SECURE FORM PROCESSING AND PROTECTION

Joe Ferguson

@JoePFerguson

https://joind.in/13441

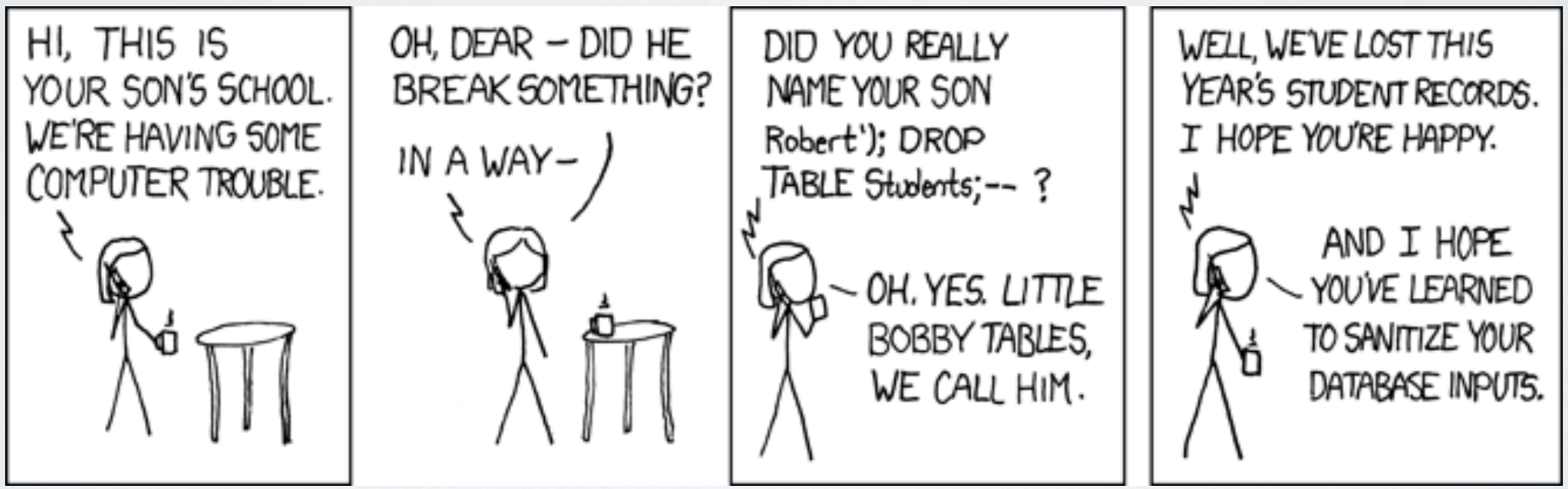
"WHAT KEEPS YOU UP AT NIGHT?"

FOR ME, IT WAS FORM PROCESSING

- for a while, at least

“HOW DO I SAFELY, SECURELY, AND RELIABLY GET INPUT FROM MY USERS?”

LITTLE BOBBY TABLES



http://xkcd.com/327/

VULNERABILITIES

CROSS SITE SCRIPTING (XSS)

"XSS enables attackers to inject client-side script into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin policy"

