

École Polytechnique de Montréal

Département de génie informatique et logiciel

LOG8371  
Ingénierie de la qualité en logiciel

TP3- Sécurité

Soumis à Mohamed-Essaddik Benyahia

Soumis par  
Julien Arès (1802992)  
Olivier Filippelli (1775057)  
Kim Thuyen Ton (1789211)  
Alex Tran (1799594)

Groupe 03 – Équipe Bravo  
Hiver 2019

19 avril 2019

## Table des matières

<b>Question 1: Analyse statique de Weka Rest</b>	<b>2</b>
a) Sommaire des résultats	2
b) Commentaires sur des vulnérabilités ou hotspots de sécurité	3
1) Dynamically executing code is security-sensitive	3
2) Logger to log exception	3
3) Class variable fields should not have public accessibility	4
4) Handling files is security-sensitive - Path.get()	5
5) Handling files is security-sensitive - FileInputStream	6
6) Using regular expression is security-sensitive	7
7) Deserializing objects from an untrusted source is security-sensitive	8
8) Using pseudorandom number generators (PRNGs) is security-sensitive	9
<b>Question 2: Analyse de Weka Rest avec Zap</b>	<b>10</b>
1) Application Error Disclosure	10
2) Buffer Overflow	11
3) X-Frame-Options Header Not Set	11
4) X-Content-Type-Options Header Missing	12
5) Web Browser XSS Protection Not Enabled	12
<b>Question 3: Comparaison entre les deux analyses</b>	<b>14</b>
<b>Références</b>	<b>17</b>
<b>Annexe</b>	<b>18</b>

**N.B.:** L'analyse de Weka Rest se concentre sur les 6 algorithmes suivants: Bayes Naïf, Bayes Net, régression linéaire, k-means, clustering hiérarchique et perceptron à plusieurs couches.

## Question 1: Analyse statique de Weka Rest

### a) Sommaire des résultats

SonarCloud produit deux rapports lorsqu'il est utilisé pour analyser du code. Les résultats de l'analyse se trouvent [https://sonarcloud.io/dashboard?id=olivierFilippelli\\_LOG8371](https://sonarcloud.io/dashboard?id=olivierFilippelli_LOG8371). Le premier rapport est fait selon le standard SANS Top 25. Il donne une note de A pour les trois critères: *Porous Defenses*, *Risky Resource Management* et *Insecure Interaction Between Component*. Le deuxième rapport est fait selon le standard OWASP Top 10 de 2017. Il donne une note de A pour les 10 premiers critères du standard, *Injection*, *Broken Authentication*, *Sensitive Data Exposure*, *XML External Entities (XXE)*, *Broken Access Control*, *Security Misconfiguration*, *Cross-Site Scripting (XSS)*, *Insecure Deserialization*, *Using Components with Known Vulnerabilities* et *Insufficient Logging & Monitoring*, et une note globale de B pour les autres critères. L'application ne comporte donc pas de risques importants pour la sécurité et peut donc être considérée sécuritaire. Cependant, il y a des vulnérabilités mineures et des *hotspots* de sécurité qui méritent d'être analysés plus en profondeur pour augmenter le niveau de sécurité de l'application.












Categories	Vulnerabilities	Security Hotspots		
		Open	In Review	Won't Fix
A1 - Injection	0 	20	0	0
A2 - Broken Authentication	0 	0	0	0
A3 - Sensitive Data Exposure	0 	6	0	0
A4 - XML External Entities (XXE)	0 	0	0	0
A5 - Broken Access Control	0 	0	0	0
A6 - Security Misconfiguration	0 	0	0	0
A7 - Cross-Site Scripting (XSS)	0 	9	0	0
A8 - Insecure Deserialization	0 	2	0	0
A9 - Using Components with Known Vulnerabilities	0 	0	0	0
A10 - Insufficient Logging & Monitoring	0 	0	0	0
Not OWASP	60 	0	0	0

Figure 1. Rapport de sécurité selon OWASP Top 10




Categories	Vulnerabilities	Security Hotspots		
		Open	In Review	Won't Fix
Porous Defenses	0 	4	0	0
Risky Resource Management	0 	4	0	0
Insecure Interaction Between Components	0 	0	0	0

Figure 2. Rapport de sécurité selon SANS Top 25

## b) Commentaires sur des vulnérabilités ou *hotspots* de sécurité

### 1) Dynamically executing code is security-sensitive

**Fichier:** src/main/java/io/swagger/api/Algorithm.java

**Criticité:** Security Hotspot - Critical

**Type de vulnérabilité:**

- OWASP Top 10 2017 Category A1 - Injection
- OWASP Top 10 2017 Category A7 - Cross-Site Scripting (XSS)
- CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')
- CWE-470 - Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')

