Sécuriser avec de bonnes pratiques de test

Journée sur la sécurité applicative – Université Laval

François Lajeunesse-Robert
2017-11-29

La force de l'engagement[MD]

Plan

- Introduction
  - Motivations et objectifs des tests
  - Établir la couverture des tests
- Prévention des vulnérabilités par le « fuzzing »
  - Préparation
  - Failles d’injection
  - Problématiques d’autorisation
  - Problématiques de logique
- Conclusion

Introduction

La force de l'engagement[MD]

Motivations

Trouver des problématiques le plus rapidement possible !

Tests vs analyse statique

Tests faits par un développeur vs tests faits par un pentesteur

Objectifs

Tests ≠ Prouver l’absence de problématiques

Tests = Identifier la présence de problématiques

Trouver toutes les problématiques? Non

Couverture de tests – Quelques chiffres

- 705 Common Weakness Enumeration (CWE) (https://cwe.mitre.org/)
- 100 cas de test de l'OWASP Testing Guide version 4 (https://www.owasp.org/index.php/OWASP_Testing_Project)
- En particulier pour le XSS :
  - 142+ variantes d'injection XSS (https://html5sec.org/)
  - Et… Encodage, techniques d'évasion, particularités des plateformes de développement, XSS Auditor, etc.

Couverture de tests – OWASP Testing Guide Categories

- Information gathering
- Configuration and Deployement Management Testing
- Identity Management Testing
- Authentifcation Testing
- Authorization Testing
- Session Managment Testing
- Input Validation Testing
- Testing for Error Handling
- Testing for weak Cryptography
- Business Logic Testing
- Client Side Testing

Couverture de tests – Une approche

Tester pour trouver les causes courantes de vulnérabilités plutôt que les vulnérabilités elles-mêmes.

Exemples de causes – Problématiques d’injection

Cause	Couverture des tests
Manque de validation des données (type et format)	Soumettre des données invalides

Exemples de causes – Problématiques d’autorisation

Cause	Couverture des tests
Manque de validation des autorisations	Tenter d’accéder à tout

Exemples de causes – Problématiques de logique

Cause	Couverture des tests
Suppositions sur le fonctionnement de l’application	Utiliser l’application de façon erratique (n’importe comment)

Couverture de tests – Fuzzing

« Fuzzing is the art of automatic bug finding, and it's role is to find software implementation faults, and identify them if possible. »

Tiré de l'article OWASP sur le fuzzing (https://www.owasp.org/index.php/Fuzzing)

Prévention des vulnérabilités par le « fuzzing »

CGI

La force de l'engagement[MD]

Préparation au fuzzing

1. Sélectionner un outil (Selenium, CasperJS, Burp, OWASP Zap, etc.).

2. Pouvoir surveiller le comportement de l'application (Erreurs, journalisation, procmon, jvm monitoring, etc.).

3. Sélectionner les tests à effectuer en fonction du contexte.

Mise en situation

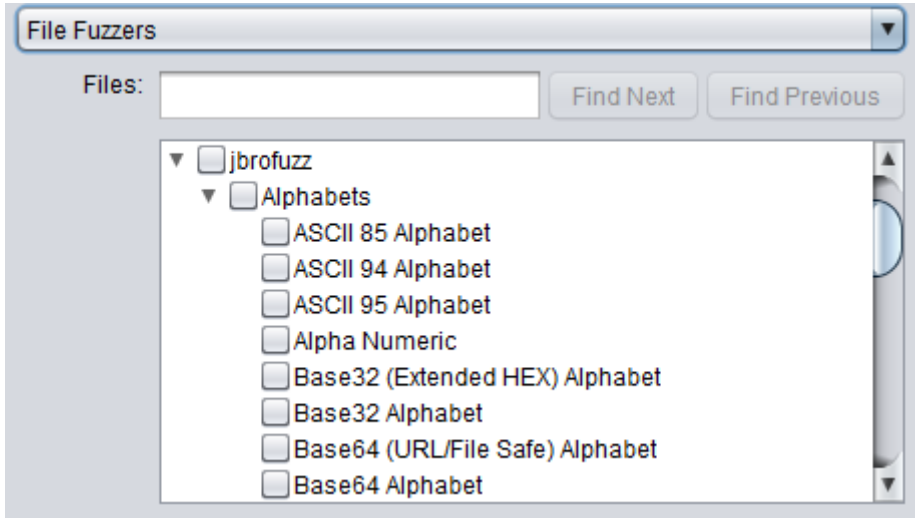
- Outil : 

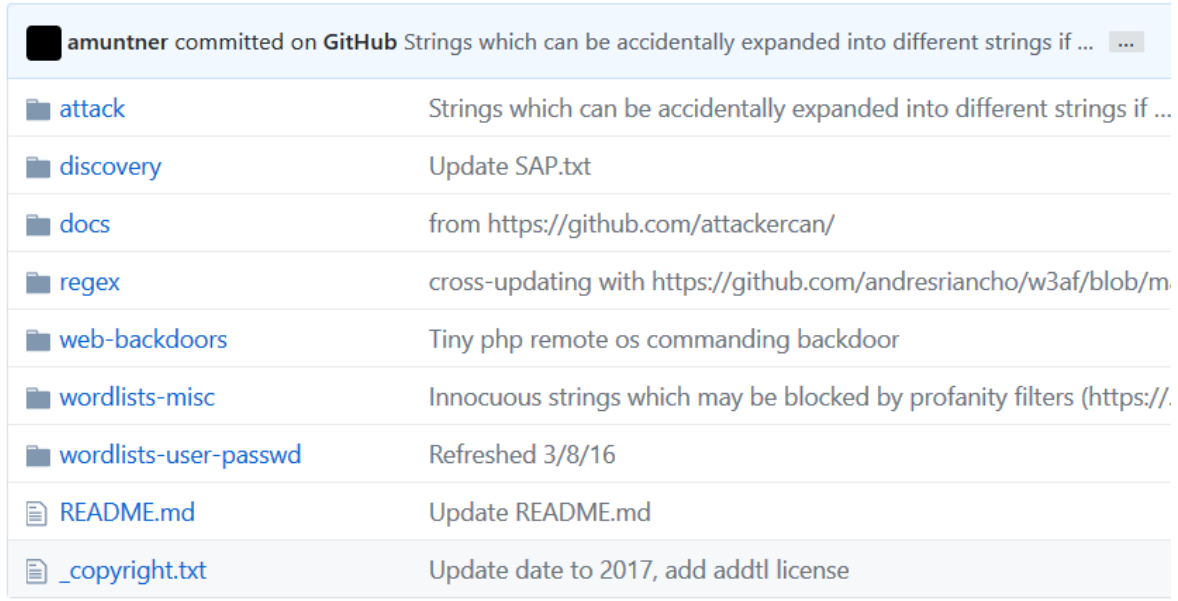
- Analyse de la réponse HTTP pour :
  - Mots clés dénotant une erreur (Unexpected, Stacktrace, Error, Exception, etc.)
  - Réflexion (contenu de la requête retourné intégralement)
  - Code de statut « intéressants »
  - Taille de la réponse

