SektionEins

The PHP Company

Secure Application Development with the Zend Framework

By Stefan Esser

© All rights reserved. SektionEins GmbH

Who?

- Stefan Esser
- from Cologne / Germany
- in IT security since 1998
- PHP core developer since 2001
- Month of PHP Bugs/Security and Suhosin
- Research and Development SektionEins GmbH

© All rights reserved. SektionEins GmbH

Part I

Introduction

© All rights reserved. SektionEins GmbH

Introduction

- Zend-Framework gets more and more popular
- Growing demand of secure development guidelines for applications based on the Zend-Framework
- Books / Talks / Seminars focus on secure programming of PHP applications without a framework
- Usage of frameworks requires different security guidelines
- Frameworks often come with own security features

© All rights reserved. SektionEins GmbH

Topics

- Central Authentication
- Central Input Validation and Filtering
- SQL Security
- Cross Site Request Forgery (CSRF) Protection
- Session Management Security
- Cross Site Scripting (XSS) Protection
- New attacks with old vulnerabilities

© All rights reserved. SektionEins GmbH

Part II

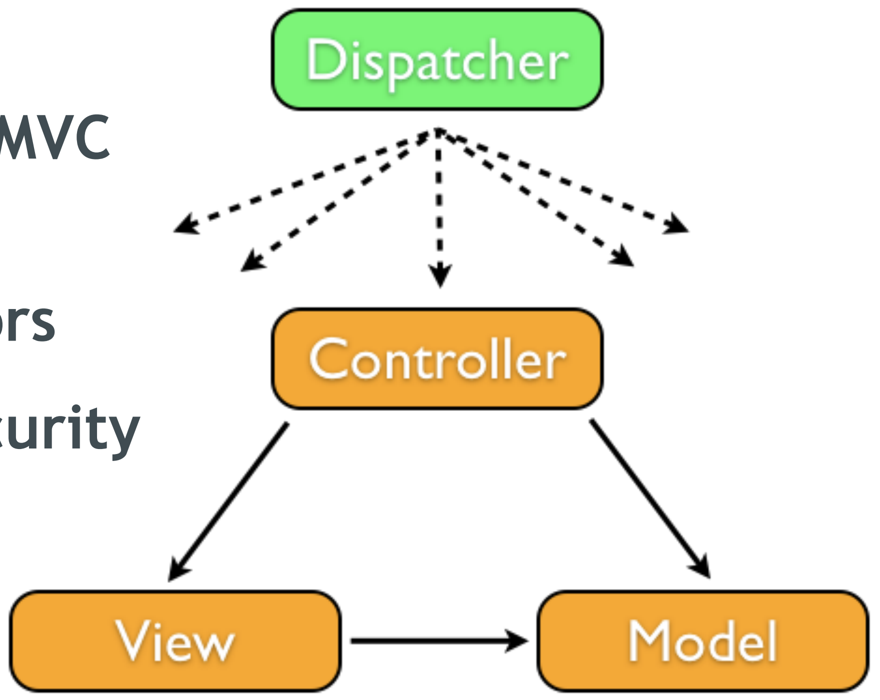
Central Authentication

and Input Validation and Filtering

© All rights reserved. SektionEins GmbH

Traditional Applications vs. Zend Framework

- Traditional applicationen have a lot of entrypoints
- ZF applications usually use the MVC design with a dispatcher
- Traditional way is prone to errors
- ZF way allows to implement security tasks in a central place
    - Input Validation and Filtering
    - Authentication



© All rights reserved. SektionEins GmbH

Front Controller Plugin

- Adding functionality to `Zend_Controller_Action`
- No class extension required
- Suitable for central tasks like authentication and input validation/filtering

```
$front = Zend_Controller_Front::getInstance();
$front->registerPlugin(new MyPlugin());
$front->dispatch();
```

