Säker Webshop

Pétur G. Hjartarson, David Dahlin
ic08ph0, ic08dd7

5 oktober 2010

Introduktion

PHP, MySQL och Apache driver många www-siter. Tack vare enkla installationer kan nu var man sätta upp en simpel http-server med php- och databasstöd. Tack vare en färdig applikation som kallas LAMP (Linux, Apache, MySQL and PHP) så kan man med en terminalrad få igång en fullskalig server. Enkelheten i att kunna sätta upp en server ger en heldel problem. Säkerhetsluckorna tas inte om hand i xAMP versionerna, utan riktar in sig på development stadiet, istället för tillförlitlighet. Detta går naturligtvis att åtgärda, med en terminalrad får man bort flera säkerhetshål */opt/lamp/lamp security*. Denna sätter lösenord på alla requests som görs till sidan, detta görs för att "hackers" inte ska kunna se känslig information om webservern, t.ex. versionsnummer eller databas-typer.

Dator Specifikation:

- *Dator:* Acer 3820TG 4Gb RAM, Intel Core i5, ATi 5820HD 1024Mb.
- *Uppkoppling:* 100/100 från Lunds stadsnät. 802.11 draft-n.
- *LAMP:* XAMPP 1.7.3a for Linux - Apache 2.2.14. MySQL 5.1.41. PHP 5.3.1. ProFTPD 1.3.2c. phpMyAdmin 3.2.4

SSL

För att kunna säkerställa att uppkopplingen till servern inte avlyssnas kan man använda sig av HTTPS. Till skillnad från vanliga HTT-protokollet som kör på port 80, så kör HTTPS på port 443. Med hjälp av *openssl* kan man generera hela certifikatskedjan. Vi har skapat våran egen CA, som verifierar vårat eget certifikat. Att man skapar sin egen CA skapar "problem" för användaren som troligtvis måste lägga till ett säkerhets undantag, med hjälp av sin user-agent kan användaren välja att tillförlita sig på våran CA. Överföringen är fortfarande lika säker, eller "lika" krypterad som om en root CA hade verifierat oss. Om användaren skulle ha våran egen skapade CA under sina root-certifikat i webläsaren, skulle denna extra åtgärd inte finnas.

Vi använder oss av 2048 bitars RSA genererad nyckel och hemsidans uppkoppling är krypterad med AES-256 med cipher block chaining. Samma kryptering hittas på ett flertal webplatser som en säker överföring krävs, t.ex. www.seb.se.

Genereringen av SSL känns igen från tidigare projekt med inriktning säkerhet. Vi använde oss dock av en metod som är anpassad för Apache och egen skapad website.[1]

PHP och HTML

Det populära skriptspråket PHP (rekursivt akronym för PHP: Hypertext Preprocessor) används genomgående i webshopen. Tack vare detta kan vi, vid förfrågningar t.ex. vid manipulation av kundvagn eller inloggning, på ett dynamiskt sätt hämta eller alterera data genom enkla mysql-förfrågningar. När användaren möter hemsidan för första gången visas två länkar i headern, "Home" och "Log in". Vid tryck på "Log in" vidarebefodras användaren till en login-ruta där man uppmanas mata in användarnamn och lösenord.

Om inte användaren är registrerad sen tidigare ges möjlighet att genomföra detta genom att klicka på länken "Registrera mig". Tanken här är att ge varje användare en kundvagn som är tilldelat varje användarkonto. På detta sätt kan användaren logga ut och fortfarande bibehålla de produkter som tillhör denne. I products.php var syftet att ge användaren en överskådlig bild av de produkter som finns tillgängliga samt deras pris och artikelnummer. När användaren bekräftar sitt val av produkt som ska adderas till kundvagnen mottagar användaren en bekräftelseruta som ska motsvara feedback. Vid tryck av "Checkout" presenteras användarens nuvarande kundvagn där pris,kvantitet och totalsumma ges.

Skulle användaren ångra antalet av en viss produkt ges möjlighet att ändra på detta. När sedan användaren klickar vidare på den gröna "Check out"-knappen vidarebefodras man till betalsidan som efterfrågar kontokortsuppgifter. Vid "Proceed" förs man sedan vidare till startsidan "Home". Betalsidan ställer inga krav på genuina kontokort, det enda som krävs för att komma vidare är att fälten består av siffror.

