Wordpress Security
Not just an oxymoron - Steve Lord
Wordpress VulnSecurity?
What you talking about, Willis?

• Who is this guy?
  • slord@mandalorian.com
  • @stevelord on twitter
  • http://www.mandalorian.com/

• I test pens and kick out the bad guys
A Word about WordPress.com that is

- It's easy to point and laugh
  - Good incident handling
  - Open responses
  - Passwords encrypted
  - 'low level root exploit' used
- Wordpress.org not affected
Who uses it?
How to spot Wordpress
Who uses it?

How to spot Wordpress

First Things First

Welcome. WordPress is a very special project to me. Every developer and contributor adds something unique to the mix, and together we create something beautiful that I'm proud to be a part of. Thousands of hours have gone into WordPress, and we're dedicated to making it better every day. Thank you for making it part of your world.

— Matt Mullenweg

Installation: Famous 5-minute install

1. Unzip the package in an empty directory and upload everything.
2. Open `wp-admin/install.php` in your browser. It will take you through the process to set up a `wp-config.php` file with your database connection details.
   - If for some reason this doesn't work, don't worry. It doesn't work on all web hosts. Open up `wp-config-sample.php` with a text editor like WordPad or similar and fill in your database connection details.
3. Save the file as `wp-config.php` and upload it.
5. Once the configuration file is set up, the installer will set up the tables needed for your blog. If there is an error, double check your `wp-config.php` file, and try again. If it fails again, please go to the support forums with as much data as you can gather.
6. If you did not enter a password, note the password given to you. If you did not provide a username, it will be admin.
7. The installer should then send you to the login page. Sign in with the username and password you chose during the installation. If a password was generated for you, you can then click on 'Profile' to change the
Common Wordpress Security Fail (and how to avoid it)
PHP Error Reporting
Start at the bottom of the barrel

- Obligatory Google Dork
  - "php fatal error" inurl:wp-content -error_log -php_errorlog
- The fix (in php.ini)
  - display_errors = Off
  - Restart HTTP server daemon
Roll Your Own Auth

Please don't

- “We can't use the standard login/registration page for our users!”
  - Enterprise Solution: Rewrite the login/registration mechanism from scratch
  - Better: Let's download a plugin that lets us change the page

- The fix:
  - *Rape* Modify `wp-login.php` HTML
  - *Pillage* Change `wp-register.php` HTML
  - *Defile* Tweak `wp-admin/wp-admin.css`
SQL Injection
Someone get mustlive on the phone quick!
SQL Injection
The 90s called and want their framework back

• Wrong

```php
<?php
    $wpdb->query("UPDATE $wpdb->posts
        SET post_title = '$title'
        WHERE ID = $id"
    );
?>
```
SQL Injection
The 90s called and want their framework back

- Less Wrong

```php
<title> = esc_sql($title);
$id = absint($id);
$wpdb->query(
    "UPDATE $wpdb->posts
    SET post_title = '$title'
    WHERE ID = $id"
);

?>
SQL Injection
The 90s called and want their framework back

• Right

```php
<?php
    $wpdb->update(
        $wpdb->posts,
        array('post_title' => '$title'),
        array('ID' => $id)
    );
?>
```
SQL Injection

Getting it right

- Useful functions
  - `esc_sql()` - escape SQL queries
  - `absint()` - convert id to positive integer
  - `$wpdb->update()`
  - `$wpdb->insert()`
  - `$wpdb->prepare()`
  - `$wpdb->get_var()`
**SQL Injection**

wpdb->prepare() hotness

```php
<?php
    $key = "some input"
    $val = 1337
    $wpdb->prepare("INSERT INTO $wpdb->postmeta
                       (post_id, key, val)
                      VALUES (%d, %s, %s)",
                      array(10, $key, $val))
);?
```
Cross-Site Scripting
When input validation gets too hard
XSS
Not just a way for appsec guys to earn ££££s

• Wrong

```php
<?php
    $foo = $_GET["echo"];  
    echo 'You submitted:' . $foo;  

?>
```

DOUBLE FACEPALM
When the Fail is so strong, one Facepalm is not enough.
XSS
Not just a way for appsec guys to earn ££££s

• Less wrong

```php
<?php
    $foo = htmlspecialchars($_GET["echo"]);
    echo 'You submitted:' . $foo;
?>
```
XSS
Not just a way for appsec guys to earn ££££s

• Right

```php
<?php
    $foo = $_GET[“echo”];
    echo 'You submitted:' . esc_html($foo);
    ?>
```
• Useful functions
  • esc_attr_e() - for translated tag attributes
  • esc_html() - for general HTML
  • esc_attr() - tag attributes
  • esc_url()
  • esc_js()
CSRF
Serious business
CSRF
Pronounced 'Sea Surf' according to the Internet

- Cross Site Request Forgery ({C|X}SRF)
  - User is tricked into what looks like action A
  - Site receives request for action B
    - Doesn't distinguish between action and intent
  - Action B happens

- e.g: http://bank.com/transfer.php?amount=10000&to=steve
**CSRF**

Nonces and other HTTP perversions

- The fix:
  - 'Nonces'
    - One-off user-specific time-limited secret keys
    - Used where actions occur (e.g. CRUD)
      - This is what POST is for, but is not exclusive
CSRF
Getting to grips with Wordpress Nonces

```php
<?php wp_nonce_field(
    $action, $name, $referrer, $echo)
?>
```

- **$action** – What you're doing (default -1)
- **$name** – Nonce field name (default _wpnonce
- **$referer** – Set referer field for validation (default true)
- **$echo** – return hidden form field? (default true)
CSRF
Verifying the Nonce

