - ☒ b. pe stivă, în ordinea inversă a transmiterii argumentelor
 - ☐ c. pe stivă, în ordinea transmiterii argumentelor
 - ☐ d. în registrele aflate libere la acel moment
-
-