#### **Description:**

Ce genre de *hotspot* de sécurité concerne la présence de réflexion ou de code exécuter dynamiquement. Ces pratiques peuvent introduire des vulnérabilités dans le code en laissant la possibilité à des utilisateurs malicieux de modifier le code. La ligne 32 du fichier contient une instance du *hotspot* potentiellement dangereux.

```
String implClass = servletContext.getInitParameter("AlgorithmApi.implementation");
if (implClass != null && !"".equals(implClass.trim())) {
    try {
        delegate = (AlgorithmService) Class.forName(implClass).newInstance();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

Dans ce cas-ci, la réflexion est utilisé pour instancier une classe. Cette pratique est potentiellement dangereuse puisqu'il est possible que le code soit alors modifier pour instancier une autre classe par un utilisateur malicieux.

#### **Recommandation de la résolution de problème:**

Comme le string permettant de déterminer la classe à instancier vient du code, il n'y a pas de risque qu'un source extérieure malveillante sélectionne une classe pouvant briser le code. Cette zone est seulement dangereuse si l'utilisateur peut modifier directement le code.

### 2) Logger to log exception

**Fichier:** src/main/java/io/swagger/api/Bootstrap.java

**Criticité:** Vulnérabilité - Minor

**Type de vulnérabilité:**

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- CWE-489 - Leftover Debug Code

**Description:**

Ce genre de vulnérabilité concerne le fait d'imprimer directement le *stack trace* d'un *Throwable* dans le *stream* par défaut. Cette pratique peut exposer des informations sensibles aux utilisateurs puisqu'il n'y a aucun contrôle sur l'information affichée. La ligne 51 du fichier contient une instance de cette vulnérabilité mineure.

```
catch (Exception e) {  
    e.printStackTrace();  
}
```

Dans ce cas-ci, la vulnérabilité est causée par l'appel à la fonction *printStackTrace* sans paramètre pour préciser un *stream*.

**Recommandation de la résolution de problème:**

Il est recommandé d'utiliser un *Logger* pour gérer les *logs* pour non seulement mieux gérer quelles informations sont affichées, mais aussi rendre les *logs* plus faciles à lire et uniformes pour les développeurs.

```
catch(Exception e) {  
    LOGGER.log("exception", e);  
}
```

### 3) Class variable fields should not have public accessibility

**Fichier:** src/main/java/io/swagger/api/ErrorResponse.java

**Criticité:** Vulnérabilité - Minor

**Type de vulnérabilité:**

- CWE-493 - Critical Public Variable Without Final Modifier

**Description:**

Ce genre de vulnérabilité concerne le fait d'exposer des variables dans le code en les mettant public ou statique. Cette pratique n'est pas suggérée, car cela permet la modification partout dans le code sans contrôle de la part du développeur. Le fichier contient une classe qui comporte plusieurs instances de cette vulnérabilité.

```
public class ErrorResponse {  
    public String actor;  
    public String errorCause;  
    public String errorType;  
    public Integer http_code;  
    public String message;  
    public String rest_params;  
    public String backtrace;  
}
```

Dans ce cas-ci, chaque attribut de la classe *ErrorResponse* est public et est donc exposé au reste du code. Cela cause donc plusieurs vulnérabilités. Non seulement il n'y a aucune méthode de validation des valeurs, la structure de la classe est complètement exposée au reste du code. Le programmeur a donc aucune garantie sur les valeurs des attributs de cette classe. Un utilisateur malicieux pourrait facilement accéder aux informations et les modifier.

### Recommandation de la résolution de problème:

Il est recommandé de mettre les attributs privés et de créer des méthodes d'accès (*get* et *set*) pour chacun d'entre-eux. Cela permet de mettre des méthodes de validation dans les méthodes d'accès et ainsi plus contrôler l'accès aux informations de la classe.

```
public class ErrorReport {  
    ErrorReport();  
  
    public String GetActor() {return actor;}  
    public void SetActor(String newActor) {actor = newActor;}  
    public String GetErrorCause() {return errorCause;}  
    public void SetErrorCause(String newErrorCause) {errorCause = newErrorCause;}  
    public String GetErrorType() {return errorType;}  
    public void SetErrorType(String newErrorType) {errorType = newErrorType;}  
    public Integer GetHttpCode() {return http_code;}  
    public void SetHttpCode(Integer newHttpCode) {http_code = newHttpCode;}  
    public String GetMessage() {return message;}  
    public void SetMessage(String newMessage) {message = newMessage;}  
    public String GetRestParams() {return rest_params;}  
    public void SetRestParams(String newRestParams) {rest_params = newRestParams;}  
    public String GetBacktrace() {return backtrace;}  
    public void SetBacktrace(String newBacktrace) {backtrace = newBacktrace;}  
  
    private String actor;  
    private String errorCause;  
    private String errorType;  
    private Integer http_code;  
    private String message;  
    private String rest_params;  
    private String backtrace;  
}
```

### 4) Handling files is security-sensitive - Path.get()

**Fichier:** src/main/java/io/swagger/api/StaticContent.java

**Criticité:** Security Hotspot - Critical

#### Type de vulnérabilité:

- OWASP Top 10 2017 Category A1 – Injection
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- SANS Top 25 - Porous Defenses
- SANS Top 25 - Risky Resource Management
- CWE-732 - Incorrect Permission Assignment for Critical Resource
- CWE-73 - External Control of File Name or Path
- CWE-20 - Improper Input Validation

- CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- CWE-538 - File and Directory Information Exposure
- CWE-403 - Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')

#### **Description:**

Comme on peut le voir à la ligne 48 du fichier StaticContent.java:

```
return Response.ok(Files.readAllBytes(Paths.get(
StringUtil.checkTrailingSlash(contextBasePath) + path))).build();
```

L'utilisation du chemin de fichier peut engendrer des risques pour le logiciel. En effet, le fait d'exposer le chemin vers le fichier ou bien d'utiliser l'entrée de l'utilisateur comme étant le chemin vers un fichier déterminé est un risque à surveiller. Par exemple, un utilisateur peut modifier le chemin d'un fichier vers un fichier malicieux puisqu'il a accès à cette information, ce qui peut provoquer une défaillance dans le système.

#### **Recommandation de la résolution de problème:**

Pour s'assurer de résoudre ce problème, les chemins vers des fichiers ne devraient pas être exposés aux utilisateurs. Ainsi, les attaquants ne peuvent pas rediriger le chemin vers un autre fichier. D'autre part, s'il n'est pas possible de supprimer l'accès, il serait également possible d'avoir un système de permissions et d'autorisations afin de limiter l'accès aux utilisateurs. On peut ainsi interdire l'accès aux utilisateurs susceptibles de causer des vulnérabilités au logiciel.

### 5) Handling files is security-sensitive - FileInputStream

**Fichier:** src/main/java/io/swagger/api/StaticContent.java

**Criticité:** Security Hotspot - Critical

#### **Type de vulnérabilité:**

- OWASP Top 10 2017 Category A1 – Injection
- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- SANS Top 25 - Porous Defenses
- SANS Top 25 - Risky Resource Management
- CWE-732 - Incorrect Permission Assignment for Critical Resource
- CWE-73 - External Control of File Name or Path
- CWE-20 - Improper Input Validation
- CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- CWE-538 - File and Directory Information Exposure
- CWE-403 - Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')

**Description:**

Comme on peut le voir à la ligne 59 du fichier StaticContent.java:

```
FileInputStream content = new FileInputStream(StringUtil.checkTrailingSlash(
contextBasePath) + path);
```

Accéder directement à un fichier peut causer un risque dont on doit surveiller. Par le fait même, manipuler un système de fichier comme FileInputStream, où cela permet de faire la lecture de flux d'octets bruts d'un fichier spécifique, peut rendre le logiciel vulnérable. Par exemple, dans notre cas, lire un fichier ayant du contenu à risque ou pas sécuriser (avec un virus infiltré par un attaquant) peut provoquer une défaillance dans le système.

**Recommandation de la résolution de problème:**

Tout d'abord, lors de la lecture de fichier, il faudrait s'assurer que les fichiers ne soient pas exposés aux personnes non autorisées et que les fichiers sont valides et sécuritaires lors de la lecture du contenu. Pour ce faire, les noms de fichier ou le contenu sensible ne devraient pas être à la portée des utilisateurs. Si tel est nécessaire, vérifier la validité du fichier avant toute action avec celui-ci. Le contenu lu dans un fichier doit également être exposé seulement aux utilisateurs autorisés.

De plus, en ce qui a trait à la création ou à l'écriture d'un fichier, le fait de s'assurer que l'on possède la permission d'accéder au fichier et que le contenu ajouté dans ceux-ci soit vérifié et approuvé serait également de mise afin d'éviter des risques de vulnérabilités. Comme pour la lecture, les fichiers créés doivent contenir un système d'autorisation afin qu'ils ne soient pas accédés par des personnes non autorisées. Ainsi, l'écriture dans un fichier sera contrôlée et sécurisée pour ne pas corrompre les données ou l'espace de stockage.

## 6) Using regular expression is security-sensitive

**Fichier:** src/main/java/io/swagger/api/data/Dao.java

**Criticité:** Security Hotspot - Critical

**Type de vulnérabilité:**

- OWASP Regular expression Denial of Service - ReDoS
- OWASP Top 10 2017 Category A1 - Injection
- SANS Top 25 - Porous Defenses
- CWE-624 - Executable Regular Expression Error
- CWE-185 - Incorrect Regular Expression

**Description:**

Comme on peut le voir à la ligne 211 du fichier Dao.java:

```
while (!Objects.equals(strictJSON, strictJSON.replaceAll("(\\\"[^\"]*)\\\"(\\.)([^\"]*\\.\\.\\.)",
"$1\\(DOT\\)$3")) {
```

Utiliser des expressions régulières peut être un risque au niveau de la performance. En effet, l'utilisation de chaînes de caractères ayant des caractères irréguliers (comme dans l'exemple ci-dessus) peut être extrêmement gourmande pour le processeur en ce qui a trait au temps nécessaire pour l'évaluation de cette chaîne. De plus, des expressions régulières peuvent mener à des attaques par déni de service (ReDoS). Ces attaques sont provoquées



par des attaquants qui peuvent forcer le programme à utiliser le plus de ressources possible afin d'évaluer ces expressions régulières, causant ainsi l'inaccessibilité pour les autres utilisateurs par manque de ressources. Enfin, un autre risque peut survenir si des expressions régulières sont employées comme entrées des utilisateurs dans l'application. Ainsi, des utilisateurs malveillants peuvent entrer des chaînes de caractères ayant des informations dangereuses pouvant corrompre le système.

#### **Recommandation de la résolution de problème:**

Tout d'abord, si l'on ne peut éviter d'utiliser des entrées utilisateurs ayant des expressions régulières, il faudrait restreindre et contrôler ces entrées afin d'accepter seulement les expressions autorisées par le système. De plus, on peut tester ces situations avec des techniques précises pouvant évaluer si les expressions régulières sont adéquates, par exemple le partitionnement par équivalence, les valeurs limites et la robustesse. Ainsi, l'utilisation de ces techniques ainsi qu'utiliser un système d'autorisations permettront de vérifier si les expressions régulières utilisées sont adéquates, sécuritaires et contrôlées.

### 7) Deserializing objects from an untrusted source is security-sensitive

**Fichier:** src/main/java/io/swagger/api/data/ModelService.java

**Criticité:** Security Hotspot - Critical

#### **Type de vulnérabilité:**

- OWASP Top 10 2017 Category A8 - Insecure Deserialization
- OWASP Deserialization of untrusted data
- CWE-502 - Deserialization of Untrusted Data

#### **Description:**

Comme on peut le voir à la ligne 199 du fichier ModelService.java:

```
cls = (Classifier) ois.readObject();
```

La *désérialisation* d'objet est une situation à surveiller puisque cette entrée peut exécuter du code non fiable pouvant mener à une source de vulnérabilité au système. Par exemple, un attaquant peut introduire du code malveillant en l'insérant dans la fonction d'un autre type d'objet que celui attendu. Ainsi, le code va être exécuté même si au final il y aura une exception qui sera appelée.

#### **Recommandation de la résolution de problème:**

Pour empêcher cette situation de se produire, il faudrait utiliser une *désérialisation* par anticipation ou utiliser un filtre pour pouvoir être certain d'avoir le bon type d'objet avant d'exécuter le code. Ainsi, contrairement à la *désérialisation* ci-dessus, on vérifie que l'objet est sécuritaire avant d'utiliser la fonction voulue, empêchant ainsi toute sorte d'attaques extérieures. De plus, on peut également restreindre l'accès aux objets sérialisés comme un fichier afin de réduire les cas de vulnérabilité.

## 8) Using pseudorandom number generators (PRNGs) is security-sensitive

**Fichier:** src/main/java/io/swagger/api/impl/Validation.java

**Criticité:** Security Hotspot - Critical

**Type de vulnérabilité:**

- OWASP Top 10 2017 Category A3 - Sensitive Data Exposure
- CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
- CWE-330 - Use of Insufficiently Random Values
- CWE-326 - Inadequate Encryption Strength
- CWE-310 - Cryptographic Issues

### Description:

Comme on peut le voir à la ligne 23 du fichier Validation.java:

```
eval.crossValidateModel(algorithm, instances, folds, new Random(1));
```

Dans ce contexte, on génère une valeur aléatoire prévisible, ce qui peut mener le logiciel à des cas de vulnérabilité. En effet, un attaquant peut éventuellement accéder à des informations confidentielles en devinant la valeur générée si cette dernière est utilisée à des fins de sécurité.

### Recommandation de la résolution de problème:

À la place d'utiliser la classe *java.util.Random* comme dans cet exemple, on peut utiliser la classe *java.security.SecureRandom* afin de générer des valeurs aléatoires sécurisées pour protéger davantage notre logiciel. Ainsi, il ne sera pas possible aux utilisateurs de prédire ou de modifier les valeurs générées puisqu'elles seront cryptées.

## Question 2: Analyse de Weka Rest avec Zap

Pour cette question, Weka REST a été déployé en utilisant des conteneurs Docker en suivant les étapes décrites à la question 2 du TP2. Par la suite, de nombreuses requêtes HTTP de type GET et POST ont été effectuées avec des paramètres différents sur les 6 algorithmes à l'étude, un "spider attack" et un "active scan" ont été réalisés pour finalement produire un rapport html (disponible en annexe et téléchargeable à l'adresse suivante <https://github.com/olivierFilippelli/LOG8371/blob/master/TP3/Zap/Zap-TP3.html>).

### 1) Application Error Disclosure

**Niveau de risque:** Moyenne

**Type de vulnérabilité:**

- CWE-200: Information Exposure
- OWASP Top Ten 2007 Category A6 - Information Leakage and Improper Error Handling

**Description:**

Ce type de vulnérabilité expose de l'information à des usagers qui n'ont normalement pas accès à ce genre d'information. Le contenu du message d'erreur d'une requête HTTP pourrait être utilisé pour lancer des attaques supplémentaires. [2]

Dans le cas de Weka REST, il y a 5 instances présentant la même erreur "Internal Server Error":

- POST <http://localhost:8080/algorithm/linearRegression>
- GET <http://localhost:8080/algorithm>
- POST <http://localhost:8080/cluster/Hierarchical>
- GET <http://localhost:8080/openapi/openapi.json>
- GET <http://localhost:8080/openapi>

**Solution:**

Il est recommandé d'implémenter des messages personnalisés pour chaque erreur avec un identifiant unique visible à l'utilisateur alors que tous les détails de l'erreur seraient envoyés directement au serveur et ne sont pas visibles pour l'utilisateur.

## 2) Buffer Overflow

**Niveau de risque:** Moyenne

**Type de vulnérabilité:**

- CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
- OWASP Top Ten 2004 Category A5 - Buffer Overflows

**Description:**

Ce type de vulnérabilité se caractérise par la copie de valeurs dans des tampons sans avoir vérifié la taille disponible. Le débordement de tampon peut être causé par des processus roulant en arrière-plan qui écrivent à l'extérieur de l'espace alloué au tampon, ce qui peut causer erreurs de segmentation ou lancer des exceptions qui peuvent arrêter le logiciel abruptement. [2]

Dans le cas de Weka REST, il y a 8 instances distinctes présentant la même vulnérabilité:

Paramètre estimatorParams:

- POST <http://localhost:8080/algorithm/BayesNet>

Paramètre datasetUri:

- POST <http://localhost:8080/algorithm/MultilayerPerceptron>
- POST <http://localhost:8080/algorithm/BayesNet>
- POST <http://localhost:8080/cluster/SimpleKMeans>
- POST <http://localhost:8080/algorithm/NaiveBayes>
- POST <http://localhost:8080/algorithm/linearRegression>

Paramètre ridge:

- POST <http://localhost:8080/algorithm/linearRegression>

Paramètre useSupervisedDiscretization:

- POST <http://localhost:8080/algorithm/NaiveBayes>

**Solution:**

Il est conseillé de réécrire les processus roulant en arrière-plan en ajoutant des méthodes vérifiant la taille maximale allouée. Ceci va nécessiter une recompilation de l'exécutable.

## 3) X-Frame-Options Header Not Set

**Niveau de risque:** Moyenne

**Type de vulnérabilité:**

- CWE-16: Configuration
- OWASP Top Ten 2017 Category A6 - Security Misconfiguration

**Description:**

Ce type de vulnérabilité est normalement introduit lors de la configuration d'un logiciel. [2] Dans le cas de Weka REST, l'entête X-Frame-Options n'est pas incluse dans la réponse à une requête HTTP pour prévenir contre les attaques de type "ClickJacking". Ceci se produit lorsqu'une requête GET est faite à l'url <http://localhost:8080/> avec le paramètre X-Frame-Options.

**Solution:**

Les navigateurs webs modernes supportent déjà l'entête X-Frame-Options. Il faut donc s'assurer que toutes les pages web de Weka REST retournent cette entête.

#### 4) X-Content-Type-Options Header Missing

**Niveau de risque:** Basse

**Type de vulnérabilité:**

- CWE-16: Configuration
- OWASP Top Ten 2017 Category A6 - Security Misconfiguration

**Description:**

Ce type de vulnérabilité est normalement introduit lors de la configuration d'un logiciel. [2] Dans le cas de Weka REST, l'entête X-Content-Type-Options de "Anti-MIME-Sniffing" n'a pas été fixée à la valeur "nosniff". Ainsi, cela permet à des anciennes versions de navigateurs web d'exécuter un "MIME-sniffing" sur le corps de la réponse, ce qui peut modifier son interprétation. Dans le cas de Weka REST, il y a 15 instances présentant cette vulnérabilité:

Paramètre X-Content-Type-Options:

- GET http://detectportal.firefox.com/success.txt
- GET http://localhost:8080/swagger-ui/swagger-ui-standalone-preset.js
- GET http://localhost:8080/openapi
- GET http://localhost:8080/swagger-ui/jguweka.css
- GET http://localhost:8080/swagger-ui/swagger-ui.css
- GET http://localhost:8080/swagger-ui/swagger-ui-bundle.js
- GET http://localhost:8080/swagger-ui/favicon-16x16.png
- GET http://localhost:8080/openapi/openapi.json
- GET http://localhost:8080/swagger-ui/favicon-32x32.png
- GET http://localhost:8080/
- POST http://localhost:8080/algorithm/MultilayerPerceptron
- POST http://localhost:8080/cluster/SimpleKMeans
- POST http://localhost:8080/algorithm/NaiveBayes
- POST http://localhost:8080/algorithm/BayesNet
- POST http://localhost:8080/algorithm/linearRegression