http://en.wikipedia.org/wiki/Cross-site_scripting

THERE IS NO STANDARD CLASSIFICATION OF XSS

TYPES OF XSS EXPLOITS

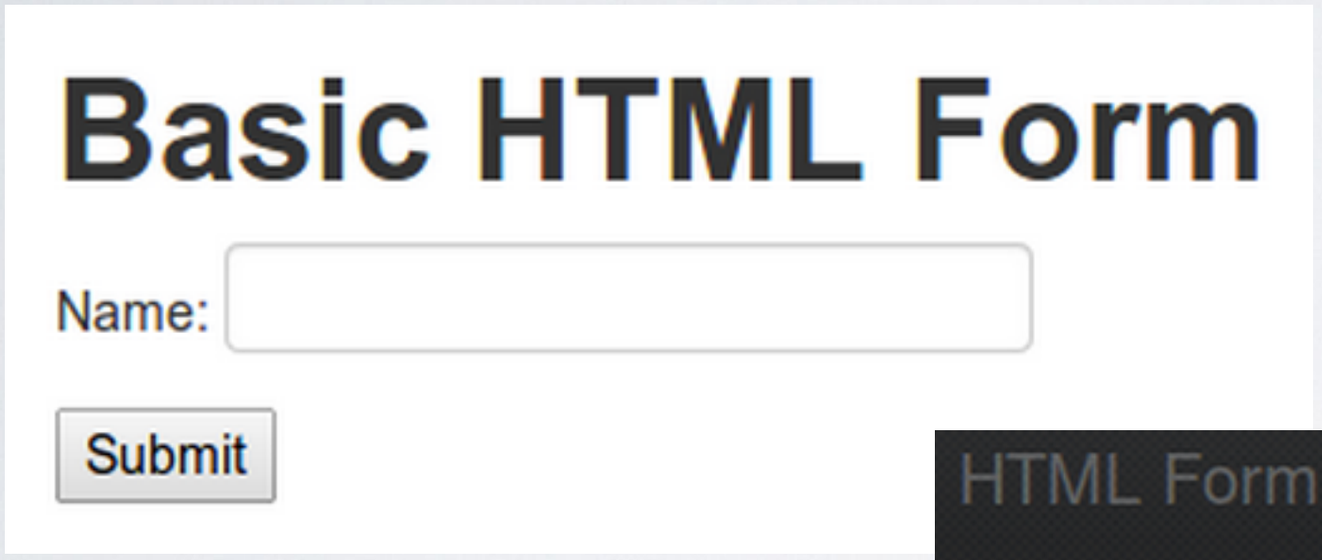
- Reflected (Non-persistent)
- Persistent

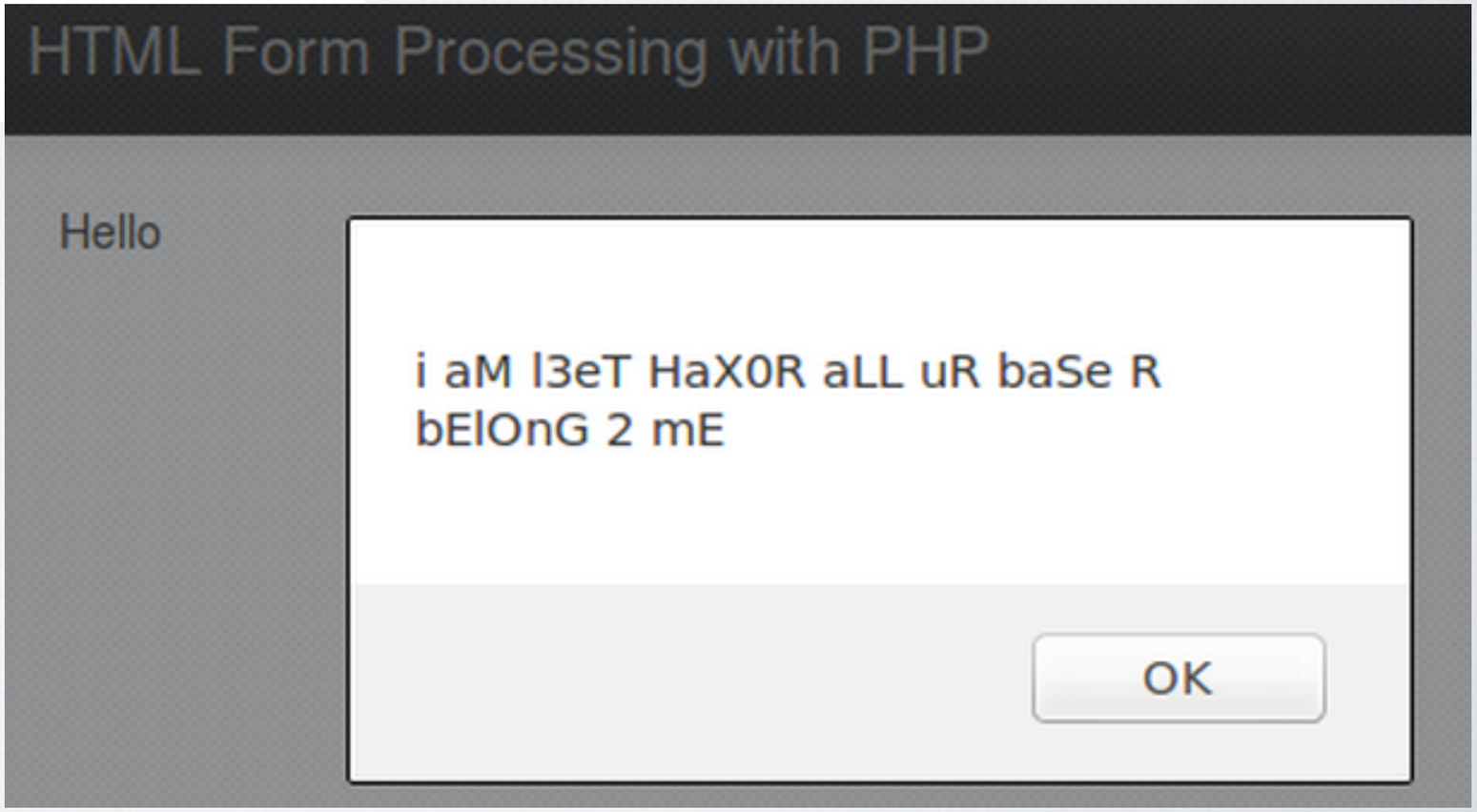
Can also be distinguished by:

- Server-side versus DOM-based vulnerabilities

REFLECTED (NON-PERSISTENT)

Data passed to the app immediately without sanitizing the data

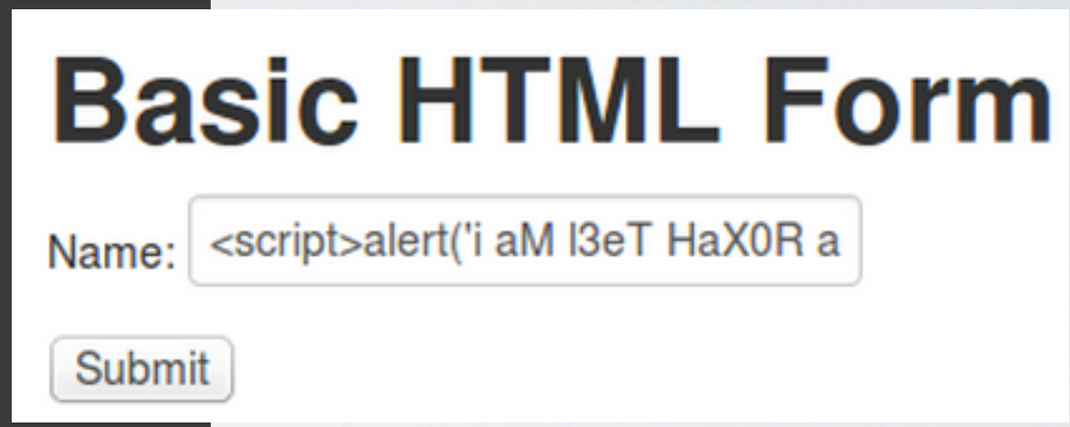




WHAT HAPPENED?

Example1.php - Secure-Form-Processing-ar

```php
<div class="container">
    <?php
    if(isset($_POST['name'])){
        //form has been submitted
        ?>
        Hello <?php echo $_POST['name']; ?>
    <?php
    } else {
        ?>
        <h1>Basic HTML Form</h1>
        <form name="basic_form" method="POST" action="#">
            <label>Name:
                <input type="text" name="name" id="name" value="">
            </label>
            <label>
                <input type="submit" id="submit" value="Submit">
            </label>
        </form>
    <?php
    }
    ?>
</div> <!-- /container -->
```



