Màster en Seguretat de les Tecnologies de la Informació i les Comunicacions

Treball de Fi de Màster

Seguretat en Aplicacions Web

EINES DE DETECCIÓ DE VULNERABILITATS EN APLICACIONS WEB – CONTROL DE DADES D'ENTRADA

Josep Villar Azorín

Directors: Jordi Duch, Robert Rallo

A Laura, Jan i Klara

Eines de detecció de vulnerabilitats en aplicacions Web

RESUM

Després de presentar les vulnerabilitats relacionades amb les dades d'entrada en les aplicacions Web, així com un estudi de les eines de detecció de vulnerabilitats disponibles actualment, aquest document proposa per una banda la tria d'una eina de codi lliure, com és Skipfish, i per una altra una eina, una aplicació Web, que proporcioni una interfície d'usuari més amigable així com un complement a l'informe que genera Skipfish, afegint explicacions i referències a les vulnerabilitats a més de propostes de solució a aquests.

ABSTRACT

After the introduction to the world of input data vulnerabilities in web applications as well as a study of some tools currently available for their detection, this paper proposes the choice of an open source tool - Skipfish – accompanied with a new tool consisting in a web application which provides a more friendly user interface adding further information on the detected vulnerabilities, including some tips to solve them.

1 ÍNDEX

Taula de continguts

1 ÍNDEX....4
2 INTRODUCCIÓ....6
3 VULNERABILITATS EN L'ENTRADA DE DADES....8
3.1 Descripció de les vulnerabilitats....8
3.1.1 Atacs d'injecció d'scripts....8
3.1.2 Atacs d'injecció de codi....12
3.1.2.1 Injecció SQL[8][9][10]....13
3.1.2.2 Injecció LDAP....16
3.1.2.3 Injecció ORM....16
3.1.2.4 Injecció XML....16
3.1.2.5 Injecció SSI....17
3.1.2.6 Injecció Xpath....18
3.1.2.7 Injecció IMAP/SMTP....18
3.1.2.8 Injecció de Codi....20
3.1.3 Injecció de comandes....20
3.1.4 Buffer overflow....21
3.1.5 Vulnerabilitat incubada....22
3.1.6 HTTP splitting/smuggling....22
3.1.6.1 HTTP splitting....22
3.1.6.2 HTTP smuggling....24
3.2 Eines de detecció de vulnerabilitats....25
3.2.1 Especificacions funcionals....25
3.2.2 Criteris d'avaluació....28
3.2.2.1 Protocols suportats....28
3.2.2.2 Autenticació....29
3.2.2.3 Gestió de sessions....29
3.2.2.4 Crawling....30
3.2.2.5 Anàlisi (Parsing)....31
3.2.2.6 Testing....31
3.2.2.7 Interfícies de comandes, control i configuració....32
3.2.2.8 Producció d'informes....33
3.2.3 Efectivitat de les eines....34
3.3 Conclusions....39
4 ESTUDI D'UNA EINA ESPECÍFICA: SKIPFISH....40
4.1 Introducció....40
4.2 Característiques....43
4.3 Instal·lació....44
4.4 Execució....44
4.5 Resultats....46
4.6 Execució d'un test....50
4.7 Àrees de millora....52
4.8 Conclusions....52
5 INTERFÍCIE WEB....53
5.1 Descripció funcional....53

5.1.1 Introducció....53
5.1.2 Presentació del menú de l'aplicació....57
5.1.3 Nova definició....57
5.1.4 Gestió de definicions....65
5.1.4.1 Executar....66
5.1.4.2 Editar....68
5.1.4.3 Eliminar....68
5.1.5 Opcions de la llista d'execucions....70
5.1.5.1 Eliminar....70
5.1.5.2 Comparar execucions triades....70
5.1.5.3 Consulta (Informe)....72
5.2 Implementació....76
5.2.1 Infraestructura utilitzada....76
5.2.2 Model de Dades....78
5.2.3 Definició de l'aplicació Web....82
5.2.4 Configuració Spring....83
5.2.5 Altres elements....91
5.3 Conclusions sobre l'eina....92
6 BIBLIOGRAFIA / REFERÈNCIES....93

2 INTRODUCCIÓ

L'ús d'Internet a la vida diària ha augmentat en els últims anys de forma exponencial, tant des del punt de vista de l'oci com des del punt de vista del comerç, els negocis i les finances.

Aquesta situació propicia també el mal ús d'aquest mitjà de comunicació global, ja sigui per interessos personals, ja sigui per fets delictius. Els beneficis que es poden obtenir poden ser molt grans, des de l'obtenció no consentida de *targets* per qüestions comercials fins directament al robatori, tant de dades sensibles (informació financera, de seguretat nacional, personal) com de diners (*phishing*).

Internet, en tractar-se d'una xarxa per definició oberta és, a priori, accessible per tothom i des de qualsevol racó del món. Així doncs, ha de ser responsabilitat dels nodes finals i les aplicacions que contenen preservar la seguretat de les dades que es comparteixen.

El principal mode d'accés de les pràctiques que porten a aquest mal ús són les dades que s'intercanvien dos agents entre sí, especialment pel que fa les dades d'entrada.

Evidentment, hi ha d'altres aspectes que han de conformar la seguretat del sistema, però les dades d'entrada poden semblar *a priori* l'element més *innocent* del conjunt i són *de facto* el principal problema. Tenir cura del control de les dades d'entrada és l'aspecte fonamental en la seguretat d'un Sistema.

Hi ha una sèrie d'organismes que vetllen per la seguretat de les aplicacions en Internet. S'ocupen d'estudiar i publicar els aspectes que posen en compromís la seguretat, les anomenades vulnerabilitats. Proporcionen informació i propostes de solució. D'entre els més reconeguts troben l'OWASP (*Open Web Application Security*).

OWASP té obert un projecte anomenat *Testing Guide*[1][2], relacionat amb els aspectes a verificar una aplicació Web, on hi ha un capítol específic pel que fa a les dades d'entrada.

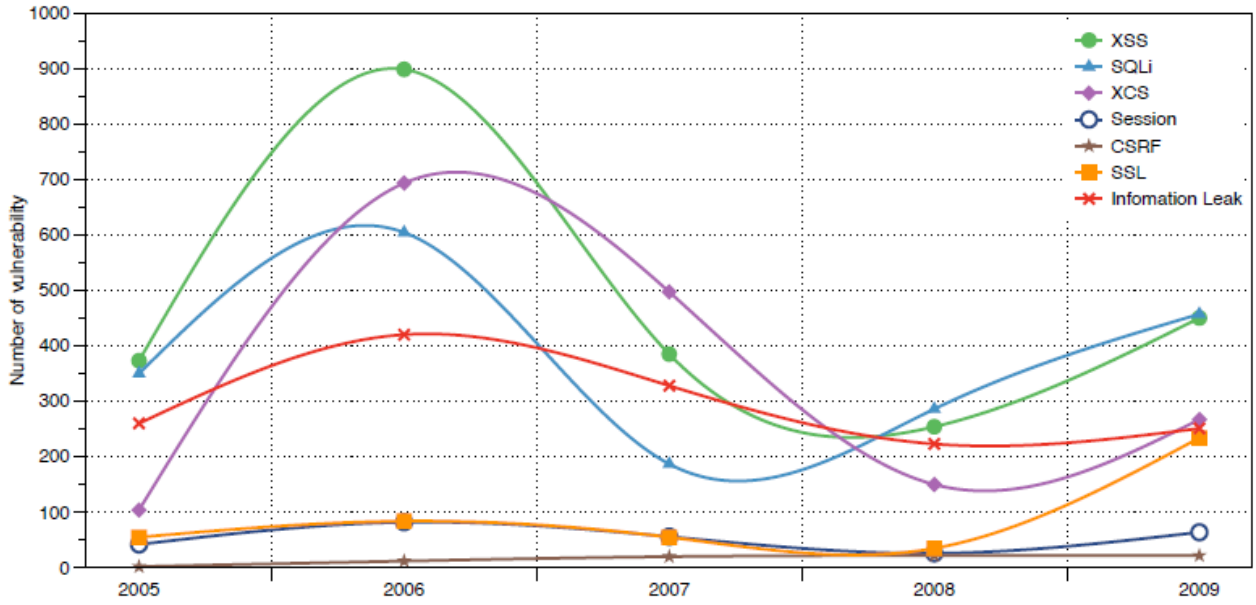
A grans trets, el lema d'aquest capítul és *input is evil.* Si una aplicació no té cura de les dades que rep, podria comprometre el sistema. Per exemple, una categoría de vulnerabilitats són les anomenades *injeccions SQL*. Depenent com està desenvolupada l'aplicació i els controls que (no) contempla, un atacant podria arribar a obtenir informació sensible o inclús autenticar-se dintre de l'aplicació com un usuari amb rols d'administració.

Per tal de poder verificar que les aplicacions Web compleixin els aspectes de seguretat racomanats per organismes com l'OWASP, hi ha disponibles una sèrie d'eines, tant comercials com de lliure accés, a tal efecte.

En aquest document inicialment es presenten les vulnerabilitats relacionades amb les dades d'entrada segons l'esmentat projecte de l'OWASP. També es presenten propostes de solució per a diferents vulnerabilitats descrites.

Lligat amb això, es presenten les característiques de les eines que actualment hi ha disponibles que, a més, haurien de complir unes especificacions funcionals[3] per les quals han d'estar sotmeses a uns criteris d'avaluació[4] .

Hi ha d'altres organismes que s'encarreguen de recollir les vulnerabilitats i publicar-les, com OSVDB (Open Source Vulnerability Database) per fer-les públiques. A tall d'exemple, aquest organisme va publicar en l'any 2009 una comparativa de les vulnerabilitats reportades. El següent gràfic mostra de forma molt clara l'aparició de vulnerabilitats en funció també de l'eclosió de l'ús d'Internet com a mitjà global de comunicació:



Es pot comprovar una pujada significativa de les vulnerabilitats l'any 2006. A partir de 'any 2007, quan OWASP publica el seu Top Ten 2007[12] (llistat de les 10 vulnerabilitats més significatives i *petides*) i es fa extensiu de forma global (existia un Top Ten 2004, però encara no semblava estar molt estès) es pot veure una caiguda de les vulnerabilitats. Evidentment, després tornen a pujar, de forma més suau, però és degut a l'evolució que hi ha de les diferents tecnologíes implicades.

A continuació es presenta una d'aquestes eines, *Skipfish*[13], en aquest cas de codi lliure. La motivació principal de triar aquesta eina és precisament la seva naturalesa d'accés universal a tothom. Dintre de les que existeixen dintre de la seva categoria sembla una de les més completes, a més de ser un projecte amb molta activitat avui en dia, cosa que assegura la seva evolució i millora.

Tot i que aquesta eina està molt ben considerada, té uns aspectes millorables en quant fa als criteris funcionals requerits, especialment en el camp de la interacció amb l'usuari i la qualitat final dels informes obtinguts.

Per aquest motiu, la última part d'aquest document consisteix en la proposta d'una aplicació que, utilitzant com a base l'eina triada, intenta millorar aquests aspectes. Proporciona a l'usuari una interfície molt més amigable, li permet emmagatzemar de forma ordenada els resultats de diferents execucions i millora els resultats obtinguts proporcionant més informació a l'usuari pel que fa a les vulnerabilitats trobades i propostes per a la seva resolució.

3 VULNERABILITATS EN L'ENTRADA DE DADES

3.1 Descripció de les vulnerabilitats

El principi fonamental que hauria de regir-se en aquest aspecte és la desconfiança total davant de les dades d'entrada (*all input is evil*).

En aquesta secció es presenta cadascuna de les diferents agrupacions de vulnerabilitats. Aquesta presentació inclou una descripció del problema, incloent-hi un exemple sempre que sigui possible, unes propostes de tècniques per combatre-les (moltes de les quals seran compartides per moltes de les diferents tipologies de vulnerabilitats) i l'impacte sobre les aplicacions que han pogut produir.

3.1.1 Atacs d'injecció d'scripts

Aquesta categoria engloba els següents punts de l'esmentat article de la *OWASP Testing Guide v3*:

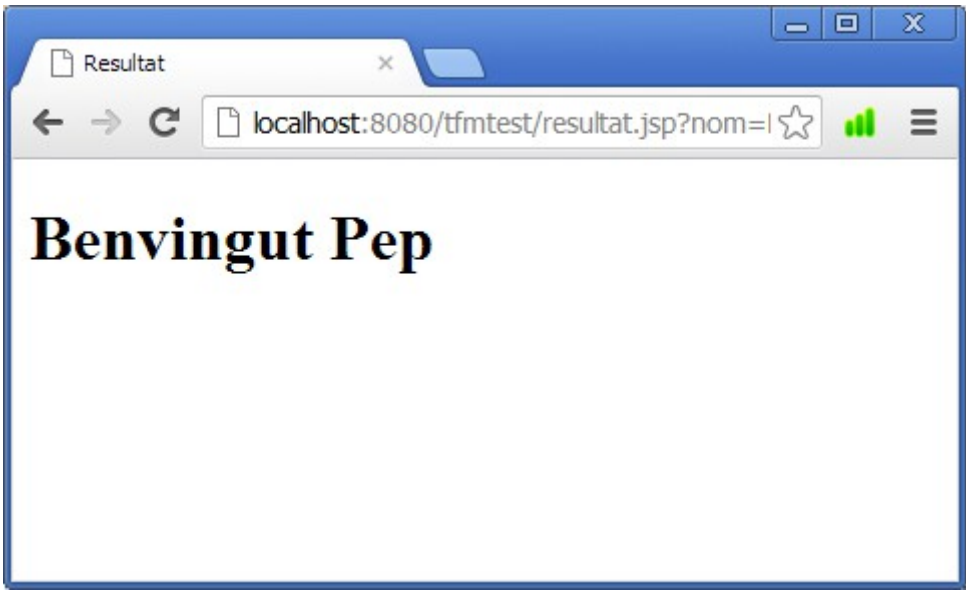
- 4.8.1 *Testing for Reflected Cross Site Scripting*[6] (OWASP-DV-001)
- 4.8.2 *Testing for Stored Cross Site Scripting* (OWASP-DV-002)
- 4.8.3 *Testing for DOM based Cross Site Scripting* (OWASP-DV-003)
- 4.8.4 *Testing for Cross Site Flashing* (OWASP-DV-004)

Els atacs d'injecció d'scripts es donen quan un atacant envia codi, generalment *javascript*, cap a l'aplicació web destí. Aquest codi és enviat mitjançant els paràmetres que pot rebre l'aplicació en la pròpia petició http, tant en el seu valor com, en determinades ocasions, en el propi nom del paràmetre. Una situació que s'ha de donar per tal que es produeixi l'atac, és que els paràmetres enviats a la pàgina acabin formant part de la resposta.

Considerem per exemple una pàgina on es demana el nom de l'usuari. Aquesta dada normalment s'introduirà mitjançant un element `HTML INPUT` (o `TEXTAREA`) i posteriorment s'efectuarà un enviament (*submit*) cap una altra pàgina on, entre d'altra informació, apareixerà el nom introduït. Si aquesta dada porta codi maliciós i no es tracta convenientment per la part servidora de l'aplicació, aquest codi no desitjat formarà part de la pàgina de destí.

A continuació es mostra el resultat:

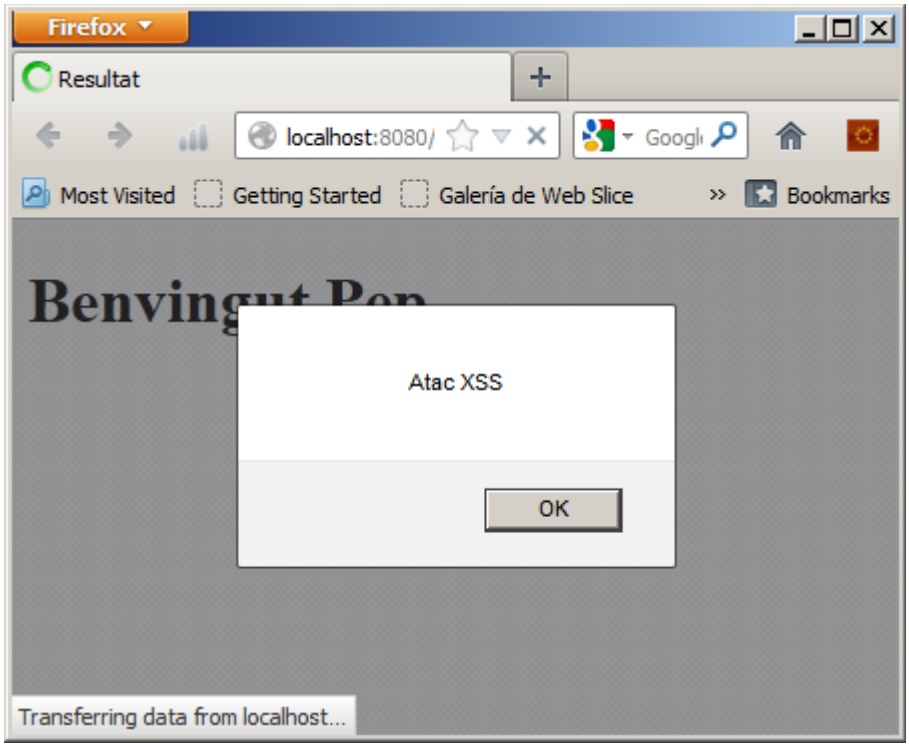
Si el contingut introduït és *correcte*, el resultat serà el següent:



Si el contingut introduït conté codi maliciós (per exemple amb el següent valor:

```
Pep</h1><script>alert('Atac XSS');</script><h1>
```

S'obté un comportament no desitjat:



El codi d'aquesta pàgina és el següent:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Resultat</title>
</head>
<body>
<h1>Benvingut <%=request.getParameter("nom") %></h1>

