CYBER SECURITY

1. forduló

Continental The Future in Motion

A kategória támogatója: Continental Automotive Hungary Kft.

Ismertető a feladathoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

- MINDEN kérdésre **van helyes válasz**.
- Olyan kérdés **NINCS**, amire az összes válasz helyes, ha mégis az összes választ bejelölöd, arra a feladatra automatikusan 0 pont jár.
- A **radio button-os kérdésekre egy helyes válasz van**.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Azokat a feladatlapokat, amelyekhez **csatolmány** tartozik, javasoljuk **NEM mobilon** elindítani, erre az érintett feladatlapok előtt külön felhívjuk a figyelmet.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Helyezéseket a 4. forduló után mutatunk**, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.
- **Badge-ket** szintén a 4.forduló után kapsz majd először.
- Ha egyszerre több böngészőből, több ablakban vagy több eszközről megnyitod ugyanazt a feladatlapot, **nem tudjuk vállalni** az adatmentéssel kapcsolatban esetlegesen felmerülő anomáliákért a felelősséget!
- A hét forduló során az egyes kategóriákban (de nem feltétlenül mindegyikben) **könnyű-közepes-nehéz kérdésekkel** egyaránt találkozhatsz majd.

Jó versenyzést kívánunk!

1.forduló

Felhívjuk figyelmedet, hogy a kategória legtöbb fordulójában - sok témakörtől eltérően - egy komplex feladattal találkozhatsz majd. Ha esetleg a heti téma nehéznek bizonyul - amivel biztosan nem leszel egyedül - , ne add fel, gyere vissza a következő héten is!

A hashelés (~hesselés vagy magyarosan hasítás) annak a folyamata, hogy kulcsokhoz eltérő értékeket generálunk. Formálisan ez azt jelenti, hogy tetszőleges hosszúságú inputhoz, azaz bemeneti adathoz egyértelműen hozzárendelünk egy rögzített hosszú kimenetet az ún. hashfüggvény/hasítófüggvény segítségével.

Gyakorlati jelentősége például, hogy alkalmas kriptográfiai hashfüggvény mellett bár a kulcs (például egy jelszó) hashelt értéke könnyen kiszámítható, de az inverz művelet, azaz a hashelt értékből az eredeti kulcs visszaszámítása csak nagyon költségesen megvalósítható (dollárszázezrek és több millió processzoróra árán).

Ennek a praktikai irreverzibilitásnak, illetve az uniform méretnek köszönhetően a jelszóadatbázisokban a jelszavakat gyakran csak hashelt alakban tárolják. Ilyenkor a bejelentkezési próbálkozással a felhasználó által megadott jelszót gyorsan lehashelik, és a tárolt hash értékkel hasonlítják össze.

Ha az adatbázis esetleg kompromittálódik, a keletkezett baj is mérséklődik, de nem tűnik el.

A használt hashelési eljárás (pl. SHA-1) ismeretében a gyakori jelszavak „password” vagy „123456” hashelt értékei gyorsan kiszámíthatóak (itt ezek pl.: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 ill. 7c4a8d09ca3762af61e59520943dc26494f8941b). A

feltört adatbázisból kikereshetőek a felhasználók, akik mellett megfelelő hash értékek vannak tárolva, és így a fiókjukba is be lehet hatolni.

Példák:

Egyik legegyszerűbb példája a hashelésnek az úgynevezett multiplikatív hashelés. A **„k"** kulcsot (szöveges input esetén a kulcs bináris/hexadecimális reprezentációját) megszorozzuk egy rögzített konstanssal (**„a"**), majd vesszük a szorzat törtrészét. A kapott **{k·a}** egy kvázi véletlenszerű érték 0 és 1 között, ezt felszorozva a céltartomány méretével, „**m"**-el (pl. rögzített n bites választ várva m = 2n) kapjuk a **[m·{k·a}]** értéket hashként

(Mivel nincs garancia arra, hogy a szorzat egészértékű, ezért vettük még a kifejezés egészrészét)

Ennek a gyakorlati implementációja egy kicsit eltérő, hiszen egyrészt a lebegőpontos kettedestörtekkel való számolás pontatlanságokhoz vezethet túl hosszú mantisszák esetén. Helyette egy speciális fixpontos rendszert alkalmazunk, amit illusztrálandó olvasható a következő példa, ahol 32 bites inputot elfogadva egy 24 bites hasított értéket számítunk ki.