http://www.phparch.com/magazine/2014-2/august/

PERSISTENT

Data passed to the app is saved by the server

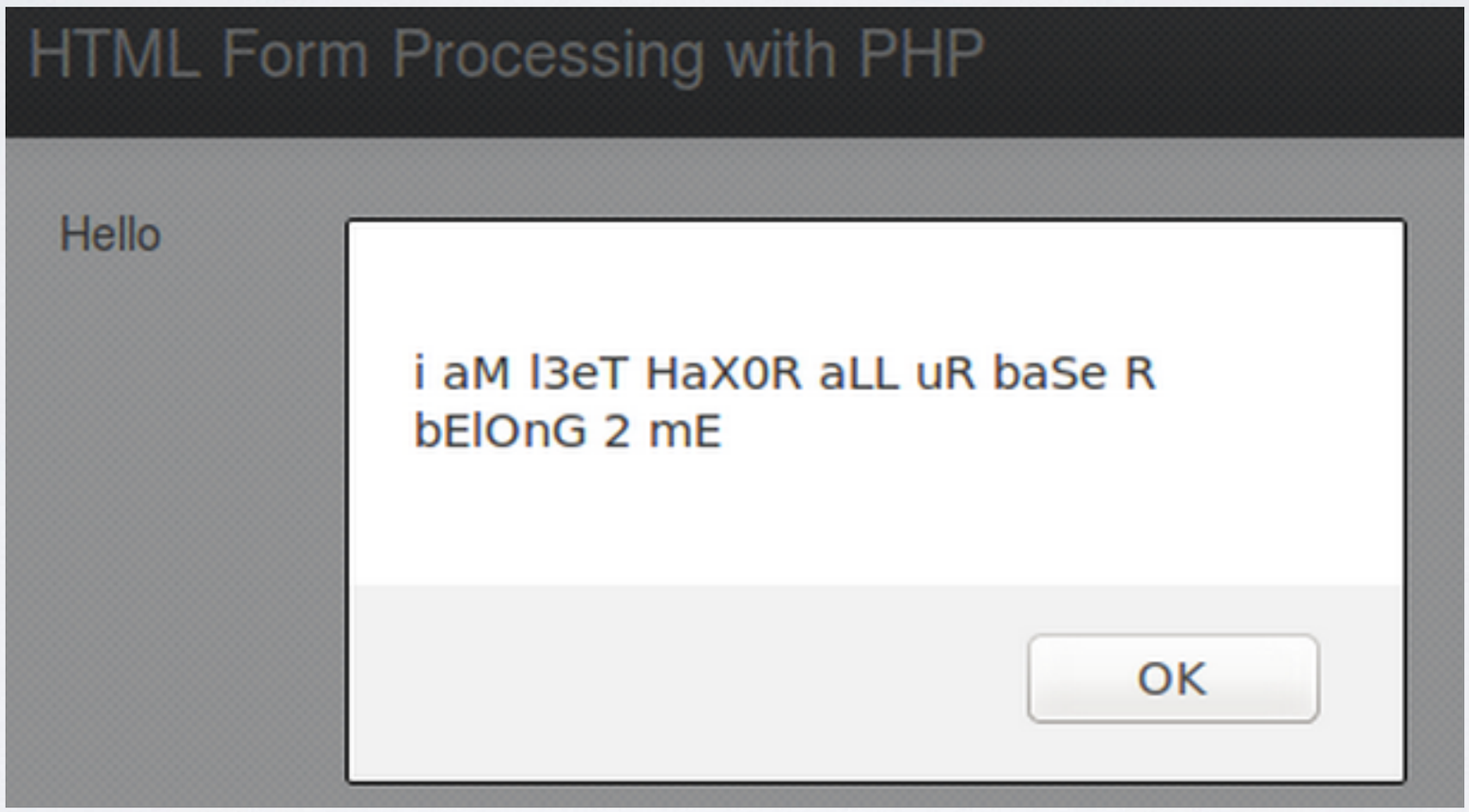
When the code to display the dynamic data is run again, the code that was inject runs again.