</body>
</html>
```

El problema està, com s'ha descrit prèviament, en el no tractament de les dades d'entrada en la pàgina de resultat. S'ha marcat en vermell el tros de codi incorrecte des del punt de vista de la seguretat.

Òbviament, aquest exemple és totalment inofensiu, però es pot arribar a incloure codi realment perillós.

Els atacs més habituals que es poden patir amb aquesta vulnerabilitat són els següents:

- Phishing. Es pot arribar a canviar el comportament o l'aparença d'un lloc vàlid (per a l'usuari inclús el certificat és correcte). Es pot superposar codi, generar un nou formulari, que enviï

dades sensibles (passwords, números de targeta, coordenades, etc...) a un tercer servidor.

- Presa de control del navegador client, permetent determinades accions sobre l'aplicació web amb els permisos que té l'usuari de l'aplicació. Es pot arribar, per exemple a *robar* l'identificador d'una sessió d'usuari, per notificar-lo a l'atacant. Aquest rep la sessió i pot accedir a l'aplicació amb aquest identificador, podent efectuar accions en nom de l'usuari *víctima*.
- Atacs de *defacement*. Canviar l'aparença d'una pàgina, substituint el seu contingut (tant textos com imatges) amb finalitats alienes a les pròpies del lloc web, com ara reivindicacions de tot tipus o desacreditació de persones o institucions.
- Es poden donar també atacs de denegació de servei distribuïts (DDoS), utilitzant de forma fraudulenta per exemple crides a serveis que potencialment poden utilitzar gran quantitats de recursos en el servidor amb tècniques AJAX.
- Explotació de vulnerabilitats conegudes pròpies del *browser*.
- Cucs XSS, es podria propagar entre diferents llocs webs o diferents pàgines d'un únic lloc on hi ha una gran afluència d'usuaris i on la informació d'aquests és compartida de forma intensiva (per exemple en una aplicació de tipus xarxa social).

Com s'ha començat a l'inici d'aquesta secció, dintre d'aquests tipus d'atacs podem contemplar algunes variants:

- XSS emmagatzemat (corresponent a OWASP-DV-002)

  En aquest cas les dades en comptes de ser mostrades directament en la pàgina de destí (encara que també podria donar-se el cas), el codi maliciós és emmagatzemat en un suport persistent, normalment una taula d'una base de dades. Potencialment és el més perillós perquè aquest codi maliciós no només està circumscrit a un únic usuari en un moment donat, si no que pot ser executat des d'una gran quantitat d'usuaris. Un exemple seria la introducció d'un comentari en un fòrum. Aquesta informació normalment s'emmagatzema per a ser mostrada posteriorment en una pàgina accessible per molts usuaris. Si no es fa una inspecció acurada del contingut, l'atac s'estendrà a tots els usuaris que visualitzin aquesta informació.

- XSS basat en DOM (*Document Object Model*, corresponent a OWASP-DV-003)

  Aquesta variant utilitza el codi injectat maliciosament per manipular el DOM en el que estan basats els navegadors actuals. Un document HTML, que és el que normalment es renderitzen en un *browser*, té una estructura i un model d'objectes que el componen. Manipulant aquests objectes es pot manipular i obtenir informació del navegador, com ara valors de cookies (especialment les de sessió). També es pot canviar l'estructura del model, modificant així l'aparença o la funcionalitat de la pàgina.

- XSS *Flashing* (corresponent a OWASP-DV-004)

  Es tracta d'un atac XSS on l'element atacat és un arxiu flash (swf). El codi amb el que s'escriu flash, ActionScript, també ha de contemplar les consideracions de seguretat al igual que han de fer les aplicacions web. Al ser susceptibles de rebre paràmetres poden patir el mateix tipus d'atac. També és cert que l'explotació d'aquestes vulnerabilitats són més difícils de dur a terme atès que examinar el seu codi no és tan directe com en el cas de les aplicacions basades en HTML. Altres elements com ara *applets* Java o components *ocx*

podrien encabir-se dintre d'aquesta categoria.

Automatitzar la detecció aquestes vulnerabilitats de forma automàtica (caixa negra) es pot fer de forma relativament fàcil, tot i que no sempre es pot arribar al 100% de la casuística. Es difícil, però, la detecció en elements *aliens* al codi HTML com ara objectes *flash*, *ocx* o *applets* Java, atesa la seva naturalesa de component *compilat*. També és difícil detecció l'atac de XSS emmagatzemat. No sempre la informació introduïda apareix en la pantalla de resultat de la navegació *atacada*.

L'estratègia per a aquesta detecció consisteix bàsicament en la detecció de formularis en les pàgines de l'aplicació a verificar. Per a cada formulari, s'han d'inspeccionar cada element INPUT i fer diferents combinacions d'enviaments (*submit*) a la URL de destí amb valors amb codi maliciós pre-establert. Per a cada crida, s'ha de comprovar que el codi maliciós està inclòs sense modificació dintre de la pàgina de resultat. També pot existir la variant on la comprovació és *atesa* per una persona que verifica els resultats en el moment en què s'està produint la verificació.

Per evitar aquesta tipologia d'atac hi ha bàsicament dues estratègies:

- Una verificació de les dades d'entrada en una capa prèvia a l'aplicació, solucions tant hardware com software com ara un *firewall d'aplicació*.
- Una correcta aplicació de determinades normes de programació segura. Les normes més importants són aquelles relacionades amb la inspecció i validació de les dades d'entrada, tant els valors com els paràmetres. Com a tècniques es poden utilitzar:
  - Supressió o alteració de caràcters que permeten l'atac (`<, >, ', “, &, …`)
  - Creació de *llistes blanques*. Només permetre una determinada llista de paràmetres amb uns valors contrastats, verificats per alguna regla o expressió regular (per exemple, un número de compte bancari ha de tenir una longitud de 20 caràcters que, a més, han de ser dígits).
  - Una mescla de les dues anteriors.

Per a l'aplicació d'aquestes tècniques és desitjable utilitzar sistemes més o menys centralitzats i no deixar en mans de cada pàgina de l'aplicació aquesta tasca. Per exemple en el món JEE existeixen els *filtres* (interface `javax.servlet.Filter`) que permeten, juntament amb els *wrappers* dels objectes *Request* i *Response*, aquesta centralització.

Es pot assignar un o varis filtres a un patró de URL. Si hi ha més d'un filtre, s'executen en cadena. Totes les peticions que segueixin aquest patró, executaran prèviament aquest filtre. El filtre pot fer les comprovacions sobre les dades d'entrada. Per facilitar la tasca, pot utilitzar una *reescriptura* dels objectes *Request* i *Response* utilitzant extensions de les classes `javax.servlet.http.HttpServletRequestWrapper` i `javax.servlet.http.HttpServletResponseWrapper`. Amb la primera es pot sobreescriure especialment el comportament dels mètodes *getParameter* i *getParameterNames*. En aquests mètodes es poden fer les comprovacions necessàries sobre les dades d'entrada i modificar-les convenientment. Amb la segona es pot modificar la sortida generada per la pàgina original, fent una inspecció del codi i eliminant el codi no desitjat.

3.1.2 Atacs d'injecció de codi

Aquesta categoria engloba els següents punts de l'esmentat article de la *OWASP Testing Guide v3*:

- 4.8.5 *Injecció SQL* (OWASP-DV-005)
- 4.8.6 *Injecció LDAP* (OWASP-DV-006)
- 4.8.7 *Injecció ORM* (OWASP-DV-007)
- 4.8.8 *Injecció XML* (OWASP-DV-008)
- 4.8.9 *Injecció SSI* (OWASP-DV-009)
- 4.8.10 *Injecció XPath* (OWASP-DV-010)
- 4.8.11 *Injecció IMAP/SMTP* (OWASP-DV-011)
- 4.8.12 *Injecció de Codi* (OWASP-DV-012)

3.1.2.1 Injecció SQL[8][9][10]

La finalitat d'aquest atac és aprofitar l'existència d'una base de dades associada a l'aplicació web que pateix l'atac.

L'atac es pot fer tant utilitzant l'estàndard en el llenguatge SQL compartit abastament per la majoria de SGBD existents, com utilitzant característiques específiques de cadascun d'ells.

Un cop més, aprofitant la falta de validació de les dades d'entrada - utilitzant-les en la construcció de sentències SQL de forma dinàmica - , és possible dur a terme accions que permetrien:

- *Information disclorure*

  Obtenció d'informació tant de la base de dades com d'arxius externs. Aquesta obtenció es pot realitzar ja sigui directament com a resposta immediata de l'aplicació (*inband*) com mitjançant un altre canal a posteriori, com via mail (*out-of-band*).

  Un exemple podria ser aprofitar una query efectuada a partir d'un paràmetre rebut a la pàgina, com ara una llista de poblacions a partir d'una província. L'atacant pot pressuposar que la base de dades subjacent és Oracle. Aleshores, podria esbrinar per exemple les taules disponibles a la base de dades per futurs atacs.

  Suposem que existeix una taula POBLACIONS on hi ha, entre d'altres, una columna CODI_PROVINCIA i DESCRIPCIO. La query s'efectuaria a partir d'un paràmetre que rebria la pàgina anomenat *província* i podria ser quelcom similar a (codi Java):

  ```
  String sql = “SELECT DESCRIPCIO FROM POBLACIONS WHERE CODI_PROVINCIA='” + request.getParameter(“provincia”) + “' ORDER BY 1”;
  ```

  En enviar el següent valor en el paràmetre *provincia*:

  ```
  08' UNION SELECT TABLE_NAME FROM ALL_USER_TABLES --
  ```

  la sentència *SQL* quedaria:

  ```
  SELECT DESCRIPCIO FROM POBLACIONS WHERE CODI_PROVINCIA='08' UNION SELECT TABLE_NAME FROM ALL_USER_TABLES --'ORDER BY 1
  ```

  Això proporcionaria la llista de poblacions de la província 08, a més del catàleg de taules de la base de dades de l'aplicació. És important la finalització `--` que insereix un comentari i

deixa sense efecte l'últim apòstrof (i la ordenació de la sentència que no és rellevant en aquest cas).

- Alteració de dades

  Inserint, modificant o esborrant informació de la base de dades. Aquest atac també es pot donar per aconseguir una denegació de serveis distribuïts (DDoS).

- Suplantació de credencials

  Si la comprovació d'un usuari del sistema es fa directament contra la base de dades a partir del codi d'usuari i la seva clau d'accés, es pot suplantar aquest usuari manipulant la potencial *query*. Suposem que el sistema disposa una taula d'usuaris (`USUARIS`) amb, com a mínim, dues columnes (`CODI` i `PASSWORD`). Les dades d'entrada provindrien de 2 paràmetres (per exemple `user` i `pass`). Així doncs, si la comprovació de l'usuari es fa mitjançant la següent query (codi *Java*):

  ```
  String sql = "SELECT * FROM USUARIS WHERE CODI='" + request.getParameter("user") + "' AND PASSWORD = '" + request.getParameter("pass") + "'";
  ```

  Si al paràmetre pass s'envia la cadena adequada, és possible saltar-se aquesta validació. Una cadena candidata (per Oracle) podria ser:

  ```
  ' OR 1=1 --
  ```

  La sentència SQL quedaria doncs (suposem que passem com a codi d'usuari *xxxx*).

  ```
  SELECT * FROM USUARIS WHERE CODI='xxxx' AND PASSWORD='' OR 1=1 --'
  ```

  El primer apòstrof tancaria la expressió de comprovació de la columna PASSWORD, després una expressió lògica `OR 1=1` sempre resulta certa i, finalment, la cadena `--` comença un comentari que permet ignorar el que queda al final de la sentència i no doni un error sintàctic (que és l'apòstrof residual).

  Aquesta sentència ens permetria, doncs, obviar la validació del password.

L'atac d'injecció SQL, de forma paral·lela als atacs XSS, poden ser emmagatzemats, és a dir, que el codi maliciós acabi inserit en una taula de la base de dades i les repercussions poden donar-se de forma massiva i diferida.

La *OWASP Testing Guide v3* contempla tenir en compte una relació de SGBDs, el més utilitzats habitualment. Això és degut a que, tot i que com s'ha comentat tots suporten l'SQL estàndard, tenen característiques específiques per a cadascuna d'elles així com una sèrie de vulnerabilitats documentades i detectades segons la versió que poden ser explotades pels atacants. Aquests SGBD són:

- Oracle
- Mysql

- SQL Server
- MS Access
- PostgreSQL

La detecció automatitzada presenta més dificultat que en el cas del XSS. Tot i que conceptualment el procediment podria resultar similar, la determinació de l'obtenció del resultat final és més difícil. Si s'ha aconseguit suplantar un usuari o obtingut més informació o provocat un error s'ha de fer en base a comparació de diferents resultats (ja sigui com a text o com a resums MD5 o SHA-1) i cerca de paraules clau, fets que poden portar a molts falsos positius.

Atès que a priori és difícil saber quines dades d'entrada són potencialment insegures, una tècnica que pot ajudar a determinar si la injecció SQL s'està efectuant realment, seria enviar paràmetres els quals portin codi maliciós però que se sàpiga que no té per què afectar al resultat. En l'exemple de l'obtenció de poblacions seria una cadena com `' AND 1 = 1 --`, quedant així la sentència

```
SELECT DESCRIPCIO FROM POBLACIONS WHERE CODI_PROVINCIA='08' AND 1=1 --'
```

Aquesta sentència continua enviant el resultat esperat.

Aleshores, es procedeix a enviar una altra cadena amb codi maliciós que potencialment pugui alterar la sortida. Continuant amb l'exemple anterior, es pot afegir un codi que torni una informació totalment diferent o que pugui provocar un error. Si optem per la primera aproximació una cadena candidata seria `' OR 1 = 1 --`, quedant així la sentència

```
SELECT DESCRIPCIO FROM POBLACIONS WHERE CODI_PROVINCIA='08' OR 1=1 --'
```

Aquesta sentència respon amb un resultat totalment diferents a l'esperat (tornaria la relació de poblacions de la província 08 i les de la resta també).

Quant a prevenir aquests atacs, a part de tenir en consideració les mateixes recomanacions que en el cas del XSS, es d'utilització obligada, per a una protecció adequada en aquest sentit, la utilització de *stored procedures* (validant les dades d'entrada) i *prepared statements*. Amb *prepared statements* no es construeixen les sentències *SQL* de forma dinàmica en funció de la informació d'entrada rebuda, si no que la sentència ja està construïda i es procedeix a assignar valors a cadascuna de les variables de la sentència, no podent-se d'aquesta forma manipular la sentència. Sobre l'exemple anterior en la taula d'usuaris (codi *Java*):

```
String sql = “SELECT * FROM USUARIS WHERE CODI=? AND PASSWORD = ?”;
PreparedStatement pstmt = connection.prepareStatement(sql);
pstmt.setString(1, user);
pstmt.setString(2, pass);
```

En aquest cas, la sentència no es manipula. Si s'envien els valors especificats anteriorment, la query no retornarà res (SQLCODE 100) i donarà per no trobat l'usuari.

De totes formes, un cop més, és important recalcar la inspecció de les dades d'entrada.

3.1.2.2 Injecció LDAP

La vulnerabilitat d'injecció LDAP conceptualment és equivalent a l'injecció SQL i les consideracions fetes sobre l'anterior vulnerabilitat són vàlides per a aquesta. En comptes d'estar tractant amb una base de dades regida per l'estàndard SQL es tracta del protocol LDAP amb el seu llenguatge de consultes.

Cal dir que la informació continguda en un sistema LDAP pot ser molt sensible atès que acostumen a ser molt utilitzats per a la gestió d'usuaris, grups i rols.

Així doncs, un atac que exploti aquesta vulnerabilitat podria:

- Accedir a contingut no autoritzat
- Evitar restriccions de l'aplicació segons el rol de l'usuari
- Recollir informació no permesa
- Afegir, modificar i esborrar objectes del sistema LDAP

3.1.2.3 Injecció ORM

Es tracta del mateix tipus d'atac que la injecció SQL amb la diferència que la vulnerabilitat rau en el codi generat per una eina ORM (Object Relational Mapping). Aquestes eines faciliten la tasca al desenvolupament en quant a la part d'accés a una base de dades i la correspondència d'aquestes amb el model d'objectes de la pròpia aplicació. Per exemple, que en obtenir les dades d'un usuari concret d'una taula d'usuaris, aquestes puguin ser mapejades de forma declarativa contra una instància d'una classe (parlant en termes Java, per exemple) que representi aquesta entitat (mapeig de columnes de la taula contra propietats del JavaBean).

3.1.2.4 Injecció XML

Aquesta vulnerabilitat està basada en la injecció de tags (o part de tags) xml en una aplicació que pugui estar basada en aquest format de fitxers.

L'explotació d'aquesta vulnerabilitat passaria per 2 fases:

- Descoberta de l'XML subjacent a l'aplicació. Forçant potencials errors d'interpretació de l'XML resultant es podria arribar, depenent de com estigui desenvolupada l'aplicació, a averiguar l'estructura de l'XML. Això es pot intentar enviar en els paràmetre d'entrada a l'aplicació caràcters que puguin provocar la malformació del document XML resultant, si aquesta entrada no ha estat validada adequadament Aquests caràcters poden ser l'apòstrof o cometa simple, les cometes dobles, els delimitadors de les etiquetes (< i >), … Per exemple, si l'aplicació intenta generar un document XML a partir d'un paràmetre:

  ```
  String xml = “<document><nom-tag1>” +
  request.getParameter(“param1”) + “</nom-tag1><id>”+calculaId()
  +”</id><nom-tag2>” + request.getParameter(“param2”) + “</nom-
  tag2></document>”;
  ```

- En intentar processar el document resultant, depenent del valor dels paràmetre rebuts *param1* i *param2,* es pot provocar un error. Si s'enviés els valors *prova1* i *prova2*, l'XML resultant seria correcte (`calculaId` és un mètode que torna un identificador intern per l'aplicació):

```
<document><nom-tag1>prova1</nom-tag1><id>9999</id><nom-
tag2>prova2</nom-tag2></document>
```

Però si s'envia el valor *prova>* i *prova2*, aleshores, el processament (*parse*) del document provocarà un error:

```
<document><nom-tag1>prova></nom-tag1><id>9999</id><nom-
tag2>prova2</nom-tag2></document> →ERROR
```

- Injecció de tags. Un cop es te el coneixement de l'XML subjacent es poden enviar valors que puguin alterar el contingut de l'XML. Si per exemple un dels nodes s'assigna de forma automàtica per l'aplicació, es pot saltar aquesta dada calculada i injectar un valor desitjat mitjançant l'inclusió de parts de tags de tancament, comentaris XML i parts de tags d'inici. Tornant a l'exemple anterior si es vulgues injectar un valor al tag id diferent a l'assignat es podria enviar mitjançant els paràmetres *param1* i *param2* els valors `prova1</nom-tag><!--` i `--><id>1111</id><nom-tag2>prova2`, el resultat seria

  ```
  <document><nom-tag1>prova1</nom-tag><!--</nom-
  tag1><id>9999</id><nom-tag2>--><id>1111</id><nom-tag2>prova2</nom-
  tag2></document>
  ```

  D'aquesta forma, s'ha obviat el valor pel tag id (9999) i s'ha introduït un valor nou (1111).

Evitar un atac que exploti aquesta vulnerabilitat passa, un cop més, la validació de l'entrada, especialment pel que fa a caràcters que poden alterar el contingut d'un document XML en la seva estructura.

3.1.2.5 Injecció SSI

Les directives SSI són executades en el servidor per proporcionar contingut dinàmic a la pàgina. Aquestes directives, per tant, permeten executar codi que és potencialment perillós atès que poden proporcionar informació del sistema i, encara pitjor, arribar a executar codi en nom de l'usuari que executa el procés corresponent al Servidor Web.

La sintaxi de les directives SSI és

```
<!--#element attribute=value attribute=value ... -->
```

Per exemple, amb la directiva

```
<!--#echo var="DATE_LOCAL" -->
```

s'obté l'hora del servidor, i amb la directiva

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

es pot incloure un arxiu (que prèviament s'ha pogut injectar al sistema). I encara pitjor, la directiva

```
<!--#exec cmd="ls" -->
```

permetria executar una comanda i mostrar el contingut.

L'atac es pot produir de forma similar a l'XSS, aprofitant la no verificació de les dades d'entrada, així com també potencialment del contingut de capçaleres HTTP de la petició (com per exemple User-Agent). Si en uns dels paràmetres s'arriba a enviar un valor

```
<!--#echo var="DATE_LOCAL" -->
```

a la pàgina resultat s'ha de buscar un patró que representi la data actual. També es pot inspeccionar per l'aparició d'algun error (depenent de cada servidor), com el que pot retornar l'Apache Web Server:

```
[an error occurred while processing this directive]
```

Dur a terme un atac automatitzat d'aquesta naturalesa estaria en la línia també dels atacs XSS.

Finalment, per tal d'evitar aquests atacs, les tècniques a utilitzar són les mateixes que pels atacs XSS.

3.1.2.6 Injecció Xpath

Conceptualment aquest atac és paral·lel al d'injecció SQL, amb la diferència que en comptes d'estar tractant amb sentències SQL que actuen contra una base de dades, es tracta amb expressions Xpath per a la consulta de documents XML.

D'aquesta forma es podria, com en l'exemple de la injecció SQL, saltar-se l'autenticació d'un usuari o l'obtenció d'informació a la qual l'atacant no hauria d'accedir. Simplement, si les dades d'entrada no estan convenientment validades, es pot enviar codi maliciós que modifiqui l'expressió XPath. Un exemple seria enviar la cadena `' or '1' = '1`. Dintre d'una expressió XPath que verifiqués una condició, juntament amb aquest codi maliciós, l'avaluació sempre seria certa. Suposem l'expressió

