**17-01-10 - Veilig een wachtwoord oversturen**

Bijna iedere webapplicatie heeft wel een inlogsysteem. Soms alleen voor de administrators, soms ook voor de bezoekers. Het maken van een inlogsysteem is niet zo ingewikkeld, maar hoe zorg je er nu voor dat niemand de inloggegevens kan stelen terwijl je deze overstuurt?

## Sniffers

De grootste bedreiging voor inlogsystemen, zijn de zogenaamde ?netwerk sniffers'. Het afluisteren met een sniffer is vrij eenvoudig. Er zijn duizenden tooltjes die het mogelijk maken al het netwerk verkeer uit te lezen. Op een thuisnetwerk is dit vaak niet zo veel, maar bij een groot bedrijf, universiteit of internetcafé kun je al snel heel wat pakketten uitlezen. Wanneer een website een eenvoudig inlogsysteem gebruikt, en het wachtwoord gewoon via HTTP verzonden wordt, kun je het wachtwoord dus ook gewoon uitlezen met de sniffer. En ja lieve mensen, ook de POST-variabelen kunnen op deze manier eenvoudig uitgelezen worden. Pas dus op met het ontwikkelen van inlogsystemen en kijk bij security verder dan alleen het server-side script (PHP, CGI, etc).

## Methode 1: SSL

De makkelijkste manier om een beveiligde verbinding op te zetten, is SSL. Bij SSL wordt alle data tussen de zender en ontvanger gecodeerd met RSA-encryptie. SSL vereist echter wel een dedicated IP-adres en een SSL-certificaat. Een dedicated IP-adres is (zoals de naam al zegt) een IP dat enkel bij jouw domein hoort. Bij de meeste webhosting-providers worden alle websites op één IP-adres onder gebracht. Voor een dedicated IP moet vaak tussen de 5 en 25 euro per jaar bijbetaald worden. Het SSL-certificaat kun je zelf kostenloos genereren, maar dit brengt wel een klein probleem met zich mee. Webbrowsers zullen de uitgever van het certificaat (jijzelf) niet herkenen en automatisch een rode balk tonen of allerlei waarschuwingen geven. Voor een administratiesysteem is dit niet zo ontzettend erg, maar voor een webshop kan dit natuurlijk niet. We willen immers niet dat onze klant allerlei waarschuwing krijgt dat hij misschien wel op de website van een oplichter is beland. Er zijn een aantal grote certificaat-uitgevers die door vrijwel alle browsers (zouden moeten) worden herkent. Onder andere: Verisign, Thawte, GeoTrust en Comodo. Verisign is verreweg de bekendste en wordt ook gebruikt door banken en andere grote instellingen. Nadeel van Verisign is de prijs. Comodo biedt op moment van schrijven de goedkoopste certificaten aan voor 12 euro, op de voet gevolgd door Geotrust met 13 euro. Voordat je een SSL-certificaat bestelt raad ik echter wel

aan eerst eens goed te kijken naar de verschillen. Sommige uitgevers bieden namelijk een gratis tweede domein aan binnen het certificaat (bijv. domein.nl & www.domein.nl) en andere bieden bijvoorbeeld ondersteuning voor mobiele telefoons. Bedenk dus van te voren goed wat je met het certificaat wilt doen. Het aantal bits encryptie geeft de sterkte van de codering aan. Hoe hoger hoe beter. Momenteel is 256 het hoogste en tevens meest gebruikte.

## Methode 2: Salt

Het doel van dit artikel is om een alternatief voor SSL onder de aandacht te brengen. Waarom? Deze methode geeft je veel meer controle over de encryptie en is bovendien gratis!

Momenteel ben ik samen met een partner hard bezig met de ontwikkeling van een CMS. Omdat we dit systeem zo breed mogelijk in willen gaan zetten is een vereist SSL-certificaat niet gewenst.

### Wachtwoord coderen

Ons inlogsysteem werkt via AJAX (javascript). Wanneer de gebruiker op de inlog-knop drukt, wordt er een XMLHttpRequest gedaan naar loginServlet.php. De gebruikersnaam en wachtwoord worden meegestuurd als POST-variabelen. So far so good, maar als iemand een sniffer op het netwerk zet, kan hij het wachtwoord dus zo uitlezen. Om ervoor te zorgen dat het wachtwoord niet zomaar uit te lezen is, kunnen we gebruik maken van de md5-codering. Deze moet dan uiteraard wel uitgevoerd worden aan de client-side! We krijgen nu dus de volgende request:

```
POST loginServlet.php
HTTP/1.1Arrayusername=username&password=md5(password)
```

Wanneer het om een 'sterk' wachtwoord gaat, kan het wachtwoord in ieder geval niet zomaar uitgelezen worden. Probleem opgelost? Echt niet! Als een hacker de md5 heeft opgepikt, kan hij alsnog handmatig de loginServlet aanroepen met deze gegevens. Hij heeft het echte wachtwoord niet nodig, aangezien je met de md5 kunt inloggen.