© All rights reserved. SektionEins GmbH

Central Authentication

```
class ForceAuthPlugin extends Zend_Controller_Plugin_Abstract
{

public function preDispatch(Zend_Controller_Request_Abstract $request)
{
    try {

        My_Auth::isLoggedIn();

    } catch (My_Auth_UserNotLoggedInException $e) {

        if (!in_array($request->getController Name(),
                      array('login','index','error'))) {

            $request->setModuleName('default')
                    ->setControllerName('login')
                    ->setActionName('index')
                    ->setDispatched(false);
            return;
        }
    }
}

}
```

© All rights reserved. SektionEins GmbH

Central Input Validation/Filtering (I)

```
$filters['index']['index'] = array(
    '*'      => 'StringTrim',
    'month' => 'Digits'
);

$filters['login']['index'] = array(
    'login' => 'Alpha',
    'pass'  => 'Alpha'
);

$validators['index']['index'] = array(
    'month'    => array(
       new Zend_Validate_Int(),
       new Zend_Validate_Between(1, 12)
    )
);

$validators['login']['index'] = array(
    'login'   => array(
       new My_Validate_Username()
    ),
    'pass'   => array(
       new My_Validate_Password()
    ),
);
```

© All rights reserved. SektionEins GmbH

Central Input Validation/Filtering (II)

```
class FilterPlugin extends Zend_Controller_Plugin_Abstract
{
public function preDispatch(Zend_Controller_Request_Abstract $request)
{
    $params     = $request->getParams();
    $controller = $request->getControllerName();
    $action     = $request->getActionName();

    @$filter = $GLOBALS['filters'][$controller][$action];
    @$validator = $GLOBALS['validators'][$controller][$action];

    $input = new Zend_Filter_Input($filter, $validator, $params);

    if (!$input->isValid()) {
        $request->setModuleName('default')
                ->setControllerName('error')
                ->setActionName('illegalparam')
                ->setDispatched(false);
        return;
    }
}
```

© All rights reserved. SektionEins GmbH

Central Integration of PHPIDS

```
class Controller_Plugin_PHPIDS extends Zend_Controller_Plugin_Abstract
{
   public function preDispatch(Zend_Controller_Request_Abstract $request)
   {
      $request = array('GET' => $request->getQuery(),
         'POST' => $request->getPost(),
         'COOKIE' => $request->getCookie(),
         'PARAMS' => $request->getUserParams());

      $init = IDS_Init::init(APPLICATION_PATH.'/config/phpids.ini');
      $ids = new IDS_Monitor($request, $init);

      $result = $ids->run();
      if (!$result->isEmpty()) {
         $compositeLog = new IDS_Log_Composite();
         $compositeLog->addLogger(IDS_Log_Database::getInstance($init));
         $compositeLog->execute($result);
      }
   }
}
```

© All rights reserved. SektionEins GmbH

Part III

Formvalidation and -filtering

© All rights reserved. SektionEins GmbH

Input Validation/Filtering in Forms (I)

- ZF - Forms use validators/filters automatically
- Validators can be added to `Zend_Form_Element` objects
- Validators can be chained arbitrarily

```php
// Create Name Element
$name = $form->createElement('text', 'name', array('size' => 40, 'maxlength' => 40));
$name->addValidator('Alpha')
     ->addValidator('StringLength', false, array(1, 40))
     ->setLabel('Name')
     ->setRequired(true);

// Message Element
$message = $form->createElement('textarea', 'message', array('rows' => 6, 'cols' => 40));
$message->setLabel('Message')
        ->setRequired(true)
        ->addFilter('StripTags');

// Create Submit Button
$submit = $form->createElement('submit', 'send');
$submit->setLabel('Absenden');

// Add all Elements to the Form
$form->addElement($name)->addElement($message)->addElement($submit);
```

© All rights reserved. SektionEins GmbH

Input Validation/Filtering in Forms (II)

- Validation of form in the corresponding Action

```
// Validation of formdata
if (!$form->isValid($this->getRequest()->getPost()))
{
   // submit variables to view
   $this->view->form = $form;
   $this->view->title = "Form 1";

   // stop processing
   return $this->render('form');
}
```

© All rights reserved. SektionEins GmbH

Part IV

SQL-Security

© All rights reserved. SektionEins GmbH

SQL-Injection in Zend Framework Applications

- ZF comes with several classes for database access
- Methods usually support Prepared Statements

```
<?php
    $sql = "SELECT id FROM _users WHERE lastname=? AND age=?";
    $params = array('Mueller', '18');
    $res = $db->fetchAll($sql, $params);
?>
```