```
String expr = “string(//user[username/text()='” + request.getParameter(“user”) + “' and password/text()='” + request.getParameter(“pass”) + “']/account/text())”;
```

en rebre tant el paràmetre *user* com *pass* la cadena esmentada, l'expressió XPath resultant seria

```
string(//user[username/text()='' or '1'='1' and password/text()='' or '1' = '1']/account/text())
```

Totes les consideracions tant des del punt de vista d'efectuar l'atac com des del punt de vista d'evitar-lo són les mateixes que per a l'atac d'injecció SQL.

3.1.2.7 Injecció IMAP/SMTP

En aquest atac l'aplicació *víctima* és típicament un sistema de WebMail que està suportat per un servei de correu IMAP/SMTP en servidors localitzats dintre de la xarxa interna de l'organització, no sent accessible des de l'exterior. L'únic punt de contacte d'aquest servei amb l'exterior és l'esmentada aplicació WebMail.

A partir dels paràmetres d'entrada que espera l'aplicació, s'executen comandes IMAP/SMTP. Un cop més, si aquesta informació d'entrada no és convenientment tractada, es corre el perill de patir un atac d'injecció de comandes IMAP/SMTP no desitjades.

L'atac automatitzat passa per les següents fases:

- Identificació dels paràmetres potencialment vulnerables i deducció de la utilitat de cadascun d'ells

  Comprovació de la resposta de la pàgina de resultat en funció de la manipulació de cadascun dels paràmetres que rep. La modificació d'un valor amb caràcters addicionals pot donar-nos pistes sobre la existència o no d'un ítem determinat (per exemple el que podria nom d'una

bústia) .

Si l'aplicació dóna com a resposta directament missatges del servei IMAP/SMTP es poden fer deduccions sobre la funcionalitat de cada paràmetre. Per exemple si hi ha un paràmetre que envia un caràcter no permès en una comanda IMAP/SMTP es podria rebre un error que podria donar una pista sobre la naturalesa del paràmetre.

- Injecció de comandes IMAP/SMTP

  Un cop determinats els paràmetres i la seva naturalesa, es pot procedir a la injecció de comandes IMAP/SMTP. Dependrà de la configuració del servidor l'execució de comandes en mode autenticat o no.

  L'estructura bàsica d'una injecció de codi IMAP/SMTP és la següent

  - Capçalera: finalització de la comanda esperada
  - Cos: Inclusió del codi injectat
  - Peu: inici de la comanda esperada

  Si per exemple s'arriba a la conclusió que hi ha una comanda la qual depèn directament d'un paràmetre (amb valor NNN):

```
FETCH NNN BODY[HEADER]
```

  aleshores substituint aquest valor per una altra seqüència, s'obtindrà una sèrie de comandes IMAP/SMTP correctes amb el codi injectat, per exemple amb

```
NNN BODY[HEADER]%0d%0aV100 CAPABILITY%0d%0aV101 FETCH NNN
```

  la seqüència de comandes resultant serà

```
??? FETCH NNN BODY[HEADER]
V100 CAPABILITY
V101 FETCH NNN BODY[HEADER]
```

Evitar aquest tipus d'atac passa bàsicament per 2 accions:

- A l'igual que en d'altre tipus d'atac (XSS, Injecció SQL) verificar i validar els paràmetres d'entrada.
- Evitar donar cap mena d'informació sobre els sistemes que donen suport al servei. Qualsevol error hauria d'expressar-se com un error genèric i no mostrar les característiques del mateix.

3.1.2.8 Injecció de Codi

Aquest atac es dóna quan una aplicació pot rebre paràmetres els quals provoquen una execució en la màquina servidora, ja sigui una comanda directe, ja sigui un arxiu que és capaç d'interpretar el servidor (una pàgina php o asp).

En el primer dels casos, evitar aquest problema passa per al filtrat de les comandes permeses. En el segon cas, per una banda, no deixar a l'aplicació arxius que poden venir a la instal·lació per defecte i que no estigui tampoc permesa la pujada directa d'arxius executables per aquestes plataformes i que siguin executats directament sense cap validació prèvia Per exemple Drupal, una aplicació PHP que, entre d'altres coses, permetia pujar imatges. Aquestes quedaven en un directori temporal, però accessibles des del navegador. L'aplicació no feia comprovacions sobre el contingut de les imatges, només si aquestes tenien alguna extensió sobre el tipus acceptat (png, gif, …). Es podia pujar un arxiu amb extensió png però que on el contingut era PHP, que és lo que interpreta el servidor que allotja l'aplicació. Aquest PHP podia contenir el codi necessari per poder executar, en nom de l'usuari que executa el procés del servidor web, qualsevol comanda del sistema operatiu.

3.1.3 Injecció de comandes

Amb aquest atac és possible enviar directament des d'una crida a una URL comandes del sistema operatiu.

Aplicacions escrites en Perl estan potencialment afectades. Aquest llenguatge permet passar informació mitjançant un *pipe* procedent de la execució d'una comanda. Així doncs, aplicacions web que reben com a paràmetre el nom d'un arxiu a veure directament, podrien potencialment ser víctimes d'aquests atacs. Suposem una aplicació escrita en Perl que rep com a paràmetre el nom d'un fitxer a visualitzar:

```
http://servidor/cgi-bin/pagina.pl?fitxer=document.txt
```

Si el que fa aquest script Perl és visualitzar el contingut del fitxer que es rep per paràmetre, aquest pot ser substituït per una comanda vàlida del sistema operatiu finalitzada amb un símbol *pipe* (|).

```
http://servidor/cgi-bin/pagina.pl?fitxer=ls%20-l|
```

Això trauria la llista del directori per defecte de l'usuari que executa el procés del servidor web o de l'intèrpret de Perl.

Realitzar un atac automatitzat en aquest sentit és força fàcil, donant com a valor als paràmetres comandes amb aquest format, per exemple, i comprovant els resultats esperats en la pàgina de resultat.

La solució al problema passa per verificar els paràmetres d'entrada, al igual que en XSS.

3.1.4 Buffer overflow

El no control d'aquesta vulnerabilitat pot portar a atacs d'execució de codi arbitrari (*shellcodes*) i inclús denegacions de servei.

Aquesta vulnerabilitat es dóna sobretot en codi escrit en llenguatges on no hi ha una protecció implícita en les posicions de memòria assignades a les variables, com per exemple el llenguatge C si en la compilació així no s'especifica. També els programes escrits en llenguatge Perl es veurien potencialment afectats per aquesta vulnerabilitat. Hi ha moltes aplicacions Web basades en CGIs escrites en llenguatge C o Perl. Aplicacions basades en llenguatges més *segurs* a aquest nivell, com ara Java (especialment si no es produeixen crides natives) no estarien afectades, a priori, per aquesta vulnerabilitat.

D'aquesta vulnerabilitat en trobem principalment tres tipus:

- Heap overflow

  El *heap* és un àrea de memòria que emmagatzema variables definides de forma dinàmica i variables globals. Si una variable que pot rebre el valor mitjançant l'aplicació Web està definida amb una mida determinada, enviat informació amb mida superior a aquesta es poden sobreescriure adreces de memòria (*access violation*) i substituir el seu contingut amb codi maliciós.

- Stack overflow

  Es tracta d'una vulnerabilitat similar a l'anterior, però desbordant la pila en comptes del *heap*. En la pila s'emmagatzema informació temporal i és utilitzada pel mecanisme de crides a procediments (pas de paràmetres, guardar-se l'adreça de retorn), així com normalment l'emmagatzemament de les variables locals a un procediment. Informar amb més informació de la prevista un paràmetre o una variable global és la forma d'explotació més freqüent d'aquesta vulnerabilitat.

- Format string

  Aquesta vulnerabilitat afecta a aplicacions escrites en C/C++ on hi ha funcions com printf, fprintf, … que admeten un primer paràmetre opcional amb el format amb el què representar el segon paràmetre utilitzant una sèrie de patrons (`%d, %s, …`). Si es crida alguna d'aquestes funcions sense el primer paràmetre opcional i l'únic paràmetre passat porta determinades cadenes que es podrien *confondre* amb patrons vàlids (especialment `%x, %p, %n`) es poden explotar les següents vulnerabilitats:

  Descobriment de l'estructura de la pila. Utilitzant els patrons `%p` i `%x` és possible conèixer l'organització i contingut de la pila.

  Control del flux de l'execució. Utilitzant el patró `%n` es pot arribar a escriure 4 bytes de dades en una adreça especificada per l'atacant. Aquests 4 bytes poden correspondre a una adreça al codi maliciós desitjat.

  Denegació de servei. Enviant una seqüència `%n%p` es pot fer fallar l'aplicació.

En general, eines automàtiques (*blackbox*) de detecció d'aquestes vulnerabilitats són força difícils d'implementar.

Un cop més, evitar atacs que s'aprofitin d'aquestes vulnerabilitats passa per la verificació exhaustiva

de les dades d'entrada, tant en el seu contingut com en la seva mida màxima permesa.

3.1.5 Vulnerabilitat incubada

Sobre aquesta vulnerabilitat ja s'ha parlat de forma indirecta en punts anteriors. Es tracta d'una vulnerabilitat que no és explotada de forma directa, si no que el seu ús es realitza a posteriori. Bàsicament es tracta de:

- Submissió d'arxius amb codi maliciós que es pot executar a posteriori. En el capítol sobre *Injecció de codi* es fa esment a l'exemple del Drupal.
- XSS emmagatzemat. La possibilitat que un codi maliciós quedi emmagatzemat en, per exemple, una base de dades i sigui explotat més tard. En el capítol sobre XSS es comenta aquest tema (un fòrum on s'introdueix un comentari que porta codi XSS incrustat i que després en ser consultat per diferents usuaris, aquests queden afectats).
- SQL emmagatzemant, també comentat en el capítol d'injecció SQL.

3.1.6 HTTP splitting/smuggling

3.1.6.1 HTTP splitting

L'HTTP splitting permet aprofitar una vulnerabilitat del protocol HTTP en la que es poden manipular les capçaleres del protocol (response headers) afegint els caràcters CR (0x0D, retorn de carro) i LF (0x0A, retorn de línia). Si l'aplicació rep com a paràmetre informació que pot ser inclosa a la capçalera, aleshores pot ser víctima potencial d'aquest atac si les dades d'entrada no son tractades adequadament. Per exemple, considerem una pàgina dintre d'una aplicació que efectua una redirecció a una altra URL:

```
<?php
header ("Location: " . $_GET['page']);
?>
```

Si es fa una petició a aquesta pàgina passant una URL en el paràmetre page, aleshores s'efectuarà una redirecció cap a la URL especificada

http://servidor/redireccio.php?page=http://www.uoc.edu

Això redireccionaria cap a la pàgina principal de la UOC. Les capçaleres de la resposta a la petició serien les següents:

```
http://localhost/test.php?page=http://www.uoc.edu

GET /test.php?page=http://www.uoc.edu HTTP/1.1
Host: localhost
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:16.0) Gecko/20100101
Firefox/16.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive

HTTP/1.1 302 Found
Date: Sun, 28 Oct 2012 19:10:26 GMT
Server: Apache/2.2.22 (Win64) PHP/5.3.13
X-Powered-By: PHP/5.3.13
Location: http://www.uoc.edu
Content-Length: 1
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
X-Pad: avoid browser bug
```

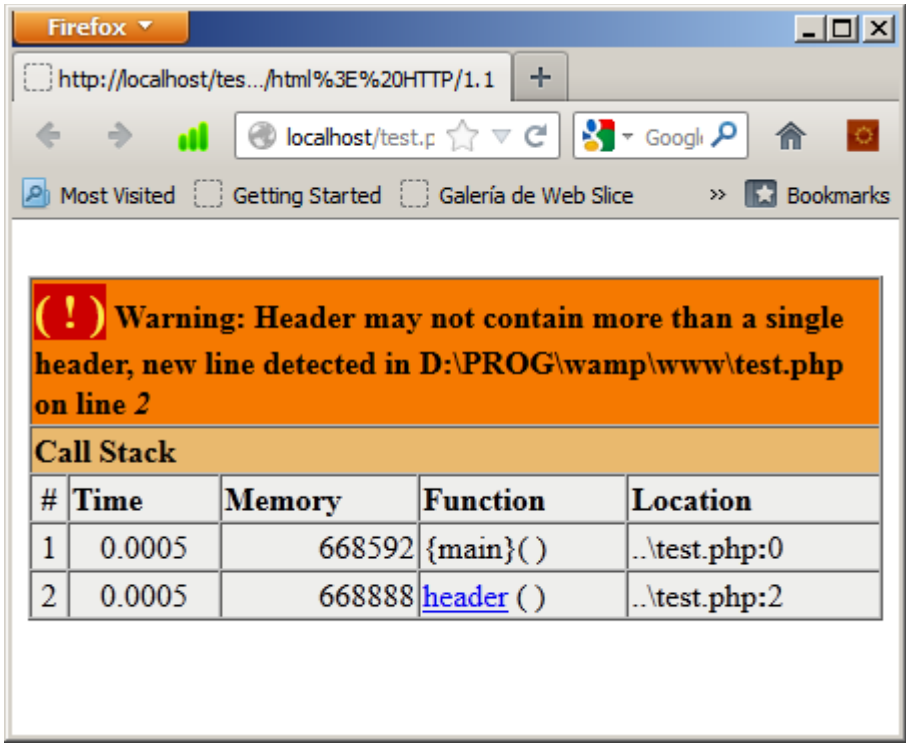
Si es canvia la crida donant al paràmetre page el valor