DATA PASSED TO THE APP IS SAVED BY THE SERVER

```
<div class="container">
    <?php
    if(isset($_POST['name'])){
        //form has been submitted
        $user = new User;
        $user->name = $_POST['name'];
        $user->save();
        ?>
        Hello <?php echo $_POST['name']; ?>
    <?php
    } else {
        ?>
        <h1>Basic HTML Form</h1>
        <form name="basic_form" method="POST" action="#">
            <label>Name:
                <input type="text" name="name" id="name" value="">
            </label>
            <label>
                <input type="submit" id="submit" value="Submit">
            </label>
        </form>
    <?php
    }
    ?>
</div> <!-- /container -->
```

INJECTED CODE RUNS AGAIN

Wherever the dynamic content is called, the injected code runs



http://www.phparch.com/magazine/2014-2/august/

SERVER-SIDE VERSUS DOM-BASED VULNERABILITIES

- Examples:
  - Single page applications (JavaScript)
  - Still need to protect these applications
  - Malicious code doesn't touch server, only DOM

WIDESPREAD XSS EXPLOITS

- Twitter September 21, 2010 "MouseOver"
  - tweeting a JavaScript function for "onMouseOver"
  - Victims would mouseover areas of a tweet that looked like highlighted areas and code would execute to tweet out the same exploit from their account.

http://en.wikipedia.org/wiki/Twitter

WIDESPREAD XSS EXPLOITS

- Facebook Early 2013 Chat & Checkin vulnerable
  - Chat: GUI for presenting a link to chat window was unfiltered / not sanitized.
  - Checkin: Attacker could post malicious scripts in pages and code would run when victims checked in to location

http://thehackernews.com/2013/04/hacking-facebook-users-just-from-chat.html

WIDESPREAD XSS EXPLOITS

- MySpace October 2005 Samy (computer worm)
  - Added an XSS on a profile that would posted to the victims own profile.
- The exploit spread like a worm virus infecting new users whenever an infected profile was viewed

http://en.wikipedia.org/wiki/Samy_%28computer_worm%29

CROSS SITE REQUEST FORGERY (CSRF)

Sending unauthorized commands
from a user that an **application trusts**

Relies on tricking a user into viewing a malicious image
or clicking on a malicious link.

CSRF CHARACTERISTICS

- Targets a site that knows about the victim
- Exploit the trust (often logged in state) of victim
- Trick victim into sending HTTP requests to target
- HTTP requests have side effects (malicious intent)

LOGIN CSRF

Used to log a user into an application

GOOGLE YOUTUBE CROSSDOMAIN SECURITY FLAW

- *.google.com was trusted

- Send a malicious SWF file to the attacker's gmail and locate the download URL

- Logged in YouTube user visits attacker's malicious page

GOOGLE YOUTUBE CROSSDOMAIN SECURITY FLAW

- Force user to authenticate and exploit a login-CSRF / session initialization vulnerability to authenticate the victim as the attacker.

- Attacker embeds the malicious SWF file to the page the victim viewing.

- Attacker now has read/write access to victim's YouTube account

http://jeremiahgrossman.blogspot.com/2008/09/i-used-to-know-what-you-watched-on.html

DYNAMIC CSRF

- Attacks can be changed based on the origin of the request.
- Dynamically created as part of an XSS exploit
- Customized payloads to specific targets
- Usually involves relying on session data getting leaked cross domain

CSRF LIMITATIONS

- Target site that doesn't check referrer header or the victim's browser supports referrer spoofing
- The attacker must target some submission point on the victim's computer (changes / reads of victim's personal information, modify bank account records, etc)

CSRF LIMITATIONS

- The attacker must determine the correct values to submit to the application
- The victim must be logged into the target application

CSRF ATTACKS ARE BLIND

REPLAY ATTACKS

http://en.wikipedia.org/wiki/Replay_attack

SCARED YET?

OF COURSE NOT!

THIS SHOULDN'T BE
THE FIRST TIME YOU
HAVE HEARD THESE TERMS

"An ounce of prevention is worth a pound of cure"

– Benjamin Franklin

CRYPTOGRAPHIC NONCE

Preventing Replay Attacks and CSRF

CRYPTOGRAPHIC NONCE

- Arbitrary number used ONCE in a cryptographic communication

- Used in HTTP digest access authentication to has the password . Nonce changes every time the 401 response is presented.

- Use to prevent replay attacks.