**Solution:**

Il faut s'assurer que l'application crée les entêtes Content-Type en incluant l'option "nosniff" pour l'entête X-Content-Type-Options.

#### 5) Web Browser XSS Protection Not Enabled

**Niveau de risque:** Basse

**Type de vulnérabilité:**

- CWE-933: Configuration
- OWASP Top Ten 2017 Category A6 - Security Misconfiguration

**Description:**

Ce type de vulnérabilité est normalement introduit lorsqu'il y a une mauvaise configuration de la sécurité d'un logiciel. [2] La protection XSS du navigateur web n'est pas activée ou est désactivée par la configuration de l'entête de la réponse X-XSS-Protection. Dans le cas de Weka REST, il y a 15 instances présentant cette vulnérabilité:

#### Paramètre X-XSS-Protection

- GET http://localhost:8080/robots.txt
- GET http://localhost:8080/algorithm/NaiveBayes
- GET http://localhost:8080/cluster/Hierarchical
- GET http://localhost:8080/cluster/SimpleKMeans
- GET http://localhost:8080/algorithm
- GET http://localhost:8080/algorithm/MultilayerPerceptron
- GET http://localhost:8080/
- GET http://localhost:8080/cluster
- GET http://localhost:8080/swagger-ui
- GET http://localhost:8080/sitemap.xml
- GET http://localhost:8080/algorithm/linearRegression
- GET http://localhost:8080/algorithm/BayesNet
- POST http://localhost:8080/cluster/Hierarchical
- POST http://localhost:8080/algorithm/NaiveBayes
- POST http://localhost:8080/algorithm/linearRegression

#### **Solution:**

Il faut s'assurer que le filtre XSS du navigateur web est activé en fixant la valeur de 1 à l'entête de la réponse X-XSS-Protection.

## Question 3: Comparaison entre les deux analyses

Vulnérabilités relevées par SonarCloud:

Dynamically executing code is security-sensitive:

- CWE-95 - Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')
- CWE-470 - Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection')

Logger to log exception:

- CWE-489 - Leftover Debug Code

Class variable fields should not have public accessibility:

- CWE-493 - Critical Public Variable Without Final Modifier

Handling files is security sensitive → Path.get():

- CWE-732 - Incorrect Permission Assignment for Critical Resource
- CWE-73 - External Control of File Name or Path
- CWE-20 - Improper Input Validation
- CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- CWE-538 - File and Directory Information Exposure
- CWE-403 - Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')

Handling files is security-sensitive - FileInputStream:

- CWE-732 - Incorrect Permission Assignment for Critical Resource
- CWE-73 - External Control of File Name or Path
- CWE-20 - Improper Input Validation
- CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
- CWE-400 - Uncontrolled Resource Consumption ('Resource Exhaustion')
- CWE-538 - File and Directory Information Exposure
- CWE-403 - Exposure of File Descriptor to Unintended Control Sphere ('File Descriptor Leak')

Using regular expression is security-sensitive:

- CWE-624 - Executable Regular Expression Error
- CWE-185 - Incorrect Regular Expression

Deserializing objects from an untrusted source is security-sensitive:

- CWE-502 - Deserialization of Untrusted Data

Using pseudorandom number generators (PRNGs) is security-sensitive:

- CWE-338 - Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)
- CWE-330 - Use of Insufficiently Random Values
- CWE-326 - Inadequate Encryption Strength
- CWE-310 - Cryptographic Issues

Vulnérabilités relevées par ZAP:

Application error disclosure:

- CWE-200: Information Exposure

Buffer Overflow:

- CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

X-Frame options Header Not Set:

- CWE-16: Configuration

X-Content-Type-Options Header Missing:

- CWE-16: Configuration

Web Browser XSS Protection not Enabled:

- CWE-933: Configuration

En observant les rapports de vulnérabilités produits par les deux outils utilisés, on remarque que les deux analyses montrent des vulnérabilités complètement différentes. On peut s'en assurer en comparant les identifiants CWE des vulnérabilités trouvées, tels que regroupés plus haut.

Cette différence peut s'expliquer par le fait que l'analyse du logiciel SonarCloud s'effectue au niveau du code source de l'application, puisqu'il s'agit d'une analyse statique, alors que le logiciel Zap procède à une analyse de l'API REST, sans accéder au code source, puisqu'il effectue des tests de pénétration. Les deux approches rappellent les concepts de tests boîte blanche et boîte noire respectivement.

SonarCloud décelé les vulnérabilités telles que les bugs, erreurs d'encryption, mauvaises odeurs présentes dans le code, alors que Zap relève les erreurs de configuration, de validation d'entrée utilisateur, et d'encapsulation de l'information, des erreurs qui sont plus proches et ont un impact possiblement plus grand du point de vue de l'utilisateur. Cette deuxième approche produit dans ce cas-ci un rapport moins détaillé et contenant moins de vulnérabilité, mais les deux rapports sont complémentaires puisque chacun des logiciels analysait une partie différente de l'application. SonarCloud n'a décelé aucune des erreurs de configuration de l'API, et Zap aucune erreur dans le formatage du code.

Ces deux méthodes ont leurs avantages et désavantages. L'analyse statique peut par exemple être appliquée à plusieurs logiciels, et se répéter de façon stratégique au cours du développement, facilitant les méthodes DevOps et d'intégration continue. Ceci la rend très efficace pour les vulnérabilités qui sont faciles à détecter de manière automatique. Par contre, pour plusieurs types des vulnérabilités, cette détection automatique est beaucoup plus complexe à effectuer, ce qui engendre un nombre augmenté de faux positifs. De plus, des problèmes liés à la configuration du logiciel ne sont pas détectés, puisque celle-ci n'est pas incluse dans le code source. Dans le cas des faux positifs, certains problèmes de sécurité ne soient pas exploitables, et il est difficile de confirmer que ces problèmes sont réellement des vulnérabilités. Des erreurs de compilation ou de dépendance non résolues pourraient également empêcher complètement l'analyse du logiciel, ce qui rend cette méthode extrêmement dépendante de l'environnement de développement. De la même manière, un problème pourrait passer inaperçu et occasionner un faux négatif si la structure du projet fait en sorte qu'il se trouve dans un module non compilé ou qui a été mis à jour