```
%0d%0aContent-Type: text/html%0d%0a%0d%0aHTTP/1.1 200 OK%0d%0aLast-
Modified: Wed, 28 Oct 2012 20:13:17 GMT%0d%0aContent-Type: text/html%0d
%0a%0d%0a<html><font color=red>ei</font></html> HTTP/1.1
```

El que s'obtindria seria una pàgina HTML que no té res a veure amb la UOC.

La solució al problema passa un cop més en verificar les dades d'entrada, observant molt especialment en aquest cas l'aparició de caràcters `0x0d` i `0x0a`.

Aquest problema també es soluciona amb una correcta configuració del Web Server i dels elements que puguin haver al darrera del mateix:



Un atac automatitzat per a aquesta vulnerabilitat es podria dissenyar de forma equivalent a la que es pot fer amb l'XSS.

3.1.6.2 HTTP smuggling

Aquesta vulnerabilitat consisteix en que es poden llençar diferents peticions degudament preparades (manipulant les capçaleres de la petició i en un ordre determinat) de forma que diferents elements pels que passen les peticions (és a dir dispositius HTTP com firewalls d'aplicació, proxies, servidors, etc) poden rebre individualment diferents tipus de petició permetent així saltar-se les validacions que poguessin efectuar aquests elements i deixant que els atacants introdueixin peticions camuflades en un element sense que un altre s'assabentés. Aquest tipus d'atac poden arribar a introducció de cross site scripting i execució de comandes del sistema operatiu.

Un dels casos més rellevants va ser el descobriment d'aquesta vulnerabilitat l'any 2005 en el servidor web Apache en versions anteriors a la 2.1.6, especialment tenint en compte que es tractava en aquell moment del servidor tenia una quota de mercat del 68% a nivell mundial.

La solució a aquest problema passa per una configuració correcta i actualitzada de tots els elements que poden intervenir en les peticions a l'aplicació final.

3.2 Eines de detecció de vulnerabilitats

El tema principal d'aquest apartat són les eines de detecció de vulnerabilitats en aplicacions web que treballen de forma més o menys automatitzada, conegudes com a *Black Box Scanners*. Particularment estarà centrat en les vulnerabilitats descrites en l'apartat anterior. N'hi ha de comercials i de programari lliure.

Aquestes eines, per norma general, no poden accedir al codi font de les aplicacions destí i han d'aplicar una sèrie de tècniques per simular els potencials atacs que poden patir aquestes.

Un dels factors amb el que aquestes eines demostren la seva fiabilitat és no només la capacitat de descobrir vulnerabilitats en una aplicació, si no també l'habilitat de poder diferenciar els coneguts falsos positius de les vulnerabilitats reals.

3.2.1 Especificacions funcionals

Amb el títol *Software Assurance Tools: Web Application Scanner Functional Specification 1.0*[3] el projecte *SAMATE* (*Software Assurance Metrics and Tools Evaluation*) del NIST (*National Institute of Standards and Technology*, una agència federal depenent del Departament de Comerç dels EUA) estableix l'especificació dels requeriments funcionals mínims que tota eina d'aquestes característiques ha de complir.

Aquests requeriments obligatoris són els següents:

- Contemplar, com a mínim, les següents vulnerabilitats (es pot observar que bona part d'elles són les tractades en aquest document en l'apartat 2):
  - XSS
  - SQL Injection
  - OS Command Injection
  - XML Injection
  - HTTP Response Splitting
  - Malicious File Inclusion
  - Insecure Direct Object Reference
  - Cross Site Request Forgery (*CSRF*)
  - Information Leakage
  - Improper Error Handling
  - Weak Authentication and Session Management
  - Session Fixation
  - Insecure Communication
  - Unrestricted URL Access
- Informar un atac que demostra la vulnerabilitat
- Especificar com s'ha produït aquest atac proporcionant la informació necessària (dades

enviades a l'aplicació i resultats obtinguts)

- Identificar la vulnerabilitat amb un nom reconegut a nivell global. Per exemple *Cross Site Scriptring* és un terme àmpliament conegut però sempre és millor referir-se a les vulnerabilitats amb nomenclatures conegudes internacionalment com *OWASP Top Ten 2010* (*A2*) o *CWE ID* (79).
- Poder identificar-se a l'aplicació destí (si s'escau) i ser capaç de mantenir la sessió durant la tasca de verificació.
- Que tingui un nivell baix de falsos positius.

Com a requeriments opcionals trobem:

- Trobar vulnerabilitats fins i tot dintre de la implementació dels següents mecanismes de defensa:

Mecanisme	***Descripció***	***Exemple***	***Vulnerabilitats afectades***
Typecasting	Conversió de la dada d'entrada en un tipus concret (numèric, booleà, ...)	`(int)($_GET['var'])` transforma `"8<script>"` al número sencer 8	XSS, Injeccions de codi, manipulació de *cookies*.
Reemplaçament de meta caràcters	Codificació de caràcters d'una *llista negra*.	Substitució en les dades d'entrada d'una sèrie de caràcters com <,>,',",... per les seves entitats HTML corresponents.	Totes les vulnerabilitats descrites en aquest document.
Restricció del rang d'entrada	Control del rang de les dades d'entrada. Interval de números, longituds de cadenes, restricció de tipus...	Utilitzant la classe `StringUtils` de Apache Commons, es limita a 4 la longitud del paràmetre `var`: `String cadenaEntrada = StringUtils.substring(request.getParameter("var"), 0, 4);`	Totes les vulnerabilitats descrites en aquest document.
Gestió d'usuaris restringida	Utilització d'usuaris específics per l'aplicació, amb els permisos justos i necessaris només per al funcionament d'aquesta. Bona política d'assignació	Definir grups específics als usuaris de la base de dades de l'aplicació només per lectura en les taules on sigui possible i restringir els accessos de	Injecció SQL, execució de comandes del SO.

Eines de detecció de vulnerabilitats en aplicacions Web

Mecanisme	***Descripció***	***Exemple***	***Vulnerabilitats afectades***
	de permisos .	modificació/inserció/esborrat només en les taules estrictament necessàries.	
Utilització de funcions criptogràfiques més segures	Per potenciar la seguretat de les aplicacions, utilitzar les versions més segures de les funcions criptogràfiques que es poden utilitzar en l'aplicació.	Utilització de l'algorisme SHA-256 en comptes del SHA-1 o MD5, atès que el primer és molt més segur. Afegir *salt* als orígens de transformacions *HASH* per dificultar l'ús de diccionaris de claus *HASH*.	Funcions criptogràfiques dèbils, *cookies* insegures.
Validació de token	Afegir un *token* únic calculat per verificar i validar l'orígen de les dades rebudes.	Afegir un identificador únic calculat en funció de les dades d'entrada. Enviar-lo com una dada més del formulari i verificar en el servidor la integritat de les dades en base a aquest *token*. Relacionat amb el punt anterior.	CSRF
Gestió de la codificació de caràcters	Tractament adequat de la forma en què estan codificades les dades d'entrada.	Simples atacs utilitzant determinades codificacions: XSS: La cadena codificada en UTF-7 : “`+ADw-script+AD4-alert('XSS')+ADsAPA-/script+AD4-`“ es reinterpretada com “`<script>alert(‘XSS’)</script>`”	XSS, Injeccions de codi, manipulació de *cookies*.
Ocultació	Ocultar informació		Qualsevol

Mecanisme	*Descripció*	*Exemple*	*Vulnerabilitats afectades*
d'informació sobre el sistema	sobre errors, identificadors de sessió, ...		vulnerabilitat que pugui donar una sortida informativa.

- No identificar vulnerabilitats que ja no són contemplades.
- Fer propostes de com evitar l'atac.
- Oferir els informes de resultat en format XML.
- Assignar un *ranking* d'importància a cada vulnerabilitat identificada.

3.2.2 Criteris d'avaluació

Amb la finalitat de poder avaluar diferents eines de detecció de vulnerabilitats, el *Web Application Security Consortium* (*WASC*) ha publicat un document-guia per ajudar en aquesta tasca, No amb un objectiu de certificar que es recullen els requeriments descrits en l'apartat anterior, si no amb la finalitat de trobar aquella eina que més s'ajusti a les necessitats de cada cas.

Per facilitar la tasca, es facilita un formulari implementat en un full de càlcul.

Aquest document, conegut com a *Web Application Security Scanner Evaluation Criteria* (*WASSEC*) recull una sèrie de punts d'avaluació d'aquestes eines:

3.2.2.1 Protocols suportats

Totes aquestes eines sempre treballen amb protocol HTTP, però han de suportar tots els protocols de comunicació relacionats:

- A nivell de transport:
  - HTTP 1.1. Utilitzat en la majoria de navegadors actuals.
  - HTTP 1.0. Encara utilitzat especialment en altres elements intermitjos com servidors proxy.
  - SSL/TLS. Les aplicacions web que intercanvien amb el client informació sensible ha d'anar encriptada.
  - HTTP Keep Alive. Amb la finalitat de millorar el rendiment, la capacitat de mantenir connexions HTTP obertes amb el client
  - Compressió HTTP. Alguns servidors web, per raons d'eficiència en la utilització de la xarxa, retornen la informació comprimida.
  - Configuració i personalització de la capçalera *User-Agent*. Capacitat de simular diferents navegadors atès que hi ha aplicacions que poden mostrar informació diferent depenent del navegador amb el què es treballa. Per exemple una aplicació pot retornar un contingut diferent depenent si el dispositiu és un ordinador o bé un mòbil o tablet.

- A nivell de proxy:
  - HTTP 1.0 Proxy
  - HTTP 1.1 Proxy
  - Socks 4 Proxy
  - Socks 5 Proxy
  - Suport a fitxers PAC (*proxy auto-config*)

3.2.2.2 Autenticació

Les eines han de donar suport als diferents mètodes d'autenticació més estàndards per poder fer el corresponent tests a les parts de les aplicacions que necessiten una autenticació prèvia:

- Basic
- Digest
- HTTP Negotiate (NTLM, Kerberos)
- Basat en formularis HTML:
  - Automatitzat
  - Mode script
  - No automatitzat (requereix intervenció humana en el moment de l'autenticació, com per exemple la introducció d'un CAPTCHA).
- Single Sign On
- Certificats client SSL
- Implementacions pròpies de l'aplicació

3.2.2.3 Gestió de sessions

Moltes aplicacions Web necessiten mantenir una sessió amb el servidor. En el cas de les eines que ens ocupen, és necessari tant en la fase de *crawling* com de testeig (*testing*). Per mantenir una sessió vàlida, les eines han de donar suport a:

- Capacitats de gestió de sessions
- Suport a diferents mètodes de mantenir el *token* que relaciona la sessió amb el navegador:
  - HTTP Cookies
  - Paràmetres HTTP
  - HTTP URL path
- Configuració de detecció del *token* de sessió
  - Automàtica
  - Manual

- Política d'actualització del *token* de sessió
  - Valor fix
  - Proporcionat en el moment d'autenticació (*login*)
  - Dinàmic

3.2.2.4 Crawling

Una part molt important d'aquestes eines consisteix en l'habilitat de navegar per tota l'aplicació a partir d'un punt inicial *(web crawling)*. S'inspecciona la pàgina resultant d'aquest punt inicial i es verifica l'existència de diferents *links* per continuar la navegació cap a ells, tot de forma recursiva. S'han de tenir en compte els següents aspectes:

- Configuració del procés de *web crawling*
  - Establiment del punt d'inici
  - Definir el(s) domini(s) o IP(s) sobre les que s'ha d'efectuar el test
  - Poder definir exclusions
    - A servidors o IPs determinades
    - URLs o patrons de URL (amb expressions regulars).
    - Algunes extensions de fitxer (podria no tenir sentit verificar les vulnerabilitats a nivell d'aplicació Web d'un arxiu .jpg, per exemple).
    - Sobre determinats paràmetres
  - Capacitat de poder evitar o limitar crides redundants
  - Suport a sessions concurrents
  - Poder especificar un temps d'espera (*delay*) entre peticions
  - Poder definir una profunditat màxima dintre de la cerca
  - Poder definir una navegació manual (*script*).
- Funcionalitat del procés de *web crawling*
  - Identificació de diferents *hostnames* dintre del procés a fi i efecte d'incloure'ls o excloure'ls del procés.
  - Suport a l'execució automatitzada de *submits* de formularis
  - Detecció de pàgines d'error i gestió de respostes HTTP tals com 404 (*not found*).
  - Suport a redireccions. Això inclou:
    - Seguiment de redireccions HTTP (Exemple: *HTTP status codes* 301, 302, 303, 307)
    - Seguiment de redireccions *Meta Refresh* (Exemple: `<meta http-equiv="refresh" content="1; URL=http://www.uoc.edu/">`)
    - Seguiment de redireccions *javascript* (Exemple: `<script type="text/javascript">window.location = "http://www.uoc.edu";</script>`)

- Identificar i acceptar *cookies*.
- Suport a peticions AJAX

3.2.2.5 Anàlisi (*Parsing*)

La fase de *parsing* de l'eina ha de contemplar els següents aspectes:

- Tipus de contingut. L'eina ha de ser capaç d'interpretar els següents tipus de contingut:
  - HTML
  - Javascript
  - VBScript
  - XML
  - Determinats fitxers plans
  - Objectes ActiveX
  - Java Applets
  - Flash
  - CSS
- Suport a diferents codificacions:
  - ISO-8859-1
  - UTF-7
  - UTF-8
  - UTF-16
- Tolerància. No ha de ser molt restrictiu, tenint en compte que, per exemple, l'HTML desenvolupat no és formalment correcte en moltes ocasions. Ha de tenir l'habilitat de poder interpretar un HTML de forma laxa.
- Personalització. Capacitat de poder personalitzar alguns del punts anteriorment descrits.

3.2.2.6 Testing

Les capacitat de test que han de suportar aquestes eines són bàsicament les extretes de la classificació d'amenaces versió 2.0 de la WASC. Les que ocupen aquest document són les següents:

- Client-side Attacks
  - Content Spoofing
  - Cross-Site Scripting
    - Reflected Cross-Site Scripting
    - Persistent Cross-Site Scripting
    - DOM-Based Cross-Site Scripting

    - Cross-Frame Scripting
    - HTML Injection
    - Cross-Site Request Forgery
    - Flash-Related Attacks
        - Cross-Site Flashing
        - Cross-Site Scripting Through Flash
        - Phishing/URL Redirection Through Flash
        - Open Cross-Domain Policy
- Command Execution
    - Format String Attack
    - LDAP Injection
    - OS Command Injection
    - SQL Injection
        - Blind SQL Injection
    - SSI Injection
    - XPath Injection
    - HTTP Header Injection / Response Splitting
    - Remote File Includes
    - Local File Includes
    - Potential Malicious File Uploads

També s'ha de permetre la configuració del test a efectuar, ja sigui quines vulnerabilitats es volen tenir en compte, afegir la informació necessària en funció de l'aplicació o bé especificar una sèrie de paràmetres que influiran en la seva execució de forma similar que es fa amb el *web crawler*:

- Quins servidors o IPs s'han de tenir en compte o no
- Quins patrons d'URL s'han de contemplar
- Quines extensions de fitxer es tindran en compte
- Suport a *cookies*
- Modificació / Establiment de capçaleres HTTP.

3.2.2.7 Interfícies de comandes, control i configuració

En aquest punt s'avaluen diferents capacitats de l'eina en quant a configuració, control i interfícies:

- Capacitats

- Permetre planificar les execucions, definint un temps inicial i podent especificar un temps màxim d'execució.
- Possibilitat de pausar i reiniciar
- Possibilitat de reaprofitar parametritzacions d'execucions anteriors per poder-les repetir, adaptar, ...
- Suport a múltiples usuaris
- Suport a execucions concurrents

- Interfícies

  Es poden contemplar les següents interfícies:

  - Aplicació *stand-alone*, instal·lada en una màquina client amb un GUI
  - La mateixa que l'anterior, però amb intèrpret de comandes
  - Interface web

- Eines d'extensió i d'interoperabilitat. S'ha de proporcionar una sèrie d'APIs o WebServices per poder integrar l'eina amb d'altres d'externes per explotar la informació que poden produir o bé definir GUIs desenvolupades *adhoc*.

3.2.2.8 Producció d'informes

El producte final de l'eina ha de ser un informe amb el resultat de les proves efectuades. S'han de tenir en compte els següents aspectes:

- Tipus d'informe
  - Resum executiu
  - Informe tècnic detallat
  - Informe delta (comparació de diferents execucions)
  - Informe de compliment (el nivell de protecció de l'aplicació, vulnerabilitats detectades, *scoring* final...). Aquest informe ha d'estar lligat a unes recomanacions o certificacions preestablertes per diferents entitats/organismes, com ara:
    - OWASP Top 10
    - WASC Threat Classification
    - SANS Top 20
    - Sarbanes-Oxley (SOX)
    - Payment Card Industry Data Security Standard (PCI DSS)
    - Health Insurance Portability and Accountability Act (HIPAA)
    - Gramm-Leach-Bliley Act (GLBA)
    - NIST 800-53

  - Federal Information Security Management Act (FISMA)
  - Personal Information Protection and Electronic Documents Act (PIPEDA)
  - Basel II
- Assessorament per a cada tipus de vulnerabilitat trobat. Donar recomanacions per a cada vulnerabilitat que s'hagi pogut detectar. Aquestes recomanacions haurien d'incloure:
  - Descripció de la vulnerabilitat
  - Identificador CVE o CWE
  - Nivell de seguretat
  - Classificació segons la puntuació establerta per la versió 2 de CVSS
  - Guia de correcció
  - Exemples de correcció
- Possibilitat de personalització dels informes
- Formats de sortida. Haurien de contemplar-se:
  - PDF
  - HTML
  - XML

3.2.3 Efectivitat de les eines

No es pot parlar de l'efectivitat en general d'aquestes eines atès el gran número d'aquestes que existeixen, tant comercials com proveïdors *Software as a Service*, així com *open source*. Només amb la llista proporcionada pel *Web Application Security Consortium* es poden trobar els següents:

Aplicacions comercials

- Acunetix WVS by Acunetix
- AppScan by IBM
- Burp Suite Professional by PortSwigger
- Hailstorm by Cenzic
- N-Stalker by N-Stalker
- Nessus by Tenable Network Security
- NetSparker by Mavituna Security
- NeXpose by Rapid7
- NTOSpider by NTObjectives

Eines de detecció de vulnerabilitats en aplicacions Web

- ParosPro by MileSCAN Technologies
- Retina Web Security Scanner by eEye Digital Security
- WebApp360 by nCircle
- WebInspect by HP
- WebKing by Parasoft
- Websecurify by GNUCITIZEN

Proveïdors de Software-as-a-Service

- AppScan OnDemand by IBM
- ClickToSecure by Cenzic
- QualysGuard Web Application Scanning by Qualys
- Sentinel by WhiteHat
- Veracode Web Application Security by Veracode
- VUPEN Web Application Security Scanner by VUPEN Security
- WebInspect by HP
- WebScanService by Elanize KG

Eines Open Source

- Arachni by Tasos Laskos
- Grabber by Romain Gaucher
- Grendel-Scan by David Byrne and Eric Duprey
- Paros by Chinotec
- Andiparos
- Zed Attack Proxy
- Powerfuzzer by Marcin Kozlowski
- SecurityQA Toolbar by iSEC Partners
- Skipfish by Michal Zalewski
- W3AF by Andres Riancho
- Wapiti by Nicolas Surribas
- Watcher by Casaba Security
- WATOBO by siberas
- Websecurify by GNUCITIZEN
- Zero Day Scan

A part de la guia d'avaluació descrita en el punt alterior, els factors a tenir en compte per determinar l'efectivitat d'aquestes eines són els següents:

- *Web Crawler* / *Parser*

  És important que el *Web Crawler*, juntament amb el *Parser*, sigui capaç de determinar el màxim possible de links dintre de cadascuna de les URLs de l'aplicació. De l'habilitat d'aquesta peça podrà dependre molt la ràtio de detecció de vulnerabilitats. Les dificultats en les que es troben aquestes eines en aquest aspecte és en els objectes que formen part de les pàgines resultants però que no són expressades en un format definit (bàsicament HTML). En una pàgina HTML és molt fàcil determinar els diferents *links*. Per a aconseguir això de forma òptima, el *parser* ha de ser força laxe en quant a la qualitat del codi HTML (no sempre es codifica al 100% seguint l'especificació). Tot i així, el repte rau en aquests altres objectes que no formen part del propi codi de la pàgina o bé que no es pot determinar de forma directa. Per exemple:

  - Objectes OCX
  - Flash
  - SilverLight
  - Applets Java
  - Crides a funcions javascript / vbscript contingudes en llibreries externes, d'on és difícil determinar pel codi si alguna acció provoca una petició a una altra URL.

En el següent quadre, extret de l'estudi *A Scale for Crawler Effectiveness on the Client-Side Hidden Web* del Departament de Tecnologies de la Informació i Comunicació de la Universitat de La Corunya[11], es pot veure una classificació dels diferents tipus d'elements que poden incorporar un enllaç i la complexitat associada a descobrir-los:

Nivell	Descripció	Complexitat	Procés d'extracció
1	Textlink	Molt baixa	Mòdul de *crawling* – Extractor d'URLs
2	JavaScript/Document.Write/Menu – Static String – Embedded JavaScript – Concatenated String - Embedded	Baixa	Mòdul de *crawling* – Intèrpret JavaScript – Analitzador de redireccions – Extractor d'URLs
3	HTML/onBody/JavaScript Redirect Javascript # - Static String - Embedded	Baixa	Mòdul de *crawling* – Intèrpret JavaScript – Extractor d'URLs
4	VBScript – Static String – Embedded	Mitjana	Mòdul de *crawling* –

Eines de detecció de vulnerabilitats en aplicacions Web

Nivell	Descripció	Complexitat	Procés d'extracció
	VBScript – Special Funcion - Embedded		Intèrpret VBS – Extractor d'URLs
5	JavaScript/Document.Write – Static String – External/Embedded Document.Write/Menu – Static String – External Menu – Concatenated String – Embedded	Mitjana / Alta	Mòdul de *crawling* – Intèrpret VBS – Extractor d'URLs
6	JavaScript/Document.Write/Menu – Concatenated String-External Applet – Static String in HTML	Mitjana / Alta	Mòdul de *crawling* – Intèrpret JavaScript avançat – Extractor d'URLs
7	JavaScript - Concatenated String – External/Embedded – Relative JavaScript – Special Function – External/Embedded – Relative Document.Write – Concatenated String – External/Embedded Document.Write – Special Function – External/Embedded Menu - Concatenated String – External/Embedded – Relative Menu - Special Function – External/Embedded Link in .java AJAX Link – Absolute	Alta	Mòdul de *crawling* – Intèrpret JavaScript avançat – Extractor d'URLs avançat Descompilador Java Analitzador de fitxers externs
8	Link in .class Applet - Static String in .class Flash - Static String in HTML/SWF Applet/Flash Redirect AJAX Link - Relative JavaScript with # - Special Function - Embedded VBScript - Special Function - Embedded	Molt alta	Very High Module of Crawling Advanced JS Interpreter Java decompiler - Flash decompiler Advanced VBS Interpreter Analyser of external files Analyser of refirects Advanced extractor of URLs Mòdul de *crawling* molt avançat – Intèrpret JavaScript avançat – Intèrpret VBS avançat – Extractor d'URLs avançat Descompilador Java

Nivell	Descripció	Complexitat	Procés d'extracció
			Descompilador FLASH Analitzador de fitxers externs Analitzador de redireccions

- Temps d'execució / Ample de banda consumit

  Es important també tenir en compte la ràtio existent entre els resultats finals de cadascuna de les eines i els recursos utilitzats per aquestes, és a dir el temps d'execució total i l'ample de banda consumit. Tot i que aquests recursos són cada cop més econòmics, s'han de tenir en compte. Per exemple, una eina que trigui molt de temps per fer la seva tasca i que el resultat sigui molt *pobre* es considerarà pitjor que una eina que trigui menys temps per obtenir un resultat equivalent.

- Detecció de les diferents vulnerabilitats

  Bàsicament és la principal finalitat d'aquestes eines. No totes les eines es comporten igual i sembla existir certa especialització en vulnerabilitats per part dels diferents fabricants. De totes formes, a nivell estadístic es pot comprovar que hi ha vulnerabilitats més fàcilment detectables que altres. En l'estudi de la Stanford University anomenat *State of the Art: Automated Black-Box Web Application Vulnerability Testing* es fa una comparativa de 8 eines de detecció de vulnerabilitats sobre una sèrie de *sites* amb vulnerabilitats reconegudes (a més d'una aplicació feta *ad-hoc* amb vulnerabilitats intoduïdes a propòsit). A nivell general, el *ranking* de vulnerabilitats detectades és el següent (només s'enumeren les que es contemplen en aquest document):

- XSS

  En aquest estudi es divideix en 3 tipus:

  - XSS directe. Es detecta de mitjana un 62.5% de les vulnerabilitats existents
  - XSS emmagatzemat. La mitjana baixa dràsticament a un 15%
  - XSS avançat (injectat via CSS o Flash). Només es detecta un 11.25%

- SQLi (tot i que pot aplicar a d'altres tipus d'injecció de codi)

  També es contemplen 2 tipus:

  - SQLi directe. Es detecta un 21.4%
  - SQLi emmagatzemat. Cap de les eines estudiades van poder detectar cap vulnerabilitat d'aquest tipus.

Una altra font que corrobora aquests resultats és l'estudi realitzat per Shay Chen anomenat Web Application Scanners Accuracy Assessment – Freeware & Open Source Scanners, on es donen uns valors similars per les vulnerabilitats esmentades per XSS tot i que dóna un resultats millors pel que fa a SQLi.

- Falsos Positius

  Un fals positiu és la detecció d'una vulnerabilitat allà o no hi és, on el propi codi de l'aplicació pot donar a determinar que hi ha una vulnerabilitat en funció dels patrons comparatius que utilitza l'eina. Evidentment és més desitjable que existeixin falsos positius que no pas falsos negatius (és a dir, obviar una vulnerabilitat real). L'habilitat de les eines per dirimir si una possible vulnerabilitat trobada és un fals positiu o no també és un aspecte important per a l'avaluació de l'efectivitat d'aquestes.

3.3 Conclusions

Tot i que a priori el percentatge de detecció de vulnerabilitats pugui semblar, a nivell global, no gaire alt, aquestes eines són de gran utilitat per a corregir problemes de seguretat en les aplicacions web, així com establir un control de qualitat pel que les aplicacions haurien de passar. Resoldre problemes per una vulnerabilitat determinada, pot comportar resoldre l'exposició a d'altres vulnerabilitats. En el cas que ocupa aquest document, que es pot focalitzar en la qualitat de les dades d'entrada, les tècniques de resolució de problemes són transversals per totes les vulnerabilitats estudiades. Un àrea de millora en aquestes eines en general és la detecció de XSS i Injecció de codi en *diferit*, o emmagatzemat en una base de dades.

Atesa la gran oferta d'aquests tipus d'eines i la especialització que cadascuna d'elles pot aportar en determinades vulnerabilitats (algunes detecten millor l'XSS i d'altres poden detectar millor injeccions de codi) els usuaris poden fer la tria de la més adient per a les seves necessitats.

Hi ha, però, alguns aspectes a tenir en compte sobre aquestes eines:

- Només tenen en compte vulnerabilitats conegudes
- Utilitzar una eina en un moment donat no és suficient. Es localitzaran les vulnerabilitats en aquell moment, però s'han d'anar utilitzant de forma periòdica.
- Els resultats s'han de revisar de forma acurada per poder distingir falsos positius.

4 ESTUDI D'UNA EINA ESPECÍFICA: SKIPFISH

4.1 Introducció

Skipfish[13] és un projecte de Google Code amb llicència Apache 2.0. Segons el seu equip de desenvolupament, es tracta d'una eina activa de reconeixement de seguretat d'aplicacions web que prepara un mapa interactiu per al lloc web objectiu duent a terme una cerca recursiva i exploracions basades en diccionaris. Aquest mapa resultant s'utilitza per aplicar una sèrie de tests de seguretat basats en vulnerabilitats conegudes. Com a producte final, s'elabora un informe en format HTML indicant les potencials vulnerabilitats detectades.

Havent un gran número d'eines d'aquestes característiques dintre del món del programari lliure, dintre dels motius per triar aquesta en particular podem trobar els següents:

- Aquesta eina està desenvolupada en C pur per aconseguir una velocitat i eficiència que no es troben en d'altres desenvolupades en llenguatges interpretats tals com Perl, Python o Java. Incorpora una gestió optimitzada del protocol HTTP, suposant poc impacte en l'ús de CPU i sent capaç d'efectuar 500+ peticions per segons en llocs webs internet, 2000+ en entorns LAN i 7000+ en instàncies locals (sempre que les aplicacions responguin adequadament, és clar).
- Proporciona heurístiques per donar suport a una gran varietat de *frameworks* de desenvolupament de llocs web (*Apache Struts, CakePHP*...) i diferents tecnologies (J2EE, PHP, .NET...) amb capacitats d'aprenentatge automàtic.
- Està basada en lògica de seguretat de vanguàrdia. De gran qualitat, donant una taxa baixa de falsos positius i capaç de detectar defectes subtils com *blind injection vectors*.

D'aquesta eina el que es proporciona és el codi, per lo qual en ser escrit en C és possible compilar-lo i executar-lo en diferents plataformes com Linux, Mac OSX i Windows (amb les llibreries de Cygwin).

Els tests de seguretat, classificats per criticitat, que contempla aquesta eina són els següents:

- Criticitat alta
    - Injecció SQL / PHP a nivell de servidor (incloent *blind vectors* i paràmetres numèrics)
    - Sintaxi SQL explícita en paràmetres GET o POST.
    - Injecció de comandes de consola a nivell de servidor (*incloent blind* vectors).
    - Injecció XML / XPath a nivell de servidor (*incloent blind* vectors).
    - Vulnerabilitat de formateig de cadenes de text.
    - Vulnerabilitats Integer overflow.
    - Acceptació del mètode HTTP PUT.

- Criticitat mitja
    - Stored and reflected XSS vectors in document body (minimal JS XSS support present).
    - Stored and reflected XSS vectors via HTTP redirects.
    - Stored and reflected XSS vectors via HTTP header splitting.
    - Directory traversal / file inclusion (including constrained vectors).
    - Assorted file POIs (server-side sources, configs, etc).
    - Attacker-supplied script and CSS inclusion vectors (stored and reflected).
    - External untrusted script and CSS inclusion vectors.
    - Mixed content problems on script and CSS resources (optional).
    - Password forms submitting from or to non-SSL pages (optional).
    - Incorrect or missing MIME types on renderables.
    - Generic MIME types on renderables.
    - Incorrect or missing charsets on renderables.
    - Conflicting MIME / charset info on renderables.
    - Bad caching directives on cookie setting responses.

- Criticitat baixa
    - Directory listing bypass vectors.
    - Redirection to attacker-supplied URLs (stored and reflected).
    - Attacker-supplied embedded content (stored and reflected).
    - External untrusted embedded content.
    - Mixed content on non-scriptable subresources (optional).
    - HTTPS -> HTTP submission of HTML forms (optional).
    - HTTP credentials in URLs.
    - Expired or not-yet-valid SSL certificates.
    - HTML forms with no XSRF protection.
    - Self-signed SSL certificates.
    - SSL certificate host name mismatches.
    - Bad caching directives on less sensitive content.

Eines de detecció de vulnerabilitats en aplicacions Web

- Avisos
    - Failed resource fetch attempts.
    - Exceeded crawl limits.
    - Failed 404 behavior checks.
    - IPS filtering detected.
    - Unexpected response variations.
    - Seemingly misclassified crawl nodes.
- Altres informacions
    - General SSL certificate information.
    - Significantly changing HTTP cookies.
    - Changing Server, Via, or X-... headers.
    - New 404 signatures.
    - Resources that cannot be accessed.
    - Resources requiring HTTP authentication.
    - Broken links.
    - Server errors.
    - All external links not classified otherwise (optional).
    - All external e-mails (optional).
    - All external URL redirectors (optional).
    - Links to unknown protocols.
    - Form fields that could not be autocompleted.
    - Password entry forms (for external brute-force).
    - File upload forms.
    - Other HTML forms (not classified otherwise).
    - Numerical file names (for external brute-force).
    - User-supplied links otherwise rendered on a page.
    - Incorrect or missing MIME type on less significant content.
    - Generic MIME type on less significant content.
    - Incorrect or missing charset on less significant content.
    - Conflicting MIME / charset information on less significant content.
    - OGNL-like parameter passing conventions.

4.2 Característiques

Segons els desenvolupadors de l'eina, aquesta té una sèrie de característiques diferenciadores que la situen en una posició d'avantatges davant d'altres productes similars, tant de codi obert com comercials:

- Alta velocitat
- Capacitats úniques de *força bruta*

  Permet la utilització de funcionalitat de *força bruta* combinada amb diccionaris que poden ser configurats per l'usuari en funció de les característiques del lloc web a examinar. Inicialment, es proporcionen quatre diccionaris (minimal, medium, complete, extensions-only). Una de les característiques importants és la capacitat d'*autoaprenentatge* en funció dels continguts inspeccionats per al manteniment automàtic dels diccionaris.

  En aquest sentit, Skipfish proporciona les següents modalitats de funcionament:

  - Sense força bruta. D'aquesta forma es comporta com la majoria d'eines bàsiques existents. No és recomanada per la limitada cobertura que proporciona. No trobaria per exemple recursos no evidents a partir de l'anàlisi del codi com ara `/admin/` o `/index.php.old`.

    L'opció consistiria en no utilitzar diccionari.

  - Força bruta *lleugera*. Amb aquesta modalitat és capaç de trobar recursos pel nom (`/admin/`) o per l'extensió (`/index.php.old`), però no per les dues característiques (`/admin.php.old`). El diccionari a utilitzar en aquest cas (dels proporcionats) és el *complete*.

  - Força bruta *normal*. Amb aquesta modalitat és capaç de detectar recursos tant pel nom com per l'extensió a la vegada. Es tracta d'una opció molt més lenta, però permet detectar més recursos. El diccionari a utilitzar en aquest cas (dels proporcionats) és el *minimal*.
- Comprovacions de seguretat d'alta qualitat

  Segons els autors, la lògica de seguretat que apliquen altres *scanners* és massa simple. Per exemple:

  - Per fer un test de XSS es limiten a injectar un codi tal com `<script>alert(1)</script>` Això pot portar tant a falsos negatius (si l'entrada està parcialment escapada) o a falsos positius (si per exemple un codi similar està inclòs dintre d'un comentari HTML)
  - Per a determinar un accés directoris fora de l'aplicació, intentar fer referències del tipus `../../../../../etc/passwd`
  - Per fer una injecció SQL, proporcionar codi específic per a diferents tecnologies i després comparar sobre patrons preestablerts per determinar l'èxit de la injecció. A la que es dóna una tecnologia no contemplada, el test en aquest aspecte és inútil.

  Skipfish es basa més en patrons de comportament que no pas en la comparació amb un conjunt de signatures específiques per cada tipus de test.

- Contemplació de problemes considerats de poca importància

  Moltes eines no contemplen una sèrie de vulnerabilitats que són considerades de poca importància o massa subtils, com per exemple discordança entre els jocs de caràcters declarats en la capçalera HTTP i el contingut real, o la discordança entre el tipus de contingut (MIME type) i el contingut real. Aquestes vulnerabilitats poden "ajudar" a produir atacs de XSS, XSSI, XSRF, etc.

  Skipfish té en compte aquests problemes potencials.

- Producció de resultats finals més senzills

  De forma intencionada Skipfish no inclou en els resultats finals una sèrie de vulnerabilitats abastament recollides en d'altres eines, tals com detecció de *cookies* no http, detecció de camps considerats *password* i que no tenen desactivada la característica *autocomplete*. La motivació per obviar aquestes vulnerabilitats és que els autors consideren que no aporten res i a causa d'afegir aquest *soroll* es produeixen informes finals més illegibles.

4.3 Instal·lació

Com s'ha comentat anteriorment, aquesta eina està escrita en C pur i és possible executar-la en diferents plataformes. Es proporciona el codi font (es pot trobar a l'adreça http://code.google.com/p/skipfish/downloads/list) i s'ha de compilar en el SO destí. Pel que fa a aquest treball, s'utilitzarà la versió 2.09b en un SO Windows 7. Per que funcioni amb aquesta plataforma, és necessari tenir instal·lades les llibreries Cygwin amb el compilador GNU C.

L'aplicació s'ha modificat per un error en el tractament de l'usuari i el password quan s'especificaven les credencials d'autenticació HTTP (opció -A). L'errada estava en que si es posava usuari i password, *requeria* també un formulari d'autenticació (opció `–auth-form`). En l'arxiu skipfish.c (situat dintre del directori src de la instal·lació) s'ha canviat la següent línia (726):