### Salt

Na uren brainstormen, kwam mijn partner met een eerste aanzet voor de echte oplossing: de salt... Hoe het dan precies geïmplementeerd moest worden, wist hij

ook nog niet, maar dit gaf wel mogelijkheden. Een salt is een serie tekens welke achter een wachtwoord geplakt worden alvorens deze te coderen. Wanneer men normaal md5 ( password ) deed, doet men nu:

```
md5 ( password + salt )
```

Om het wachtwoord te valideren, moet dan aan de serverkant (1) het wachtwoord uit de database gehaald worden, (2) op dezelfde manier gecodeerd worden, dus md5 ( DBpassword + salt ) en dan (3) vergeleken worden met de userinput.
Zolang deze salt uniek is voor iedere inlogpoging, is het probleem opgelost. Wanneer men steeds dezelfde salt gebruikt, is de lol er natuurlijk ook af, want dan zitten we weer op hetzelfde probleem als bij md5 ( password ).

## De unieke salt

Om er zeker van te zijn dat de salt uniek is en bovendien niet aangepast kan worden door de client, moeten we deze laten bepalen door de server. De server genereert dus de salt en stuurt deze over naar de client. Let op: dit pakket kan ook afgeluisterd worden! Sterker nog, het is zeer waarschijnlijk dat dit pakket afgeluisterd zal worden als de andere request ook wordt afgeluisterd. Zolang de salt echter uniek is en de client geen invloed heeft op deze salt, is dit echter geen probleem. We gaan namelijk nu met het volgende javascript (!) aan de slag:

```
md5 ( md5 ( password ) + salt )
```

We kunnen ook md5 ( password + salt ) doen, maar dan komen we straks in de problemen, aangezien ik in de database md5 ( password ) heb opgeslagen. Ik kan dus nooit meer controleren of de user-input klopt met het wachtwoord in de database. We moeten dus echt dit krijgen:

```
Database: md5 ( password )ArrayInput: md5 ( md5 ( password ) +
salt )Arraymd5 ( database + salt ) == input
```

We kunnen de salt in dit geval het beste opslaan in een session. We kunnen de salt ophalen via AJAX (loginServlet.php) of deze
automatisch in de login-pagina laten zetten met PHP. Voor deze situatie is het logisch om de salt in de Servlet te laten generen, zodat de functionaliteit op één plaats blijft.

### Niet te kraken?

Zolang md5 nog niet echt gekraakt is, is dit script absoluut veilig. De hacker kan de salt uitlezen en de md5 die verzonden wordt door de gebruiker. Om hier de md5 van het wachtwoord uit te halen, ben je jaren bezig. Het wachtwoord is namelijk gecodeerd in 32 tekens. Een md5 van 32 tekens + salt kraken is een enorme klus. Als je dan toch wilt bruteforcen, kun je het beste een script schrijven dat md5 ( md5 ( password ) + salt ) uitvoert. We weten namelijk de salt al en het wachtwoord is ongetwijfeld een stuk kleiner dan 32 tekens. En alle combinaties van 1 t/m 31 tekens uitproberen duurt (bijna) even lang als alle combinaties met 32 tekens proberen. Het is wel belangrijk dat gebruikers sterke wachtwoorden hebben (meer dan 8 tekens en een combinatie van 8 letters en cijfers). Het kraken van het wachtwoord is dus absoluut onmogelijk met de huidige technologie. En mocht md5 ooit gekraakt worden, dan passen we gewoon het script aan, zodat het werkt met een ander coderingsmechanisme, zoals sha.
Het inloggen met de md5 zelf is ook onmogelijk geworden. De gegenereerde md5 bevat namelijk ook de unieke salt. En deze salt wordt door de server bepaald en opgeslagen in een sessie. De sessie is voor de gebruiker niet aan te passen.

## Conclusie

Het salt-systeem is 100% veilig, op voorwaarde dat er 'sterke' wachtwoorden worden gebruikt. Daarnaast moet men niet vergeten dat pakketten na het inloggen ook afgelezen kunnen worden. Dit betekent dat wanneer je een nieuwe gebruiker aanmaakt via het administratie-paneel, je deze ook via het salt-mechanisme zou moeten versturen. Helaas is het dan onleesbaar voor de server, omdat md5 een onomkeerbaar proces is. Alleen md5-en is ook niet genoeg, omdat een hacker genoeg heeft aan deze md5. Ons systeem is er namelijk op ontwikkeld dat de md5 + salt verzonden wordt. Een hacker kan een salt laten generen zoals iedere gebruiker en deze vervolgens achter zijn gesloten md5-string plakken. Nu hoeft hij dit geheel alleen nog een keer te md5-en en dan kan hij inloggen. Bij het aanmaken van gebruikers kunnen we dit probleem eenvoudig omzeilen door het wachtwoord te laten generen door het script op de server en deze te mailen naar de gebruiker. En deze mail zou dan wel encrypted moeten zijn met PGP.

De kans dat iemand af zit te luisteren terwijl je inlogt is afhankelijk van de locatie. Zolang we dit thuis op een goed beveiligd netwerk doen, is er niets aan de hand. Wanneer we een risico-locatie zitten (internetcafé, etc.), moeten we er wel rekening mee houden dat we getapt kunnen worden. Daarom raad ik aan SSL of het bovengenoemde salt-systeem te gebruiken. Voor de salt-gebruikers raad ik

daarnaast aan om geen gebruikers aan te maken op een risico-locatie. Doe dit gewoon even thuis, zodat je zeker weet dat je geen risico loopt.