Prévenir les problématiques d’injection : Tester le format

Prévenir les problématiques d'injection

- Approche 1 – Dictionnaires de valeurs

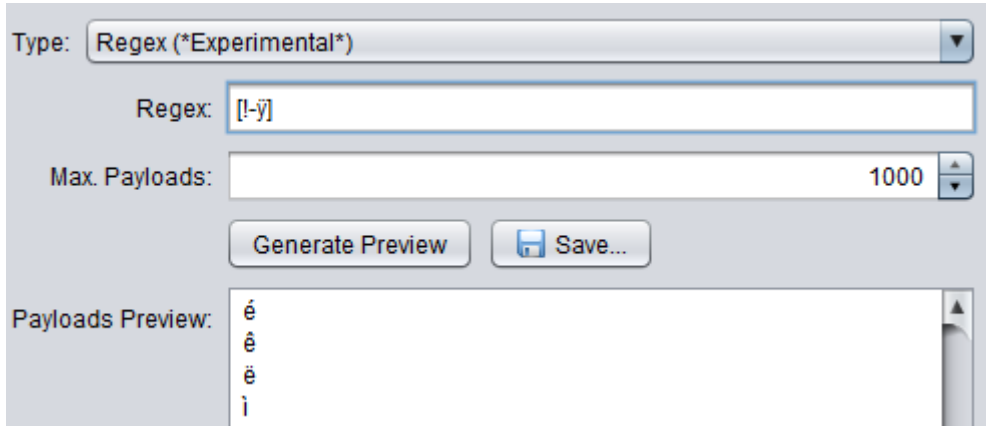


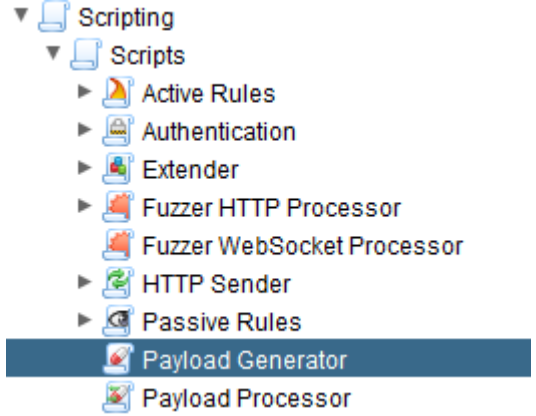


Source images: OWASP Zap, https://github.com/fuzzdb-project/fuzzdb

Prévenir les problématiques d’injection

- Approche 2 – Génération d’un jeu de valeurs personnalisé





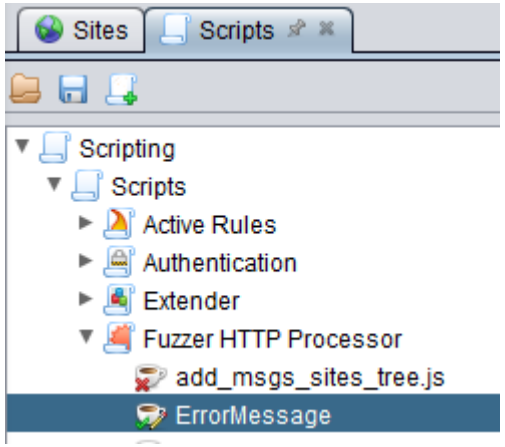
Source images: OWASP Zap

Prévenir les problématiques d’injection

Démo

Prévenir les problématiques d'injection – Démo

- Recherche de mots clés dans la réponse HTTP :
  - ZAP Script → Fuzzer HTTP Processeur

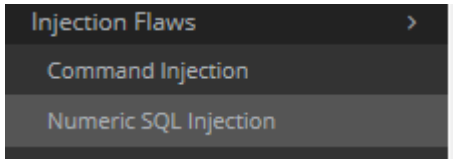



```
var keywords = [/Unexpected/i, /Stacktrace/i, /Error/i, /Exception/i];

// Called after receiving the fuzzed message from the server
function processResult(utils, fuzzResult){
     keywords.forEach(function (keyword) {
          if (fuzzResult.getHttpMessage().getResponseBody().toString().match(keyword) !== null)
               fuzzResult.addCustomState("Key Custom State", "Message Contains " + keyword + " keyword")
     });


     // Returns true to accept the result, false to discard and not show it
     return true;
}
```

Script : https://github.com/FrancoisLR/ZAPScripts/blob/master/httpfuzzerprocessor/ErrorMessage.js