```
if (auth_user && auth_pass) {
```

per

```
if (auth_type != AUTH_BASIC && auth_user && auth_pass) {
```

4.4 Execució

Aquesta eina s'executa des d'una finestra de consola, amb línia de comandes. Les opcions disponibles són les següents:

```
Authentication and access options:

  -A user:pass      - use specified HTTP authentication credentials
  -F host=IP        - pretend that 'host' resolves to 'IP'
  -C name=val       - append a custom cookie to all requests
  -H name=val       - append a custom HTTP header to all requests
  -b (i|f|p)        - use headers consistent with MSIE / Firefox / iPhone
  -N                - do not accept any new cookies
  --auth-form url   - form authentication URL
  --auth-user user  - form authentication user
  --auth-pass pass  - form authentication password
  --auth-verify-url -  URL for in-session detection

Crawl scope options:
```

```
  -d max_depth     - maximum crawl tree depth (16)
  -c max_child     - maximum children to index per node (512)
  -x max_desc      - maximum descendants to index per branch (8192)
  -r r_limit       - max total number of requests to send (100000000)
  -p crawl%        - node and link crawl probability (100%)
  -q hex           - repeat probabilistic scan with given seed
  -I string        - only follow URLs matching 'string'
  -X string        - exclude URLs matching 'string'
  -K string        - do not fuzz parameters named 'string'
  -D domain        - crawl cross-site links to another domain
  -B domain        - trust, but do not crawl, another domain
  -Z               - do not descend into 5xx locations
  -O               - do not submit any forms
  -P               - do not parse HTML, etc, to find new links

Reporting options:

  -o dir          - write output to specified directory (required)
  -M              - log warnings about mixed content / non-SSL passwords
  -E              - log all HTTP/1.0 / HTTP/1.1 caching intent mismatches
  -U              - log all external URLs and e-mails seen
  -Q              - completely suppress duplicate nodes in reports
  -u              - be quiet, disable realtime progress stats
  -v              - enable runtime logging (to stderr)

Dictionary management options:

  -W wordlist     - use a specified read-write wordlist (required)
  -S wordlist     - load a supplemental read-only wordlist
  -L              - do not auto-learn new keywords for the site
  -Y              - do not fuzz extensions in directory brute-force
  -R age          - purge words hit more than 'age' scans ago
  -T name=val     - add new form auto-fill rule
  -G max_guess    - maximum number of keyword guesses to keep (256)

  -z sigfile      - load signatures from this file

Performance settings:

  -g max_conn     - max simultaneous TCP connections, global (40)
  -m host_conn    - max simultaneous connections, per target IP (10)
  -f max_fail     - max number of consecutive HTTP errors (100)
  -t req_tmout    - total request response timeout (20 s)
  -w rw_tmout     - individual network I/O timeout (10 s)
  -i idle_tmout   - timeout on idle HTTP connections (10 s)
  -s s_limit      - response size limit (400000 B)
  -e              - do not keep binary responses for reporting

Safety settings:

  -l max_req      - max requests per second (0.000000)
  -k duration     - stop scanning after the given duration h:m:s
```

Com es pot veure, aquestes opcions són molt extenses, per lo que a continuació es descriuen algunes que, fora de les que poden ser les més evidents, són de gran utilitat i a tenir en compte:

- Limitació de la profunditat de l'arbre de *crawling*. Les opcions -c (limitar al fill immediat) i -x (limitar el total de profunditat) permeten ajustar l'scan en llocs web que poden ser molt extensos.

- Comprovacions addicionals SSL. Amb l'opció -M es proporciona informació d'avisos sobre continguts seguir i no segurs dintre d'una mateixa pàgina.
- Especificar un browser determinat. Amb l'opció -b es pot indicar si s'està actuant com un Internet Explorer, un Mozilla Firefox o un iPhone. No es limita a canviar la capçalera *User-Agent*, si no que inclou d'altres específiques per cadascun d'ells.
- No acceptació de noves cookies. En un sistema autenticat on l'estat és guardat per una *cookie*, és possible que durant el procés d'*scan* es cridi a una funcionalitat de sortir de l'aplicació (si no s'ha especificat evitar-la amb l'opció -X) i reescriure o eliminar la *cookie*. Per evitar això es poden utilitzar les opcions -N i -C.
- Especificar dominis de confiança per excloure'ls dels tests. Això es pot obtenir amb l'opció -B.
- Reducció de l'ús de memòria. Per generar l'informe final, Skipfish guarda a memòria RAM mostres dels documents trobats. Quan s'estan verificant llocs web molt extensos, el consum de memòria pot ser molt alt. Utilitzant l'opció -e s'eliminen els continguts binaris (no ASCII) sense impactar en la qualitat del resultat final.
- Per una execució més ràpida, es pot indicar amb l'opció -P que Skipfish no analitzi l'HTML obtingut per tal d'extreure els enllaços utilitzant així només el sistema de força bruta.

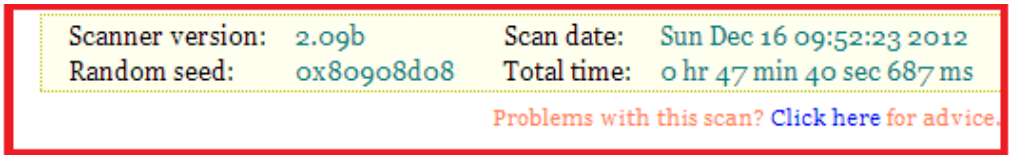
4.5 Resultats

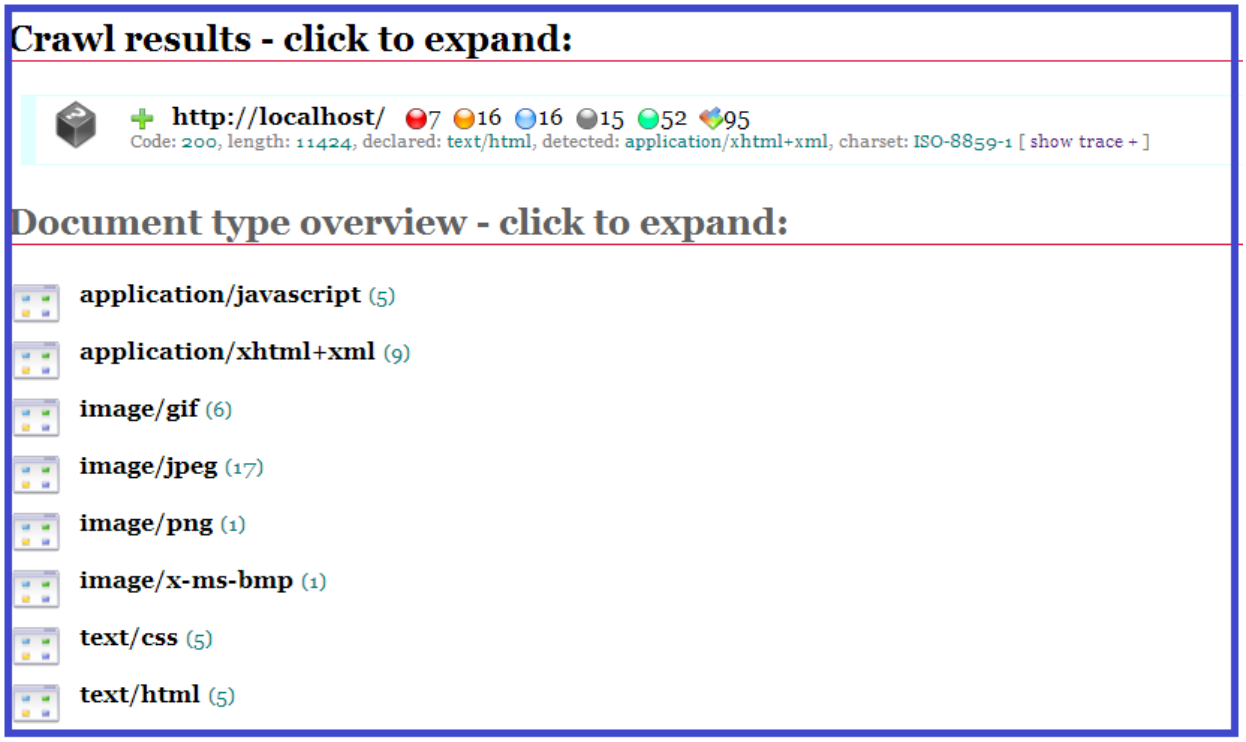
Un dels pocs arguments obligatoris que s'ha d'especificar en l'execució d'Skipfish és el directori de destí dels resultats (opció `-o <directori>`). En aquest directori es troba típicament:

- Una pàgina html (index.html) que consisteix en l'informe. Aquest html sempre és igual, independentment del test realitzat i el seu contingut depèn d'arxius javascript que es descriuen a continuació.
- Una sèrie de recursos gràfics (arxius .png)
- Directoris i subdirectoris on s'emmagatzemen les peticions efectuades i els seus resultats
- Uns arxius javascript que són realment el resum del resultat. Els més destacats són:
  - summary.js. Conté 4 variables javascript informant de la versió utilitzada, el temps emprat en l'scan, la data de l'escan i la llavor utilitzada.
  - samples.js. Conté 2 variables javascript en format JSON (JavaScript Object Notation):
    - mime_samples: conté la relació de URLs agrupades per MIME-types. Entre d'altra informació, per cada url s'especifica el subdirectori (comentats en el tercer punt d'aquesta llista) on hi ha les dades relatives a la petició i a la resposta efectuada.
    - issue_samples: conté la relació de vulnerabilitats detectades. Per a cadascuna d'elles s'informa de la criticitat, el tipus de vulnerabilitat, la URL i la referència als subdirectoris on es poden trobar les dades de petició i resposta.

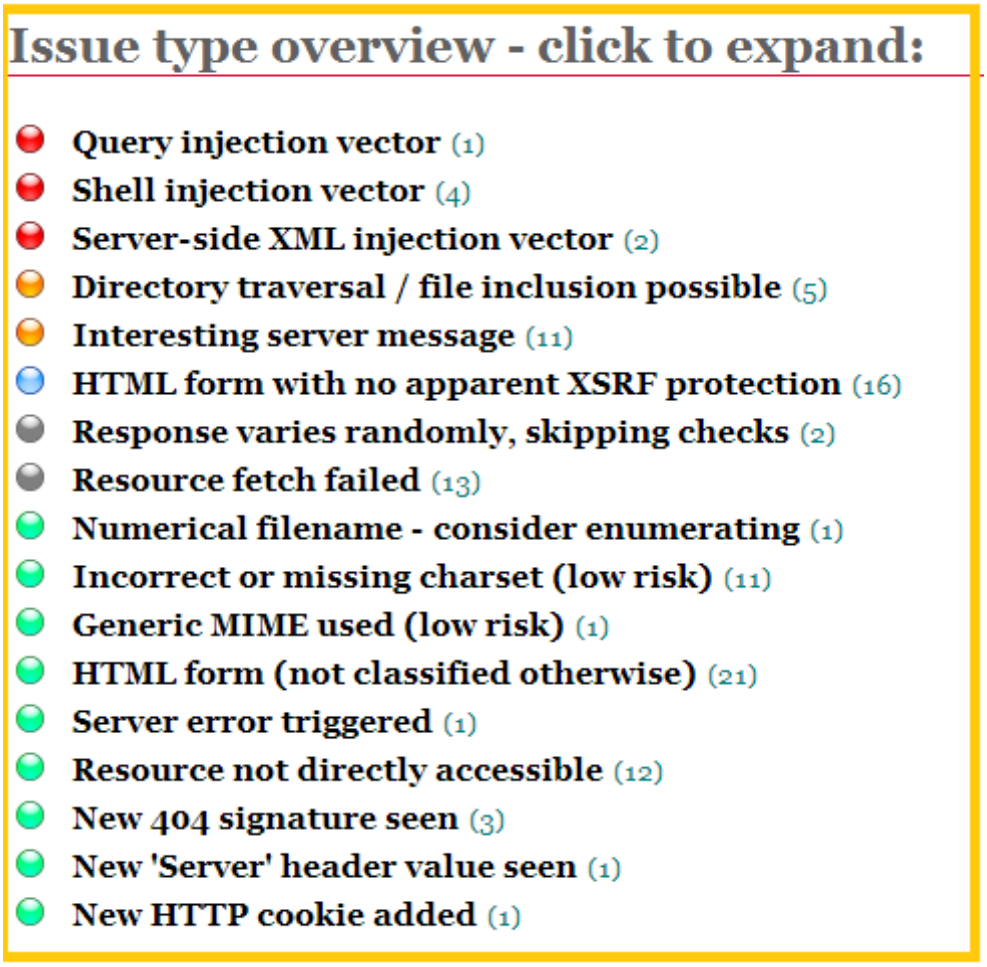
En les següents imatges es mostren parts d'un informe d'Skipfish on es relacionen aquests elements:

summary.js





mime_types de samples.js



issue_samples de samples.js

Com es pot veure, es mostra la relació de mime-types trobats durant la inspecció del lloc web. Cadascun d'ells es pot desplegar i es poden veure les diferents urls trobades:

Es mostra la URL (és *linkable*), la seva mida i la traça de la petició:

