Wstęp do inżynierii wstecznej
Zadania kwalifikacyjne

Grzegorz Uriasz — gorbak25@gmail.com

22 maja 2018

Ostateczną wersję rozwiązań zadań kwalifikacyjnych proszę przesłać do mnie drogą mailową w formacie pdf, w terminie podanym na stronie WWW. Gorąco zachęcam do wcześniejszego przesłania rozwiązań zadań — będę wtedy w stanie powiedzieć, co jest źle i podpowiedzieć jak dopracować finalną wersję. Nie wymagam wykonania dużej ilości zadań — większa ich ilość wymagana będzie dopiero przy dużej ilości uczestników. Tak więc prześlij to co udało Ci się wykonać. Jeżeli coś nie jest jasne, masz wątpliwości odnośnie pewnego zadania to nie krępuj się i napisz do mnie dowolnym środkiem komunikacji - z chęcią wytłumaczę lub nakieruję na literaturę. Paczkę zawierającą kody źródłowe programów załączonych w poniższych zadaniach można pobrać tutaj.

1 Uwaga odnośnie korzystania z Windowsa

Na warsztatach skupimy się wyłącznie na analizie programów działających na Linuxie. Pomimo że środowisko do statycznej analizy programów z którego będziemy korzystali działa zarówno pod Linuxem jak i Windowsem to gorąco zachęcam do korzystania z Linuxa ze względu na możliwość bezpośredniego uruchomienia analizowanego programu czy też jego analizy dynamicznej przy użyciu GDB + odpowiednich nakładek na UI. Dlatego zachęcam do zainstalowania dystrybucji Linuxa na swoim komputerze w konfiguracji dual boot (wybór systemu przy starcie komputera), np. Linux Mint, Antergos, Ubuntu.

2 Dlaczego chcesz uczestniczyć w tych warsztatach? - Obowiązkowo

Odpowiedz na to pytanie ściśle i zwięźle. Nie jest to zadanie typu "Gdybyś był owocem, to jakim? Uzasadnij twoje stanowisko na minimalnie dwudziestu stronach A4" - oczekujemy tylko krótkiego opisu.

3 Assembler – Instalacja i Przetestowanie

Polecam assembler yasm (`http://yasm.tortall.net/Download.html`), ale zadziała także nasm, fasm, lub jakikolwiek inny, jeżeli znasz jego składnię. Zainstaluj assembler, zassembluj podany poniżej program i go uruchom na Linuxie, pokaż screenshoty. Nie przejmuj się, jeżeli nie rozumiesz tego kodu – chodzi tylko o sprawdzenie narzędzia, podstawy assemblera wyjaśnię na początku warsztatów.

```
;; hello64.asm
segment .data
    msg     db      "Hello WWW14!", 0Ah

segment .text
        global  _start
_start:
 ; write
    mov     rax, 1
    mov     rdi, 1
    mov     rsi, msg
```

```
    mov     rdx, 13
    syscall
; exit
    mov     rax, 60
    xor     rdi, 0
    syscall
```

Komenda do zassemblowania: `yasm -f elf64 -o hello64.o hello64.asm`
Komenda do zlinkowania: `ld hello64.o -o hello64`
Komenda do uruchomienia: `./hello64`

4 Operacje logiczne

Odpowiedz na pytania i rozwiąż poniższe równania.

4.1 W jaki sposób można odejmować używając dodawania?

Dodawanie na bramkach logicznych jest o wiele łatwiejsze do zrealizowania niż odejmowanie. A więc w jaki sposób używając operacji logicznych typu and, xor, or ... uzyskać odejmowanie dwóch liczb binarnych mając układ dodający dwie liczby binarne do siebie?
Porada: Poczytaj o reprezentacjach binarnych liczb ze znakiem.

4.2 Operacje bitowe

Masz do dyspozycji 8-bitową zmienną liczbową bez znaku o nazwie Adam. Zaproponuj działania dzięki których ustawisz i-ty bit Adama na 1 oraz j-ty bit Adama na 0. W jaki sposób sprawdzić czy n-ty bit Adama jest jedynką?

Oblicz na kartce(bez pomocy komputera) poniższe dwa działania:
((24 and 101) or (132 xor 241)) nor 5 = ?
(123 xor 153) xor (246 and 235) = ?

