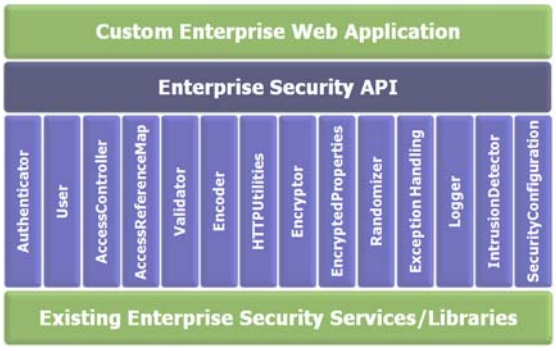
ierie des Protocoles et Logiciels Sécurisés

ESAPI : Enterprise Security Application Programming Interface

Version 2



Yacine Challal

23/22/2014

Table des matières

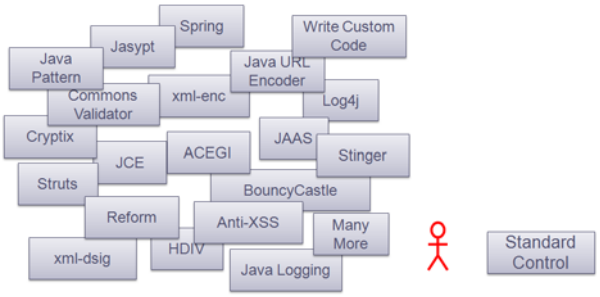
I

Enterprise Security Application Programming Interface

A. Introduction

Problématique

Il existe plusieurs librairies qui offrent des fonctionnalités de sécurité comme l'authentification, la vérification des inputs et des outputs, etc. (Voir figure suivante)



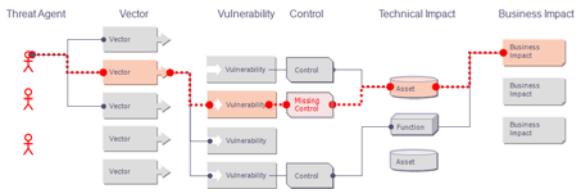
Librairies pour logiciel sécurisé

Néanmoins, ces librairies ne sont pas intégrées, intuitives et parfois ne sont pas correctes. De ce fait il est souvent complexe aux développeurs d'exploiter ces librairies, notamment dans des projets d'envergure avec des environnements logiciels éthérogènes.

B. Comprendre le cheminement d'une attaque

Le risque

Un risque est un chemin d'un agent de menace à un actif métier comme illustré sur la figure suivante :



Chemin d'un risque

Dans ce cheminement on retrouve les éléments suivants :

- Vecteur – une attaque utilisée par un attaquant ou la possibilité qu'offre un agent de menace non malicieux qui cause une vulnérabilité exploitable.
- Vulnérabilité – une faiblesse dans un système informatique - généralement un contrôle oublié ou compromis
- Impact technique – le résultat technique direct suite à une attaque réussie
- Impact métier – l'effet résultat sur le métier de la cible.

Une analyse de risque revient à retrouver ces chemins à travers ce modèle où l'impact est tellement important qu'il serait nécessaire d'apporter de nouveaux contrôles efficaces ou de corriger les contrôles existants.

C. Enterprise Security API

Éléments de Motivation

- Il est difficile de mettre en oeuvre des contrôles de sécurité corrects
  - Nécessité d'une attention particulière au détail.
    Une erreur introduit des vulnérabilités
    Souvent ça nécessite de fortes compétences en développement et en sécurité.
- Absence d'une approche unifiée unique
  - Au sein d'un même langage
  - A travers différents langages.

Objectifs de ESAPI

- Construction d'un ensemble de contrôles de sécurité commun aux langages les plus utilisés aujourd'hui
  - Avoir des interfaces communes à travers différents langages tant que possible.
  - Fournir au moins une implémentation de référence simple pour chaque contrôle de sécurité pour servir d'exemple ou être utilisé
  - Facilement extensible
- Développement d'une API de sécurité Open Source qui est:
  - Scalable à l'échelle d'une entreprise
  - Assez large dans la couverture des contrôles de sécurité
  - License libre (BSD, Creative Commons)