```php
<?php wp_nonce_field(  
    if ( empty($_POST) || !
    wp_verify_nonce($_POST['name'],  
    'action') )  
    {  
        die ('Bad nonce.');  
    }  
    else  
    {  
        // process form data  
    }  
?>
```
**CSRF**

Some extra value

- When in admin
  - Use `check_admin_referer()`
- When not in admin
  - Check referer generally
- AJAX submission?
  - `$nonce = wp_create_nonce('action');`
  - `&ajax_nonce=$nonce`
  - `check_ajax_referer('action');`
3rd Party Plugins/Themes
Would you trust code written by these guys?

THE HACKERZZ GROUP
WE WILL RULE THE WORLD :)
3rd Party Plugins/Themes
A quick and dirty sanity check

- Did you write it yourself?
- Did you get it from Wordpress.org?
- Have you had direct contact with the author?
- Did you have to pay for it?
- Have you got the 'pro' version?
- Has the author released an update in the past year?
- Is it compatible with current wordpress?

- The more you answered no to, the more you need to audit *all* of the code
3rd Party Plugins/Themes
A quick and dirty sanity check

- Check for code obfuscation
  - `find . | xargs grep -i base64 > base64.txt`

- Check for links to external sites
  - `find . | xargs grep '\<[[:alpha:]]*:][/[^/]*>' > urls.txt`

- Check for potentially malicious content
  - `find . | xargs grep -Ei 'iframe|src|javascript:|eval|include' > dodgy.txt`
3rd Party Plugins/Themes
A quick and dirty sanity check

• Use the previous slide as a starting point
  • Things can be hidden anywhere
    – Don't assume a .gif is a .gif until you've seen it in a text/hex editor
  • Make sure you cover all code (php, JS) and data

• http://wpmu.org/why-you-should-never-search-for-free-wordpress-themes-in-google-or-anywhere-else/
Miscellaneous Mistakes
Entering the mouth of madness
Miscellaneous Mistakes
Entering the mouth of madness
Can the user do that?
Authentication != Authorization

```php
<?php current_user_can($capability); ?>
```

- `$capability` – the capability you're checking for e.g. 'manage_options'
- Use this everywhere if you don't want public access
- Options for more granularity
  - Role scoper plugin
  - Members plugin
- User levels deprecated in 3.0
Exec() and it's kin
Here be dragons

- exec(), passthru(), proc_*, shell_exec(), system(), popen() and backticks (`) are evil
- Do not use them
Exec() and it's kin
Here be dragons

- If you must use them
  - Don't use user-input for arguments
  - Set safe_mode_exec_dir in php.ini
  - Specify the full executable path
  - Use escapeshellcmd() on $cmd before execution
Exec() and it's kin
Here be dragons

• If you must use pass them user-supplied input
  • Set safe_mode_exec_dir in php.ini
  • Specify the full executable path
  • Use escapeshellcmd() on $cmd before execution
  • Use escapeshellarg() on arguments before execution
Exec() and it's kin
Here be dragons

- If you must use pass them user-supplied input
  - Set safe_mode_exec_dir in php.ini
  - Specify the full executable path
  - Use escapeshellcmd() on $cmd before execution
  - Use escapeshellarg() on arguments before execution
- Consider a career change
Remote File Include (RFI)
Or week 2 of Learn PHP in 21 days

```php
$inc = $_GET['inc'];
include($inc);
```?

- Don't do it. Ever.
- Use switch/case with hardcoded (from a config file) values
Fun with .htaccess
A few bits to take away

Order Allow,Deny
Deny from all

<!-- Files ~ "\.(css\|jpe?g\|png\|gif\|js)\$">
    Allow from all
</Files>

ServerSignature Off

- Limits access to specific file extensions
- Add your own extensions as needed
- Tells Apache not to report version
Fun with .htaccess
Add to /wp-admin/.htaccess

<Files ~ "\.(php)$">
Order Deny, Allow
Allow from 127.0.0.1
Deny from all
</Files>

- Limit /wp-admin/ access to localhost
- Access via SSH tunnel
- Change/Add IP for remote access from fixed network
Testing Wordpress
Yes, you can

- Useful tools
  - Plecost
    - http://code.google.com/p/plecost/
  - Netsparker
    - http://www.mavitunasecurity.com/
  - Acunetix (free edition, XSS only)
    - http://www.acunetix.com/
  - Burp Suit Pro
    - http://www.portswigger.net/
  - OpenVAS (with local checks)
Before you go live
Things to do

● Some ideas
  ● Use rewrite rules to redirect wp-login.php and /wp-admin to SSL only
  ● Lock down wp-admin, phpmyadmin etc.
  ● Minimise use of 3rd party plugins and themes

● Must do's before going live
  ● Audit your own code
  ● Audit 3rd party plugins and themes
After you go live

Things to do

- Audit plugin/theme upgrades prior to application
  - At least have a security process
- App test on major upgrades
- Read the changelog
  - Hunt the bug
  - Verify the fix
- Use liberal volumes of common sense
Thanks for having me
It keeps me off the streets

This presentation brought to you by DJ Shadow, UNKLE, Death in Vegas and Caffeine. Lots of sweet, sweet caffeine. My next talk will be at Bsides London on Breaking, Entering and Pentesting on April 20th and at DC4420 that evening about evading defences. CC-NC-SA ©2011 Mandalorian.