Kriptográfia és biztonság

Mivel projektünkben intenzív hálózati kommunikációt használunk, és felhasználók adatait, jelszavait kezeljük, ezért kiemelten fontos a biztonság kérdése.

Alapvető tény, hogy felhasználók jelszavait nem szövegként kell tárolni egy adatbázisban, hiszen ha egy támadó fél hozzájutna az adatokhoz, akkor egyből tudná a bejelentkezési adatokat. Ennek elkerülése érdekében komplex, SHA512-es titkosítást használunk.

Ez egy olyan algoritmus neve, amely egy bemeneti karakterláncot látszólag véletlenszerűen átalakít egy hexadecimális értékké. Azonban ez a folyamat koránt sem véletlenszerű, hiszen ha a folyamatot többször hajtjuk végre ugyanazzal a bemenettel, mindig ugyanazt a hexadecimális eredményt fogjuk kapni, valójában egy egyértelmű hozzárendelésről beszélünk.

Minden eltérő bemeneti karakterlánc esetén egy teljesen más hexadecimális értéket (ún. hash-t) kapunk, ha ezekre ránézünk, kizárt dolog, hogy az eredeti jelszót megismerjük:

“House of Swords”: c8eb916e211f7e9062ec5a367bd4d756fea4a3d2462dc567b32a0fcfaf509050e71dacd61ba48296d3ce8b2bbe9fb558d47f5d38efb35294bb613d78981ecff9

“House of Sword”: f05d153fba12b96b5428ec75345ece77518af0bc9ba9afacd779cfcca76f1831fdd0d04639304ad8309c7d7cce15ddea4ef8d376b7af2068e8d1880b5ffde36a

Ahogy az jól látszik, ha a bemenet csak egy karakterrel tér el, már akkor is teljesen különböző eredményt kapunk.

Azonban ez önmagában nem teljesen megfelelő a véleményünk szerint, hiszen az egyszerűbb jelszavakat egy “Dictionary attack”, avagy egy “Szótár támadás” segítségével így is könnyedén fel lehet törni.

Ez ellen úgy tudunk védekezni, hogy a jelszó titkosítása előtt a bemenet végéhez ún. “Salt” és “Pepper” karakterláncokat fűzünk.

A “Salt” karakterlánc az egy (a mi esetünkben 20 karakter hosszú) véletlenszerűen generált, mindenféle karakterből álló lánc, amelyet a titkosított jelszó mellett tárolunk. A “Pepper” egy véletlenszerű, kis- vagy nagybetűs karakter az angol ábécéből, amit nem tárolunk el. Miután hozzáfűztük a jelszó végéhez ezt a két karakterláncot, az után következik a titkosítás, majd az eltárolás.

Ezzel azt érjük el, hogy ha a felhasználó egy egyszerűbb jelszót ad meg, ami szerepel egy adathalászok által ismert adatbázisban, akkor sem fogják tudni egyből feltörni, hiszen az adathalászok ezeket a jelszavakat és a hozzájuk tartozó hexadecimális hash-t tárolják, de ezek a szavak a “Salt” nélkül lettek titkosítva, így nem fognak egyezni.

A “Pepper” ugye nem került tárolásra, de akkor honnan tudjuk, hogy melyik karaktert fűztük a jelszó végére titkosítás előtt? Igazából nem tudjuk, erre a megoldást a “Brute-force” módszer jelenti, avagy minden esetben, amikor a felhasználó be akar jelentkezni, a beírt jelszó végére hozzáfűzzük a “Salt”-ot, és az így keletkezett szó végére egyesével kipróbáljuk az 52 lehetséges betűt, majd egyesével az 52 szót titkosítjuk. Ha valamelyik hexadecimális érték egyezik az adatbázisban lévővel, akkor tudjuk, hogy jó jelszót írt be a felhasználó, ellenkező esetben pedig téveset. Ez a módszer programidőben nem szignifikáns, hiszen egy karakterláncot összehasonlítani egy másikkal a másodperc töredéke alatt lehetséges, 52-vel megcsinálni ugyanezt pedig a másodperc 52 töredéke alatt lehet, ez még mindig elenyésző időben.

Ez úgy javít a biztonságon, hogy ha a támadó is “Brute-force” módszerrel szeretné feltörni a jelszót, akkor 52-szer több időbe fog neki telni, hiszen a végén lévő karaktert is fel kell törniük, amire 52 lehetőség van (a-Z).

Így a végleges jelszó, illetve a hozzátartozó hash valahogy így néz ki:

“House\_of\_Swords” + “1aGtPGDRvCDpShGZyjTR” + “M” = “House\_Of\_Swords1aGtPGDRvCDpShGZyjTRM”

f60578adeef9efe963b407822234334ed4895cf7f0d1c7ad9a83e4c0787e21c0c9410abe2345bc88583599c67ca3c8b1a6c66a597f4748a1736aef34a0f5605e