Prévenir les problématiques d’injection – Démo

- WebGoat SQLi

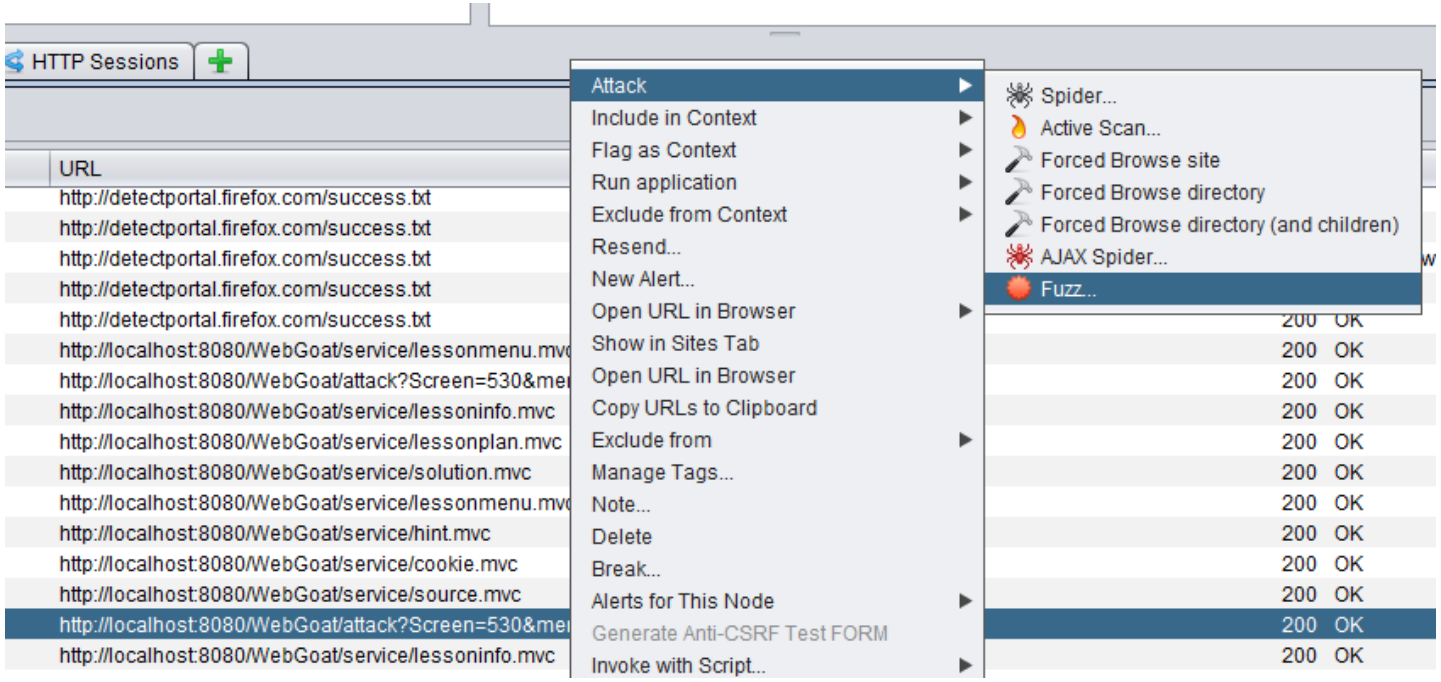


- Faire une requête

```
GET http://localhost:8080/WebGoat/attack?Screen=530&menu=1100&station=101&SUBMIT=Go! HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: http://localhost:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
```

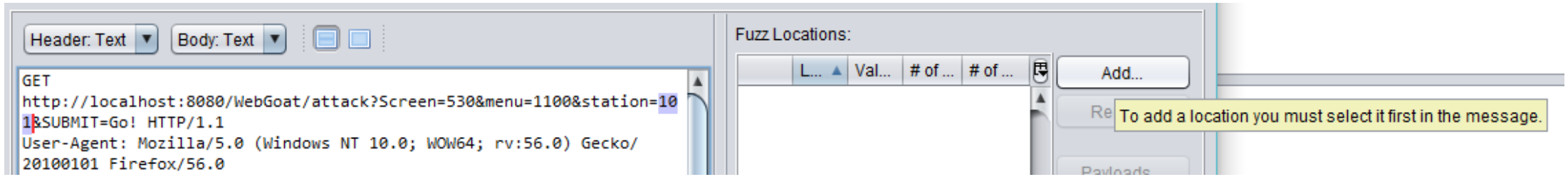
Prévenir les problématiques d'injection – Démo

- Lancer le « fuzzer »



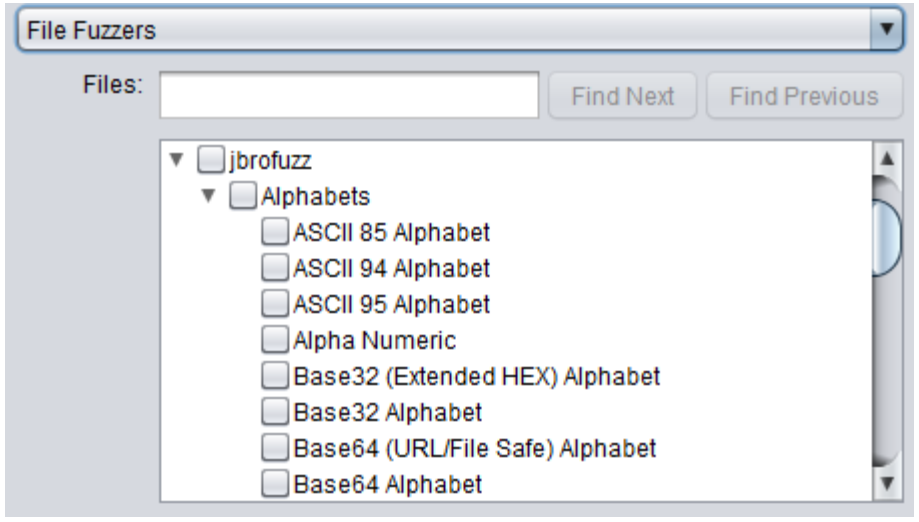
Prévenir les problématiques d’injection – Démo

