Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Kan jeg få salt på min hash?

Vigtig information omkring adgangskode opbevaring.
Gennemgår hvad en salted hash er, i forhold til en almindelig hash, og hvilke sikkerhedsrisici der er ved begge dele.
Til dig der har et site hvor folk kan oprette sig med brugernavn og password

Skrevet den **02. Feb 2009** af **radion** I kategorien **Programmering / PHP** | ★★★☆☆

Opbygning:

Hvad er en hash-værdi?

En hashværdi er en envejskryptering, dvs. det kan ikke dekrypteres.
En hashværdi er unik, dvs. til hver tekststreng, findes der kun én hashværdi.

For at starte ved starten, så handler det her i bund og grund om håndtering af adgangskoder.
Hvis du driver et site, hvor folk kan skrive en adgangskode, for at komme ind på lukkede dele af sitet, så er denne artikel for dig.

Når man gemmer adgangskoder, er det vigtigt at sikkerheden er i orden, der er ikke meget sjov i at gemme en hemmelig adgangskode, hvis stedet den opbevares i er frit tilgængeligt.
Ofte er databasen, tekstfilen eller hvor man nu gemmer sine brugeres adgangskoder, ikke frit tilgængelig, men hvis en række liste med adgangskoder skulle falde i de forkerte hænder, så er det vigtigt at de ikke umiddelbart kan anvendes.

Derfor er det en god politik at gemme hash-værdier af en adgangskode, i stedet for at gemme selve adgangskoden.

I PHP er der indbygget forskellige algoritmer til at udregne, bl.a. MD5 og SHA1.
I denne artikel, vil jeg tage udgangspunkt i MD5, men, alle problemstillinger såvel som løsninger gælder naturligvis også for SHA1, og andre hashalgoritmer

Når man kører en adgangskode gennem MD5, vil man få en hashværdi ud på 32 karakterer. Hvis du kører en DVD på 4.7 GB gennem MD5, vil du stadig få en hashværdi ud på 32 karakterer.

> ***MD5 afleverer altid en hashværdi på 32 karakterer***

Hvorfor bruge en hash-værdi
Sikkerhedshensyn
Jeg har et site, hvor folk kan oprette brugernavn og adgangskode
I stedet for at gemme deres adgangskode i databasen, så gemmer jeg hashværdien af deres adgangskode.
Det giver et ekstra element af tryghed, da jeg ikke kan se deres rigtige kode, ingen af mine medarbejdere kan se deres rigtige kode, og hvis databasen skulle falde i forkerte hænder, er der ingen andre der umiddelbart kan gennemskue adgangskoderne.

Hvis min adgangskode er "heste" så vil den tilsvarende hashværdi blive gemt i databasen, altså MD5("heste") der vil outputte "33c760c9e54eae0ffbb4c5ff696dfbc6"

Problematik angående rainbowtables
Hash-værdier kan ikke dekrypteres, men der er store projekter i gang, hvor man udregner hash-værdier på alle mulige forskellige ord, og tilfældige bogstavkombinationer, og gemmer disse i en tabel, hvor man bagefter kan søge.
Disse tabeller kaldes rainbowtables.
Dvs. at hvis jeg har fanget mig en adgangskode-database hvor adgangskoderne er envejskrypteret med fx MD5, så kan jeg snuppe den hash, og søge i min foretrukne rainbowtable efter den, og få vist hvad den oprindelige tekst er.

Et eksempel er http://www.gdataonline.com der er en kæmpe online rainbowtable, hvor alle frit kan søge på hash-værdier, og findes de i databasen, får man vist den oprindelige værdi.

