

Defending against Users

International PHP Conference

May 3rd, 2005. Amsterdam, the Netherlands

Derick Rethans <dr@ez.no>

<http://derickrethans.nl/talks.php>



What is Wrong Here?

```
<?php
    $sql = "
        SELECT card_num, card_name, card_expiry
        FROM    credit_cards
        WHERE   uid = '{$_GET['uid']}'
    ";
?>
```

<http://example.com/script.php?uid=42>

```
SELECT card_num, card_name, card_expiry
FROM    credit_cards
WHERE   uid = '42'
```

!

<http://example.com/script.php?uid=42'%20or%20''='>

```
SELECT card_num, card_name, card_expiry
FROM    credit_cards
WHERE   uid = '42' or ''=''
```

What is Wrong Here?

```
<html>  
  <head>  
    <title>Example</title>  
  </head>  
  <body>  
    Name: <?php echo $_GET['name']; ?>  
  </body>  
</html>
```

http://example.com/script.php?name=derick

Name: derick

!

http://example.com/script.php?name=<script>alert('!');</script>



```
<?php
    $uid = (int) $_GET['uid'];
    $sql = "
        SELECT card_num, card_name, card_expiry
        FROM    credit_cards
        WHERE   uid = '{$uid}'
    ";
?>
```

http:

```
SELECT card_num, card_name, card_expiry
FROM    credit_cards
WHERE   uid = '42'
```

: -)

http:

```
SELECT card_num, card_name, card_expiry
FROM    credit_cards
WHERE   uid = '42'
```