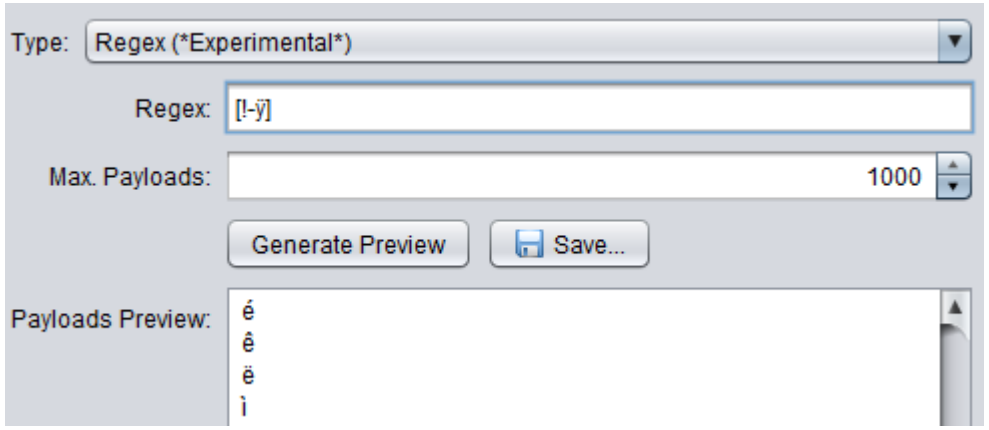
- Sélectionner l’endroit où « fuzzer »



Prévenir les problématiques d’injection – Démo

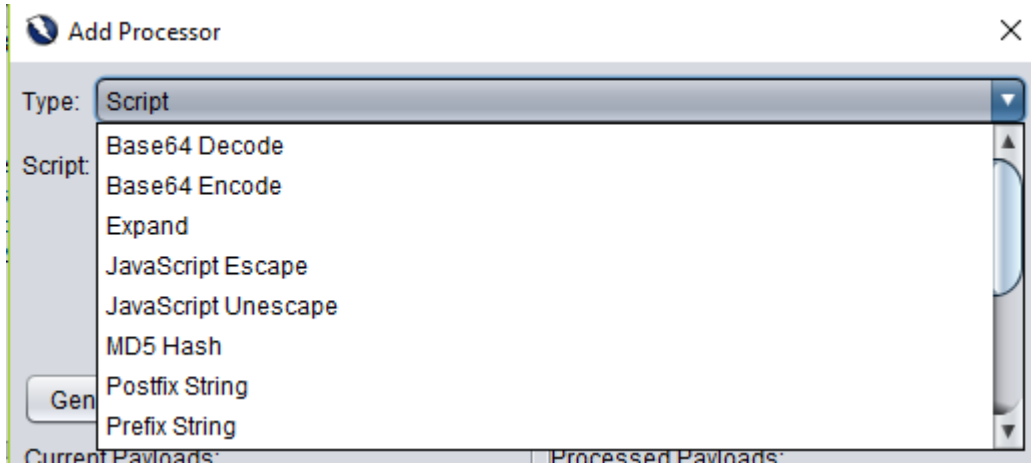
- Sélectionner quoi « fuzzer »