Søger jeg på "33c760c9e54eae0ffbb4c5ff696dfbc6" i gdataonline.com får jeg vist at det er hashværdien til "heste"

> ***Dvs. en hashværdi, er kun sikker så længe den ikke er blevet udregnet og lagt tilgængelig i en rainbowtable.***

Hvad er en salted hash
Salted hash er stadig en hashværdi, men, en hashværdi hvor man har tilføjet noget ekstra til den oprindelige tekst. Denne tilføjelse kalder man salt.
Ideen bag salted hash er at inden man udregner hashværdien af en given tekst, så lægger man en ny tekst til den oprindelige, og får derfor en helt ny hashværdi.

altså i stedet for

```
$hash = MD5($adgangskode);
```

Bliver det til

```
$hash = MD5($salt . $adgangskode);
```

Eksempel
Vi så før at hashværdien for "heste" er "33c760c9e54eae0ffbb4c5ff696dfbc6"
Nu tilføjer vi lidt ekstra til min adgangskode "heste", lad os tilføje "fiske" som salt

I stedet for at udregne hashværdi for "heste" udregner jeg nu hashværdien for "fiskeheste" altså

$hash = MD5("fiske" . "heste"); //05899dbfac0ba4743513dce0fb50d40b

Pseudokoden for at gemme en brugers kode kunne se således ud:

```
<pseudokode>

Modtag adgangskoden
Generer tilfældig "salt"
Udregn hash udfra adgangskode + salt
Gem hashværdien
Gem salt

</pseudokode>
```

Det er vigtigt at vi også gemmer "salt" da vi skal bruge det igen, næste gang brugeren logger på, og vi skal checke om denne har skrevet den rigtige adgangskode

Pseudokoden for at checke om en brugers kode er rigtig kunne se således ud:

```
<pseudokode>

Modtag brugernavn
Modtag adgangskode
Find salt og adgangskodens hashværdi udfra brugernavn
Udregn ny hash af (brugerens indskrevne adgangskode + salt fra databasen)
Check ny hash op mod hashværdi fra databasen

</pseudokode>
```

Er salted hash sikker?
Mere sikker end almindelig hash, fordi en person der vil cracke hashen, er nødt til at genudregne alle hashværdier for hele hans rainbowtable, da alle værdier skal have saltet indsat i enden, inden der bliver fundet hashværdier.
Derfor kræver det langt flere ressourcer at cracke en adgangskode der er saltet, end en der ikke er.
Og naturligvis skal man bruge forskellig salt til alle adgangskoder man opbevarer.

Jeg vil nu opstille et kodeeksempel der genererer saltede hashværdier

```
<?
//Venligst udlånt af www.phpsec.org
//http://phpsec.org/articles/2005/password-hashing.html
//hvor lang en streng skal min salt være, maks I dette eksempel er 32
define('SALT_LENGTH', 9);

function generateHash($plainText, $salt = null)
{

    if ($salt === null)
```

```
    {
//Hvis der ikke er defineret noget salt, så skal der udregnes en tilfældig
salt-værdi
        $salt = substr(md5(uniqid(rand(), true)), 0, SALT_LENGTH);
    }
    else
    {
    //hvis $salt er udfyldt, så udtrækkes saltet fra adgangskoden
        $salt = substr($salt, 0, SALT_LENGTH);
    }
    //Der returneres en 41 karakter lang streng der består af
    // salt & md5(salt . kode)
    return $salt . md5($salt . $plainText);
}

?>
```

Der er 2 måder at anvende ovenstående kode på.

1. når en bruger skal have gemt en ny adgangskode, så kaldes funktionen som følger:

```
$nykode = generateHash($brugerkode);
```

$nykode vil så indeholde saltstrengen + hashværdien af (salt + brugerens kode)
Dette lægges i databasen.

2. Når brugerens adgangskode så skal checkes, så kaldes funktionen på følgende måde, med $brugerkode som den adgangskode brugeren lige har skrevet, og $dbkode som den hashværdi der er gemt i databasen.

```
$checkkode = generateHash($brugerkode,$dbkode);
```

$checkkode vil nu kunne sammenlignes med $dbkode

Links

PHPSEC's artikel om selv samme emne, kodeeksemplet er hentet derfra
http://phpsec.org/articles/2005/password-hashing.html

Online rainbowtable hos gdataonline.com
http://gdataonline.com/seekhash.php

MD5 hos PHP.net
http://php.net/md5

SHA1 hos PHP.net
http://php.net/sha1

------artikel changelog-----
23-12-2006 kl. 22:39 | ´´ ændret til " så koden skulle være c&p venlig. Ændret i Synopsis