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    Name: <?php echo htmlentities($_GET['name']); ?>
  </body>
</html>
```

<http://example.com/script.php?name=derick>

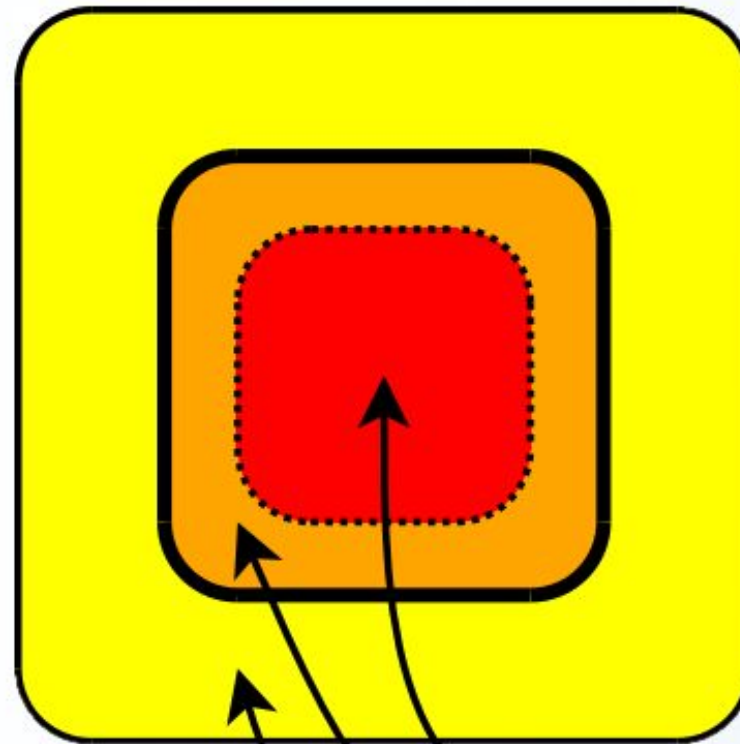
Name: derick

:-)

[http://example.com/script.php?name=<script>alert\('!!'\);</script>](http://example.com/script.php?name=<script>alert('!!');</script>)

<script>alert('!!');</script>

