IT Security Coding Contest 2010

2010. 12. 03.

Általános szabályok:
A verseny során bármilyen segédeszközt használhattok a feladatok megoldásához.

A feladatok megoldását csak abban az esetben fogadjuk el, ha az alábbi anyagokat is megkapjuk:

- A megoldó program vagy programok használatának leírása, valamint a futtatásukhoz szükséges előfeltételek (operációs rendszer, futtató környezet (.net, jre,...), stb.)
- A megoldó program vagy programok bináris (futtatható) formában – működőképesen! (Amennyiben van valamilyen függőség, azokat a dll-eket, könyvtárakat, stb. is kérjük!)
- A megoldó program vagy programok forráskódja
- A fordításhoz szükséges információk (használt fordító megnevezése, pontos verziója, használt kapcsolók, stb.)
- A megoldás elvét leíró dokumentáció
- A csapat tagjainak nevét tartalmazó txt fájl

Minden egyes feladat megoldását (azaz a fenti anyagokat mind) külön csomagolt (zip/rar/tar.gz,…) állományban várjuk.

Az értékelés során általános irányelvként figyelembe vesszük a megoldás eredményességét, hatékonyságát, valamint megvalósítás igényességét. Részpontokat is adunk, így érdemes félkész megoldásokat is beadni részletes leírással.

1. Feladat - Statisztikai alapú kódtörő program készítése - (90p)

Az Magyar Titkosszolgálat egy olyan program készítését tűzte ki célul, ami a lehető legnagyobb hatékonysággal képes visszafejteni egy tetszőleges nyelv egy legalább 10000 karakteres, betűnként kódolt szövegét a forrásnyelv (beszélt nyelv, nem programozási) ismeretében. A Titkosszolgálat egy olyan programot vár, aminek

- első paramétere, egy a forrásnyelven íródott legalább 10000 karakter hosszúságú, txt formátumban tárolt értelmes szöveget tartalmazó fájl,
- második paramétere egy legalább 10000 karakter hosszúságú betűnként kódolt, txt formátumban tárolt szöveget tartalmazó fájl,
- harmadik paramétere pedig a dekódolt kimeneti fájl elérési útja legyen.

Paraméterezés:
`StatDecrypt language_sample.txt encrypted.txt <kimeneti fájl elérési újta>`

A feladat megoldását statisztikai alapokon történő betűazonosítás módszerével kérjük. A *brute force* elvű megoldásokat nem tudjuk elfogadni.

Segédadatok:

- **Letölthető**: www.hackerverseny.hu → a Dokumentumok menü alatt
- **Fontos:** A csomagolt fájlban az *encrypted.txt* dokumentumban a sorvége jeleket nem kódoltuk. (ASCII = 10) A dokumentum több, mint 30000 karaktert tartalmaz. A Robinson Crusoe regény bizonyos fejezeteit kódoltuk le.
- A *robinson_crusoe.txt* fájl több, mint 20000 karaktert tartalmaz, így tehát használható a betanítás alapjául, de természetesen nem kötelező az átadott anyagokat használni a tesztelésre.

Pontozás:

- Algoritmus hatékonysága, gyorsasága szerint
- Nem a megadott anyagokkal, általunk generált más fájlokkal történő tesztelés után az eredeti szövegtől mért hamming-távolság függvényében.
- A megoldás elvét leíró dokumentáció alapján

2. Feladat - RSA dekódolás – (150p)

Adott egy angol nyelven íródott ASCII formátumú szöveg, amit RSA-val kettő, maximum N db számjegyből álló prímszámot használva byte-onként kódoltak le. Tudjuk, hogy a kódolt szöveget tartalmazó, *.rsa* kiterjesztésű fájlban egy kódolt karaktert 2N byte-on tárolnak számként. Tehát pl.: egy adott karaktert 2 db maximum 3 karakter hosszúságú prímmel kódolva 6 karakteren tárolódik. Ha a kódolás után generálódott szám helyiértékeinek száma kisebb, mint 2N, 0-kal balról kiegészítésre kerül és, mint szám íródik ki a fájlba. (pl.: ha az „a” karaktert egy 3-jegyű és egy 2-jegyű számmal kódoljuk, az eredmény „053223”-hez hasonló string lesz).

