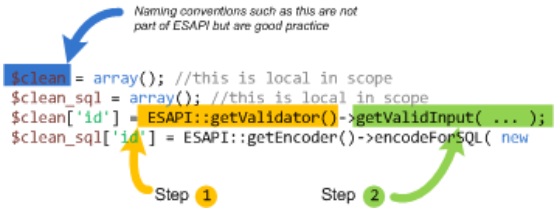
OWASP ESAPI for PHP Project Manager

***Abstract*—Don't write your own security controls! Reinventing the wheel when it comes to developing security controls for every PHP web application leads to wasted time and massive security holes. OWASP Enterprise Security API (ESAPI) for PHP helps software developers guard against security related design and implementation flaws. ESAPI for PHP is designed to make it easy to retrofit security into existing applications, as well as providing a solid foundation for new development.**

***Index Terms*—OWASP, ESAPI, PHP, XSS, SQLi**



-

-

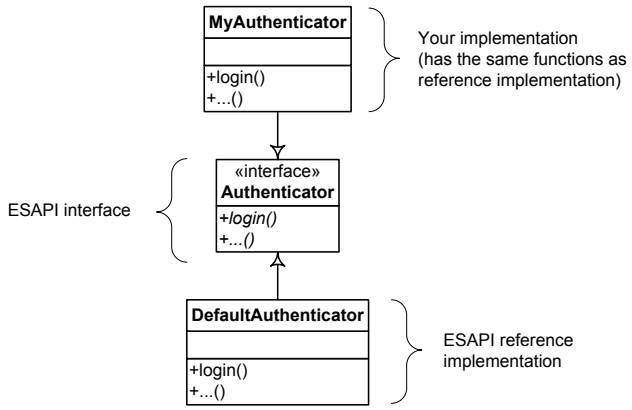
-

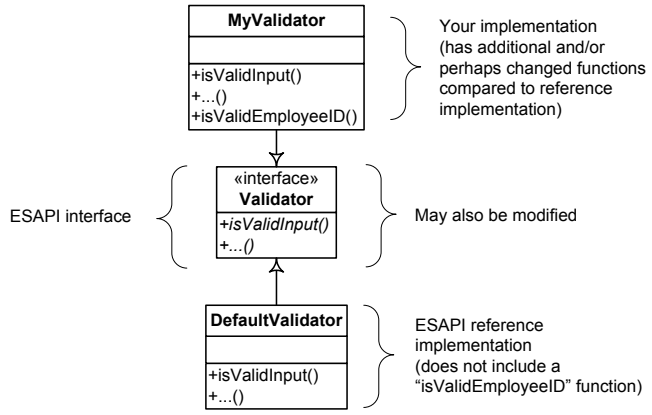
A. For example:

```
...
require_once dirname(__FILE__) . '/../Authenticator.php';
...
//your implementation
class MyAuthenticator implements Authenticator {
...
```

B. Developers would call ESAPI in this example as follows:

```
...
$ESAPI = new ESAPI();
$myauthenticator = new MyAuthenticator();

//register with locator class
ESAPI::setAuthenticator($myauthenticator);
$authenticator = ESAPI::getAuthenticator();
$authenticator->login(...); //use your implementation
...
```





A. For example:

```
...
require_once dirname(__FILE__) . '/../Validator.php';
...
//reference implementation
class DefaultValidator implements Validator {
...
//not defined in Validator interface
function isValidEmployeeID($eid) {
...
```

B. Developers would call ESAPI in this example as follows:

```
...
$ESAPI = new ESAPI();
$validator = ESAPI::getValidator();
$validator->isValidEmployeeID(1234);
...
```

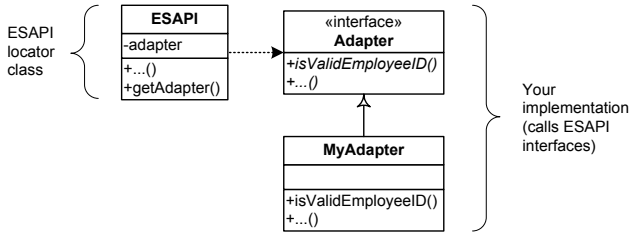
A. For example:

```
...
class ESAPI {
...
//not defined in ESAPI locator class
private static $adapter = null;
...
//new function
public static function getAdapter() {
```

```
    if ( is_null(self::$adapter) ) {
      require_once
dirname(__FILE__).'/adapters/MyAdapter.php';
      self::$adapter = new MyAdapter();
    }

    return self::$adapter;
  }

  //new function
  public static function setAdapter($adapter) {
    self::$adapter = $adapter;
  }



  ...
  //new interface
  interface Adapter {

    function getValidEmployeeID($eid);
    function isValidEmployeeID($eid);

  }



  ...
  require_once dirname ( __FILE__ ) . '/../Adapter.php';

  //new class with your implementation
  class MyAdapter implements Adapter {

  //for your new interface
  function getValidEmployeeID($eid) {
    //calls reference implementation
    $val = ESAPI::getValidator();
    //calls using hardcoded parameters
    $val->getValidInput(
      "My Organization's Employee ID",
      $eid,
      "EmployeeID", //regex defined in ESAPI config
      4,
      false
      );
  }


  //for your new interface
  function isValidEmployeeID($eid) {
    try {
      $this->getValidEmployeeID($eid);
      return true;
    } catch ( Exception $e ) {
      return false;
    }

  }
```

B. Developers would call ESAPI in this example as follows:

```
  ...
  $ESAPI = new ESAPI();
  $adapter = ESAPI::getAdapter();
  $adapter->isValidEmployeeID(1234);
  ... //no other ESAPI controls called directly
```



- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____
- _____