Multiple Levels of Defense



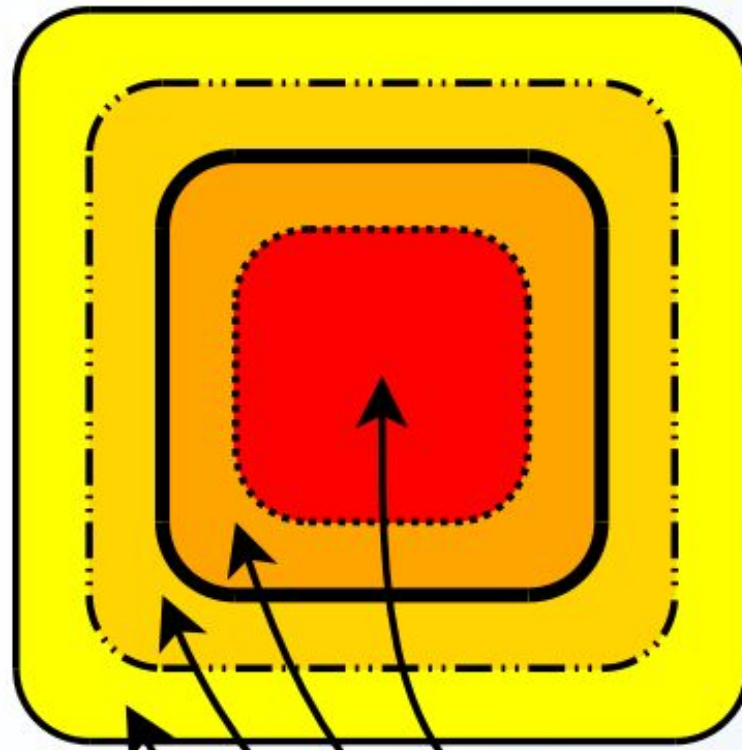
Your Application

Your Input Validation

mod_security

- Apache module
- Can be used to avoid certain attack
- Is not PHP specific

Multiple Levels of Defense++



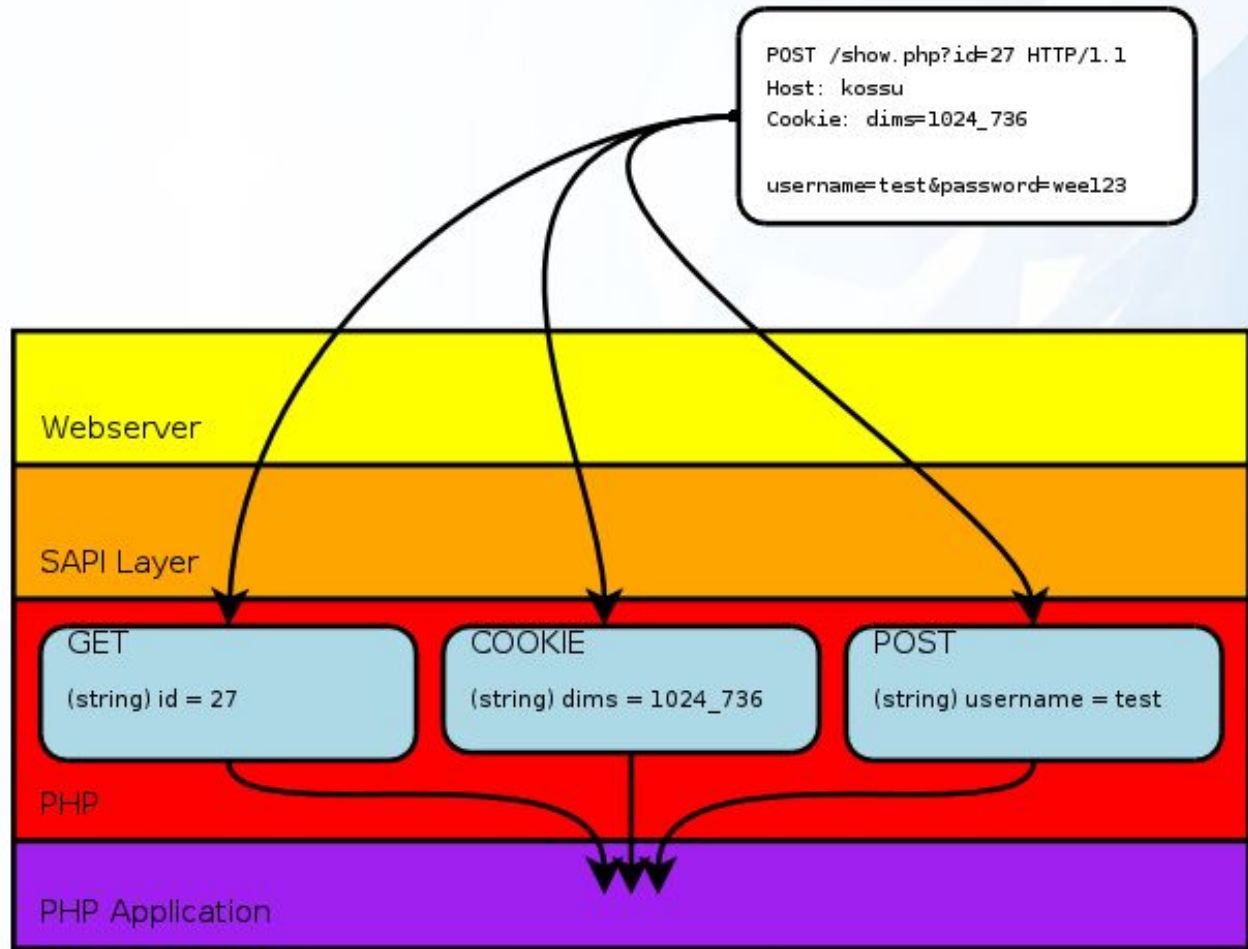
Your Application

Your Input Validation

SAPI Input Filter

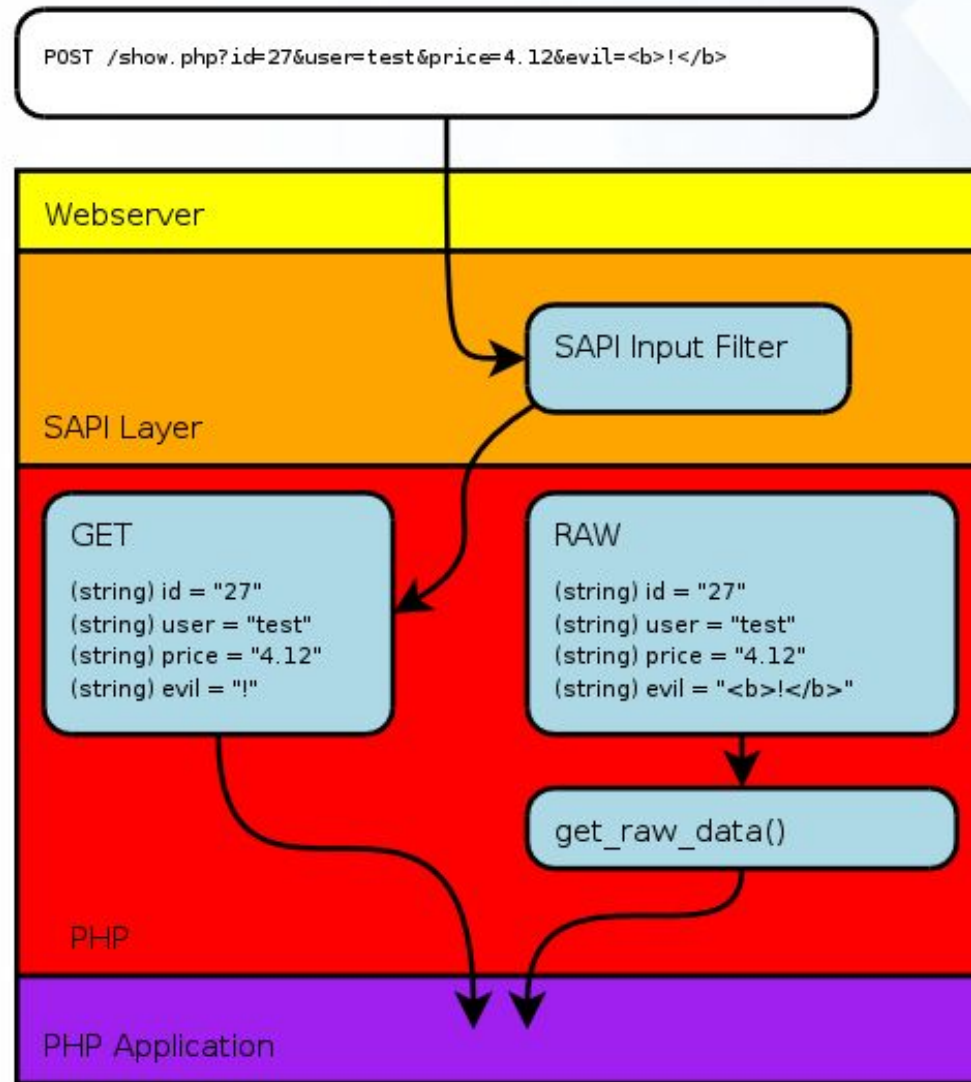
mod_security