Feladat:

- **Letölthető**: www.hackerverseny.hu → a Dokumentumok menü alatt
- Dekódolja a kapott N2.rsa, N3.rsa, N4.rsa, N5.rsa, N8.rsa és N10.rsa fájlokat.
    - A fájlok nevében található szám a kódolásra használt prímek maximális helyiértékszámát határozza meg (maximum ennyi számjegyből állnak a kódolásra használt prímek – kevesebb lehet).
    - A fájl első sora a két prím szorzatát tartalmazza, ami a nyilvános kulcs egyik része.
    - A fájl második sora a kódolásra használt nyilvános kulcs másik része.
    - A fájl harmadik sorától a kódolt szöveg olvasható.
- A kapott információk alapján dekódolja a fájlokat.
    - Határozza meg a *p* és *q* prímszámokat, melyek szorzata adja a kulcs első részét (*n*).
    - Határozza meg a titkos kulcs második paraméterét is (*d*).
    - Minél nagyobb N-re oldja meg a feladatot.
- **Megjegyzés:** A kódolt szövegek angol nyelvű, az élettel kapcsolatos idézeteket tartalmaznak. Az eredeti szövegek tartalmazzák a „life”, vagy a „Life” szavak valamelyikét.
- **Fontos:** Prímszámokat tartalmazó letölthető táblázat nem használható. Az N-jegyű prímszámok előállítására a jegyek számával paraméterezhető segédprogram elkészítése szükséges, ami kimenetként egy txt-fájlba, soronként helyezi el a meghatározott prímeket. Ezt a kimenetet lehet paraméterként átadni a programnak.
- **A program paraméterezése:**
    - első paraméterként (vagy –if kapcsolóval) a bemeneti kódolt fájl
    - második paraméterként (vagy –n kapcsolóval) N értéke
    - harmadik paraméterként (vagy –p kapcsolóval) a prímeket soronként tartalmazó txt fájl
    - negyedik paraméterként (vagy –of kapcsolóval) pedig a dekódolt kimeneti text fájl elérési útja legyen.
- **A kimenet**
    - első sora *p*-t
    - második sora *q*-t
    - harmadik sora a titkos kulcs második paraméterét (*d*)
    - negyedik sora a dekódolt szöveget tartalmazza

Pontozás:

- Attól függően, mekkora N-ig sikerült dekódolni a fájlok tartalmát
- A prímelőállítás hatékonysága (algoritmus, gyorsaság)
- A faktorizáció-probléma megoldásának módja (algoritmus, gyorsaság)
- A titkos kulcs meghatározásának megoldási módszere
- Mivel 4 byte kevés nagyobb N-ek esetén a feladat megoldásához. Szükséges egy BigInteger/BigNum osztály használata is. Ennek megvalósításának módja is beleszámít a pontozásba.

Leadásra várjuk az egyes részfeladatok megoldási elvét leíró részletes dokumentációt, valamint a dekódolt fájlokat is.

3. Feladat – Log-adatok csoportosítása – (80 pont)

A központi naplózó szerverre tömegével érkeznek be különböző források, applikációk log üzenetei. A logserver azonos formátumra alakítja az üzeneteket, ezáltal az üzenetek szétválaszthatók változó (timestamp, dátum, idő, host, üzenet, stb.) és konstans részekre, melyek egésze a naplósor jelentéstartalmát adja.
Az üzemeltető nagyszámú eszköz mellett könnyen több millió log sorral találhatja szembe magát. A nagymennyiségű adat miatt, a sorok szűrése, és a sorok elemzése nehéz feladat. (A való életben pl. ilyen elemzőeszközökkel is figyelik a banki átutalásokat, majd gyanús tranzakció esetén a rendszer beállított szabályok szerint riasztást generál.)

Ehhez a tevékenységhez adhat hatékony kiindulási alapot az alábbi feltételeket teljesítő program megírása:

- A bemenetére adott logállomány sorait (bemenő paramétere a logokat tartalmazó text formátumú fájl neve) beolvassa, majd felis8émeri az azonos típusú üzeneteket.
- Ezeket típus szerint bontva megjeleníti a képernyőn az adott típusú sor előfordulásával és az adott csoportra történő szűrést megvalósító – automatikusan, a program által generált – **regexp** kifejezésekkel együtt.
- Plusz pont jár azért, ha az alkalmazás képes fő és altípusokra való bontásra is.

A program ezáltal lehetőséget ad, hogy könnyedén szűrjük a naplóállományt a számunkra kiemelt fontosságú sorokra, így azok könnyebb elérése hatékonyabb elemzést tesz lehetővé.

Forrásadat letölthető:

- **Letölthető**: www.hackerverseny.hu → a Dokumentumok menü alatt
- *Megj.:* a letölthető logállományok valós rendszerből származtatott anonym-izált adatok unix-os formátumban (a sorvége jelekre érdemes figyelni).
- Életszerű továbbá, hogy hibás sorok érkeznek be a logserver-re, hogy az üzenet nem egy sorban tárolódik.
- *Egy logsor felépítése:*
  ellenőrző összeg <whitespace> timestamp <ws> facility (milyen alrendszer adja fel a jogot) <ws> severity (az üzenet súlyossága) <ws> dátum <ws> idő <ws> host ip <ws> üzenet

A versenyző feladata a programot olyan általánossággal elkészíteni, hogy az bármilyen méretű, az itt leírt formátumban tárolt fájlra működjön abban az esetben is, ha a tesztelésre használt naplófájl bizonyos attribútumoknál (severity, facility, host ip, stb.) az átadott állományban nem szereplő értékeket tartalmaz.

