# Abuso di PHP

Discuteremo di:

- Costrutti particolari
- Ottimizzazione/profiling
- Gestione degli errori/exception
- Sicurezza
- Web stuff (smarty/xajax)

lesion@autistici.org

# Costrutti e funzioni particolari

- possono facilitare di molto la vita del programmatore;
- poco conosciuti, difficile trovarne casi d'uso

- operatore $$
- chiamate a funzioni da stringhe
- foreach con riferimento
- get_user_function
- debug_backtrace
- register_tick

# Costrutti particolari (operatore $$)

Attraverso questo operatore e' possibile fare riferimento ad una variabile o una funzione il cui nome non e' conosciuto a priori:

```
<?php
  $foo = 'contenuto di a';
  $bar = 'foo';
  echo $$bar;
?>
```

```
contenuto di a
```

# Costrutti particolari (operatore $$)

Vediamone alcuni utilizzi:

```php
<?php

  $page   = $_POST['page'];
  $option = $_POST['option'];
  $lang   = $_POST['lang'];
  $id     = $_POST['id']
  $user   = $_POST['user'];
  $pwd    = $_POST['pwd'];
  $altro  = $_POST['altro'];

?>
```

```php
<?php

  foreach( $_POST as $key => $value )
  {
    $$key = $value;
  }

?>
```

**ATTENZIONE:** nel secondo caso tutti gli elementi dell'array $_POST vengono considerati quindi occhio a non sovrascrivere variabili

# Costrutti particolari (chiamate a funzioni da stringhe)

Un'altra cosa interessante e' la chiamata di funzioni il cui nome e' contenuto all'interno di una variabile:

```
<?php

  if( $hashType == 'md5' )
  {
    $ret = md5( $string );
  }
  elseif( $hashType == 'sha1' )
  {
    $ret = sha1( $string );
  }

?>
```

```
<?php

  $ret == $hashType( $string );

?>
```

# Costrutti particolari (foreach con riferimento)

Dal PHP5 e' possibile accedere in scrittura agli elementi di un array passato ad un foreach utilizzando l'operatore &, vediamo come questo ci permette di semplificare del codice:

```php
<?php

  foreach( $array as $key => $element )
  {
    $array[$key] = 'nuovoValore';
  }

?>
```

```php
<?php

  foreach( $array as &$element )
  {
    $element = 'nuovoValore';
  }

?>
```

# Funzioni particolari (get_defined_functions)

- Perche' i linguaggi interpretati spaccano
- Autoreferenzialita'

```php
<?php

  function testMe(){ return; }

  $funzioni = get_defined_functions();
  $funzioniUtente = $functions['user'];

  //Array ( [0] => testme )
  print_r( $funzioniUtente );

?>
```

- Restituisce 2 array di funzioni dichiarate dall'utente e interne al php (internal, user)
- Un'idea furba e' farci una TEST SUITE

# Funzioni particolari (debug_backtrace)

```php
<?php

  function testMe( $arg )
  {
    var_dump ( debug_backtrace() );
  }

  testMe( 'arg' );

?>
```

Restituisce un backtrace dello stack PHP

# Funzioni particolari (register_tick_function)

- Un buffo concetto del PHP e' il tick. Un tick è un evento che si verifica ogni N istruzioni di basso livello eseguite dal parser.
- E' possibile chiamare una o piu' funzioni allo scatenarsi di questo evento attraverso la funzione register_tick_function.
- Il valore di N e' specificato usando la direttiva declare.