- Prep. Statements operating error allows SQL injection
- ZF also has escaping functions for dynamic SQL queries
- Lack of escaping leads to SQL injection

© All rights reserved. SektionEins GmbH

SQL-Injection + PDO_MySQL = Danger

- Traditionally MySQL allows only a single SQL query
  - ext/mysql – stops multi-queries completely
  - ext/mysli – has separate function `mysql_multi_query()`

- *ATTENTION: PDO_MySQL doesn‘t have this limitation*

- SQL injection in ZF Applicationen using PDO_MySQL is more dangerous than in applications using the traditional MySQL interfaces

© All rights reserved. SektionEins GmbH

Zend_Db - Escaping

```
function quote($value, $type = null)
```

- Always the right escaping – one function instead of many
- ATTENTION: strings are put between single quotes

```
function quoteIdentifier($ident, $auto=false)
```

- Escaping function for identifiers
- Traditional PHP applications have to implement their own
- ATTENTION: strings are put between single quotes

© All rights reserved. SektionEins GmbH

Zend_Db_Select

- To create somewhat dynamic SQL SELECT queries
- Uses Prepared Statements internally as much as possible
- SQL injection possible if wrongly used
    - ATTENTION especially at **WHERE** and **ORDER BY**

```
// Build this query:
//   SELECT product_id, product_name, price
//   FROM "products"
//   WHERE (price < 100.00 OR price > 500.00)
//     AND (product_name = 'Apple')

$minimumPrice = 100;
$maximumPrice = 500;
$prod = 'Apple';

$select = $db->select()
             ->from('products',
                    array('product_id', 'product_name', 'price'))
             ->where("price < $minimumPrice OR price > $maximumPrice")
             ->where('product_name = ?', $prod);
```

© All rights reserved. SektionEins GmbH

Part V

Cross Site Request Forgery (CSRF) Protection

© All rights reserved. SektionEins GmbH

Cross Site Request Forgery (CSRF) Protection

- CSRF protections are usually based on secret, session dependent form tokens
- Zend-Framework has `Zend_Form_Element_Hash` which is such a token with built-in validator
- Forms can be safeguarded against CSRF by adding the form element

```
$form->addElement('hash', 'csrf_token',
                  array('salt' => 's3cr3ts4ltG%Ek@on9!'));
```

© All rights reserved. SektionEins GmbH

Automatic CSRF Protection

- Protection has to be applied manually
- By extending the `Zend_Form` class it is possible to create a new form class with automatic CSRF protection

```
<?php
class My_Form extends Zend_Form
{
    function __construct()
    {
        parent::__construct();
        $this->addElement('hash', 'csrf_token',
                    array('salt' => get_class($this) . 's3cr3t%Ek@on9!'));
    }
}
?>
```

© All rights reserved. SektionEins GmbH

Part VI

Session Management Security

© All rights reserved. SektionEins GmbH

SektionEins

Session Management Configuration

- Configuration has big impact on session security
- SSL applications muse be secured with the *secure* flag
- Own session name / session storage for each application
- Hardening against XSS with the *httpOnly* flag
- Setting the maximum lifetime

```
<?php
Zend_Session::setOptions(array(
  /* if SSL server */         'cookie_secure'   => true,
  /* own session name */      'name'            => 'mySSL',
  /* own session storage */   'save_path'       => '/sessions/mySSL',
  /* hardening against XSS */ 'cookie_httponly' => true,
  /* short lifetime */        'gc_maxlifetime'  => 15 * 60
                         ));
Zend_Session::start();
?>
```

© All rights reserved. SektionEins GmbH

Session Fixation and Session Hijacking

- **Session Fixation**
  - Gets slightly harder with session validation/strict session handling
  - Stopped by regenerating a new session id on each status change
    - session_regenerate_id(true);
  - Best implemented in the login/logout module
- **Session Hijacking**
  - Only stoppable by using SSL (to stop network sniffing)
  - *httpOnly* cookies protect against session id theft by XSS
  - Session validation only of limited use