```
=== REQUEST ===

GET /WebGoat/css/layers.css HTTP/1.1
Host: localhost
Accept-Encoding: gzip
Connection: keep-alive
User-Agent: Mozilla/5.0 SF/2.09b
Range: bytes=0-399999
Referer: http://localhost/
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Cookie: JSESSIONID=91B9611B529F2CCF7FF4D489C9548470


=== RESPONSE ===

HTTP/1.1 200 Contenido Parcial
Server: Apache-Coyote/1.1
Cache-Control: private
Expires: Thu, 01 Jan 1970 01:00:00 CET
Accept-Ranges: bytes
ETag: W/"219-1334915848000"
Last-Modified: Fri, 20 Apr 2012 09:57:28 GMT
Content-Range: bytes 0-218/219
Content-Type: text/css
Content-Length: 219
Date: Sun, 16 Dec 2012 08:13:37 GMT
#lessonTitle       {position:absolute;left:94px;top:75px;width:690px;height:22px;z-index:1;float:
right;font-size: 20px;color: #FFFFFF;}
#hMenuBar {position:absolute;left:245px;top:108px;width:538px;height:22px;z-index:2;}


=== END OF DATA ===
```

De la mateixa forma, les vulnerabilitats detectades són desplegables i es pot veure cadascuna d'aquella tipologia detectades:

4.6 Execució d'un test

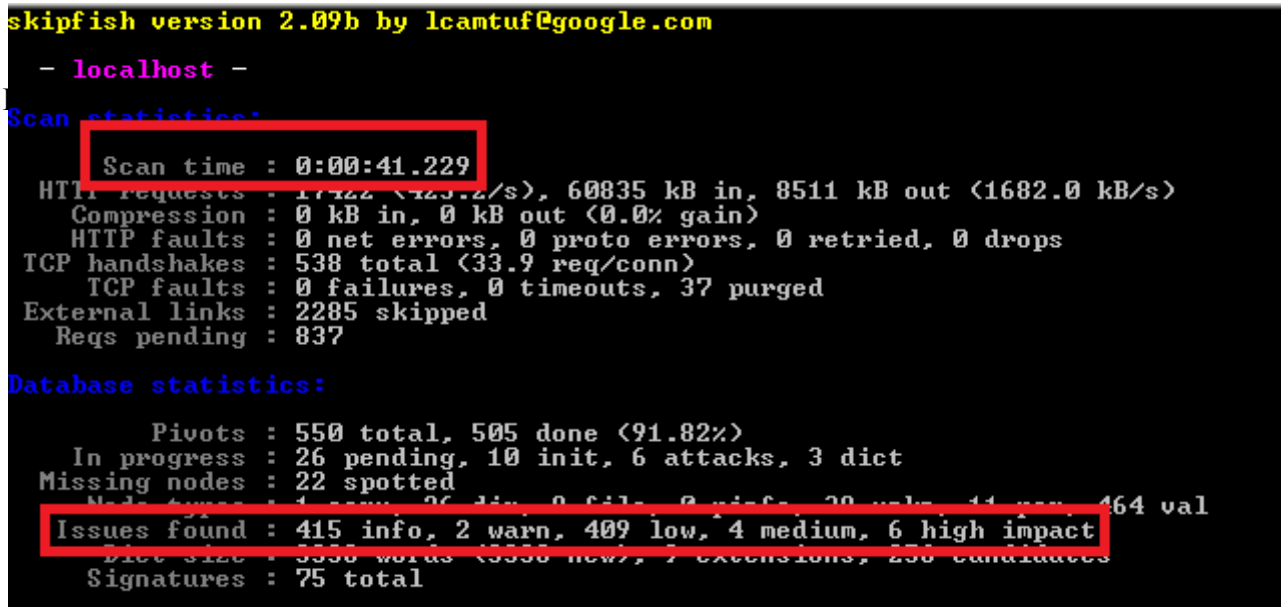
Per provar aquesta eina, s'utilitzarà una aplicació Web, desenvolupada per OWASP, anomenada WebGoat (https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project). Aquesta aplicació és una aplicació JEE deliberadament insegura dissenyada per ensenyar i demostrar vulnerabilitats. Està estructurada en 30 lliçons i inclou vulnerabilitats tals com:

- Cross-site scripting
- Control d'accés
- Seguretat en *threads*
- Manipulació de paràmetres *hidden*.
- *Cookies* de sessió vulnerables.
- Injecció SQL de tipus 2 (*blind*)
- Injecció SQL numèrica
- Injecció SQL alfanumèrica

Després d'instal·lar-la i posar-la en marxa, es procedeix a executar un test amb *Skipfish*, mitjançant la línia de comandes, utilitzant les opcions mínimes:

```
skipfish.exe -o resultat -A webgoat:webgoat  http://localhost/WebGoat/attack
```

Ja només en menys d'un minut d'execució va trobar fins al 6 vulnerabilitats d'alt impacte:



Després de mitja hora funcionant, el test finalitza donant aquests resultats:

- **Query injection vector** (11)
- **Shell injection vector** (6)
- **Server-side XML injection vector** (5)
- **Directory traversal / file inclusion possible** (7)
- **Interesting server message** (16)
- **XSS vector in document body** (1)
- **HTML form with no apparent XSRF protection** (19)
- **Response varies randomly, skipping checks** (2)
- **Resource fetch failed** (14)
- **OGNL-like parameter behavior** (2)
- **Incorrect or missing charset (low risk)** (11)
- **Generic MIME used (low risk)** (1)
- **HTML form (not classified otherwise)** (25)
- **Server error triggered** (1)
- **Resource not directly accessible** (10)
- **New 404 signature seen** (3)
- **New 'Server' header value seen** (1)
- **New HTTP cookie added** (1)

4.7 Àrees de millora

Skipfish és un projecte viu que està en constant evolució. De fet, durant l'elaboració d'aquest document, va aparèixer una nova *release*. Una de les principals motivacions d'aquesta evolució és que l'autor és conscient de determinades mancances i àrees de millora. Algunes d'aquestes són les següents:

- Millora en la detecció de *buffer overflow.*
- Una detecció de XSS més acurada. Les comprovacions que es fan al respecte són força *rudimentàries* (segons l'autor). A més, l'eina no disposa d'un motor d'scripting del tot adequat per avaluar expressions.
- Efectuar comprovacions sobre elements no HTML com ara PDFs, Flash, Applets Java, etc...
- Afegir comprovacions de força bruta sobre *passwords* i noms de fitxers basats en números.
- Contemplar altres tipus d'autenticació. Actualment només contempla autenticació BASIC i basada en formulari. Caldria afegir autenticacions addicionals com ara NTLM i autenticació DIGEST.
- Suport a per a l'ús de proxies HTTP i SOCKS.
- Possibilitat de rellençar a partir d'un punt si aquesta queda aturada per qualsevol causa.
- Sistema d'instal·lació més automatitzat. Actualment, la "instal·lació" consisteix en descomprimir una carpeta que conté els fonts i compilar-los. Encara que només s'afegís una instrucció tipus *make install*, ja seria una millora.
- El tema que ocupa el següent capítol d'aquest document: una millora en la interfície d'usuari i en els informes generats.

4.8 Conclusions

Efectivament sembla una eina força efectiva i eficient. Fent un test sense cap mena de preparació especial (determinar un diccionari adient, afinant els paràmetres més adients, …) el número de vulnerabilitats trobades és força alt tenint en compte les efectivitats globals d'aquestes eines segons es descriu en el capítol 3 d'aquest document.

Tot i la seva efectivitat i eficiència els aspectes al meu criteri que serien molt millorables serien tant la interacció amb l'usuari (la interfície de línia de comandes és molt complexa) i una millora en la presentació dels resultats.

Per solventar aquests problemes, en aquest document es presentarà a continuació una eina consistent en una interfície web que permeti a l'usuari utilitzar i aprofitar *Skipfish* de forma més còmoda i útil.

5 INTERFÍCIE WEB

5.1 Descripció funcional

5.1.1 Introducció

Després de presentar l'eina *Skipfish*, a continuació es proposa una aplicació que consisteix en una interfície web que permeti utilitzar-la a l'usuari de forma més fàcil i poder obtenir a l'hora uns resultats amb informació addicional que serà d'utilitat per entendre el problema i la seva possible resolució.

Aquesta nova interfície Web té com a objectius el desenvolupament dels següents punts inicials:

- Una pàgina principal que permet accedir, mitjançant un menú, a totes les funcionalitats de l'aplicació.

  Els casos d'ús d'aquesta pantalla són:

  - L'usuari demana des d'un navegador l'adreça de l'aplicació
  - L'aplicació presenta el menú principal.

A partir d'aquí es descriuen els casos d'ús que es deriven dels anteriors:

- **Alta d'una nova definició**. Considerem una definició com la configuració necessària per a llençar una execució. En aquesta pantalla apareixen totes les opcions de línia de comandes disponibles en *Skipfish*. Aquesta pantalla s'ha agrupat en els diferents apartats en què estan organitzats els paràmetres (veure apartat 4.4). Els casos d'ús d'aquesta pantalla són:
  - L'usuari ha fet *click* en l'opció de menú *Nova definició*
  - L'aplicació presenta el formulari d'alta de definició
  - L'usuari introdueix, com a mínim, una descripció per a la nova definició i la URL d'inici de l'aplicació a realitzar els tests.
  - Sub-cas 1:
    - L'usuari prem el botó acceptar
    - L'aplicació efectua les verificacions tant a nivell local com a nivell de servidor
    - Sub-cas 1.1:
      - Les validacions són correctes, el sistema calcula un nou identificador intern i insereix a la base de dades la nova definició i presenta la pantalla de llistat de definicions disponibles.
    - Sub-cas 1.2:
      - Les validacions són incorrectes, el sistema informa a l'usuari del problema i presenta el formulari d'alta de definició mantenint les dades prèviament

introduïdes per l'usuari.

- Sub-cas 2:
  - L'usuari prem el botó cancel·lar
  - L'aplicació demana confirmació per abandonar l'edició
  - Sub-cas 1.1:
    - L'usuari accepta.
    - El sistema mostra la llista de les definicions disponibles en el sistema.
  - Sub-cas 1.2:
    - L'usuari rebutja.
    - El sistema presenta el formulari d'alta de definició mantenint les dades prèviament introduïdes per l'usuari.

- **Consulta de les definicions**. Es tracta d'una pantalla per mostrar la llista de les execucions disponibles en el sistema. Aquesta pantalla, a més de dades identificatives de cada definició en cadascuna de les seves files, es proporcionen accessos (*links*) per a les diferents opcions disponibles per a una definició. Els casos d'ús d'aquesta funcionalitat són els següents:
  - L'usuari ha fet *click* en l'opció de menú *Llistat de definicions*
  - L'aplicació calcula la llista de definicions disponibles buscant-les a la base de dades
  - L'aplicació presenta la pantalla de llistat de definicions.

- **Execució de tests en funció de les definicions disponibles**. Es tracta d'una extensió del cas d'ús *Consulta de les definicions*:
  - Un cop obtinguda la llista, l'usuari fa *click* sobre el *link* *Executar.*
  - El sistema demana confirmació
  - Sub-cas 1.1:
    - L'usuari *accepta*.
    - El sistema presenta la pantalla de *Execució de definició.* En aquesta pantalla es pot especificar una descripció per a la definició i una llista de correus electrònics on notificar.
    - L'usuari prem el botó *Acceptar*
    - El sistema demana confirmació.
      - Sub-cas 1.1.1:
        - L'usuari accepta.
        - El sistema llença l'execució de la línia de comandes corresponent a la definició triada i mostra una pantalla informant que s'ha sol·licitat l'execució

de la definició. Aquesta execució es fa en paral·lel al funcionament de l'aplicació. Quan finalitza executa les següents operacions:

- Actualitzar l'estat a la base de dades
- Si s'han informat e-mails de notificació, s'envien a les adreces corresponents.

- Sub-cas 1.1.2:
  - L'usuari rebutja.
  - El sistema manté a l'usuari en la mateixa pantalla de llistat de definicions.

- Sub-cas 1.2:
  - L'usuari rebutja.
    - El sistema presenta la pantalla de *Execució de definició*. En aquesta pantalla es pot especificar una descripció per a la definició i una llista de correus electrònics on notificar.

- **Eliminació de definicions disponibles**. Es tracta d'una extensió del cas d'ús *Consulta de les definicions*:
  - Un cop obtinguda la llista, l'usuari fa *click* sobre el *link* *Eliminar.*
  - El sistema demana confirmació
  - Sub-cas 1.1:
    - L'usuari accepta.
    - El sistema esborra la definició triada
    - El sistema torna a executar el cas d'ús *Consulta de les definicions*.

  - Sub-cas 1.2:
    - L'usuari rebutja.
    - El sistema manté a l'usuari en la mateixa pantalla de llistat de definicions.

- **Consulta de les execucions efectuades d'una definició**. Es tracta d'una extensió del cas d'ús *Consulta de les definicions*.
  - Un cop obtinguda la llista, l'usuari fa *click* sobre el *link* *Execucions*
  - El sistema calcula la llista de les execucions disponibles d'una definició.
  - Es presenta una pantalla amb el llistat de les definicions calculades. Per a cada fila es mostra una *checkbox*, la seva descripció (donada en el moment de sol·licitar l'execució), les dades d'inici i finalització i 2 accions representades en 2 *links*: *Eliminar* i *Consultar*.

- **Eliminació d'execucions de definicions disponibles**. Es tracta d'una extensió del cas d'ús

Consulta de les execucions d'una definició:

- Un cop obtinguda la llista, l'usuari fa *click* sobre el *link* *Eliminar.*
- El sistema demana confirmació
- Sub-cas 1.1:
  - L'usuari accepta.
  - El sistema esborra l'execució triada
  - El sistema torna a executar el cas d'ús *Consulta de les execucions d'una definició*.

- Sub-cas 1.2:
  - L'usuari rebutja.
  - El sistema manté a l'usuari en la mateixa pantalla de llistat d'execucions d'una definició.

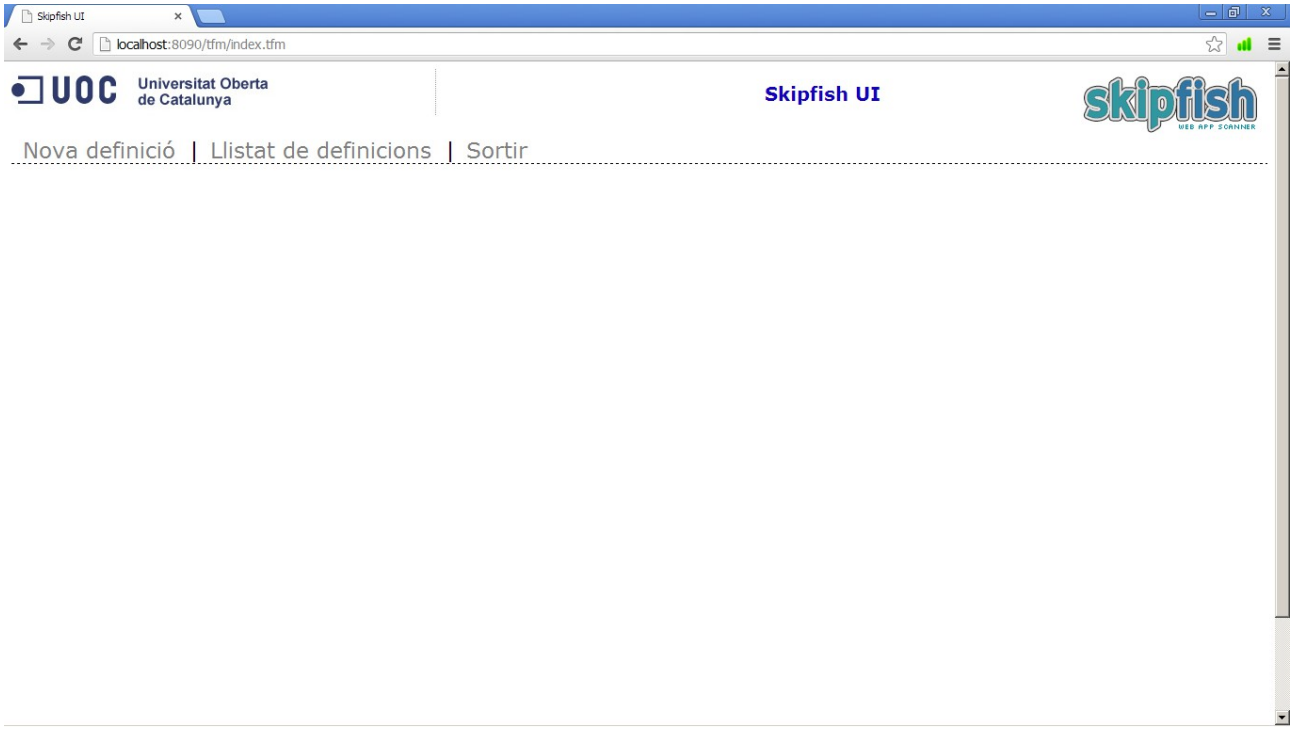
- **Consulta d'una execució**. Elaboració d'informes més estesos respecte als obtinguts per *Skipfish. Es tracta d'una extensió del cas d'ús Consulta de les execucions d'una definició*:
  - Un cop obtinguda la llista, l'usuari fa click sobre el *link* *Consultar*.
  - El sistema calcula el nou informe
  - El sistema presenta una pantalla amb l'informe calculat.

- **Comparativa de diferents execucions d'una definició**. Es tracta d'una extensió del cas d'ús *Consulta de les execucions d'una definició*:
  - Un cop obtinguda la llista, l'usuari marca els *checkboxes* de 2 (i només 2) de les execucions disponibles.
  - El sistema fa aparèixer un nou botó*:* Comparar execucions
  - El sistema calcula el nou informe comparatiu
  - El sistema presenta una pantalla amb l'informe comparatiu calculat.
- **Eliminació d'execucions de definicions disponibles**. Es tracta d'una extensió del cas d'ús *Consulta de les execucions d'una definició*:
  - Un cop obtinguda la llista, l'usuari fa *click* sobre el *link* *Eliminar.*
  - El sistema demana confirmació

A continuació es descriuen les pantalles de l'aplicació en base als casos d'ús descrits.

5.1.2 Presentació del menú de l'aplicació

En accedir a aquesta aplicació des d'un navegador (http://servidor/tfm/index.tfm) es presenta la següent pàgina d'inici:



Es pot veure un menú amb 3 opcions:

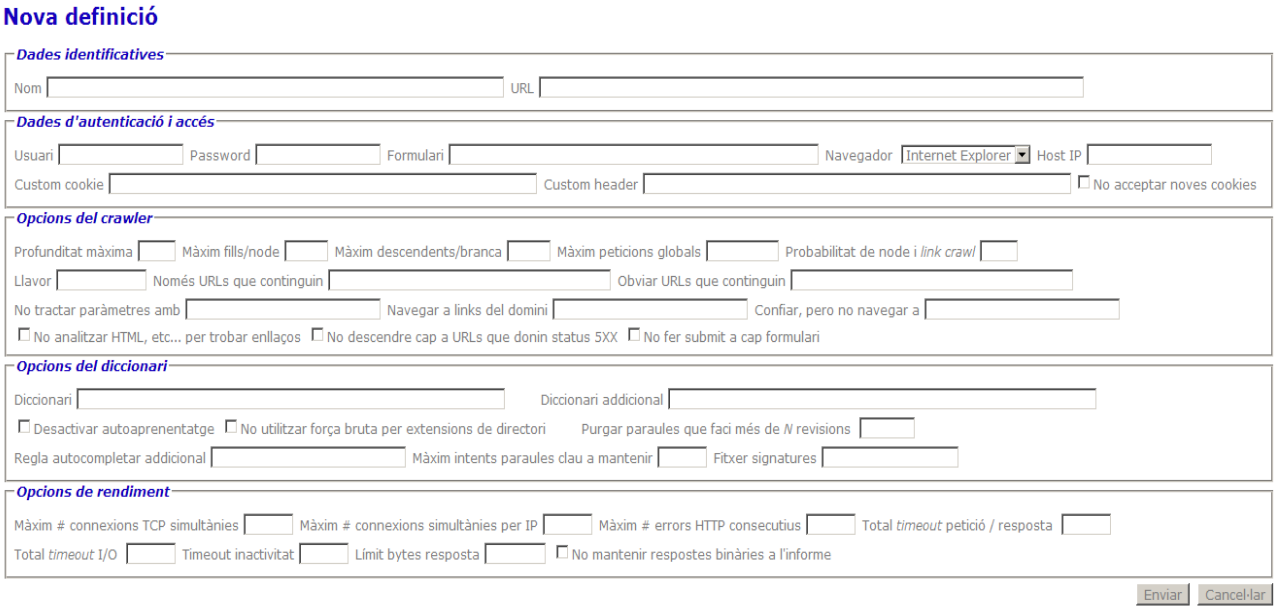
- Nova definició
- Llistat de definicions
- Sortir

5.1.3 Nova definició

Com s'ha pogut comprovar en la descripció de l'eina Skipfish, aquesta presenta una interfície en mode línia de comandes que pot arribar a ser molt complexa i difícil de gestionar si es volen aprofitar al màxim les possibilitats que ofereix.

A tal efecte es proporcionarà un formulari que permeti donar d'alta i modificar diferents tests. El formulari per donar d'alta / mantenir una definició s'accedeix des del punt de menú

Aquest formulari és el següent:



En el moment de donar d'alta un nou test, com a informació obligatòria s'haurà de proporcionar una descripció identificativa del test i la URL d'inici del lloc web a validar.

La resta de dades aniran lligades a les diferents opcions que ofereix l'eina i estaran sotmeses a validacions lògiques de compatibilitat entre elles. En altres casos en funció de la configuració introduïda, les execucions del test utilitzaran unes opcions o unes altres. Per exemple, si s'especifica un usuari i un password, però no la URL d'un formulari d'autenticació, aleshores es considerarà utilitzar l'opció -A, en cas contrari, s'utilitzarien les opcions --auth-form, --auth-user, --auth-path.

La relació de camps, relacionats amb les opcions d'Skipfish que representen, que disposarà aquest formulari de manteniment de tests és la següent:

Dades d'autenticació i accés:

Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
Nom	-	Text	Cap	No ha de ser buit	
URL	Paràmetre obligatori URL de Skipfish	Text	Cap	Ha de ser una URL formalment correcta	URL d'inici del lloc web a fer el test
Usuari	Part de -A o bé --auth-user	Text	Cap	No ha de ser buit si el camp Password no és buit	Usuari per accedir a l'autenticació si el lloc web ho requereix
Password	Part de -A o bé --auth-pass	Text	Cap	No ha de ser buit si el camp Usuari no és buit	Password per accedir a l'autenticació si el lloc web ho requereix
IP de HOST	-F	Text	Cap	Si no és buit ha de complir l'expressió regular: `\w+\s*[=]\s*\d{1,3}[.]\d{1,3}[.]\d{1,3}[.]\d{1,3}[.]`	Assumeix que HOST té com a adreça IP l'especificada
Custom cookie	-C	Text	Cap	Si no és buit ha de complir l'expressió regular `\w+\s*[=].+`	Afegir la cookie especificada en totes les peticions
Custom header	-H	Text	Cap	Si no és buit ha de complir l'expressió regular `\w+\s*[=].+`	Afegir la capçalera HTTP especificada en totes les

Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
					peticions
Navegador	-b	Llista desplegable	Valors: <buit> IE Firefox iPhone		Realitza les peticions com si fos un dels navegadors especificats
Acceptar noves cookies	-N	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció

Opcions del crawler:

Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
Profunditat màxima	-d	Text	16	Numèric, més gran que zero	
Màxim número de fills a indexar per node	-c	Text	512	Numèric, més gran que zero	
Màxim número de descendents a indexar per branca	-x	Text	8192	Numèric, més gran que zero	
Màxim número de peticions globals	-r	Text	100000000	Numèric, més gran que zero	
Probabilitat de node i *link crawl*	-p	Text	100	Numèric >0 && <=100	
Llavor	-q	Text	Buit	Si no buit, número	

Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
				hexadecimal superior a zero	
Només seguir URLs que continguin la cadena	-I	Text	Buit		
Obviar URLs que continguin la cadena	-X	Text	Buit		
No tractar paràmetres que continguin la cadena	-K	Text	Buit		
Navegar a links que apuntin al domini	-D	Text	Buit	Ha de complir l'expressió `\w*`	
Confiar, però no navegar cap a links del domini	-B	Text	Buit	Ha de complir l'expressió `\w*`	
No analitzar HTML, etc... per trobar enllaços	-P	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció
No descendre cap a URLs que donin com a status 5XX	-Z	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció
No fer submit a cap formulari	-O	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció

Opcions del diccionari:

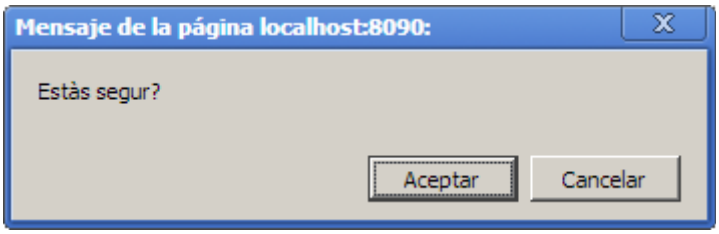
Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
Diccionari	-W	Text	Buit	Si no buit ha de representar un path vàlid	Si no s'especifica cap, s'utilitza el que ve per defecte a la instal·lació (skipfish.wl copiat al directori principal de la instal·lació).
Diccionari addicional (només lectura)	-S	Text	Buit	Si no buit ha de representar un path vàlid	
Desactivar auto-aprenentatge	-L	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció
No utilitzar força bruta per extensions de directori	-Y	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció
Probabilitat de node i *link crawl*	-p	Text	100	Numèric >0 && <=100	
Purgar paraules que faci més de N revisions	-R	Text	Buit	Si no buit, número superior a zero	
Regla d'auto-completar addicional	-T	Text	Buit	Si no és buit ha de complir l'expressió regular `\w+\s*[=].+`	
Màxim número d'intents de paraules clau a	-G	Text	256	Numèric superior a zero	

mantenir					
Utilitzar el fitxer de signatures	-z	Text		Si no buit ha de representar un path vàlid	

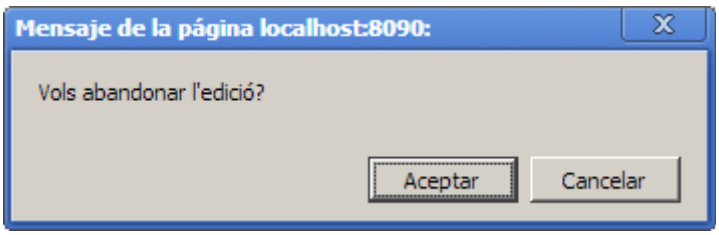
Opcions de rendiment:

Etiqueta	Opció relacionada	Tipus	Valor per defecte	Validacions	Observacions
Màxim número de connexions TCP simultànies	-g	Text	40	Numèric superior a zero	
Màxim número de connexions simultànies per IP	-M	Text	10	Numèric superior a zero	
Màxim d'errors HTTP consecutius	-f	Text	100	Numèric superior a zero	
Total timeout petició/ resposta	-t	Text	10	Numèric superior a zero	
Total timeout entrada/ sortida	-w	Text	10	Numèric superior a zero	
Timeout de inactivitat	-i	Text	10	Numèric superior a zero	
Limit en bytes de la mida de la resposta	-s	Text	400000	Numèric superior a zero	
No mantenir respostes binàries a l'informe	-e	Check-box	Desmarcat		Si marcat, s'assumeix que s'utilitza l'opció

En prémer el botó *Enviar* es demana confirmació de l'alta:



De la mateixa forma, en polsar el botó *Cancel·lar* també es demana confirmació:



En tots dos casos, després de realitzar l'acció triada (Enviar / Cancel·lar) es torna a la llista de definicions disponibles.

Cada definició s'emmagatzema en una taula de base de dades i té com a identificador un UUID. Les columnes d'aquesta taula són les següents:

- ID. Identificador únic. No introduït per l'usuari
- NAME. Descripció de la definició donada per l'usuari
- CREATION_DATE. Data de creació / modificació. No introduït per l'usuari.
- URL. Adreça d'inici del test
- AUTH_USER. Usuari si el lloc web a fer el test ho requereix
- AUTH_PASS. Password de l'usuari anterior
- URL_FORM. Adreça del formulari d'autenticació, si les credencials no són demanades pel propi navegador (per exemple en autenticacions de tipus BASIC).
- VERIFY_FORM. Adreça de verificació de sessió vàlida.

La resta de columnes, corresponent a la resta d'opcions que es poden especificar en una execució d'Skipfish, tenen la següent nomenclatura, per tal de poder gestionar aquesta informació de forma més automatitzada:

PARM*C*_*O*

On PARM és un prefix, *C* indica si el nom de l'opció és majúscula (U, d'*uppercase*) o minúscula (L, de *lowercase*) i *O* és el nom de l'opció. Per exemple, l'opció

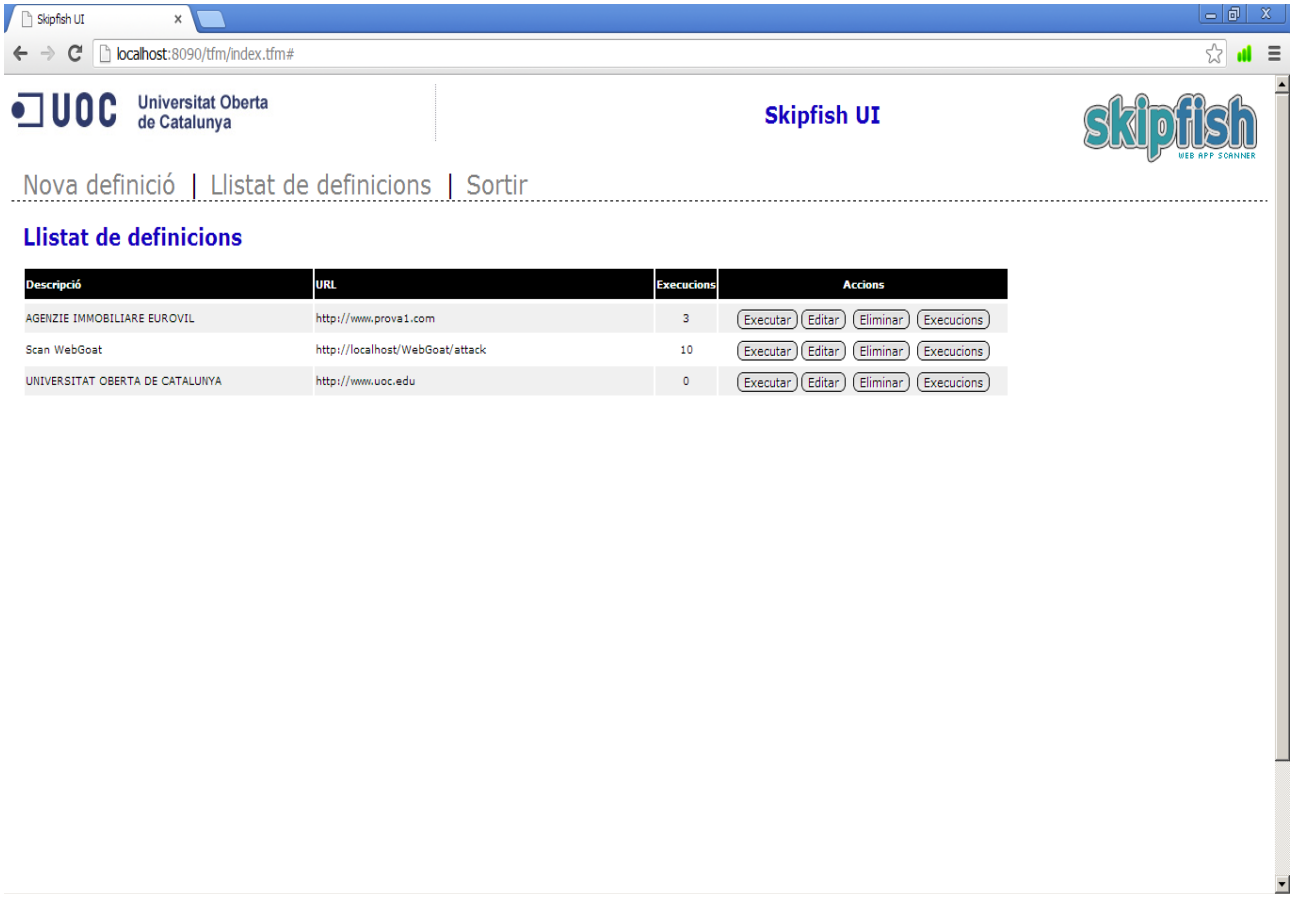
```
-Q              - completely suppress duplicate nodes in reports
```

es correspondria a la columna PARMU_Q, mentre que l'opció