- Sits between PHP and the webserver
- Is used while fetching data from users sources
- Can be used to filter data
- Prohibit data from entering PHP
- Written as a C extension to PHP
- Server wide filter



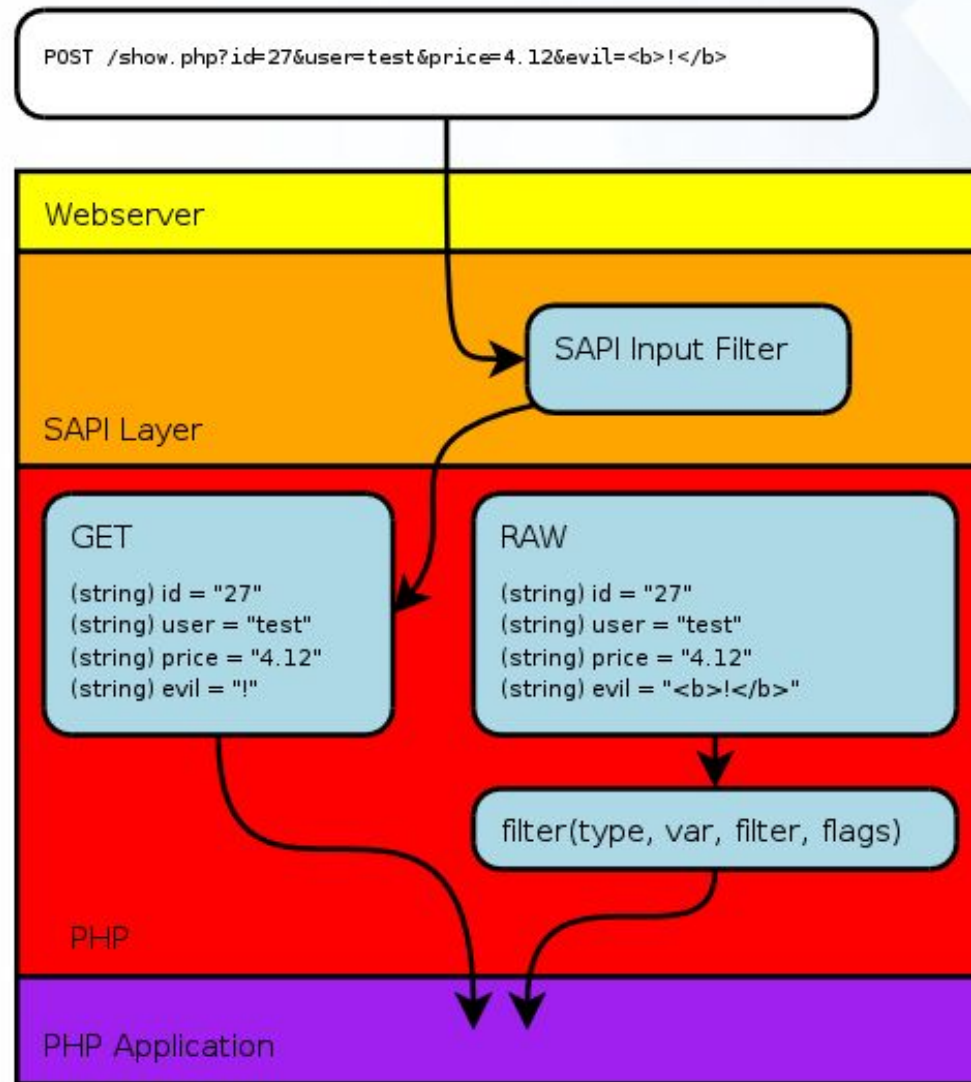
First Idea of an Input Filter

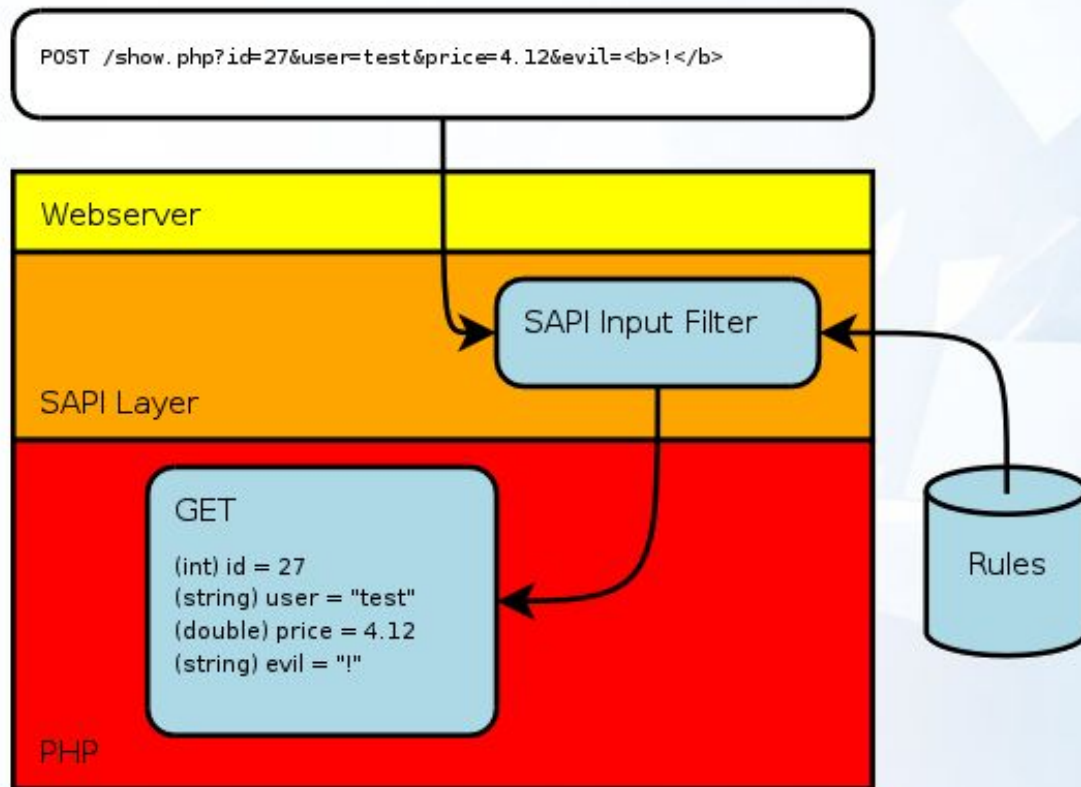
Share your information

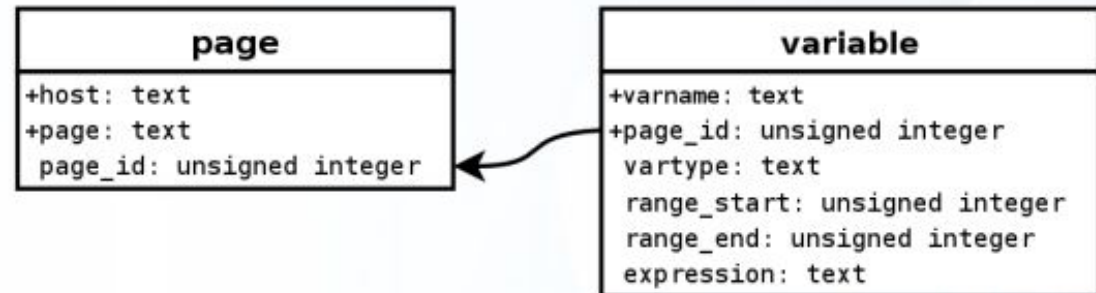


Second Idea of an Input Filter

Share your information







For each page you can define:

- Which variables have a filter assigned
- What the type of that variable should be ¹
- For integers: what range the variable should be in
- For text: what filters should be run on the data ²

¹ bool, long, double or string

² not implemented yet

Registering an input filter

In your extension's PHP_MINIT_FUNCTION:

```
sapi_register_input_filter(sqlite_sapi_input_filter);
```

The filter function's signature:

```
unsigned int sqlite_sapi_input_filter(  
    int arg,                /* 0 = POST, 1 = GET, 2 = COOKIE */  
    char *var_name,        /* The original variables name */  
    char **val,            /* The original variables value */  
    unsigned int val_len,  /* The original variables length */  
    unsigned int *new_val_len /* The length of the modified  
                               * variable */  
)
```

The function should return:

- 0: if the variable should be *discarded*
- 1: if the variable should be *registered*

In your extension's PHP_MINIT_FUNCTION:

```
zend_hash_init(&(SQLITE_FILTER_G(filter_rules)), 0, NULL,
              (dtor_func_t) sqlite_filter_rule_dtor, 1);

if (read_rules(TSRMLS_C) != SUCCESS) {
    return FAILURE;
}
```

Reading rules:

```
static int read_rules(TSRMLS_D)
{
    int          res = SUCCESS;
    char         *err = NULL;
    int          sqlite_ret;
    sqlite_vm    *vm;
    const char   *tail;
    const char   **row_data, **col_names;
    int          column_count, i;

    /* Open SQLite database */
    SQLITE_FILTER_G(sdb) =
        sqlite_open("sqlite_filter.sqlite", 0600, &err);

    if (SQLITE_FILTER_G(sdb) == NULL) {
        sqlite_freemem(err);
        return FAILURE;
    }
}
```

- 1. Retrieve page name from server context
- 2. Create a hash key to do the rule matching
- 3. Locate the rule in the hash
- 4. Create a new variable container (ZVAL) and fill it with the original data
- 5a. If we have a matching rule, use it
- 5b. Otherwise fall back (*strip_tags()*)
- 6. Select the correct array where to save too (GET, POST, COOKIE)
- 7. Register the modified variable
- 8. Deny PHP to register the original variable

5a. If we have a matching rule, use it

```
if (rule) {
    switch (rule->type) {
        case IS_LONG:
            convert_to_long(new_var);
            if (rule->range_start != -1 &&
                Z_LVAL_P(new_var) < rule->range_start)
            {
                Z_LVAL_P(new_var) = rule->range_start;
            }
            if (rule->range_end != -1 &&
                Z_LVAL_P(new_var) > rule->range_end)
            {
                Z_LVAL_P(new_var) = rule->range_end;
            }
            break;
        case IS_DOUBLE:
            convert_to_double(new_var);
            break;
        case IS_BOOL:
            convert_to_boolean(new_var);
            break;
        case IS_STRING:
            /* FIXME: Use 'expression' filter here */
            break;
    }
}
```

5b. Otherwise fall back (*strip_tags()*)

```
{  
    Z_STRLEN_P(new_var) =  
        php_strip_tags(Z_STRVAL_P(new_var), Z_STRLEN_P(new_var),  
            NULL, NULL, 0);  
}
```

Other example default filters might be:

- Deny the variable totally
- Use *htmlentities()*
- Strip everything not in [A-Za-z0-9]

- Better page matching
- Implement 'expression' rules
- Implement different fallback algorithms
- Make the database location a setting
- Release it through PECL



These slides:

<http://derickrethans.nl/talks.php>

SQLite Input Filter:

http://derickrethans.nl/sqlite_filter.php

Questions?: <mailto:dr@ez.no>