© All rights reserved. SektionEins GmbH

Session Validation (I)

- Recognizes valid sessions by checking additional information
- Often recommended to stop session fixation / hijacking - but only limited usefullness
- Zend Framework supports session validators
    - Zend_Session_Validator_HttpUserAgent

```
<?php
try {

    Zend_Session::start();

} catch (Zend_Session_Exception $e) {

    // Zend_Session::regenerate_id() support is broken
    session_regenerate_id(true);

}
Zend_Session::registerValidator(new Zend_Session_Validator_HttpUserAgent());
?>
```

© All rights reserved. SektionEins GmbH

Session Validation (II)

- Be careful when checking additional information fields
- User-Agent HTTP header checks problematic since atleast Microsoft Internet Explorer 8
- Accept HTTP header checks already a problem with earlier versions of Internet Explorer
- Checking the client IP not possible for users of big proxies / companies / ISPs
  - Limit check to class C / B / A network
  - Better compatibility with SSL sites

© All rights reserved. SektionEins GmbH

Session Validation - Valididating the Client IP

```
class Zend_Session_Validator_RemoteAddress extends Zend_Session_Validator_Abstract
{
    /**
     * Setup() - this method will get the client's remote address and store
     * it in the session as 'valid data'
     *
     * @return void
     */
    public function setup()
    {
        $this->setValidData( (isset($_SERVER['REMOTE_ADDR'])
            ? $_SERVER['REMOTE_ADDR'] : null) );
    }
    /**
     * Validate() - this method will determine if the client's remote addr
     * matches the remote address we stored when we initialized this variable.
     *
     * @return bool
     */
    public function validate()
    {
        $currentBrowser = (isset($_SERVER['REMOTE_ADDR'])
            ? $_SERVER['REMOTE_ADDR'] : null);

        return $currentBrowser === $this->getValidData();
    }
}
```

© All rights reserved. SektionEins GmbH

Part VI

Cross Site Scripting (XSS) Protection

© All rights reserved. SektionEins GmbH

XSS in Zend Framework Applications

- Symfony comes with an automatic output escaping
- Zend Framework comes without such things
- stopping XSS is task of the programmer
- XSS vulnerabilities occur in the “view“

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title><?php echo $this->title; ?></title>
</head>
<body>
<h2><?php echo $this->headline; ?></h2>
<ul>
<li><a href="<?php echo $this->link; ?>">Link 1</a></li>
</ul>
</body>
</html>
```

© All rights reserved. SektionEins GmbH

Protection against XSS (I)

- Traditionally: Two alternatives
  - Outputting only clean values

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title><?php echo $this->escape($this->title); ?></title>
</head>
<body>
<h2><?php echo $this->escape($this->headline); ?></h2>
<ul>
<li><a href="<?php echo urlprepare($this->link); ?>">Link 1</a>
</li>
</ul>
</body>
</html>
```

© All rights reserved. SektionEins GmbH

Protection against XSS (II)

- **Traditionally: Two alternatives**
  - Assigning only clean values

```
$entityFilter = new Zend_Filter_HtmlEntities();
$urlFilter    = new My_Filter_Url();

$this->view->title    = $this->escape("Seite 1");
$this->view->headline = $entitiyFilter->filter($this->getRequest()
                                                     ->getPost('link'));
$this->view->link     = $urlFilter->filter($this->getRequest()
                                                 ->getPost('link'));
```

© All rights reserved. SektionEins GmbH

Safeguarding with Zend_View_Helper

- Traditional solutions prone to erros - any mistake == XSS
- Automatic scanning for failure to apply filters is hard
- Prohibit direct output of variables
- Output only via `Zend_View_Helper`
- XSS protection becomes task of `Zend_View_Helper`

```
<form action="action.php" method="post">
    <p><label>Your Email:
<?php echo $this->formText('email', 'you@example.com', array('size' => 32)) ?>
    </label></p>
    <p><label>Your Country:
<?php echo $this->formSelect('country', 'us', null, $this->countries) ?>
    </label></p>
    <p><label>Would you like to opt in?
<?php echo $this->formCheckbox('opt_in', 'yes', null, array('yes', 'no')) ?>
    </label></p>
</form>
```