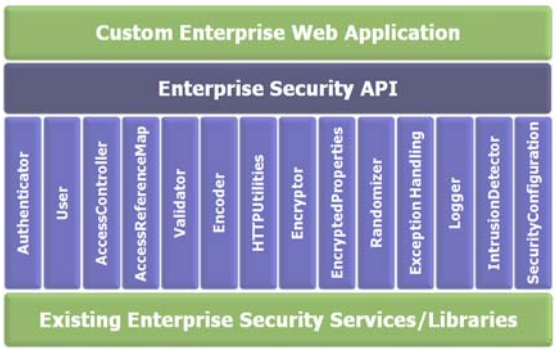
Remarque : Qualités attendues pour une Enterprise Security API

Un développeur a besoin de contrôles de sécurité qui soient:

- Standards
- Centralisés
- Organisés
- Intégrés
- De Haute qualité
- Intuitifs
- Testés
- Résout le problème de contrôles oubliés et/ou compromis

Aperçue générale

La figure suivante illustre l'ensemble des interfaces qui compoent ESAPI



Interfaces d'ESAPI

D. Encodage et échappement des Outputs

L'encodage est le processus de transformation de données d'un format à un autre. L'interface ESAPI Encoder contient un nombre de méthodes pour décoder des inputs et encoder des output pour les rendre fiables pour les interpreteurs (SQL, HTML, etc.). C'est un mécanisme puissant contre les attaques de type injection et XSS. L'encodage concerne la traduction de certains caractères spéciaux en des caractères équivalents qui ne sont plus significatifs pour l'interpreteur.

Exemple

Par exemple, utiliser un encodage HTML avant l'envoie d'input “untrusted” à un navigateur protège contre XSS.

```
ESAPI.encoder().encodeForHTML(input)
```

E. Validation des inputs

Validation des Input est le processus qui assure qu'un programme opère sur des données propres, correctes et utiles Par exemple, si on arrive à utiliser une entité qui encode du HTML avant l'envoie au navigateur , ceci permettrait de prévenir la majorité des attaques XSS.

Fondamental

L'interface ESAPI Validator définit un ensemble de méthodes pour canoniser et valider “untrusted input”.

ESAPI.validator().getValidInput(String context,String input,String type,int maxLength,boolean allowNull)

F. Authentification

L'authentification est le processus qui permet de déterminer si une entité est bien celle prétendue. L'interface ESAPI Authenticator définit un ensemble de méthodes pour générer et gérer des comptes et identifiants de sessions. L'objectif de cette interface est d'encourager les développeurs à protéger les crédentiels au mieux.

```
ESAPI.authenticator().createUser(username, password,
password)
ESAPI.authenticator().getUser(username).enable();
ESAPI.authenticator().getUser(username).unlock();
User user = ESAPI.authenticator().login(HTTPServletRequest,
HTTPServletResponse);
//Définir les valeurs des variables UsernameParameterName
et PasswordParameterName dans ESAPI.properties car la
méthode login se base sur ces variables pour récupérer leur
valeurs de la requête HTTP
User user = ESAPI.authenticator().logout;
```

G. Contrôle d'accès

Le contrôle d'accès est le processus qui définit les privilèges de chaque utilisateur dans le système. L'interface ESAPI AccessController définit un ensemble de méthodes qui peuvent être utilisées dans un large éventail d'applications pour renforcer le contrôle d'accès. Dans la majorité des applications, le contrôle d'accès doit être effectué dans de multiples et différents endroits à travers les diverses couches applicatives. Cette interface fournit un contrôle d'accès au niveau des URL, fonctions métier, données, services et fichiers. L'implémentation de cette interface nécessite l'accès à l'objet current User (Authenticator.getCurrentUser()) pour déterminer les rôles et permissions. L'implémentation aura aussi besoin d'information sur les ressources accédées. En utilisant ces deux info, l'implémentation retourne la décision de contrôle d'accès.

Exemple

```
try {
    ESAPI.accessController().assertAuthorized("businessFunction", runtimeData);
    // execute BUSINESS_FUNCTION
} catch (AccessControlException ace) {
... attack in progress
}
```

Exemple de contrôle d'accès

H. Accès indirect aux objets

L'interface AccessReferenceMap permet d'associer des références indirectes aux

références directes des objets du système.

RandomAccessReferenceMap est une implémentation de cette interface avec des références indirectes aléatoires.

I. Gestion de sessions

La gestion de sessions est le processus qui permet de garder la trace de l'activité de l'utilisateur à travers les sessions d'interaction avec le système. L'interface ESAPI HTTPUtilities est une collection de méthodes qui fournit une sécurité supplémentaire relative à HTTP requests, responses, sessions, cookies, headers, et logging.

