Bonus Reading - Hash Tables and Security

How Not to Store Passwords

1. Bad Solution 1: Plain text password
    - So hackable.
2. Bad Solution 2: Encryption [sha1(password), md5(password)]
    - Very open to brute force and rainbow tables.
    - Aynı şifrelere sahip birden çok kişi varsa bir şifre kırıldığında hepsi kırılır çünkü bütün encrpyted string'ler aynıdır.
3. Bad Solution 3: Encryption with Fixed Salt [sha1(FIXED_SALT + password)]
    - Fixed salt kullanmak oldukça riskli.
4. Bad Solution 4: Encryption with Per User Salt [sha1(PER_USER_SALT + password)]
    - Salt'ları database'de tutmak riskli.
5. Hashing
    - Basic bir hashing algorithm kullanıyorsak basit bir Google aramasıyla bile kırılabilir. Yine rainbow table saldırısına ve brute force'a açık.

How to Store Passwords

- Hashing + Salting.
- A salt is a random string, unique for each user. Hash'i oluştururken her bir kullanıcı için de tamamen random (dynamic secret mantığı) bir string oluştururuz. Dolayısıyla birileri aynı şifreyi kullansa bile hash aynı oluşmuyor. Bu yöntem bizi rainbow table saldırılarından korur fakat brute force'a hala açıktır. SALT'ları database'de yada ENV olarak tutmak güvensizdir.

Bcrypt

- Bcrypt şöyle çalışıyor:

```
hash(salt + just_entered_password)
```

- Örneğin ruby bcrypt hashing mekanızmasını kullanıyor. Bu MD5'e çok daha yavaş çalışmak üzere **hazırlanmış** bir algoritma. Avantajı ise: MD5'ı kırmaya çalışan bir saldırgan saniyede 150.000 şifreyi test edebilirken, bcrypt kullanarak şifre kırmaya çalışırsa saniyede sadece 500 şifre deneyebilecek. Ayrıca saldırganların bcrypt için rainbow table oluşturması gereken süre göz önünde bulundurulursa - mümkün değil.
- Peki bcrypt salt'ı random oluşturuyorsa şifreler nasıl kıyaslanıyor login esnasında? Yani random bir şeyi tekrar üretemeyiz sonuçta?
- BCrypt defines its own == method, which knows how to extract that "salt" value so that it can take that into account when comparing the passwords. Bcrypt is not an encryption algorithm, it is a hash algorithm. You cannot reverse a hash (1000 ve 100'ün mod'unun aynı şey olması örneği gibi). It is provably impossible.

```
$2a$12$8vxYfAWCXe0Hm4gNX8nzwuqWNukOkcMJ1a9G2tD71ipotEZ9f80Vu

$bcrypt_id$log_rounds$128-bit-salt184-bit-hash
```

- Bcrypt random oluşturduğu salt'ı hash'in içine koyuyor. Yani salt'ı store etmemize gerek yok. Brcryp'in check yapan fonksiyonu salt'ı hash içerisinden nasıl çıkartacağını bildiği için bizim salt'ları store etmemize gerek yok.
- Kaynak: http://dustwell.com/how-to-handle-passwords-bcrypt.html

Hash Collision

- Hash collision dediğimiz şeyle MD5 ve SHA1'ın kırılması aynı şey. İki farklı data için aynı hash üretiliyorsa bu collision durumu oluyor. Yani güvenli bir dosya ile virüslü bir dosyanın hash'i aynı üretilirse bu çok tehlikeli bir durum. Bilgisayarlar çok hızlandığı için artık bunu yapmak mümkün oldu ve hızlı bilgisayarların bir sonucu olarak MD5 kırıldı.