Sessioner

När sessioner används i PHP utan omtanke kan eventuella säkerhetshot föreligga. Till exempel s.k "Session fixation" där man tvingar användaren att nyttja en fördefinerad sessionsid. För att komma till bukt med detta låter vi servern generera ett sessions identitet. Även vid inloggning så genereras en ny session identitet.

php.ini

I php.ini görs bestämelser huruvida php ska tolkas på webservern. Vi har tagit *expose_php = off* så att versionen av php inte ska synas i headern av paketet.

Vi har *global_variables = off*, som förövrigt är off deafult sedan PHP5.

MySQL

MySQL var formellt inget krav i projektet. Dock beslutades det tidigt in i projektet att nyttjandet av mysql skulle underlätta hanteringen av produkter och användare avsevärt. Ur säkerhetsaspekt kan detta innebära en risk t.ex mot SQL-injektioner. För att gardera oss mot detta har vi tillämpat s.k prepared statements i de fall som varit nödvändiga.[4] Ett exempel ses nedan som är taget ur products.php:

/*Inserts the selected product to cart*/

$ stmt = mysqli_prepare($ db,"INSERT INTO ShoppingCart (userid,itemID,item, price, amount) VALUES (?,?,?,?,?)");

mysqli_stmt_bind_param($ stmt,"issii",$ id,$ itemID,$ name,$ price,$ amount);

mysqli_stmt_close($ stmt);

Nuvarande utformning på databasen är:

members(id, username, password)

Products(itemID, name, price)

ShoppingCart(userid, itemID , item , price , amount)

Tidigare MySQL erfarenhet har hämtats från kursboken i Databasteknik. [2]

Hashen

Givetvis sparas inte lösenorden i tabellen members i klartext. Vi hittat en säker metod som genererar ett salt, som bereonde på användarens namn och lösenordslängd, konkatinerar slump saltet till lösenorded, som vi sedan hashar med SHA1 algoritmen som finns i php. När vi sedan dekrypterar hashen så kör vi in hashen från databasen som extra in-parameter i vår hashfunktion.[3]

Explicit msqli_connect()

I vanliga fall så anger man server, namn och lösenord i mysqli_connect(). För att göra så att dessa uppgifter inte står i klartext i någon php-fil så deklarerar vi variabelnamn i httpd.conf och tilldelar dem i httpd.conf filen. När vi sedan anropar mysqli_connect() i php-filen så anropar vi de deklarerade variablerna från httpd.conf filen. På så sätt ansluter vi explicit till våran MySQL server, utan att visa inloggningsuppgifterna.

Apache

Apache är idag världens mest använda http-server. Tack vare enkel konfiguration och stort gemenskap stöd på internet är det lätt att komma igång och att få hjälp med. Tack vare XAMPP behövdes inga stora konfigurationer för att fungera.

Många anvädnbara moduler inkluderas i grund konfigurationsfilen. Det blir å andra sidan mycket onödiga moduler som laddas in.

httpd.conf

httpd.conf bestämmer hur Apache ska köras. Där anger man server root mappen, alltså var man ska anlända på den lokala datorn om man slår in websideadressen. Man anger också vilken fil som ska laddas för användaren, i vårat fall: *index.php*. Här laddar vi ssl modulen som ska kunna hantera krypterad trafik. Vi laddar däremot inte alla SSL inställningar i den filen, utan inkluderar /opt/lamp/etc/httpd-ssl.conf. Vi sätter upp en virtuel host och tillåter port 433.

Javascript

Javascript nyttjas i inmatningsfält och dialogrutor. Det mesta är taget taget från intensiva och spontanta forumsökningar. Källor kan därmed ej redovisas.

CSS

CSS är för amatörer, vi kan skiten men vi orkade inte göra det, det är autogenererat, No coding. [5]

Referenser

[1] *Setting up SSL certificate on Apache* - http://www.flatmtn.com/article/setting-ssl-certificates-apache

[2] Database systems: The Complete Book, Second Edition (International Edition). Prentice Hall 2009. ISBN: 0-13-135428-0

[3] PHP - SHA1, comment: 94326 - http://www.php.net/manual/en/function.sha1.php

[4] PHP: MySQL Manual - http://php.net/manual/en/book.mysql.php

[5] Button Maker - http://css-tricks.com/examples/ButtonMaker/