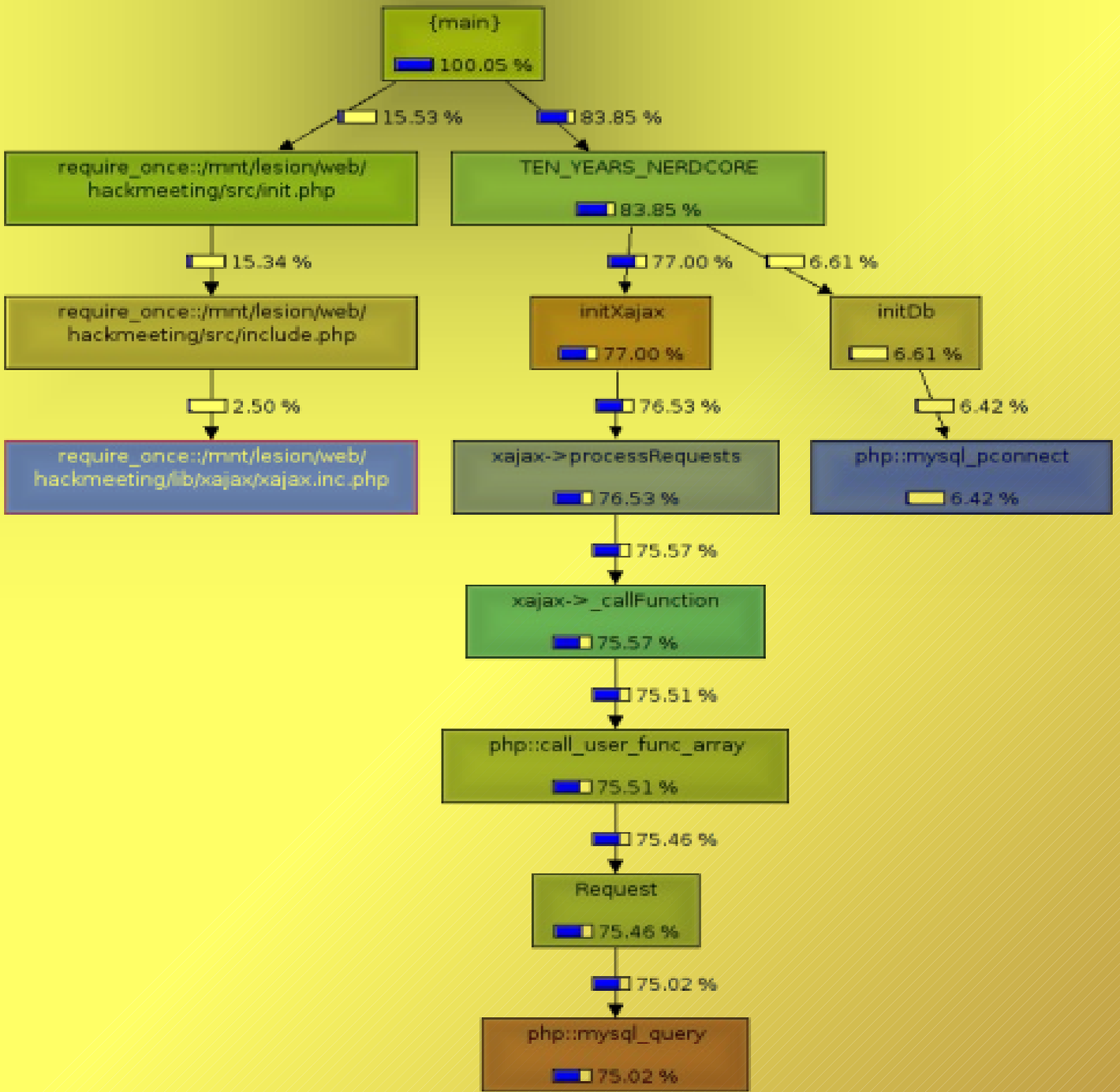
# Ottimizzazione e profilazione del codice

- L'ottimizzazione del codice e' un argomento decisamente vasto
- Da valutare specificatamente caso per caso perche' coinvolge moltissime variabili
- L'ottimizzazione seria prevede:
  - l'analisi dei colli di bottiglia
  - il tuning di webserver/db/fs ecc.
  - l'uso di moduli di caching appositi (eaccelerator, xcache)
  - compressione dell'output
- Qui di seguito invece soltanto pippe giocose

# Ottimizzazione e profilazione del codice

- Fasi del PHP (interpretazione/parsing, esecuzione del “byte code”)
- Differenze nell'ottimizzazione di queste due fasi
- Strumenti utili per la profilazione (xdebug, apd, kcachegrind)

# Ottimizzazione e profilazione del codice (kcachegrind)



# Argomento a scelta

- Template engine: smarty
- Ajax stuff: xajax
- Hack strani: gif upload/sessions's vuln
- Gestioni degli errori, eccezioni