25-12-2006 kl. 01:45 | grundet julen er denne artikel gjort gratis. Hvis nogle af jer der har betalt, er utilfredse med at have betalt, så skal I nok få jeres 5 point tilbage, læg et svar i http://www.eksperten.dk/spm/752356

svar til the_email 25/12 2006 01:52 | true, men, har ikke taget det med for ikke at forvirre mere end højst nødvendigt.
Desuden er meningen måske ikke udtrykt klart nok. Meningen er at til hver tekststreng hører der KUN én hashværdi
en streng kan altså ikke have 2 forskellige hashværdier.
taget til revidering :)
Men, jeg kan love dig jeg tager det op i en kommende artikel, omhandlende adgangskodeopbevaring generelt :)

reply på o-zone:
tak for de fine ord, og for en forklarende opfølgning overfor andre brugere, men, jeg hæfter mig ved at du skriver, "hvis man bruger samme salt til alle brugere skal man kun omregne rainbowtableen een gang"
Jeg ved ikke om det er myntet på min kode, men, jeg vil bare nævne at min kode netop lægger op til forskellig salt til alle brugere.

Kommentar af the_email d. 25. Dec 2006 | 1

Fin artikel. Du påstår dog at der findes en unik MD5-hash til hver tekststreng. I og med at en MD5-hash er på 32 karakter, fra 0-9 og a-f, må der være 16^32 forskellige strenge. Der er altså ca. 3,403*10^38 forskellige MD5-hash'es. Det er ganske vidst utroligt mange, men dog ikke uendeligt :-)

Kommentar af md_craig d. 27. Mar 2007 | 2

Der er dog lige et par ting, kald det misforståelser om du vil i artiklen...

Først og fremmest er en hash ikke unik i den forstand at den har en 1-1 relation med en tekststreng. i teorien vil mange forskellige kombinationer af strenge kunne producere den samme hash (Også kendt som Hash collision), dette skyldes naturligvis at en hash har en fast størelse som bringer os til misforståelse nr. 2.

En MD5 hash levere ikke 32 karaktere. men derimod en 128 bits værdi, i dette tilfælde representeret i Hexadecimal.

Ang. det med at lagre saltet med hashen er mindre sikkert, det er det self. men i det øjeblik at man skal til at "genbygge" sine tabeller for at finde en collision, så kan man stadig tage det mere roligt end hvis man som med "33c760c9e54eae0ffbb4c5ff696dfbc6" bare kan gå ind på: http://md5.rednoize.com/ taste hashen ind og trykke enter.

Kommentar af greew d. 01. Jan 2007 | 3

Ganske udmærket artikel - jeg bruger også selv hashing af kodeord.
Som mclemens gør opmærksom på mener jeg også at det ikke er helt holdbart at opbevare saltet i

databasen, hvis man bruger som argumentation at databasen bliver stjålet. Men dette er også det eneste lille men jeg har. Ellers er det en helt fin artikel!

Kommentar af o-zone d. 03. Mar 2007 | 4

Super fin pointe. Jeg hasher altid mine brugeres passwords, og jeg har lige checket lidt ud på den online reainbowtable du henviser til. Heldigvis var det kun et par stykker der "faldt igennem" - men nogle gange kræver det jo ikke mere end et enkelt lille hul at lægge hele systemet ned!
I øvrigt hasher jeg passwords med clientside javascript, så de bliver krypteret hos brugeren INDEN de bliver sendt over http! ... I skal huske at det kun næsten er nok at gøre det serverside, hvis I har en eavesdropper hængende på linien!

mclemens: 7-9-13 mener jeg IKKE bør indgår i nogens sikkerhedsprocedurer, og at der ikke findes nogen 100% sikre systemer kan da vel aldrig blive til et argument for at lade være med at gøre sig umage!!