après la dernière indexation du code par l'outil d'analyse. L'approche de l'analyse statique permet, puisqu'effectuée pendant le développement, de réduire les coûts de correction des problématiques. De l'autre côté, les tests de pénétration tels que ceux effectués par Zap permettent d'augmenter la certitude des risques après le déploiement du logiciel, mais entraînent des coûts de correction plus grands, et oblige des efforts manuels afin d'effectuer les tests. [1]

En observant les avantages et désavantages des deux solutions utilisées dans le cadre de ce travail, aucun des deux logiciels ne permet à lui seul de garantir la qualité du code et du produit final, et les deux, ou des solutions équivalents, devraient être utilisés conjointement afin d'assurer du mieux possible un produit de qualité: l'analyse statique tout au long du développement, et les tests de pénétration chaque fois qu'une version stable est prête au déploiement.

# Références

[1] Polytechnique Montréal, *Notes de cours*, S.D.

[2] Common Weakness Enumeration, 2018. [En ligne]. Disponible:

<https://cwe.mitre.org/index.html>

# Annexe

## ZAP Scanning Report

### Summary of Alerts

Risk Level	Number of Alerts
<a href="#">High</a>	0
<a href="#">Medium</a>	3
<a href="#">Low</a>	3
<a href="#">Informational</a>	0

### Alert Detail

Medium (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	http://localhost:8080/algorithm/linearRegression
Method	POST
Evidence	HTTP/1.1 500 Internal Server Error
URL	http://localhost:8080/algorithm
Method	GET
Evidence	HTTP/1.1 500 Internal Server Error
URL	http://localhost:8080/cluster/Hierarchical
Method	POST
Evidence	HTTP/1.1 500 Internal Server Error
URL	http://localhost:8080/openapi/openapi.json
Method	GET
Evidence	Internal Server Error
URL	http://localhost:8080/openapi
Method	GET
Evidence	Internal Server Error
Instances	5
Solution	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user.
Reference	
CWE Id	200
WASC Id	13
Source ID	3

Medium (Medium)	Buffer Overflow
Description	Buffer overflow errors are characterized by the overwriting of memory spaces of the background web process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer) and other registers causes exceptions, segmentation faults, and other process errors to occur. Usually these errors end execution of the application in an unexpected way.
URL	http://localhost:8080/algorithm/BayesNet
Method	POST
Parameter	estimatorParams
Attack	POST http://localhost:8080/algorithm/BayesNet HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: application/json Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----1471854840170777951432989130 Origin: http://localhost:8080 Content-Length: 207064 Connection: keep-alive Host: localhost:8080

URL	http://localhost:8080/algorithm/BayesNet
Method	POST
Parameter	estimatorParams
Attack	POST http://localhost:8080/algorithm/BayesNet HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----19136200271530477873564879713 Origin: http://localhost:8080 Content-Length: 41605 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/MultilayerPerceptron
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/algorithm/MultilayerPerceptron HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----1799801460918405990382301749 Origin: http://localhost:8080 Content-Length: 22402 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/NaiveBayes
Method	POST
Parameter	useSupervisedDiscretization
Attack	POST http://localhost:8080/algorithm/NaiveBayes HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----199079267774208501596948710 Origin: http://localhost:8080 Content-Length: 6040 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/BayesNet
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/algorithm/BayesNet HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----19136200271530477873564879713 Origin: http://localhost:8080 Content-Length: 41608 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/linearRegression
Method	POST
Parameter	ridge
Attack	POST http://localhost:8080/algorithm/linearRegression HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----19592259215072254771113355178 Origin: http://localhost:8080 Content-Length: 10775 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/BayesNet
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/algorithm/BayesNet HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: application/json Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----1471854840170777951432989130 Origin: http://localhost:8080 Content-Length: 207067 Connection: keep-alive Host: localhost:8080

URL	http://localhost:8080/cluster/SimpleKMeans
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/cluster/SimpleKMeans HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/x-arrf Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----17542832771972984692332047420 Origin: http://localhost:8080 Content-Length: 22645 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/linearRegression
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/algorithm/linearRegression HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----19592259215072254771113355178 Origin: http://localhost:8080 Content-Length: 10776 Connection: keep-alive Host: localhost:8080
URL	http://localhost:8080/algorithm/NaiveBayes
Method	POST
Parameter	datasetUri
Attack	POST http://localhost:8080/algorithm/NaiveBayes HTTP/1.1 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0 Accept: text/uri-list Accept-Language: en-CA,en-US;q=0.7,en;q=0.3 Referer: http://localhost:8080/ Content-Type: multipart/form-data; boundary=-----199079267774208501596948710 Origin: http://localhost:8080 Content-Length: 6041 Connection: keep-alive Host: localhost:8080
