Inżynieria wsteczna - zadania kwalifikacyjne

21. Wakacyjne Warsztaty Wielodyscyplinarne

1. Funkcja (2 pkt)

Poniższy kod przedstawia funkcję zapisaną w asemblerze x86-64 w konwencji wywołań System V AMD64: (czyli takiej, jaka jest używana w Linuxie na x86-64)

```
mov rax, rdi
mov rcx, rsi
and rax, rcx
not rax
mov rdx, rdi
not rdx
not rcx
and rdx, rcx
sub rax, rdx
shr rax, 1
and rdi, rsi
add rax, rdi
retn
```

Jako rozwiązanie należy podać równoważny kod w C lub C++. Dodatkowo, należy przekształcić te obliczenia do prostszej formy - powyższa funkcja liczy coś prostego, ale trochę na około. Rozwiązanie należy dokładnie (ale zwięźle) opisać - co, skąd i dlaczego.

2. Błąd kompilatora? (2 pkt)

Kompilując poniższą funkcję C++:

```
#include <cstdint>

uint32_t mod_1337(uint32_t a) {
   return a % 1337;
}
```

używając kompilatora g++ 15.1 na architekturę x86-64 otrzymamy następujący kod asemblera:

```
mod_1337(unsigned int):
        mov edx, edi
        mov eax, 2284010283
        imul rdx, rax
        mov eax, edi
        shr rdx, 32
        sub eax, edx
        shr eax
        add eax, edx
        shr eax, 10
        imul edx, eax, 1337
        mov eax, edi
        sub eax, edx
        ret
```

(patrz https://godbolt.org/z/en5zfW13n).

Wygenerowany kod asemblera nie zawiera w sobie instrukcji dzielenia ani modulo. Dlaczego? Czy jest to błąd w kompilatorze? Czy powyższy kod jest poprawny? Dlaczego kompilator wygenerował go w ten sposób?

Jako rozwiązanie należy odpowiedzieć na powyższe pytania.

3. Czytnik obrazków (6 pkt)

W zadaniu dany jest program `img_reader`, który wyświetla obrazki w nieznanym formacie `.img` oraz przykładowy obrazek `mnt.img`. Celem zadania jest zrozumienie formatu używanego przez program i obrócenie `mnt.img` zgodnie z ruchem wskazówek zegara o 90 stopni.

Tak jak w poprzednich zadaniach, rozwiązanie należy dokładnie (ale zwięźle) opisać - co, skąd i dlaczego. Pamiętajcie, że im lepiej ponazywacie zmienne i funkcje tym mniej będzie trzeba dodatkowo opisywać.

Przykład uruchomienia programu:

```
chmod +x img_reader
./img_reader mnt.img
```

Na systemach z glibc starszym niż 2.38 należy użyć załączonych bibliotek:

```
chmod +x ./ld-linux-x86-64.so.2
LD_LIBRARY_PATH=. ./ld-linux-x86-64.so.2 ./img_reader mnt.img
```

Program i biblioteki:
https://drive.google.com/drive/folders/1h52cMl4UcZJxWJnAyqG-CNrIj_pzDKqq.

Jako rozwiązanie należy przesłać:

- Program generujący obrócony `mnt.img`.
- Zwięzły, ale dokładny opis formatu (może być albo osobny, albo jako komentarze w kodzie).

Podpowiedzi i uwagi:

- Do rozwiązania nie jest wymagana analiza kodu maszynowego z programu `img_reader`, można go potraktować jako black-box - tylko zmieniać plik wejściowy i oglądając rezultaty zrozumieć format danych.
- Najprościej będzie użyć języka Python. Przydadzą się funkcje `struct.pack` oraz `struct.unpack`. Dodatkowo, za pomocą biblioteki Pillow można pomóc sobie w analizie formatu rysując dane z pliku.
- Format używa kompresji, ale dość prostej i stworzonej na potrzeby tego zadania.
- Zewnętrzna warstwa kompresji to *deflate*, jest wiele bibliotek które potrafią go zdekompresować (np. zlib). Ale uwaga! Nie oznacza to, że w zadaniu wystarczy po prostu wrzucić cały plik od tak do dekompresora deflate i tyle.

Przydatne materiały

W rozwiązywaniu powyższych zadań pomóc mogą:

- Opis działania instrukcji x86: https://www.felixcloutier.com/x86/.
- Netwide Assembler: https://nasm.us/.
- Dowolny edytor szesnastkowy.

Wszelkie pytania do zadań i samych zajęć proszę kierować na adres redford@dragonsector.pl.