Legyen a példajelszavunk **k="Key"**, ennek ASCII-reprezentációja hexadecimálisan **0x4B6579**.

Ezt tízes számrendszerbe váltva 4941177-t kapunk.

A rögzített konstansnak vegyük a következő számot, **a32 = 2654435769.**

/Kitérő: Miért ezt a számot választjuk? Ez a szám nem más, mint 2^{32} / φ egészrésze, ahol φ az aranymetszés száma, azaz 1.61803.... A választás hátterében az áll, hogy ez a konstans „szórja szét" leghatékonyabban az egymást követő kulcsokat.

Hasonlóan a természetben is megfigyelhető, hogy növények 360° / φ azaz cirka 222,5/137,5°-os szögben egymáshoz növesztenek új leveleket, ezzel maximalizálva a befogott napfényt.

Az ilyen típusú konstanssal elvégzett multiplikatív hashelést Fibonacci-hashelésnek nevezzük.

Fontos megjegyezni, hogy ez nem egy kriptografikus hash, de egyszerűség okából most a feladatokban ezt fogjuk annak használni./

Az is fontos szempont, hogy ebben a példában kizárólag csak 32 bit méretű, előjel nélküli egész számokkal (uint32) dolgozunk; a szorzandót, a szorzót és átmenetileg a szorzatot is ilyen számokként kezeljük. Ebből adódóan, ha a szorzás eredménye esetleg nagyobb lenne, mint a legnagyobb uint32-ben reprezentálható szám, a túlcsordulás normális és elvárt, így a túlcsordult eredménnyel dolgozunk tovább. A túlcsordulást egy moduló művelettel szimulálhatjuk.

Elvégezve a szorzást, a kulcsunkat a konstanssal megszorozva és a túlcsordulást modulóval imitálva a következőket kapjuk:

$(4941177_{10} * 2654435769_{10}) \bmod 2^{32} = 1416725873_{10} = 0101\ 0100\ 0111\ 0001\ 1000\ 0101\ 0111\ 0001_2$

Az utolsó 8 bitet elhagyva a kapott szám $0101\ 0100\ 0111\ 0001\ 1000\ 0101_2$, amiből kapjuk a végső hashértéket: 0x547185

Felhasznált idő: 00:00/20:00 **Elért pontszám:** 0/10

1. feladat 0/10 pont

A Continental egy kollégája a fent bemutatott Fibonacci-hashelést a következő változtatásokkal implementálta:

- 32 bites helyett 64 bites inputot fogad el
- 24 bites helyett 48 bites hashértéket számol

A rendszert kipróbálandó a „Conti123" jelszót adta meg.

Helyes implementáció esetén mi lesz ennek a 48 bites eltárolt hexadecimális hashértéke?

Válasz

- ○ 0x6813E87C4643
- ◉ 0xB7B6A47B4517

Ez a válasz helyes, de nem jelölted meg.

- 0xC8A8E4443214
- 0x12B8E68C32EF

Magyarázat

Először számoljuk ki a jelszó hexadecimális reprezentációját. A jelszó karaktereinek az alábbiak az ASCII-tábla alapján az értékeik:

C 0x43

o 0x6F

n 0x6E

t 0x74

i 0x69

1 0x31

2 0x32

3 0x33

Ennek konkatenált értéke 0x436F6E7469313233, azaz 4859223969216148019_{10}

A példánál megismert szorzószám helyett most ki kell számolnunk a64 = [2^{64}/ φ]-t, ez [11400714819323198485,951] = 11400714819323198485.

(4859223969216148019_{10} * $11400714819323198485_{10}$) mod 2^{64} = 13237949004049273391

Ez binárisan $1011011110110110101001000111101101000101000101111101001000101111_2$

Ennek az utolsó 16 bitét levágva kapjuk: $1011011110110110101001000111101101000101000101 1_2$

Ez nem más, mint 201995071472919_{10} = **0xB7B6A47B4517**

Természetesen mindez a számolás elvégeztethető egy programmal is, egy példaimplementáció így nézne ki C++-ban:

```
//Constants
const unsigned long a_64 = 11400714819323198485;
const int q_64 = 48;

unsigned long FiboHash_64(unsigned long k)
{
    return (k * a_64) >> (64 - q_64);
}
```