Exemple

Eviter l'attaque de type “session fixation”

```
ESAPI.httpUtilities().changeSessionIdentifier();
```

J. Top 10 des attaques OWASP et leur contremesures ESAPI

OWASP Top Ten 2010	OWASP ESAPI
A1: Injection	Encoder
A2: Cross-Site Scripting (XSS)	Encoder, Validator
A3: Broken Authentication and Session Management	Authenticator, User, HTTPUtilities
A4: Insecure Direct Object	AccessReferenceMap,
A5: Cross-Site Request Forgery	User (CSRF Token)
A6: Security Misconfiguration	Security Configuration
A7: Insecure Cryptographic Storage	Encryptor
A8: Failure to Restrict URL Access	AccessController
A9: Insufficient Transport Layer	HTTPUtilities
A10: Unvalidated Redirects and	AccessController

OWASP Top10 attacks vs. ESAPI interfaces

D'lala Online

II

A. Développement d'une application Sécurisée avec ESAPI

Environnement de développement

Vous allez développer une application sécurisée en s'appuyant sur l'application ESAPI Swingset Interactive.

Swingset implémente les contrôles de sécurité offerts par ESAPI et vous fournit des tutotiels et exemples expliquant l'utilisation des différentes interfaces définies dans ESAPI pour mettre en oeuvre ces contrôles de sécurité.

Question 1

Afin de vous familiariser avec ESAPI, lancez l'exécution de Swingset sur Tomcat à partir d'Eclipse puis ouvrez l'application dans un navigateur :

https://localhost:8443/Swingset/main

Question 2

Expliquer comment se font les contrôles suivants en précisant les appels de méthodes nécessaires à leur mise en oeuvre ainsi que la configuration sous-jacente :

- Création d'un utilisateur
- Login
- Logout
- Récupérer l'utilisateur actuel de la session
- Vérifier si un utilisateur peut accéder à une page web, Fichier, exécuter une méthode
- Eviter une Injection SQL
- Vérifier qu'une donnée entrée est conforme à un format spécifique
- Eviter une attaque de type XSS

  Eviter une attaque CSRF (Cross Site Request Forgery)
- Créer et utiliser un "crypto-token"
- Chiffrer puis déchiffrer une donnée sensible

B. Application de vente en ligne sécurisée : D'lala Online

Description de l'application D'lala Online

L'application D'lala Online permet à un utilisateur authentifié d'effectuer une recherche de produits dans une base de donnée.

A l'issue de cette recherche, l'utilisateur commande les produits qu'il souhaite acheter en payant en ligne avec sa carte bancaire CIB.

La commande lui sera livrée à l'adresse de livraison fournie lors de la création de son compte utilisateur.

Question 1

Elaborez le MCD correspondant à la base de donnée de cette application.

Indice :

> *Il n'est pas nécessaire de faire un MCD trop détaillé, car l'objectif du Mini-Projet est de mettre en application les contrôle de sécurité.*
>
> *On se contentera des classes : Produit, Commande, Client*

Question 2

Après la traduction du MCD en modèle relationnel, créez la base de données sous MySQL en utilisant phpMyAdmin déjà installé dans la VM

Question 3

Développez une version de D'lala Online (sans contrôles de sécurité) avec JSP permettant de faire une recherche de produits et de passer une commande en payant en ligne avec la carte bancaire.

Question 4

Analysez les risques de sécurité et citez les attaques potentielle contre cette application. Entre autres attaques, expliquez comment un attaquant peut mener les attaques suivantes contre votre application et à quels niveaux de l'application :

- Cross Site Scripting
- Injection
- Fixation de Session
- Usurpation d'Identité
- Cross Site Request Forgery
- Vol de numéro de carte bancaire
- etc.

Contrôles de Sécurité de D'lala Online

Etant donné l'analyse de risques et de vulnérabilité effectuée précédemment, on souhaite développer une nouvelle version de D'lala Online qui soit sécurisée.

Question 5

Rajouter à la version précédente tous les contrôle de sécurité avec ESAPI qui permettent :

- créer un nouveau client
- authentifier un client
- gérer la session d'un client et éviter les CSRF, et fixation de session

- canoniser et valider les Inputs et éviter les XSS
- encoder les output et éviter les XSS et Injection

On souhaite rajouter une option à D'lala Online permettant à un client d'offrir des cadeaux à d'autres clients enregistrés sur D'lala Online

Question 6

Développez une telle version en utilisant un CryptToken.