Veritabanında Parola Saklamak için Algoritma Seçiminde Yapılan Yaygın Hatalar

Klasik güvenlik anlayışının temelinde olduğu gibi bilgi güvenliği felsefesinde de tedbir almada iki temel çeşit vardır. Bunlardan biri; bir güvenlik ihlalinin gerçekleşmesinin önüne geçilmesi için alınan tedbir, diğeri ise bir güvenlik ihlalinin gerçekleştiği durumlarda hasarı en aza indirecek ve olası zincirleme ihlallerin önüne geçecek olan tedbirdir. Bu iki çeşit tedbiri somut bir örnekle anlatmak gerekirse; tedbir çeşitlerinden biri arabanın lastik basıncının, fren sisteminin, lastik diş derinliğinin kontrol edilmesi, diğeri ise hava yastıklarının konması ve sürücünün emniyet kemerini takmasıdır. İlk tedbirler kazanın olmasını engellemeye yönelik iken, ikinci tedbirler ise kaza olduktan sonra meydana gelebilecek can kayıpları ve yaralanmaları engellemeye yöneliktir.

Web uygulamasının güvenliğinin sağlanmasının yanı sıra, Web uygulamasına ait parolaların veritabanında saklanması ve bu saklama yönteminin güvenliği de oldukça önemlidir. Web uygulamasındaki veya bütün bilişim altyapısını oluşturan öğelerden birindeki bir açık kullanılarak güvenlik ihlalinin oluştuğu durumlarda, parolaları ele geçiren bir saldırganın bu bilgiyi kullanamamasını sağlamak için düz metine kıyasla güvenli olduğu bilinen kriptografik özet (Hash) yöntemleri kullanılır. Bunların başında gelen popüler algoritmalardan biri de günümüzde MD5 algoritmasıdır. Hash kullanmadaki amaç, ele geçen parolaların doğrudan kullanılmasını önlemektir. Ancak sadece MD5 ile Hash edilmiş parolalar dahi günümüzde yeterli güvenlik seviyesi sağlamamaktadır. Bu yüzden kırılması daha zahmetli ve zor olan Salt kullanma yöntemi ile parolalar depolanır. Parola depolama işlemi için Hash tek yönlü algoritması düşünülürken genel bir hata "hızlı" bir algoritma seçmektir. Hızlı bir algoritma ile Hash edilen parolalar yine Rainbow Table ve Brute Force işlemleri ile yavaş algoritmalara oranla daha hızlı kırılabilirler. Hızlı algoritmalar parola depolamak amaçlı değil, kriptografik işlemler için kullanılmalıdır. Ağ iletişimi içinde veri bütünlüğü tespiti için kullanılan algoritmalar (Checksum) buna örnek gösterilebilir.

Yazılımlarda parolaları saklamak için hızlı algoritma kullanmak, kimi zaman bu algoritmalara Salt değerleri ile yaklaşılsa dahi, saldırganların parola kırma işini etkili bir ölçüde zorlaştırmamaktadır. Parolaları saklamak için kullanılan algoritma az kullanılan, güçlü ve yavaş algoritmalardan seçildiğinde, kullanımın yaygın olmaması nedeniyle sistem altyapısının değişmesi ve yeni platformlara taşınması gerektiği durumlarda sorunlarla karşılaşılabilir. Bu durum da yine kullanılabilirlik-güvenlik terazisindeki kullanılabilirlik kefesini etkilemektedir. Az kullanılan algoritmalar için hali hazırda parola kırma programları ve Rainbow Table (bir çeşit önceden kırılmış parola değerleri) bulmak zor olduğundan, saldırganı çok kullanılan bir algoritmaya oranla daha çok mesai beklemektedir.

Bir saldırgan sistemde ele geçirdiği binlerce hesabı kırmak için hıza ihtiyaç duyar. Ancak, bir Web uygulamasına giriş yapmak isteyen bir kullanıcı için algoritmanın hızı kullanılabilirlik dengesini çok değiştirmez. Çünkü kullanıcı parolasını yazdığı anda, parola doğruysa bu işlemi sisteme 1 defa yaptırır. Fakat bir saldırgan bu işlemi duruma göre binlerce veya milyonlarca defa yapmaya mahkumdur.

Bu durumda parola saklamak için kullanılan Hash algoritmasının yavaş ve güçlü olması önerilmektedir. MD5 örneğine bakıldığında, MD5 yerine daha yavaş ve güçlü olduğu bilinen SHA-512 kullanmak, burada ifade edilen bilgilere göre daha güvenli bir seçim olacaktır. Bir parola Hash işlemine tabii tutulmadan önce içine birçok Salt değerinin ve tekrarlı (belki binlerce defa) iterasyonlarının sokulduğu Hash fonksiyonları yazılmalıdır. Böylece üst üste farklı şekillerde Hash işlemine tabii tutulan parolayı deneme yanılma yöntemleriyle geri elde etmek zorlaşacak ve algoritma yavaş bir hal alacaktır.

Deneme yanılma (Brute Force) yöntemlerinin yanı sıra “doğum günü saldırısı” adı altında, algoritmanın kendisine yönelik saldırılar da vardır. Bu saldırıların temeli 366 kişiden en az 2 kişinin doğum tarihlerinin aynı gün olmasına dayanır. Kriptografik özet çıkarma işleminde giren verinin boyutu ne olursa olsun elde edilen çıktı belli matematiksel limitlere dayanır. Bu da iki farklı girdinin aynı çıktıyı vermesine neden olabilir.

Aşağıdaki tablo Hash algoritmalarının hız ve doğum günü saldırısına karşı olan süseptibilitelerine dair bilgiler içermektedir.