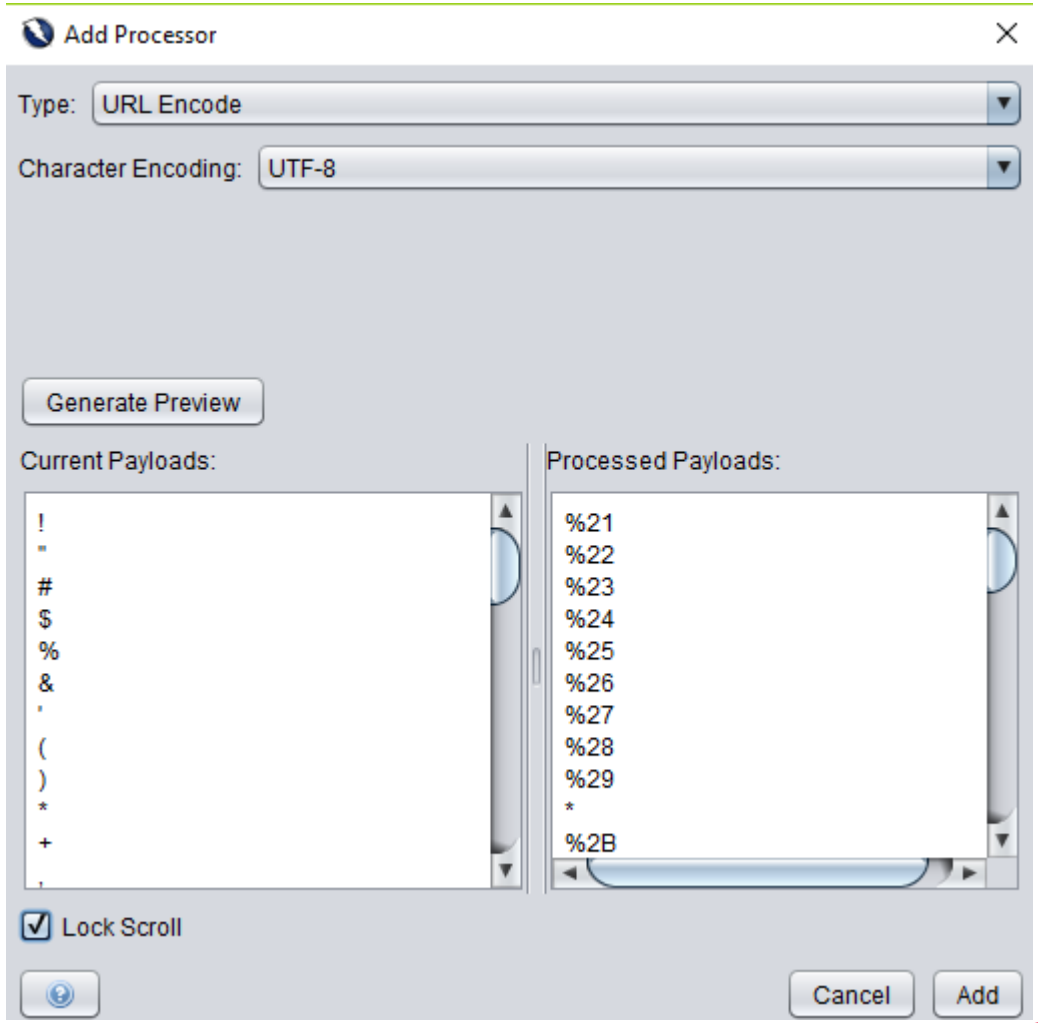




Prévenir les problématiques d'injection

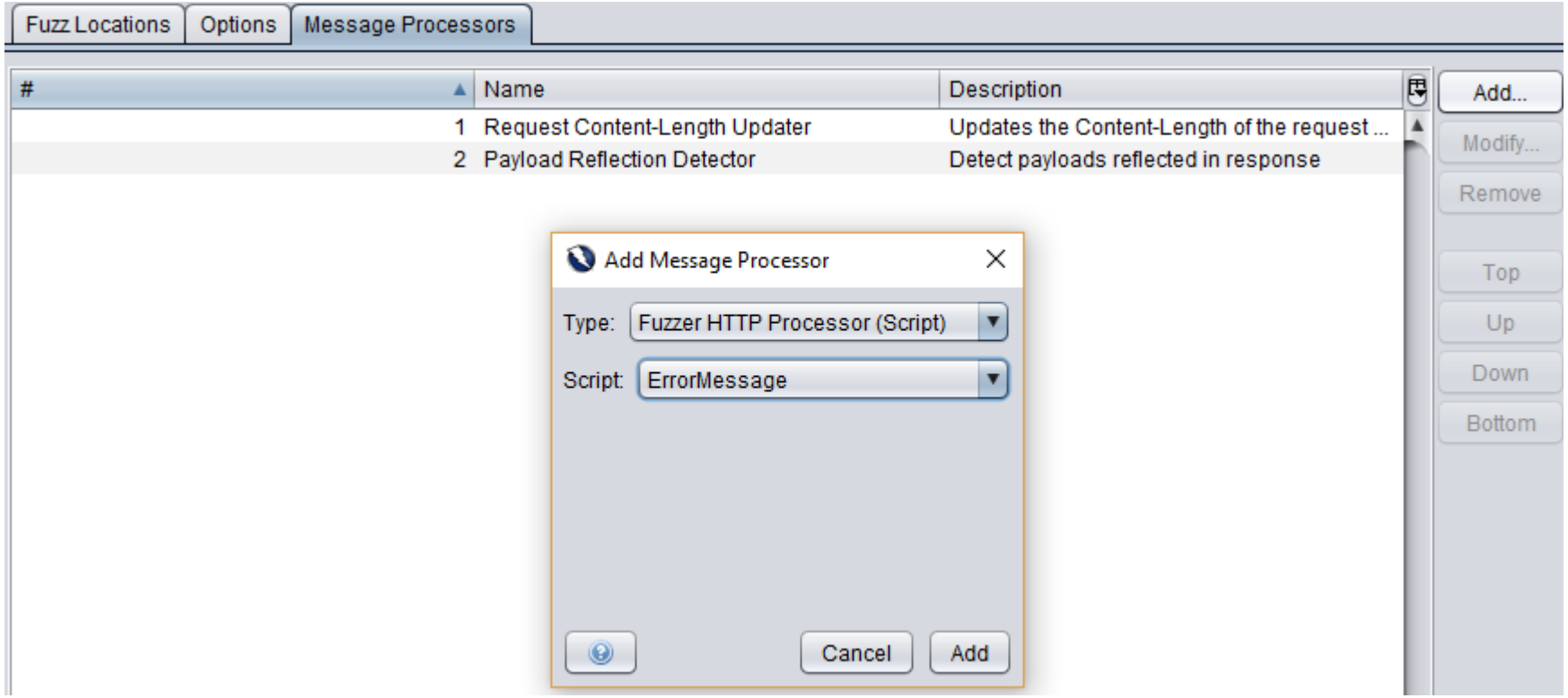
- Sélectionner le bon processeur





Prévenir les problématiques d'injection – Démo

- Sélectionner le ou les bons « processors »



Prévenir les problématiques d’injection – Démo

- Rouler et analyser le résultat du « fuzzer »

Size Resp. Body	Highest Alert	State	Payloads
1,392 bytes			%23
1,392 bytes			%24
1,392 bytes			%25
1,392 bytes			%26
1,448 bytes		Message Contains /Unexpected/i keyword	%27
1,392 bytes			%28
1,392 bytes			%29
1,392 bytes			*

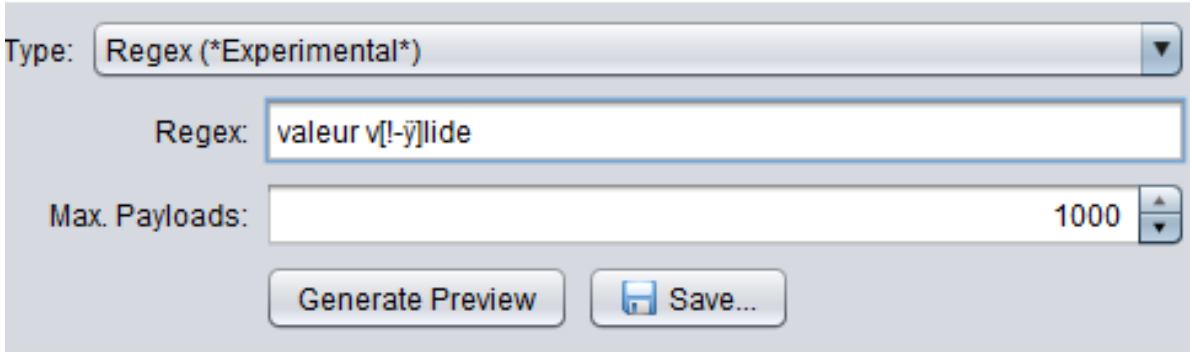
Prévenir les problématiques d’injection

Attention

[!-ÿ][!-ÿ] = 49 284 possibilités

Prévenir les problématiques d’injection
Être plus efficace

- Modifier quelque peu des valeurs valides



Source image: OWASP Zap

Prévenir les problématiques d'injection
Être plus efficace (suite)