Leadásra várjuk az algoritmus működését leíró dokumentumot is.

4. Feladat – Jogosultságalapú dokumentum-megosztó webalkalmazás - (100 pont)

Írjon egy olyan web-alkalmazást, amellyel dokumentumokat oszthat meg a regisztrált felhasználók közt. A dokumentumokat csak a bejelentkezett felhasználók láthatják, és tölthetik le, **autentikáció nélkül erre semmilyen módon ne legyen lehetőség**.
A program garantálja az adatok csak arra jogosult felhasználók általi hozzáférését! A védelem ne legyen megkerülhető! A védelmet a webalkalmazásnak kell megvalósítania!

Egy dokumentum lehet *privát*, amit csak az adott felhasználó lát, és *publikus*, amit minden bejelentkezett felhasználó láthat.
A rendszerbe két szerepkör létezzen: *user* és *admin*.

- az *admin* látja minden felhasználó *privát* és *publikus* dokumentumait is egyaránt
- *admin* jogosultsággal rendelkező felhasználó tölthet fel új dokumentumot (a saját nevében), és törölheti bármely (bármely user által feltöltött) dokumentumot.
- kizárólag az *admin* tud új felhasználót regisztrálni
- a *user* felhasználó lát minden *publikus* dokumentumot
- a *user* képes feltölteni új dokumentumot és törölheti az általa feltöltötteket
- a *privát* dokumentumokat csak a feltöltő személy és *admin*-ok láthatják

További kritériumok:

- a webalakalmazás nem használhat adatbázis rendszert
- nem használható semmilyen publikus keretrendszer
- nincs semmilyen konfigurációs jog a szerveren. (nem állítható be BASIC auth.)
- Nem telepíthető semmilyen külső modul, plugin a szerverre, azon kívül, amit az alap telepítő csomag tartalmaz.
- apache esetében nem használható htaccess
- csak HTTP vagy HTTPS protokollal érhetik el a felhasználók a rendszert
- a fileok HTTP-vel, vagy HTTPSel elérhető helyen tárolhatóak
- csak Adobe PDF, MS doc, és MS Excel fileok tölthetőek fel

Beadandó:

- Rövid telepítési útmutató, rendszerspecifikáció
- Itt is elvárjuk az általatok alkalmazott megoldás ismertetését, és annak indoklását miért ezt választottátok, szerintetek ez hogyan garantálja a kiírásban szereplő követelmények teljesülését. Valamint a megoldásotok mindennapi működéséhez milyen üzemeltetési, biztonsági előfeltételek szükségesek.
- Annak leírása, hogy a program miképpen garantálja, hogy az adatok csak autentikált felhasználók által férhetnek hozzá az adatokhoz? Miért nem lehet megkerülni a védelmet?

Pontozás:

- Az értékelésnél a fenti dokumentumok és a forrás alapján megállapítjuk milyen technológiai megoldásokkal, milyen szinten teljesíti a koncepciótok a „védelem ne legyen megkerülhető” feltételt.
- A kód átvizsgálása hatékonysági és biztonságtechnikai szempontok alapján.

5. feladat – Sztegano-hacking 2010 – (80 pont)

Látogass el az http://itscc10.pet-portal.eu címre, amely weboldalon többféle szteganográfiai módszerrel is rejtettünk információt. A cél, hogy az összes információmorzsát megtaláld, és a többféle rejtési algoritmus megfejtéséből összeálljon a teljes elrejtett adat. Segítségképpen eláruljuk, hogy a rejtett információ minden darabja 7 bites ASCII-kódolású. Felhívjuk továbbá arra is a figyelmet, hogy egyes algoritmusokat csalétekként alkalmaztunk, tehát a velük elrejtett információ nem képezi a megtalálandó titok részét. A csalétkek felfedezése és visszafejtése is pontot ér.
Részpontokat is adunk, érdemes a félig kész megoldásokat is beadni! Érdemes az ötleteiteket is leírni, hogyha nem vagytok biztosak, mert a részpontszámok erre is vonatkoznak!

Pontozás:

Rendre egyre több pont jár az előrehaladásért:

- rejtési algoritmus megállapítása
- algoritmus leírása
- valós/csalétek rejtett adatról van-e szó
- a rejtett adat megadása.