Instances	10
Solution	Rewrite the background program using proper return length checking. This will require a recompile of the background executable.
Other information	Potential Buffer Overflow. The script closed the connection and threw a 500 Internal Server Error



Reference	<a href="https://www.owasp.org/index.php/Buffer_overflow_attack">https://www.owasp.org/index.php/Buffer_overflow_attack</a>
CWE Id	120
WASC Id	7
Source ID	1

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	<a href="http://localhost:8080/">http://localhost:8080/</a>
Method	GET
Parameter	X-Frame-Options
Instances	1
Solution	Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).
Reference	<a href="http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx">http://blogs.msdn.com/b/ieinternals/archive/2010/03/30/combating-clickjacking-with-x-frame-options.aspx</a>
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	<a href="http://detectportal.firefox.com/success.txt">http://detectportal.firefox.com/success.txt</a>
Method	GET
Parameter	X-Content-Type-Options
Instances	1
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.  If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.  At "High" threshold this scanner will not alert on client or server error responses.
Reference	<a href="http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx</a> <a href="https://www.owasp.org/index.php/List_of_useful_HTTP_headers">https://www.owasp.org/index.php/List_of_useful_HTTP_headers</a>
CWE Id	16
WASC Id	15
Source ID	3

Low (Medium)	Web Browser XSS Protection Not Enabled
Description	Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server
URL	<a href="http://localhost:8080/robots.txt">http://localhost:8080/robots.txt</a>
Method	GET
Parameter	X-XSS-Protection
URL	<a href="http://localhost:8080/algorithm/NaiveBayes">http://localhost:8080/algorithm/NaiveBayes</a>
Method	GET
Parameter	X-XSS-Protection
URL	<a href="http://localhost:8080/cluster/Hierarchical">http://localhost:8080/cluster/Hierarchical</a>
Method	GET
Parameter	X-XSS-Protection
URL	<a href="http://localhost:8080/cluster/SimpleKMeans">http://localhost:8080/cluster/SimpleKMeans</a>
Method	GET
Parameter	X-XSS-Protection

URL	http://localhost:8080/algorithm
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/algorithm/MultilayerPerceptron
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/cluster
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/cluster/Hierarchical
Method	POST
Parameter	X-XSS-Protection
URL	http://localhost:8080/algorithm/NaiveBayes
Method	POST
Parameter	X-XSS-Protection
URL	http://localhost:8080/swagger-ui
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/algorithm/linearRegression
Method	POST
Parameter	X-XSS-Protection
URL	http://localhost:8080/sitemap.xml
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/algorithm/linearRegression
Method	GET
Parameter	X-XSS-Protection
URL	http://localhost:8080/algorithm/BayesNet
Method	GET
Parameter	X-XSS-Protection
Instances	15

Solution	Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.
Other information	<p>The X-XSS-Protection HTTP response header allows the web server to enable or disable the web browser's XSS protection mechanism. The following values would attempt to enable it:</p> <p>X-XSS-Protection: 1; mode=block</p> <p>X-XSS-Protection: 1; report=http://www.example.com/xss</p> <p>The following values would disable it:</p> <p>X-XSS-Protection: 0</p> <p>The X-XSS-Protection HTTP response header is currently supported on Internet Explorer, Chrome and Safari (WebKit).</p> <p>Note that this alert is only raised if the response body could potentially contain an XSS payload (with a text-based content type, with a non-zero length).</p>
Reference	<a href="https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet">https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet</a> <a href="https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/">https://blog.veracode.com/2014/03/guidelines-for-setting-security-headers/</a>
CWE Id	933
WASC Id	14
Source ID	3

Low (Medium)	X-Content-Type-Options Header Missing
Description	The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.
URL	http://localhost:8080/algorithm/MultilayerPerceptron
Method	POST
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/swagger-ui-standalone-preset.js
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/openapi
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/jguweka.css
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/cluster/SimpleKMeans
Method	POST
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/swagger-ui.css
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/swagger-ui-bundle.js
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/algorithm/NaiveBayes
Method	POST
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/favicon-16x16.png
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/openapi/openapi.json
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/swagger-ui/favicon-32x32.png
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/algorithm/BayesNet
Method	POST
Parameter	X-Content-Type-Options
URL	http://localhost:8080/
Method	GET
Parameter	X-Content-Type-Options
URL	http://localhost:8080/algorithm/linearRegression
Method	POST
Parameter	X-Content-Type-Options
Instances	14
Solution	Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.  If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.
Other information	This issue still applies to error type pages (401, 403, 500, etc) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.  At "High" threshold this scanner will not alert on client or server error responses.

Reference	<a href="http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx">http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx</a> <a href="https://www.owasp.org/index.php/List_of_useful_HTTP_headers">https://www.owasp.org/index.php/List_of_useful_HTTP_headers</a>
CWE Id	16
WASC Id	15
Source ID	3