- En fonction des validations côté client

```
<select name="station">
  <option value="101">Columbia</option>
  <option value="102">Seattle</option>
  <option value="103">New York</option>
  <option value="104">Houston</option>
</select>
```

```
<form action="/action_page.php">
Country code: <input type="text" name="country_code"
pattern="[A-Za-z]{3}" title="Three letter country code">
<input type="submit">
</form>
```

```
function isNumeric(_str) {
    try {
        parseInt(_str);
        return true;
    } catch(e) {
        return false;
    }
}
```

Sources images: WebGoat, https://www.w3schools.com/tags/att_input_pattern.asp

CGI

Prévenir les problématiques d’autorisation: Énumération des ressources

Prévenir les problématiques d’autorisation

- Énumérer les ressources (statiques et endpoints)

```
>dir /s /b .extract > filelist.txt
```

```
http://localhost:8080/WebGoat/css
http://localhost:8080/WebGoat/database
http://localhost:8080/WebGoat/fonts
http://localhost:8080/WebGoat/images
http://localhost:8080/WebGoat/index.jsp
http://localhost:8080/WebGoat/js
http://localhost:8080/WebGoat/lessons
http://localhost:8080/WebGoat/lesson_content.jsp
http://localhost:8080/WebGoat/main.jsp
http://localhost:8080/WebGoat/META-INF
http://localhost:8080/WebGoat/mfe_target
http://localhost:8080/WebGoat/plugins
http://localhost:8080/WebGoat/plugin_extracted
http://localhost:8080/WebGoat/plugin_lessons
http://localhost:8080/WebGoat/reportBug.jsp
http://localhost:8080/WebGoat/sideWindow.jsp
http://localhost:8080/WebGoat/uploads
http://localhost:8080/WebGoat/users
http://localhost:8080/WebGoat/WEB-INF
http://localhost:8080/WebGoat/webgoat.jsp
http://localhost:8080/WebGoat/webgoat_challenge.jsp
http://localhost:8080/WebGoat/css/animate.css
http://localhost:8080/WebGoat/css/font-awesome.min.css
http://localhost:8080/WebGoat/css/img
```


CGI

Prévenir les problématiques d’autorisation

Comparer les sessions – Les codes de statuts

GET	http://localhost:8080/WebGoat/css/layers.css	200	200
GET	http://localhost:8080/WebGoat/css/lesson.css	200	200
GET	http://localhost:8080/WebGoat/css/main.css	200	200
GET	http://localhost:8080/WebGoat/css/menu.css	200	200
GET	http://localhost:8080/WebGoat/css/webgoat.css	200	200
GET	http://localhost:8080/WebGoat/database	404	200
GET	http://localhost:8080/WebGoat/database/database.prp	200	200
GET	http://localhost:8080/WebGoat/fonts	404	404
GET	http://localhost:8080/WebGoat/fonts/FontAwesome.otf	200	200
GET	http://localhost:8080/WebGoat/fonts/fontawesome-webfont.eot	200	200

Prévenir les problématiques d’autorisation

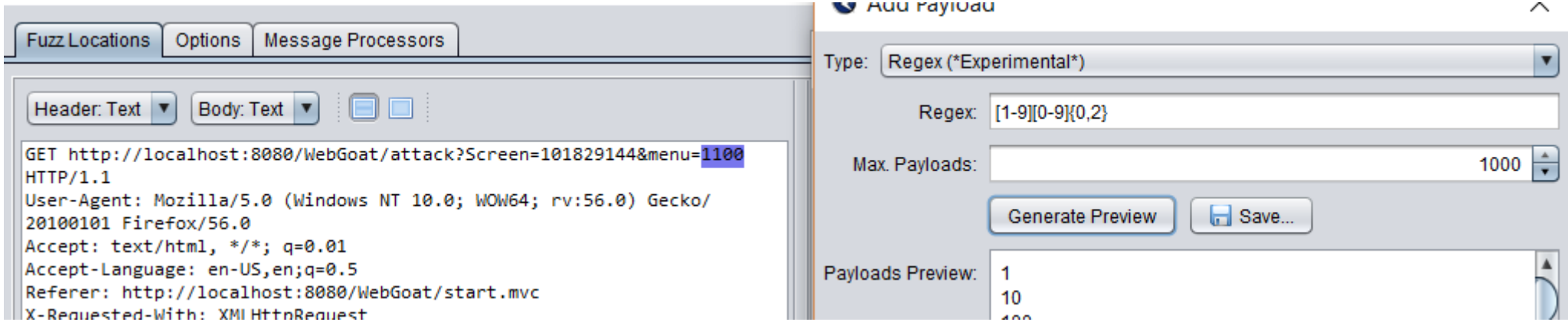
Comparer les sessions – Les tailles des réponses

Id	Req. Timestamp	Method	URL	C...	Reason	RTT	Size Resp. Body	Highest Alert	Note	Tags
387	27/11/17 14:47:20	GET	http://localhost:8080/WebGoat/plugin_less...	200	OK	16...	784,789 bytes	Low		Comment
390	27/11/17 14:47:48	GET	http://localhost:8080/WebGoat/logout.mvc	200	OK	65...	2,000 bytes	Medium		SetCookie, Co...
397	27/11/17 14:48:12	GET	http://localhost:8080/WebGoat/logout.mvc	200	OK	15...	2,000 bytes	Medium		Comment
398	27/11/17 14:48:12	GET	http://localhost:8080/WebGoat/css/main.css	200	OK	16...	19,986 bytes	Low		Comment
399	27/11/17 14:48:12	GET	http://localhost:8080/WebGoat/css/animate...	200	OK	16...	59,837 bytes	Low		Comment
400	27/11/17 14:48:12	GET	http://localhost:8080/WebGoat/plugins/boot...	200	OK	16...	99,961 bytes	Low		Upload, Comm...
402	27/11/17 14:48:12	GET	http://localhost:8080/WebGoat/css/font-awe...	200	OK	16...	17,780 bytes	Low		Comment
408	27/11/17 14:48:42	GET	http://localhost:8080/WebGoat/plugin_less...	200	OK	31...	3,150 bytes	Medium		Form, Passwor...

