OWASP ESAPI 设计模式

本文探讨 3 种常见的 OWASP 企业安全 API（ESAPI）设计模式。其目的是独立于开发语言之外，也就是说本文档中所述的设计模式可以应用于 ESAPI 所有语言版本。OWASP ESAPI Toolkits 的设计初衷是使各个开发环境的开发人员都能获得功能强大操作方便的安全控件。

我们欢迎您的反馈

ESAPI 的进一步开发通常通过邮件列表讨论和不定期研讨会的形式展开，欢迎大家对 ESAPI 的完善提出意见和建议。有关 API 和本文档的任何问题请发邮件至 owasp-esapi@lists.owasp.org。

版权与许可

目录

图片

ESAPI 概述

OWASP ESAPI Toolkits 的设计初衷是使各个开发环境的开发人员都能获得功能强大使用方便的安全控件。各版本 OWASP ESAPI 的基本调用方式都相同，如下图所示：

图 1：ESAPI 的工作原理介绍

除了语言方面的差异，所有的 OWASP ESAPI 版本都具有如下相同的基本设计结构：

- 都有一整套安全控件接口，并且这些接口不包含任何应用层的逻辑。例如，在这些接口中定义了发送给不同安全控件的参数类型，他们不包含任何私有信息或逻辑控制。
- 每个安全控件接口都有一个参考示例。示例使用的类中实现了安全控件接口中定义的方法，因此包含一定的逻辑控制。不过这些应用逻辑并不是基于特定组织或特定应用的，同时在这些参考示例类中也不包含任何私有信息或逻辑控制。例如，基于字符串的输入验证。
- 程序开发者可以有选择的实现自己的安全控件接口。可能这些接口类中的应用逻辑是由您的组织开发的或者为您公司定制的，也就是说这些应用逻辑可以被设计成是针对某项应用或者是针对某个公司的，在这些类中也可能拥有您或者您的公司开发的专有信息或者专有逻辑，例如：企业级身份认证。

有三种方法可以来为每个安全控件添加您自己的应用：“嵌入式”单例模式，“拓展”单例模式和“拓展”工厂模式。本文接下来的部分就来探讨这三种设计模式，同时对于一些情况来说可能适合于使用不止一种模式来解决问题。

嵌入式单例模式

ESAPI 安全控件接口中包含一个通常被称为“定位器”的 ESAPI 类。这个类用于获取一个安全控件中的已实例化的单例对象，这些实例化的单例对象被调用来执行安全检查（如进行访问控制检测）或产生安全效用（如生成审计日志）。

“嵌入式”单例模式指的是，开发者可以使用自己的实现代码，来替换已有安全控件的参考示例。否则，ESAPI 程序接口保持不变。

例如：

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

开发者对 ESAPI 的调用如下所示：

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

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

上述例子的 UML 如图 2 所示：

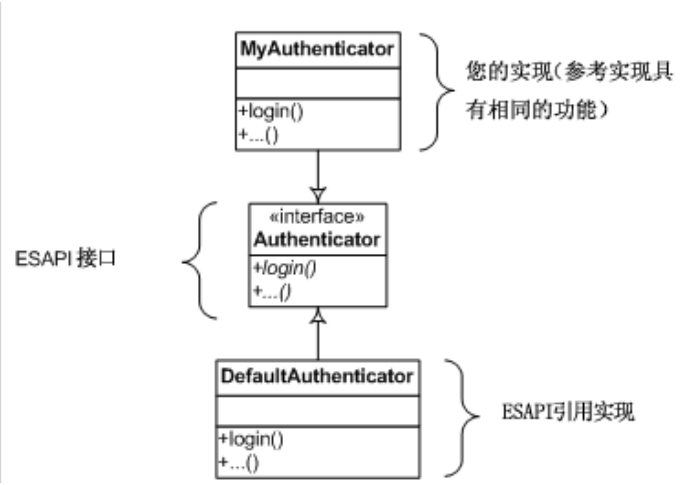


图 2：嵌入式单例模式示例

采用这种方法的优点是可以在 ESAPI 和您自己的程序代码中实现松耦合。

缺点则是需要开发者理解如何根据自己公司或者应用的需要使用正确的参数来调用 ESAPI 的方法。

拓展的单例模式

虽然 ESAPI 的安全控件引用实现即可进行安全检测而且能依据公司或者应用的需要产生一些安全效用，我们还是需要降低使用门槛，使开发人员无需对 ESAPI 如何通过传入和公司或者应用相关的参数执行过程工作原理做更多了解。

拓展的单例模式指的是开发者自己实现相应的安全控件，来替代已有安全控件的引用实现。并支持添加、修改、删除方法相应的安全控件接口。

例如：

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

在本例中开发者可以按下面的方式调用 ESAPI：

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

上述例子的 UML 如下图所示：

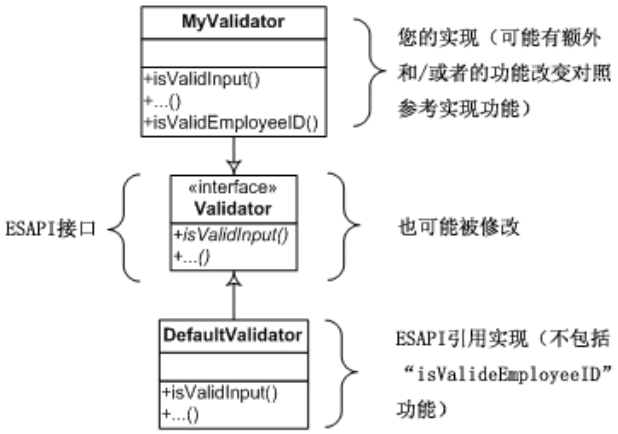


图 3：拓展的单例模式示意图

采用这种方法的优点是开发人员并不需要非常了解如何根据公司或者应用的需求调用带有具体参数的 ESAPI 函数，优点还包括：减少或消除开发人员偏离你的组织或者应用规则的风险时能够调用 ESAPI 函数。