EXAMPLE NONCE IN PHP

```php
<?php
ini_set('default_mimetype', 'text/plain');
ini_set('default_charset', 'ISO-8859-1');
define('NONCE_SECRET', 'MemphisPHP!');

require_once('NonceUtil.php');

print "generating a nonce with a 1 second lifetime.\n";
$nonce = NonceUtil::generate(NONCE_SECRET, 1);

print "check nonce (nonce should be valid): ";
$r = NonceUtil::check(NONCE_SECRET, $nonce);
var_dump($r);

print "\n";

print "generating a nonce with a 1 second lifetime.\n";
$nonce = NonceUtil::generate(NONCE_SECRET, 1);

print "wait 2 seconds.\n";
sleep(2);

print "check nonce (nonce should be invalid): ";
$r = NonceUtil::check(NONCE_SECRET, $nonce);
var_dump($r);

```

https://github.com/timostamm/NonceUtil-PHP

USING WORDPRESS & NONCE CURIOUS?

- WordPress has it's own internal NONCE System
- It isn't a true NONCE since you can use it more than once.
- More info:
  - https://www.getpantheon.com/blog/nonce-upon-time-wordpress
  - Written by Cal Evans

PREVENTING XSS

HTMLENTITIES()

- Convert all applicable characters to HTML entities
- This function is identical to htmlspecialchars() in all ways, except with htmlentities(), all characters which have HTML character entity equivalents are translated into these entities.

http://php.net/htmlentities

```php
<div class="container">
    <?php
    if(isset($_POST['name'])){
        //form has been submitted
        ?>
        Hello <?php echo htmlentities($_POST['name']); ?>
    <?php
    } else {
        ?>
        <h1>Basic HTML Form</h1>
        <form name="basic_form" method="POST" action="#">
            <label>Name:
                <input type="text" name="name" id="name" value="">
            </label>
            <label>
                <input type="submit" id="submit" value="Submit">
            </label>
        </form>
    <?php
    }
    ?>
</div> <!-- /container -->
```

FILTER_VAR()

- Filters a variable with a specified filter
- Returns the filtered data, or FALSE if the filter fails.
- Example Filters:
  - FILTER_VALIDATE_EMAIL
  - FILTER_VALIDATE_INT

http://php.net/manual/en/function.filter-var.php

SANITIZE WITH FILTER_VAR()

- Sanitize incoming or outgoing data
- Example Filters:
  - FILTER_SANITIZE_EMAIL
  - FILTER_SANITIZE_STRING
  - FILTER_SANITIZE_NUMBER_INT
  - FILTER_SANITIZE_URL

http://php.net/manual/en/filter.filters.sanitize.php

```php
<div class="container">
    <?php
    if(isset($_POST['name'])){
        //form has been submitted
        $salutation = filter_var($_POST['salutation'], FILTER_SANITIZE_STRING);
        $name = filter_var($_POST['name'], FILTER_SANITIZE_STRING);
        $greeting = 'Hello ' . $salutation . ' ' . $name;
        echo $greeting;
    } else {
        ?>
        <h1>Sanitizing Data</h1>
        <form name="basic_form" method="POST" action="#">
            <label>Salutation:
                <select name="salutation" id="salutation">
                    <option value="Hello">Hello</option>
                    <option value="Mrs.">Mrs.</option>
                    <option value="Mr.">Mr.</option>
                </select>
            </label>
            <label>Name:
                <input type="text" name="name" id="name" value="">
            </label>
            <label>
                <input type="submit" id="submit" value="Submit">
            </label>
        </form>
    <?php
    }
    ?>
</div> <!-- /container -->
```

MANY FRAMEWORKS HAVE THIS BUILT IN

ANGULARJS

- Angular calls it XSRF
- Server needs to set a JavaScript readable cookie "X-XSRF-TOKEN"
- Unique per user and be verifiable by the server

https://docs.angularjs.org/api/ng/service/$http

ZEND

- Zend\Escaper contains methods for escaping output

- Zend\Filter contains common data filters

- Zend\Form\Element\Csrf Protection is achieved by adding a hash element to a form and verifying it when the form is submitted.