Source image: OWASP Zap

Prévenir les problématiques d’autorisation: Énumération des identifiants

Prévenir les problématiques d’autorisation



Source images: OWASP Zap

Prévenir les problématiques d'autorisation

Optimiser l'énumération

Identifiants séquentiels

```
at/attack?Screen=1075773632 HTTP/1.1
 NT 10.0; Win64; x64; rv:57.0) Gecko/2
```

Max integer (JAVA & .NET) : 2 147 483 647

Identifiants aléatoires

```
29aaab9f-3f27-43aa-9216-94f882eb0a7c
```

GUID : 2^122 bits possibilités !!!!

```
SELECT id FROM USERS
```

Prévenir les problématiques de logique

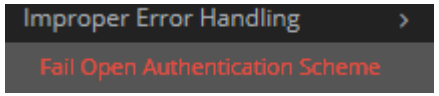
Ajout / retrait de paramètres

Prévenir les problématiques de logique

Démo

Prévenir les problématiques de logique – Démo

- WebGoat Improper Error Handling

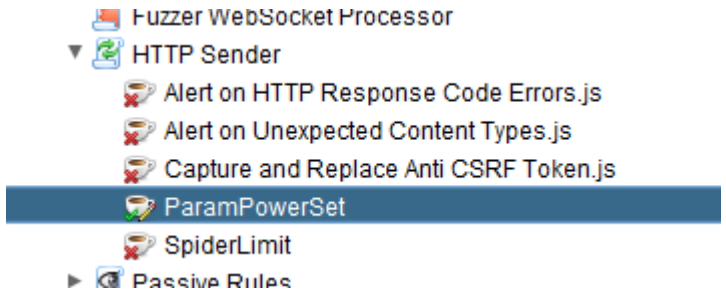


- Faire une requête

```
GET http://localhost:8080/WebGoat/attack?Screen=279&menu=1000&Username=webgoat&Password=sdfsdfewr&SUBMIT=Login
1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: http://localhost:8080/WebGoat/start.mvc
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DF4E23A4CC0594FC773E04AA3C5878A5
Connection: keep-alive
Host: localhost:8080
```

Prévenir les problématiques de logique – Démo

- Activer le script testant toutes les possibilités

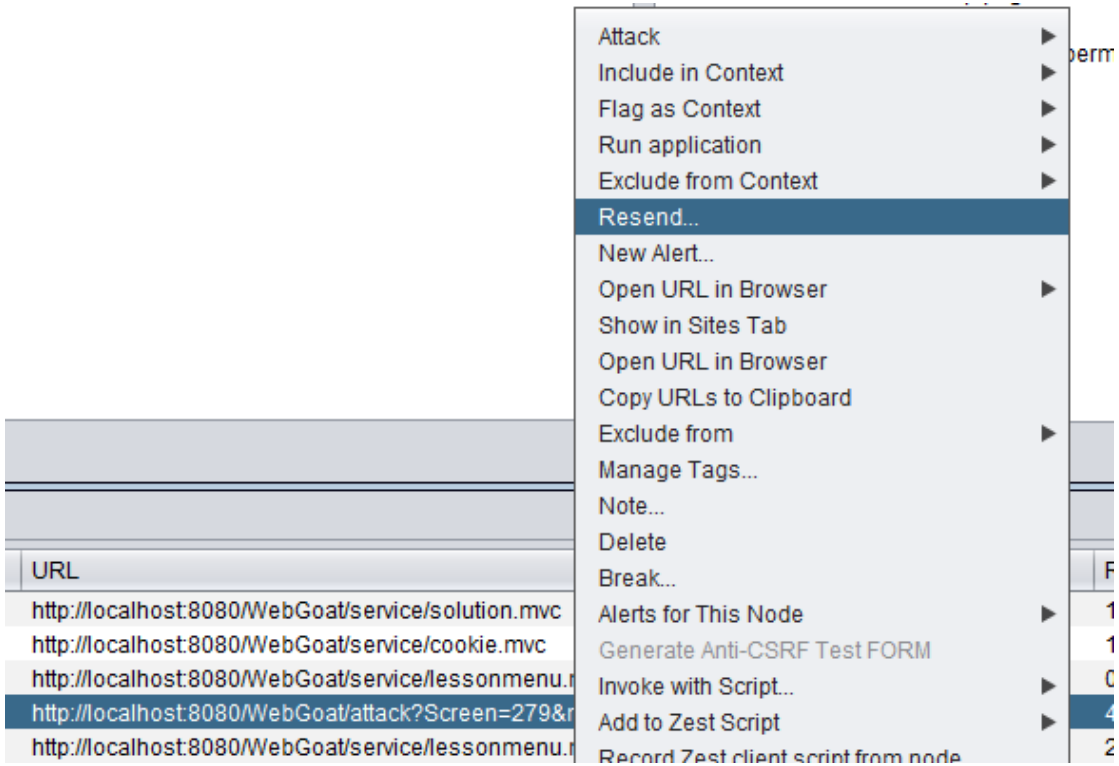


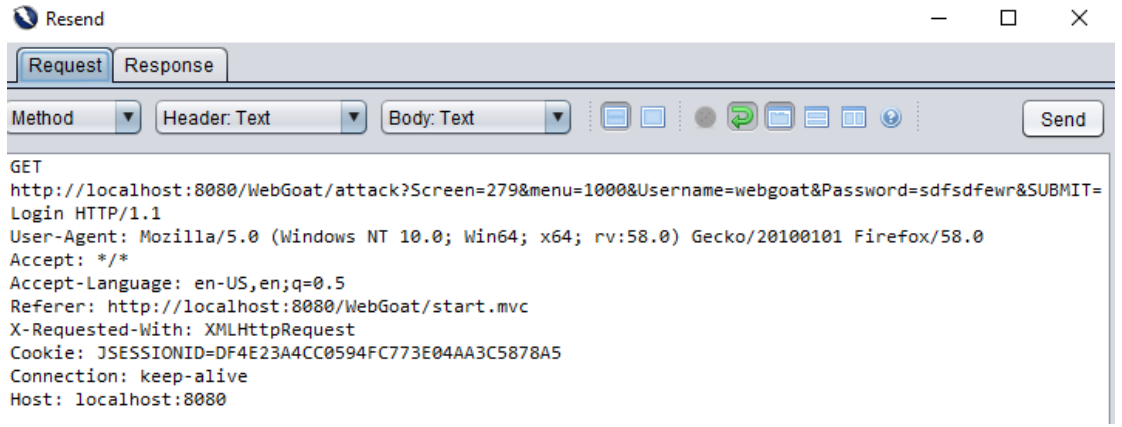

```
powerSet(reqParams).forEach(function (_combination) {
    var newMsg = msg.cloneAll();
    urlParamsTreeSet.clear();
    formParamsTreeSet.clear();
    print(_combination);

    _combination.forEach(function (elem) {
        switch(elem[0]) {
        case "URL":
            urlParamsTreeSet.add(elem[1]);
            break;
        case "FORM":
            formParamsTreeSet.add(elem[1]);
            break;
        }
    });
```