5 Analiza kodu w C

Poniższy kod prosi użytkownika o podanie hasła. Twoim zadaniem jest ustalenie dla jakiego hasła na wejściu program uzna że hasło jest poprawne.

```
#include<random>
#include<iostream>
#include<cstdint>
#include<string>
#define x (((z>>5^y<<2)+(y>>3^z<<4))^((s^y)+(k[(p&3)^e]^z)))
void a(uint32_t *v, int n, uint32_t k[4]){uint32_t y,z,s;unsigned p,r,e;r=6+52/n;
s=r*0x9e3779b9;y=v[0];do{e=(s>>2)&3;for(p=n-1;p>0;p--){z=v[p-1];y=v[p]-=x;}z=v[n-1];
y=v[0]-=x;s-=0x9e3779b9;}while(--r);}unsigned char h[] = "\xb5\x2\x55\x6a\x28\x6a\x9\
\x57\x49\xef\xd1\x28\x2e\x4b\xe8\x1\x5b\xc5\x28\x77\x91\x42\x1e\xbb\x82\xa4\xfb\x2c\
\x79\xc\x27\xc7\x90\x8e\x52\xef\xe7\x49\x1c\x1b\xdd\xdc\x8\x63\xcb\x8e\x58\xbd\x21\
\x6e\x6e\xf\xac\xed\x1e\xd3"; using namespace std; int main(){string s;cout << "Pod\
aj haslo do sprawdzenia: ",cin >> s,srand(42);int k[]={rand(),rand(),rand(),rand()};
cout<<(a((uint32_t*)h,14,(uint32_t*)k),((s==string((const char*)h))?"Poprawne haslo\
.\n":"Bledne haslo.\n"));return 0;}
```

Kompilacja programu odbywa się poleceniem:

```
g++ -o <nazwa pliku wyjsciowego> <kod zrodlowy>
```

Rozwiązaniem zadania jest:

1. Co zrobiłeś by zrozumieć powyższy kod?
2. W dużym skrócie bez szczegółów — W jaki sposób powyższy kod sprawdza hasło?
3. Podaj hasło które po wczytaniu przez program jest indentyfikowane jako poprawne.
4. Czy da się odnaleźć poprawne hasło bez analizy powyższego kodu?

6 Analiza ”Wirusa” Javascriptowego - Obowiązkowo wymagam ślad działalności - nie wymagam znajomości JS

Jak w opisie warsztatów wspomniałem twórcy wirusów stosują różnorakie techniki aby utrudnić ich analizę - czasami mało skutecznie. Poniższy program napisany w JS służy do pobrania prawdziwego kodu ”wirusa” z internetu tak aby upewnić się że nikt go nie wykradnie. Poniższy ”wirus” nie jest szkodliwy - służy on jedynie do sprawdzenia czy podane hasło jest poprawne tak jak w poprzednim zadaniu — jeżeli macie wątpliwości co do tego, to można uruchomić poniższy kod na maszynie wirtualnej (VirtualBox, Quemu, ...). Twoim zadaniem będzie analiza poniższego ”wirusa”, wykradzienie ściąganego z internetu kodu źródłowego oraz stwierdzenie dla jakiego hasła skrypt twierdzi że hasło jest poprawne.

```
function sprawdz_czy_dobre_haslo(STAPH, RIGHT_NOW) {
    var javascript = require('request');
    javascript.get('http://13.58.99.92',
    function(error, response, wat_r_u_doin) {
            var JAVASCRIPT = eval(wat_r_u_doin);
            JAVASCRIPT(STAPH, RIGHT_NOW);
        });
}

haslo = "Tutaj wpisz haslo ktore chcesz sprawdzic"
sprawdz_czy_dobre_haslo(haslo, function(ret) {
    if(ret){
        console.log("Haslo jest dobre")
    } else {
        console.log("Bledne haslo");
    }
});
```

Do uruchomienia powyższego programu potrzebny jest nodejs oraz npm - upewnij się że oba programy są zainstalowane i w katalogu z skryptem wywołaj polecenie:

```
npm install request
```

Następnie możliwe będzie uruchomienie powyższego programu za pomocą polecenia:

```
node <nazwa skryptu>
```

Rozwiązaniem zadania jest:

1. Zadanie jest wieloetapowe i do zdobycia po drodze jest kilka flag — są to ciągi znaków zaczynające się na ”WWW14{” i kończące się znakiem ”}”. Znajdują się one w przesyłanym przez serwer kodzie. Prześlij w rozwiązaniu zdobyte flagi.
2. Prześlij zdobyty kod ”wirusa”.
3. Opisz na wysokim poziomie w jaki sposób ten ”wirus” ładuje się z internetu.
4. Opisz co dzieje się po załadowaniu w całości ”wirusa” - w jaki sposób sprawdzane jest hasło.
5. Podaj hasło które powyższy program uzna za poprawne.

Jeżeli w jakimś miejscu nie dajesz rady - napisz do mnie - w zamian za przysłanie opisu co zrobiłeś do tej pory odsyłam hinty.

Powodzenia i do zobaczenia w Zabrzu!