Med hensyn til det med at gemme saltet sammen med hashen - radion skriver jo selv i artiklen at det "bare" vil kræve at hackeren genregner hele sin rainbowtable. Det er selvfølgelig ikke en umulig opgave - men det giver da stadig væsentlig bedre sikkerhed end hvis du bruger en usaltet hash.
Ulempen ved at bruge samme salt til alle pws er at når først saltet er fundet, så behøver hackeren kun at omregne rainbow tabellen een gang, for at knække ALLE dine brugeres passwords (dem af dem der er dårlige nok til at dukke op i en rainbow table i hvert fald).

Til sidst vil jeg lige huske jer på at sikker omgang med jeres brugeres passwords jo ikke kun handler om JERES systems sikkerhed. Mange brugere benytter samme password flere forskellige steder (ja det er dumt - men sådan ER verden altså indrettet), så hvis først den er faldet et sted, er der åbnet op mange steder. Hvis I overhovedet ikke hasher brugernes passwords, og I bruger en ukrypteret linie (altså f.eks. alm. http), så sender I brugernes password i klartekst hver eneste gang de logger på, så eavesdroppere har fri adgang på alle hylder! Har I hashet den (og især MED salt) så kan de pws I sender ikke bruges til squat ud over jeres system. Så mclemens og andre der sender pws i klartekst: i mine øjne skylder I jeres brugere at spænde hjelmen en smule mere ind (no offence)

Kommentar af qtax87 (nedlagt brugerprofil) d. 04. Jan 2007 | 5

Okay artikel men du kan bare:
md5(md5(md5(md5(md5(md5(md5(md5(md5(md5("din streng"))))))))))

Kommentar af pillpopper d. 13. Jan 2007 | 6

Rimligt god viden, at have/få samt en virkelig god og udførlig formulering.
;))

Kommentar af terrak d. 24. Dec 2006 | 7

Jeg gav 5 point for denne artikel, udelukkende pga. den morsomme overskrift. Indholdet var okay. Jeg savner måske lidt omkring positive/negative sider ved at bruge MD5 og SHA1, som nævnes i artiklen.

Kommentar af thomaxz d. 10. Jan 2007 | 8

Kommentar af shetter d. 07. Jan 2007 | 9

Kommentar af mclemens d. 04. May 2007 | 10

o-zone skrev - "mclemens: 7-9-13 mener jeg IKKE bør indgår i nogens sikkerhedsprocedurer, og at der ikke findes nogen 100% sikre systemer kan da vel aldrig blive til et argument for at lade være med at gøre sig umage!!" ... Nej, jeg bruger ikke 7-9-13 på den måde og koder så sikkert som jeg kan, men man håber da at man ikke har overset noget, tror du misforstår mig!! Generalt er jeg enig i de ting du skriver. Dog vil jeg lige kommentere lidt: "... Med hensyn til det med at gemme saltet sammen med hashen - radion skriver jo selv i artiklen at det "bare" vil kræve at hackeren genregner hele sin rainbowtable. ..." Hvis den var henvendt til min kommentar om opbevaring af salt eller ens salt, så var det jeg skrev en general kommentar om salt og ikke henvendt til artiklens skribent.

... ..."Hvis I overhovedet ikke hasher brugernes passwords, og I bruger en ukrypteret linie (altså f.eks. alm. http), så sender I brugernes password i klartekst hver eneste gang de logger på, så eavesdroppere har fri adgang på alle hylder! Har I hashet den (og især MED salt) så kan de pws I sender ikke bruges til squat ud over jeres system." - Det vil så sige at når de sender et kodeord, som bliver hashet ... så kan det ikke opsnappes, men et der ikke bliver hashet kan godt ??? Altså umiddelbart ville jeg jo tro at det der sendes fra brugeren kunne opsnappes, uanset om det hashes på serveren eller ej... At du så kører clientside hashing også, gør jo så at det kodeord der sendes er hashet i forvejen, men ikke desto mindre kan det jo stadig opsnappes, hvis vi ikke snakker krypteret linje - ligesom ikke hashede kodeord - eller tager jeg fejl ?