http://framework.zend.com/manual/current/en/modules/zend.form.element.csrf.html

ZEND CSRF PROTECTION

```php
<?php

use Zend\Form\Element;
use Zend\Form\Form;

$csrf = new Element\Csrf('csrf');

$form = new Form('project-create');
$form->add($csrf);
```

http://framework.zend.com/manual/current/en/modules/zend.form.element.csrf.html

ZEND ESCAPING OUTPUT

```php
<?php
$escaper = new Zend\Escaper\Escaper('utf-8');
?>

<h4>Project</h4>

<div class="well clearfix">
    <div class="col-md-8">
        <p><strong>Title</strong>:
            <?php echo $escaper->escapeHtml($project->title); ?></p>
        <p><strong>Description</strong>:
            <?php echo $escaper->escapeHtml($project->description); ?></p>
        <p><strong>Status</strong>:
            <?php echo $escaper->escapeHtml($project->status.title); ?></p>
        <p><strong>Priority</strong>:
            <?php echo $escaper->escapeHtml($project->priority.title); ?></p>
        <p><strong>Owner</strong>:
            <?php echo $escaper->escapeHtml($project->owner.username); ?></p>
    </div>
```

http://framework.zend.com/manual/current/en/modules/zend.escaper.escaping-html.html

SYMFONY

- Generate CSRF Token (Symfony\Component\Form\Extension\Csrf\CsrfProvider)
  - {{ csrf_token('authenticate') }}
- Twig Template can default to automatic escaping
  - If disabled: {{ user.username|e }}

SYMFONY CSRF PROTECTION

```
{# ... #}
    <form action="{{ path('login_check') }}" method="post">
    {# ... the login fields #}

        <input type="hidden" name="_csrf_token"
               value="{{ csrf_token('authenticate') }}">

    <button type="submit">login</button>
</form>
```

http://symfony.com/doc/current/cookbook/security/csrf_in_login_form.html

SYMFONY ESCAPING OUTPUT

If the escaper extension is enabled, escaping is automatic.
Otherwise you can use :

```
@extends('layouts.default')

@section('content')
    <h4>Project</h4>

    <div class="well clearfix">
        <div class="col-md-8">
            <p><strong>Title</strong>: {{ project.title|e }}</p>
            <p><strong>Description</strong>: {{ project.description|e }}</p>
            <p><strong>Status</strong>: {{ project.status.title|e }}</p>
            <p><strong>Priority</strong>: {{ project.priority.title|e }}</p>
            <p><strong>Owner</strong>: {{ project.owner.username|e }}</p>
        </div>
```

http://twig.sensiolabs.org/doc/templates.html

SLIMPHP

- Slim-Extras - Slim Authentication and XSS Middlewares

- Slim\Extras\Middleware\CsrfGuard

https://github.com/codeguy/Slim-Extras

LARAVEL

- Query Builder uses PDO parameter binding to protect against SQL injection
- Automatically handles CSRF when using Form::open
- Escape output by using {{{ $input }}} in Blade

LARAVEL CSRF PROTECTION

```
@extends('layouts.default')

@section('content')
    <div class="well">
        {{ Form::open(array('action' => 'ProjectsController@store')) }}

        <h4>Create New Project</h4>

        <div class="form-group {{ ($errors->has('title')) ? 'has-error' : '' }}">
            {{ Form::text('title', null, array('class' => 'form-control', 'placeholder' => 'Title')) }}
            {{ ($errors->has('title') ? $errors->first('title') : '') }}
        </div>
```

LARAVEL CSRF PROTECTION

```
<div class="well">
<form method="POST" action="http://simplelance.dev/projects" accept-charset="UTF-8">
    <input name="_token" type="hidden" value="MxFONpFcVcg8qiRDqM0tTq5kFBNxNWUnhtV4rzhm">

    <h4>Create New Project</h4>

    <div class="form-group ">
        <input class="form-control" placeholder="Title" name="title" type="text">

    </div>
```