© All rights reserved. SektionEins GmbH

Part VII

unserialize() and User Input

© All rights reserved. SektionEins GmbH

SektionEins

unserialize() and User Input

- **Never use unserialize() on user input !**
  - Properties can be filled arbitrarily – even private ones
  - __destruct() and __wakeup() methods will be executed
  - autoloader allows loading arbitrary objects

- **Zend Framework comes with a lot of classes**
  - combination of classes allow hijacking the control flow

© All rights reserved. SektionEins GmbH

unserialize() - Example Exploit

- **Classes of Zend-Framework allow**
  - Upload of arbitrary files
  - Execution of arbitrary PHP Code (ZF >= 1.8.0)
  - Sending SPAM Emails (ZF >= 1.8.0)
  - Inclusion of arbitrary files (ZF >= 1.9.0)

© All rights reserved. SektionEins GmbH

Zend_Queue_Adapter_Activemq

```php
class Zend_Queue_Adapter_Activemq extends
Zend_Queue_Adapter_AdapterAbstract
{
    ...
    /**
      * Close the socket explicitly when destructed
      *
      * @return void
      */
     public function __destruct()
     {
         // Gracefully disconnect
         $frame = $this->_client->createFrame();
         $frame->setCommand('DISCONNECT');
         $this->_client->send($frame);
         unset($this->_client);
     }
```

Zend_Queue_Adapter_Activemq

_client

© All rights reserved. SektionEins GmbH

Zend_Queue_Stomp_Client_Connection

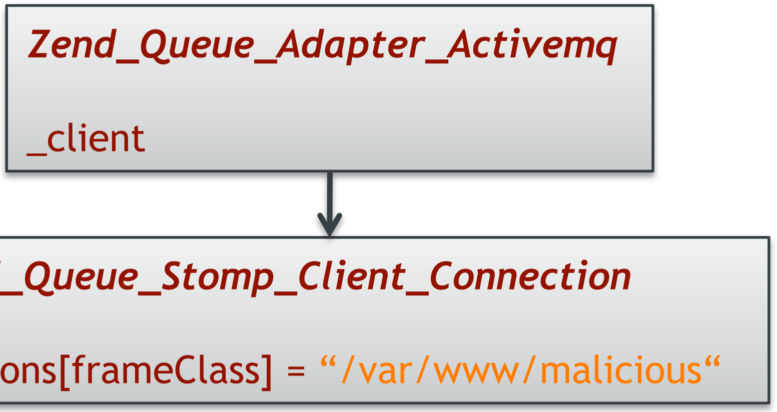
```
class Zend_Queue_Stomp_Client_Connection
    implements Zend_Queue_Stomp_Client_ConnectionInterface
{
    ...
    public function getFrameClass()
    {
        return isset($this->_options['frameClass'])
            ? $this->_options['frameClass']
            : 'Zend_Queue_Stomp_Frame';
    }

    public function createFrame()
    {
        $class = $this->getFrameClass();

        if (!class_exists($class)) {
            require_once 'Zend/Loader.php';
            Zend_Loader::loadClass($class);
        }

        $frame = new $class();
        ...
```

Zend_Queue_Stomp_Client_Connection

_options[frameClass]

© All rights reserved. SektionEins GmbH

Combined


Zend_Queue_Adapter_Activemq
_client
Zend_Queue_Stomp_Client_Connection
_options[frameClass] = “/var/www/malicious“

```
O:27:"Zend_Queue_Adapter_Activemq":1:{s:
36:"\0Zend_Queue_Adapter_Activemq\0_client";O:
34:"Zend_Queue_Stomp_Client_Connection":1:{s:11:"\0*\0_options";a:1:
{s:10:"frameClass";s:18:"/var/www/malicious";}}}
```

© All rights reserved. SektionEins GmbH

Questions ?

Get Audited by Web Security Experts

http://www.sektioneins.com

© All rights reserved. SektionEins GmbH