Hash algoritmaları hız karşılaştırma tablosu.*

Algoritma	Çıktı boyutu (bit)	Kelime Boyutu (bit)	Bilinen Çakışma *(Doğum günü saldırısı)*	Göreceli Hız Örneği (MiB/s)
MD5	128	32	Var	255
SHA-0	160	32	Var	-
SHA-1	160	32	Teorik olarak (2^{51}) adet var.	153
SHA2-256/224	256/224	32	Yok	111
SHA2-512/384	512/384	64	Yok	99

Yaygın diğer algoritmalar ve hızları (MiB/s)*

CRC32	253
Adler32	920
Tiger	214
Whirlpool	57
RIPEMD-160	106
RIPEMD-320	110
RIPEMD-128	153
RIPEMD-256	158

**Veriler http://www.cryptopp.com/benchmarks.html adresinden alıntıdır.*

“parola” Kelimesinin Değişik Kriptografik Özet Algoritmalarındaki Sonuçları	
Girdi	(parola) 70:61:72:6f:6c:61 (uzunluk=6 byte)
Adler32	08d90280
CRC32	6ff263f0
Haval	5603729ec7c2d1ee8c14dc9b329041fd
MD2	b7d609ef500e4648458136b988b54e2b
MD4	79c7489d2ac6e8248f8274883347e742
MD5	8287458823facb8ff918dbfabcd22ccb
RipeMD128	e8ea149078368d688e02c59d3ede9cf8
RipeMD160	7dbab4974c4a32532c16b306468bf2759afe47e3
SHA-1	83592796bc17705662dc9a750c8b6d0a4fd93396
SHA-256	a80b568a237f50391d2f1f97beaf99564e33d2e1c8a2e5cac21ceda701570312
SHA-384	386cc41d1595dae584a9ba804161a3c71613ade48e4cd32712e205c1bb32ee252ff6d1d3ae31ab33b2636302a75d7e1b
SHA-512	6226ff0e50b5313f287a6904ecf242b67d00d28bd211ddae51e8f044d24de0defd4daaa32eecac9bb13f9d2fe462941838937f16613aafdd075075ef9dfe7b64
Tiger	363de1028525a467e5e9b6ad3cfa00e36b38e2a490f75d1f
Whirlpool	3d926e55cbaedf1344d24791be43c46a8c0c8f7f2dc831a8864d46497df7ac4b92cf880469379a80b142a2b012d4104bd665bbd535817b4a5b3448ef467b172a

Kriptografik özet algoritmalarının parola depolaması için kullanılmasının doğası gereğince var olan bir sistemin depolama algoritmasının değiştirilmesi de “yavaş” olan yöne doğru “kolay” yapılabilmektedir. MD5 algoritması ile Salt kullanılmadan gerçekleştirilmiş bir parola özeti, ileri bir zamanda tercih edildiğinde kolaylıkla Salt eklenerek de depolanabilir. Bunun için depolanan MD5 parola özetinin bir daha MD5 algoritmasına bir Salt değeri ile sokulması yeterli olacaktır. Ancak hali hazırda Salt eklenerek oluşturulmuş bir parola özeti sadece düz MD5 algoritması kullanılacak bir sisteme güncellenemez.

Yavaşlatılacak olan parola depolama sistemi için bir örnek:

Hızlı sistem örneği:
Kullanıcı -> (Kullanıcı bir parola kelimesi seçer) -> parola
Sistem -> (Seçimi MD5 algoritmasına sokar) -> 8287458823facb8ff918dbfabcd22ccb

Hızlı sistemin yavaşlatılması örneği:
Kullanıcı -> (Kullanıcı bir parola kelimesi seçer) -> parola
Sistem -> (Seçimi MD5 algoritmasına sokar) -> 8287458823facb8ff918dbfabcd22ccb
Yavaşlatma Sistemi -> (Var olan çıktıya Salt değeri ekler) -> md5($salt_degeri+8287458823facb8ff918dbfabcd22ccb)
Güncelleme işlemi -> (Bütün parola özetleri yeni değere göre güncellenir -> "Güncel ve daha yavaş parola özetleri"

Salt değeri eklenmiş veya başka algoritmalarla iterasyona sokulmuş bir parola özeti daha "hızlı" yöne doğru geliştirilemez. Bunun nedeni kriptografik özet algoritmalarının tek yönlü olmasıdır. Yavaşlatma örneğinde eklenen Salt değeri aslında MD5 algoritmasının bir iterasyonudur. Yaygın kullanılan Salt ile depolama işlemi kullanıcının girdiği parola kelimesi ile işleme sokularak yapılan kriptografik özet işlemidir.

Yukarıdaki örneğe benzer olarak MD5 algoritmasının çıktıları SHA-512 algoritmasına veya tercih edilen başka bir algoritmayla işleme sokularak, parola depolama için kullanılacak algoritma daha yavaş hale getirilebilir. Bu algoritmalar kendi içinde bir algoritma oluşturabilecek şekilde farklı iterasyonlar ve kombinasyonlarla bir araya getirebilir. Kriptografik açıdan benzer olmasa da, 3DES (Triple DES) algoritması fikir olarak buradaki anlatım ile benzerlik taşımaktadır. 3DES şifreleme algoritması, DES algoritmasının 2 düz 1 ters iterasyonu ile meydana getirilmiştir. Kriptografik özet algoritmaları için ters iterasyon mümkün olmasa da yazılımcılar farklı algoritmaları ve salt değerlerini kullanarak kendilerine özgün yeni algoritmalar oluşturabilir ve saldırganların işlerini zorlaştırabilirler.

Gökhan Muharremoğlu
Bilgi Güvenliği Uzmanı

gokhan.muharremoglu@iosec.org