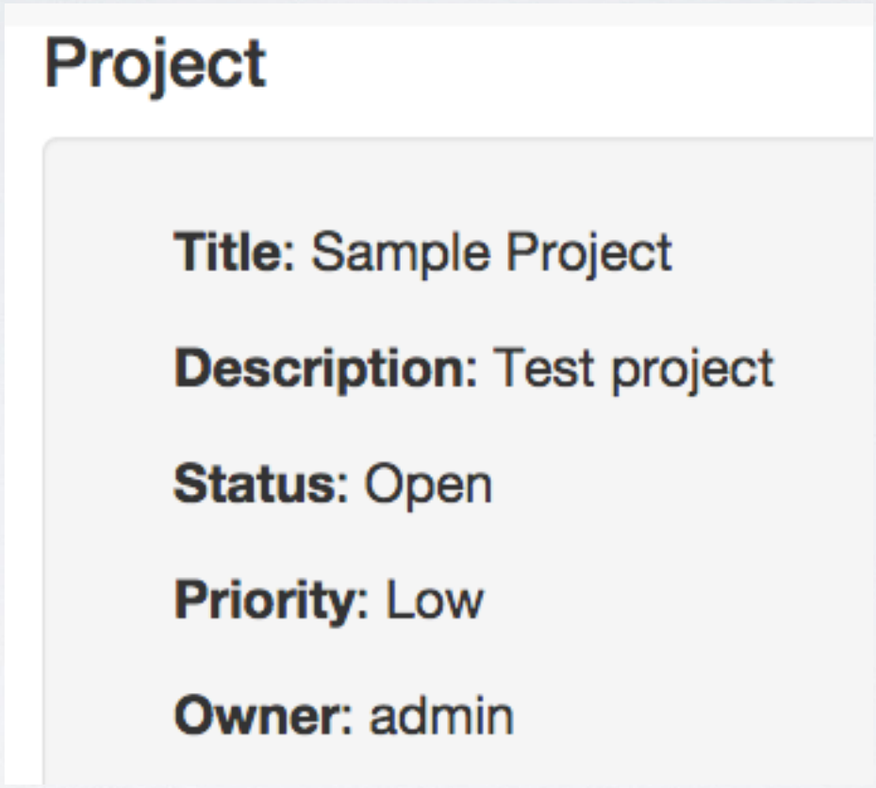
LARAVEL CSRF PROTECTION

```
70  /*
71   |--------------------------------------------------------------------------
72   | CSRF Protection Filter
73   |--------------------------------------------------------------------------
74   |
75   | The CSRF filter is responsible for protecting your application against
76   | cross-site request forgery attacks. If this special token in a user
77   | session does not match the one given in this request, we'll bail.
78   |
79  */
80
81  Route::filter('csrf', function()
82  {
83      if (Session::token() !== Input::get('_token'))
84      {
85          throw new Illuminate\Session\TokenMismatchException;
86      }
87  });
```

LARAVEL ESCAPING OUTPUT

```
1   @extends('layouts.default')
2
3   @section('content')
4       <h4>Project</h4>
5
6       <div class="well clearfix">
7           <div class="col-md-8">
8               <p><strong>Title</strong>: {{{ $project->title }}}</p>
9               <p><strong>Description</strong>: {{{ $project->description }}}</p>
10              <p><strong>Status</strong>: {{{ $project->status->title }}}</p>
11              <p><strong>Priority</strong>: {{{ $project->priority->title }}}</p>
12              <p><strong>Owner</strong>: {{{ $project->owner->username }}}</p>
13          </div>
```

LARAVEL ESCAPING OUTPUT



OTHER FRAMEWORKS

- Check the documentation for best practices!

XSS TESTING TOOLS

- Acunetix Web Vulnerability Scanner
  - http://www.acunetix.com
- IBM Security AppScan
  - http://www-03.ibm.com/software/products/en/appscan
- Burp Suite
  - http://portswigger.net/burp
- OWASP Zed Attack Proxy Project
  - https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

LINKS

- Examples & Links:
    - https://github.com/svpernova09/Secure-Form-Processing-and-Protection-Talk
- http://en.wikipedia.org/wiki/Cross-site_scripting
- http://en.wikipedia.org/wiki/Cross-site_request_forgery
- http://securingphp.com
- "HTML Form Processing with PHP" Article:
    - http://www.phparch.com/magazine/2014-2/august/
- Leave me feedback: https://joind.in/13441