Script : https://github.com/FrancoisLR/ZAPScripts/blob/master/httpsender/ParamsPowerSet.js

Prévenir les problématiques de logique – Démo

- Lancer la requête manuellement





Prévenir les problématiques de logique

```
POST http://localhost:8080/WebGoat/attack?Screen=1075773632&menu=1000 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 46
Cookie: JSESSIONID=D9473324754E5EF15AFFCAEFA88ACCA0
Connection: keep-alive
Host: localhost:8080
```

```
Username=webgoat&Password=wwaeqwe&SUBMIT=Login
```

2^x possibilités

où x est le nombre de paramètres

Prévenir les problématiques de logique
Éviter l’explosion combinatoire

- Tester uniquement sur les cas pertinents

```
<td>
  <input name="Username" value="" type="TEXT" required="">
</td>
</tr>
<tr>
<td>…</td>
<td>
  <input name="Password" value="" type="PASSWORD" required="">
```

Conclusion

La force de l'engagement[MD]

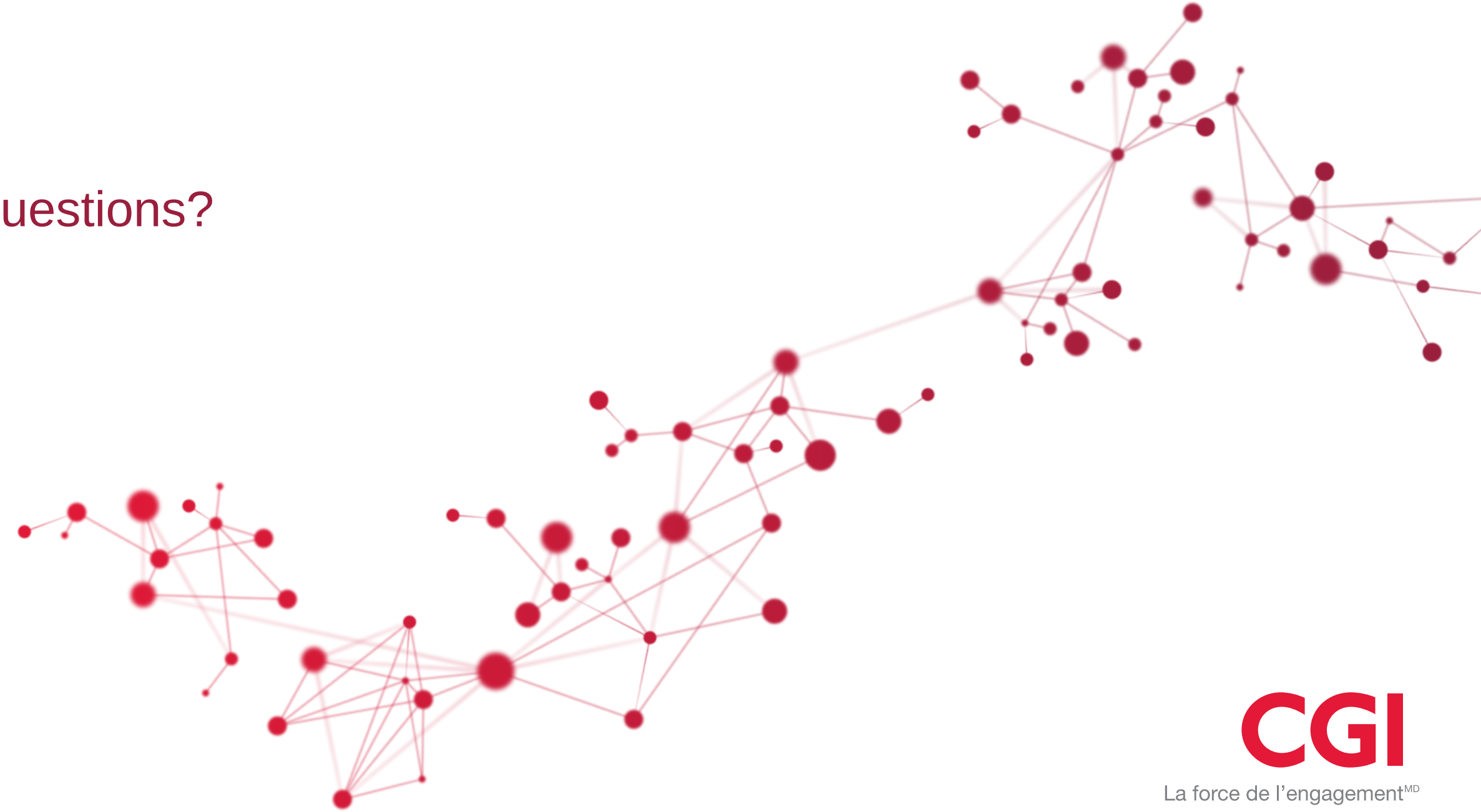
Sécuriser avec de bonnes pratiques de test

- Inscrire les tests en cours de développement comme une étape supplémentaire de validation;

- Tester pour trouver les causes courantes de vulnérabilités;

- Tirer profit du fait d'exécuter les tests tôt dans le cycle :
  - Identifier et corriger les comportements anormaux;
  - Rapidité d'exécution des tests;
  - Limiter au maximum les cas de tests en se concentrant sur les cas limites.

Questions?



CGI

La force de l'engagement[MD]