采用这种方法的不足之处在于 ESAPI 和应用实现之间产生了紧耦合：你需要维护已经修改的安全控件参考实现以及已修改的安全控件的接口（因为 ESAPI 的版本将不断更新）。

拓展的工厂模式

虽然 ESAPI 安全控件参考实现能够进行安全检测而且能够起到您公司或者应用要求安全效用，仍然有必要消除程序员使用不同于公司策略或者应用规则的风险。频繁的人员流动可能需要考虑这个问题，另一个例子是大力实施编码标准。

拓展的工厂模式指的是一个新的安全控件接口及其实现，这个新的接口实现反过来调用 ESAPI 安全控件的参考实现，或者用户自己重写的实现。在这种模式下，ESAPI 的定位器会首先被调用，以便获得新的安全控件的单独实例。事实上这个实例调用的是 ESAPI 安全控件的参考实现或者由用户自己重写的实现 。

例如，在 ESAPI 的定位器类中：

. . .

```
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;
}
```

新的安全控件类的接口如下：

```
...
//new interface
interface Adapter {

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

}
```

新的安全控件类的实现如下：

```
...
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;
    }

}
```

在本例中开发者可以按照如下方式调用 ESAPI：

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

上述例子的 UML 如下图所示：

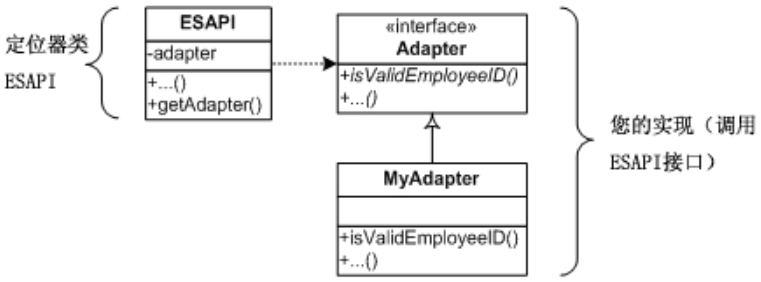


图 1: 扩展的工厂模式示意图

使用这种方法的优点和上面提到的拓展的单例模式中的相同，同时相对拓展的单例模式而言，增加了 ESAPI 和您的程序之间的松耦合特性。

此种模式的不足之处在于需要额外维护已修改的 ESAPI 的定位器类（因为 ESAPI 的版本将不断更新）。

下一步去哪儿

OWASP 是 WEB 应用安全领域的前沿网站。OWASP 网站包含了许多项目，论坛、博客、报告、工具和论文等资料。此外，OWASP 每年举办两次大型 Web 应用安全会议，已在八十多个地方设立分会。您可以在这里找到 OWASP 的 ESAPI 项目主页 http://www.owasp.org/index.php/ESAPI

如果您是 ESAPI 的使用者，下面的项目可能对您有所帮助：

- OWASP 应用安全验证标准 (Application Security Verification Standard ， ASVS) 项目 - http://www.owasp.org/index.php/ASVS
- OWASP Top 10 风险项目 - http://www.owasp.org/index.php/Top_10
- OWASP 代码审计指南（Code Review Guide ）- http://www.owasp.org/index.php/Category:OWASP_Code_Review_Project
- OWASP 测试指南（Testing Guide） - http://www.owasp.org/index.php/Testing_Guide
- OWASP 合规性项目（Legal Project） - http://www.owasp.org/index.php/Category:OWASP_Legal_Project

同时作为 ESAPI 的用户，下面的网站也可以给您更全面的信息：

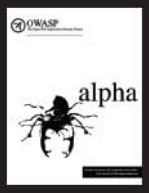
- OWASP - http://www.owasp.org
- MITRE - Common Weakness Enumeration – Vulnerability Trends, http://cwe.mitre.org/documents/vuln-trends.html
- PCI Security Standards Council - publishers of the PCI standards, relevant to all organizations processing or holding credit card data, https://www.pcisecuritystandards.org
- PCI Data Security Standard (DSS) v1.1 - https://www.pcisecuritystandards.org/pdfs/pci_dss_v1-1.pdf

THE BELOW ICONS REPRESENT WHAT OTHER VERSIONS ARE AVAILABLE IN PRINT FOR THIS TITLE BOOK.

ALPHA: "Alpha Quality" book content is a working draft. Content is very rough and in development until the next level of publication.

BETA: "Beta Quality" book content is the next highest level. Content is still in development until the next publishing.

RELEASE: "Release Quality" book content is the highest level of quality in a books title's lifecycle, and is a final product.



ALPHA
PUBLISHED

This page is intentionally blank

YOU ARE FREE:

to share - to copy, distribute and transmit the work

to Remix - to adapt the work

UNDER THE FOLLOWING CONDITIONS:

Attribution. You must aatribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

OWASP
The Open Web Application Security Project

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible," so that people and organizations can make informed decisions about application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.