COMPUTERWORLD

Huskeliste til en sikker side

I denne artikel kan du læse hvilke ting som er meget vigtige hvis du gerne vil have en sikker side. Det er ikke en afhandling men mere en Problem > Løsningsmodel. Der er også små eksempler inkluderet i artiklen.

Skrevet den **25. Feb 2009** af **sungdk** I kategorien **Programmering / PHP**

FORMÅLET
Denne artikel er baseret udfra mine egne oplevelser med nogle sider jeg har programmeret i PHP. Jeg har været ude for at nogle brugere på siderne misbrugte funktionerne som de ikke havde lov til, og derfor har jeg nu besluttet mig for at skrive denne artikel om ting du skal huske hvis du vil lave en sikker hjemmeside.

Jeg ved godt at artiklen er lidt overfladisk, men den er også mere tænkt som en slags huskeliste. Desuden er der sikkert andre metoder til at sikre siden, men disse nedenstående metoder har sikret mine sider imod misbrug.

SERVEREN ER NR. 1
Vil du lave en sikker side så skal du starte med at konfigurere opsætningen på serveren. For at undgå sql-injection er der en funktion ved navn "magic_gpc" som skal slås til. Når denne er slået til så er det ikke længere muligt at bruge '-tegnet. Et kendt sql-injection eksempler er brugen af: 1' OR '1' = '1.

Dvs. har man ikke sikret sig imod sql-injection så kommer der til at stå dette hvis man prøver at logge ind:

```
<?php
//database_forbindelse_inden

$result = mysql_query("SELECT * FROM tabel WHERE brugernavn = '1' OR '1' = '1' && kodeord = '1' OR '1' = '1'");
exit;
?>
```

Dvs. Misbrugeren kan nu logge sig ind på din side, uden egentlig at have oprettet en bruger eller angivet korrekt brugernavn og kodeord.

En anden vigtig ting er at rette register_globals til OFF. Ved at sætte denne funktion på OFF, så specificerer man sine variables. Dvs. $_POST['felt'] kun kan komme fra en form med method med POST. Havde register_globals derimod stået på ON, så ville $felt kunne indeholde information fra forms, arrays og meget andet.

SESSIONS ELLER COOKIES?
Session og cookies er begge en slags små "pakker" som kan gemme information. I almindelig login-scripts, er det meget normalt at folk enten benytter sig af Cookies eller Session - Men det ene er mere sikkert end det andet!

Session og cookies har altså det samme formål, minder meget om hinanden, men alligevel er der forskel.

Den største forskel er at cookies bliver gemt på din egen computer som små filer, og sessions bliver gemt på selve serveren som du er inde på. Man skal så vidt muligt enhver brug af cookies når det kommer til login-scripts, da man kan manipulere med cookies, og det er muligt at se indholdet af en cookie.
Vil du der imod lave et "gem login" eller på anden måde genkende brugeren selvom browseren er lukket ned og startet op igen, så er det cookies du skal have fat i. Cookies har den fordel at de kan sættes til først at blive automatisk fjernet fra din computer efter et vis antal tid/dage/år osv. MEN! Samtidigt vil jeg sige, at skal man lave et en cookie i forbindelse med "gem login"-funktionen, så skal man altid først md5 (eller på anden måde) skjule sit kodeord.

Eksempel på md5-envejskryptering:

```
<?php
$str = 'hej hej';
$str_md5 = md5($str);

echo "Her er hej hej med md5: $str_md5";

exit;
?>
```

XSS (Cross Server-Scripting)
XSS er et andet "hul" som folk ofte glemmer at sikre sig imod. Løsningen på XSS er heldigvis ret simpel, men har man nu en hjemmeside bestående af mange 100 sider kan det godt være et større arbejde. Men det er jo bl.a. derfor at jeg har skrevet denne artikel, så folk netop husker at lave sikker programmering lige fra starten.

Navnet "Cross Server Scripting", siger lidt om hvad dette hul handler om. Her har "misbrugeren" nemlig mulighed for at gemme et script i din database/tabel, og når det aktuelle felt bliver trukket ud i en ganske ren echo-streng som fx:

```
<?php
//database_forbindelse_inden

$result = mysql_query("SELECT * FROM tabel");
$row = mysql_fetch_array($result);

echo $row['xss-feltet'];

exit;
?>
```

I $row['xss-feltet'] kunne misbrugeren evt. have tilføjet dette javascript som ved åbning straks vil åbne en alert/popup med teksten "Din sikkerhed er ikke i top":

```
<SCRIPT LANGUAGE="JavaScript">
alert ("Din sikkerhed er ikke i top")
</script>
```

Når brugeren åbner denne side hvor dette database-udtræk finder sted, så kan det være at misbrugeren har indtastet en kode så din egen bruger bliver slettet, sender ufrivilligt informationer til misbrugeren, sletning af hjemmesiden og mere andet. Kortsagt så er dette noget som man skal være særlig opmærksom på.

Løsning var som sagt simpel, og det handler om at få PHP til ikke at udføre HTML/programmering som bliver udskrevet via echo.

Her er et eksempel på en løsning:

```
<?php
//database_forbindelse_inden

$result = mysql_query("SELECT * FROM tabel");
$row = mysql_fetch_array($result);

//strip_tags fjerner alt kode fra database udtrækket, undtagen <br> og (I dette eksempel)
$renset_echo = strip_tags($row['xss-felt'],"<br>[i]");
echo $renset_echo;

exit;
?>
```

[i]Man kan også bruge htmlentities() og nogle andre funktioner.

SIKRE SIG IMOD IFRAME-HACK (SPECIFIK XSS)
Er misbrugeren rigtig slem, så kan personen finde på at oprette en side på en anden server som indeholder en skjult iframe. Denne iframe kunne fx se således ud:

```
<iframe src="http://www.din-egen-side.dk/slet_bruger.php" frameborder="0" marginwidth="0" marginheight="0"></iframe>
```

Her skal man forestille sig, at brugeren allerede har logget sig ind på din hjemmeside, og lige nu er der en session som indeholder brugeren unikke id fra databasen.

Kigger man i iframe-koden, så kan man se at "misbrugeren" prøver at henvise brugeren til "