.... "Så mclemens og andre der sender pws i klartekst: i mine øjne skylder I jeres brugere at spænde hjelmen en smule mere ind (no offence)" Jeg mener ikke at der er den store forskel på hashing eller ej, men mere krypteret linje eller ej. Den eneste forskel jeg føler på hashing(med uden salt) eller ingen hashing, er hvorvidt din sikkerhed er iorden så de ikke får adgang til din db og kodeordene. Opsnapning af kodeord kan ske med eller uden hash, ikke ?-). Nåh, nu bruger jeg ikke rigtig selv login systemer - men har kun et meget begrænset login punkt hvor der er et mindre indhold, så det er ikke min hovedpine p.t., men ja ssl krypteret forbindelse er selvfølgelig at foretrække, hvis behovet er derefter - tør du godt logge på eksperten m.fl. når dit kodeord sendes åbent?. Jeg kan selvfølgelig godt se dit problem med brugere der bruger ens kodeord, men mon ikke man skulle få brugere til at holde op med det - i det mindste når det kodeord de afsender kan opsnappes inden det bliver hashet på serveren. (brugere kan ikke selv vælge kodeord under mit brug, så det med ens kodeord sker ikke medmindre det er tilfældigt). Husk, min kommentar går på det sikkerhedsmæssige i hashing og ikke det sikkerhedsmæssige i hvorvidt linjen er krypteret eller ikke krypteret ;o) (no offence).

... --- ... --- ... Postet 28/12 2006 16:24: Fint skrevet og god ide. Du starter op med, at folk burde bruge salt og hash istedet for blot hash, fordi så kan det hashede kodeord ikke lige omregnes til det ikke hashede kodeord (eller et ord der giver samme md5 værdi). Da de hashede kodeord ikke ligger frit tilgængelig, som du selv nævner, skal der være en brist for at få fat i det hashede kodeord.

Risikoen ligger så i, at den der har fået adgang til md5 kodeordet også har fået adgang til saltet. Så kan brugeren sætte cpu'en til at beregne på, hvilke ord der i forbindelse med saltet giver den rigtige md5 hash... Det ville være ude i ekstremer, og det er heller ikke sikkert at personen, der får fat i det hashede kodeord også får fat i saltet.

Det eneste jeg mener er at salt ikke i alle tilfælde gør det mere sikkert, såfremt man har et site, hvor der er huller i sikkerheden. Jeg hasher ikke selv, men låser istedet sikkerhedsmæssigt, så der ikke kommer brister (7-9-13). Jeg ser ikke hashing som en sikkerhed, hvis saltet kan risikere at ryge ud samtidig, men ellers er saltet selvfølgelig en klar fordel.

Ved brug af salt ville jeg vælge noget fast salt (variabelt er selvfølgelig også ok) og gemme det eksternt fra db'en og så include saltet istedet for at hente det fra db'en - fordi hvis kodeordene i db'en ryger, ville der være stor risiko for at saltet også gjorde, hvis det lå i db'en.

Rainbowtables er vel ok, de minder os om at md5 ikke er sikkert, hvis md5 "dehashes" ved beregning og sammenligning på resultater fra md5 hashede ord.

Kommentar af windcape d. 04. Jan 2007 | 11

godt skrevet, vil klart anbefale den til ALLE :-) og påpege at det samme gælder også sha1 :) . | Coldgate: hvilket er komplet stupidt, og har absolut intet med at sætte salt på. Derudover så at md5 kryptere en md5 gør det faktisk nemmere at cracke den. Flere gange md5 hjælper heller ej mod rainbow tables.

Kommentar af dragothica d. 01. Feb 2007 | 12

Ganske god lille artikel. Vi bør måske nævne muligheden for også at benytte sig af mysql funktionen password(). Denne kan dekrypteres, men gemmer man en hash værdi i en mysql krypteret form, er det ekstra arbejde for 'hackeren' og derved bliver sikkerheden større.

Kommentar af emilfs d. 09. May 2007 | 13

Du ved hvad du taler om!

Kommentar af superfisker d. 07. Jul 2008 | 14

Super fedt :D
Som emilfs siger, så ved du hvad du taler om!

Helt igennem fed artikel, som klart kan anbefales !