```
-u              - be quiet, disable realtime progress stats
```

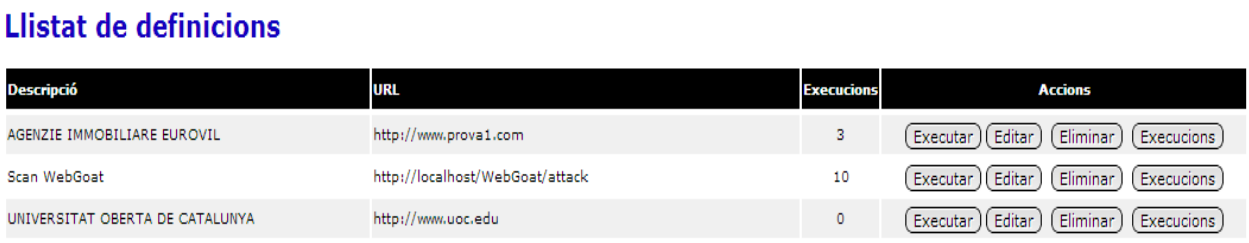
es correspondria a la columna PARML_U.

5.1.4 Gestió de definicions



Per poder gestionar les definicions inicialment es presentarà la llista de les disponibles. S'accedeix des del punt de menú

Cada fila tindrà accions associades a la definició:



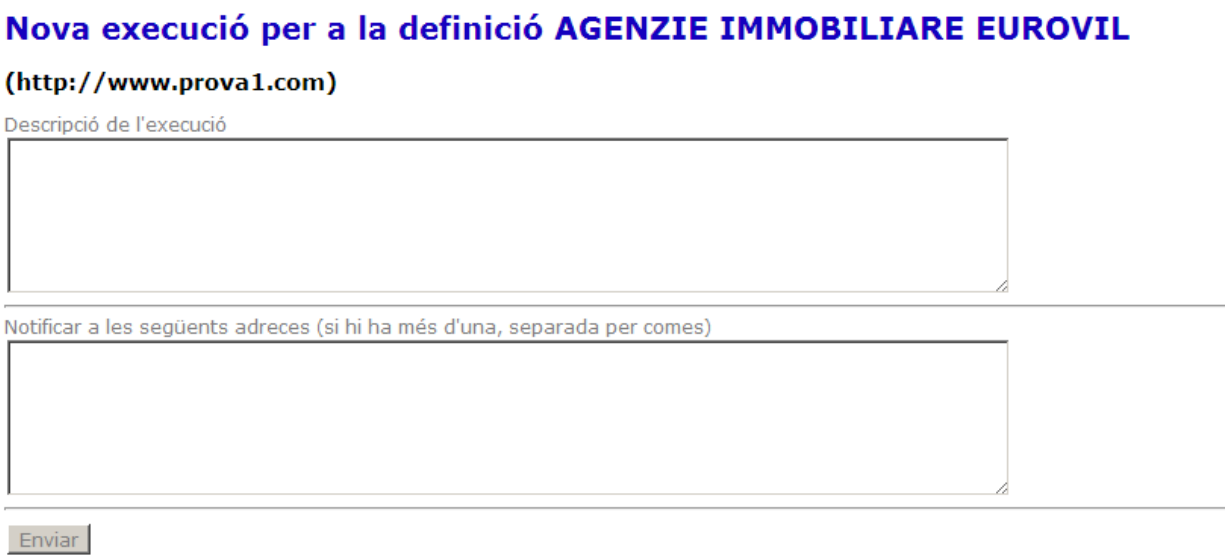
Les columnes de cada fila són:

- Descripció. Nom identificatiu que s'ha proporcionat a la definició
- URL. Adreça inicial d'accés per a realitzar els test.
- Execucions. Número d'execucions efectuades per a aquella definició
- Accions. Llistat d'accions disponibles per a cada definició. Aquestes són
    - Executar. Execució d'un test a partir de la definició.
    - Editar. Modificació de la definició
    - Eliminar. Esborrat de la definició i de les seves execucions
    - Execucions. Consulta i gestió de les execucions de la definició.

A continuació es detalla cadascuna d'aquestes accions:

5.1.4.1 Executar

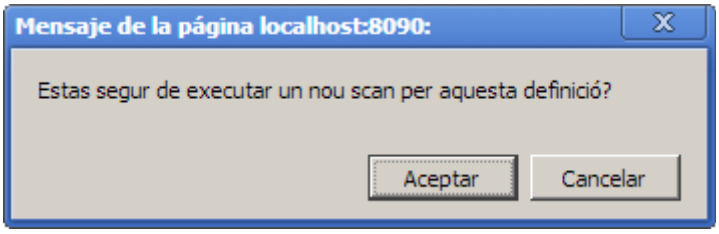
En fer click sobre el botó *Executar*, aquesta acció porta a un nou formulari:



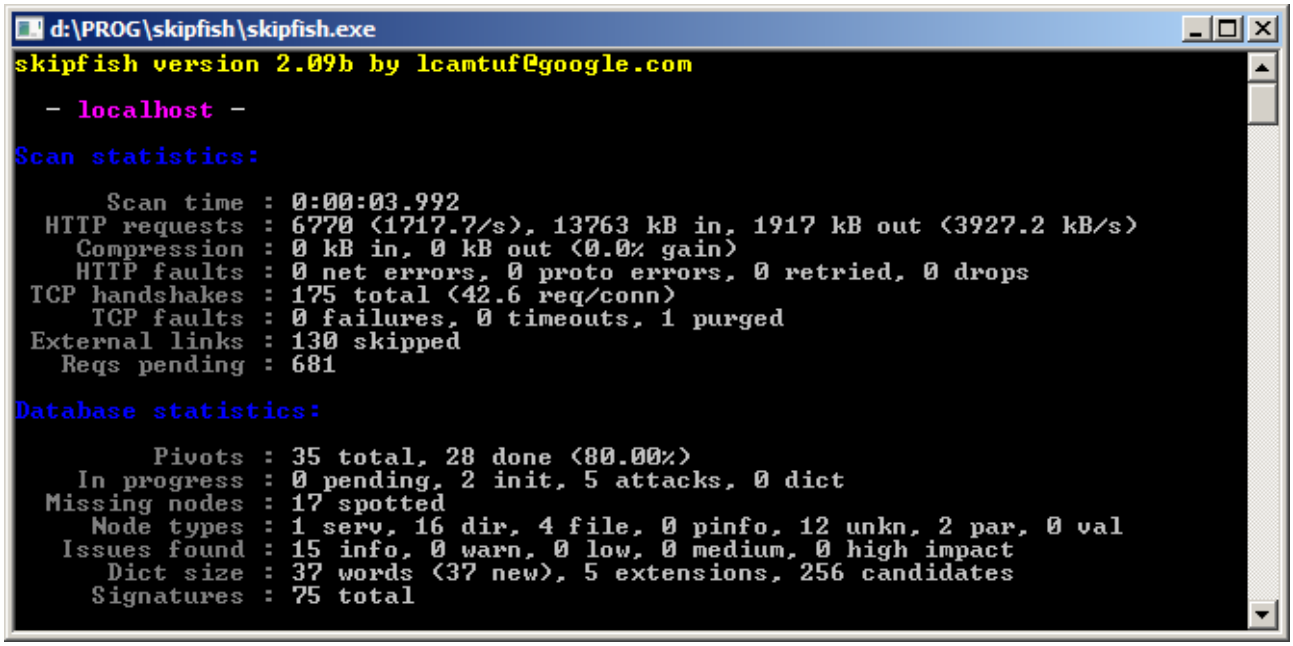
En primer terme es presenta el títol del formulari: *Nova execució per a la definició XXXXX*, on *XXXXX* és el nom identificatiu que s'ha donat a la definició. A continuació, per informar de quin lloc web és el que s'està comprovant, es mostra la URL d'inici.

L'usuari pot introduir opcionalment una descripció de l'execució per poder-la identificar i una sèrie d'adreces de correu electrònic per notificar la finalització de l'execució.

En prémer el botó *Enviar*, es procedeix a l'execució del test, després de demanar confirmació:



A partir de les dades de la pròpia definició, es construeix la línia de comandes que s'executarà. Aquesta línia de comandes s'executarà en un thread independent de l'aplicació en una finestra (cmd.exe en Windows, una *shell* en sistemes *nix):



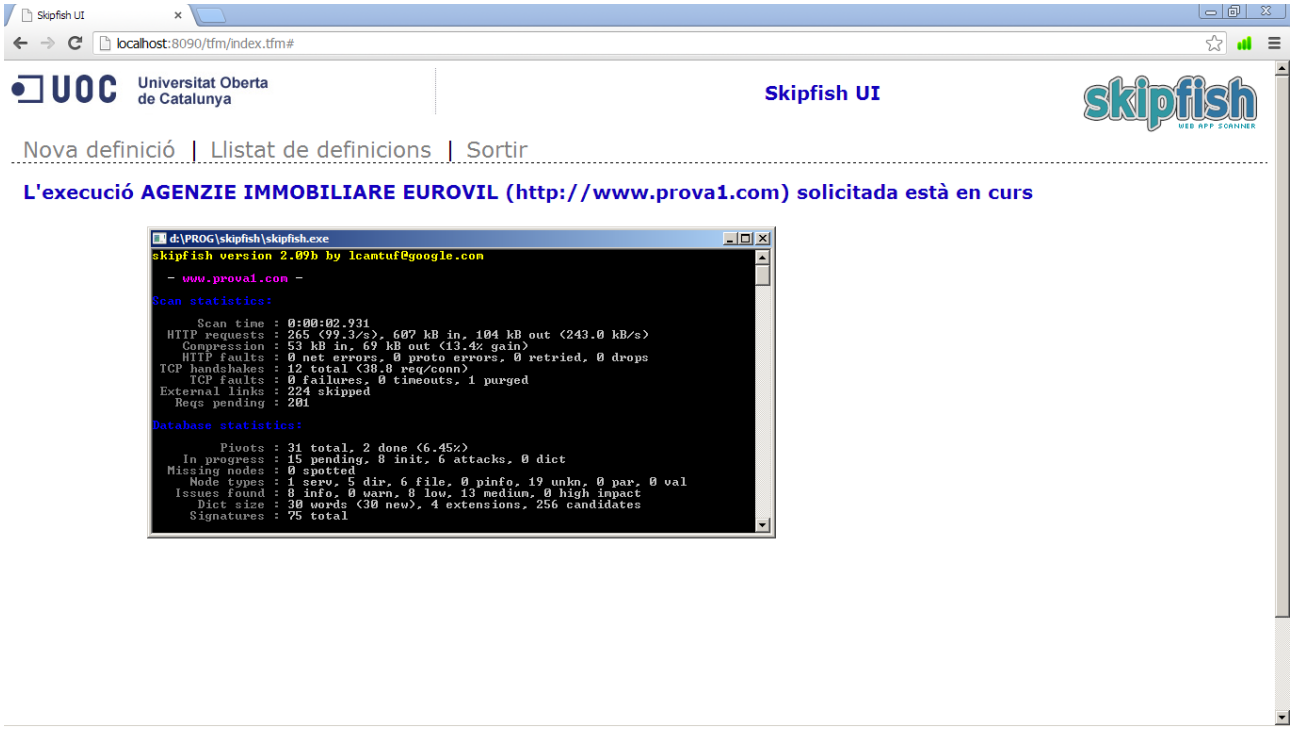
Aquí es pot veure que el que s'està executant és *Skipfish*. Un paràmetre que no apareixia en el manteniment d'una definició és el directori de destí. En els detalls tècnics d'implementació d'aquesta aplicació es podrà veure que hi ha un paràmetre de l'aplicació que és el directori base (*path*). Cada execució disposarà d'un identificador únic (un UUID) que, juntament al directori base formarà el nom de la carpeta de destí de l'execució.

Cada execució quedarà emmagatzemada en una taula de base de dades, amb les següents columnes:

- ID – Identificador únic (UUID) de l'execució
- SCAN_ID – Identificador de la definició a la que pertany
- STARTED – Data/Hora d'inici de l'execució
- FINISHED – Data/Hora de finalització de l'execució

- RESULT – Return code de la línia de comandes després d'executar Skipfish
- DESCRIPTION – Descripció donada per l'usuari
- EMAILS – Llista d'adreces de correu electrònic a enviar notificació de la finalització de l'execució.

En llençar l'execució, es presenta una pàgina informant que l'execució s'ha enviat sense problemes:



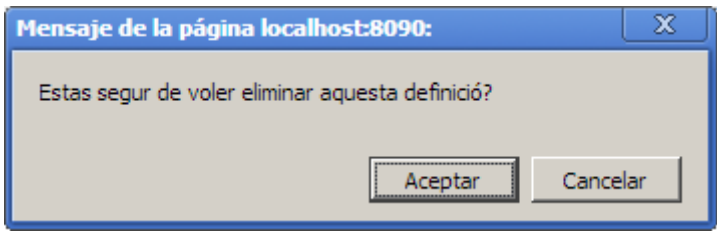
5.1.4.2 Editar

En fer click sobre el botó *Executar*, aquesta acció porta al mateix formulari que per l'alta, però amb els camps informats amb les dades de la definició i amb un títol diferent (*Edició de la definició XXXX*, on *XXXX* és la descripció donada a la definició).

El funcionament i les validacions són les mateixes que per l'alta.

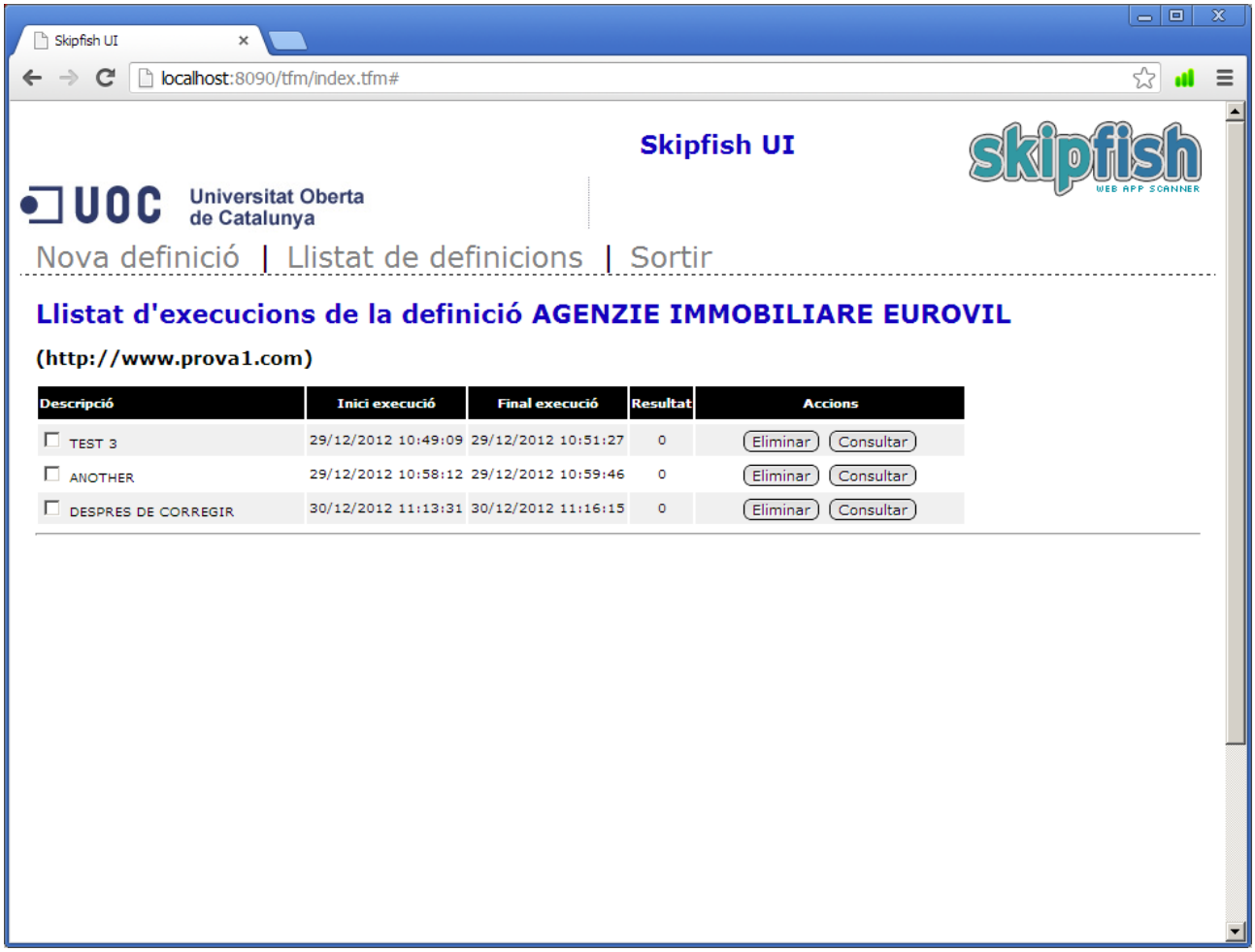
5.1.4.3 Eliminar

En fer click sobre el botó *Eliminar*, aquesta acció demana confirmació per a l'esborrat de la definició triada i les seves execucions:



Si es prem *Acceptar*, es procedeix a l'esborrat i si es prem *Cancel·lar*, es cancel·la l'acció. En ambdós casos, es torna a presentar la llista de definicions.Execucions

En fer *click* sobre el botó *Execucions*, aquesta acció porta a una pàgina amb les execucions de la definició triada:



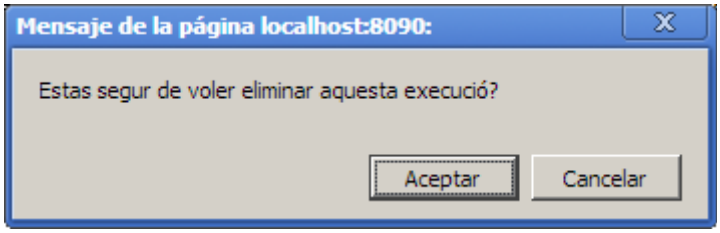
Aquesta pàgina mostra una llista amb les execucions, per a cada fila es mostren les següents columnes:

- Descripció. Nom donat per l'usuari a l'hora de demanar l'execució del test. Va acompanyada d'un *checkbox* que serveix per poder triar l'execució (juntament amb una altra) per a la seva comparació.
- Inici execució. Data/hora de la petició d'execució del test.
- Final execució. Data/hora de finalització del test
- Resultat. Return code de l'execució del test.
- Accions. Accions possibles que es poden fer sobre l'execució. A part de les visibles (Eliminar, Consultar), hi ha una tercera que apareix a la part inferior de la llista quan s'han triat 2 (i només 2) execucions. Aquesta acció és la de comparar 2 execucions. A continuació es detallen cadascuna d'aquestes accions.

5.1.5 Opcions de la llista d'execucions

5.1.5.1 Eliminar

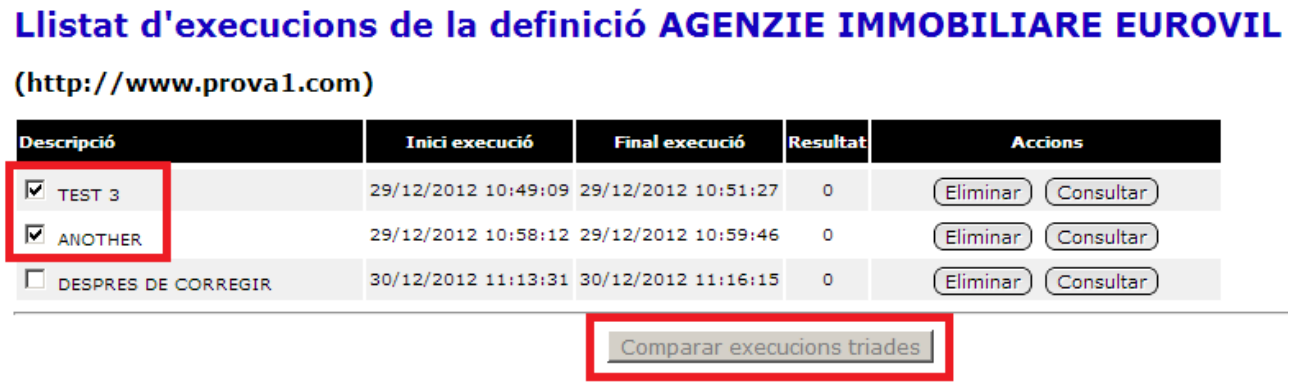
En fer click sobre el botó *Eliminar*, aquesta acció demana confirmació per a l'esborrat de l'execució triada:



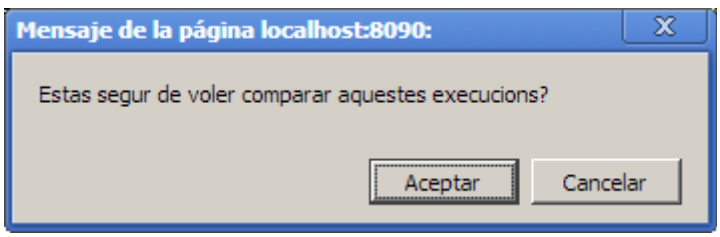
Si es prem *Acceptar*, es procedeix a l'esborrat i si es prem *Cancel·lar*, es cancel·la l'acció. En ambdós casos, es torna a presentar la llista d'execucions de la definició en curs.

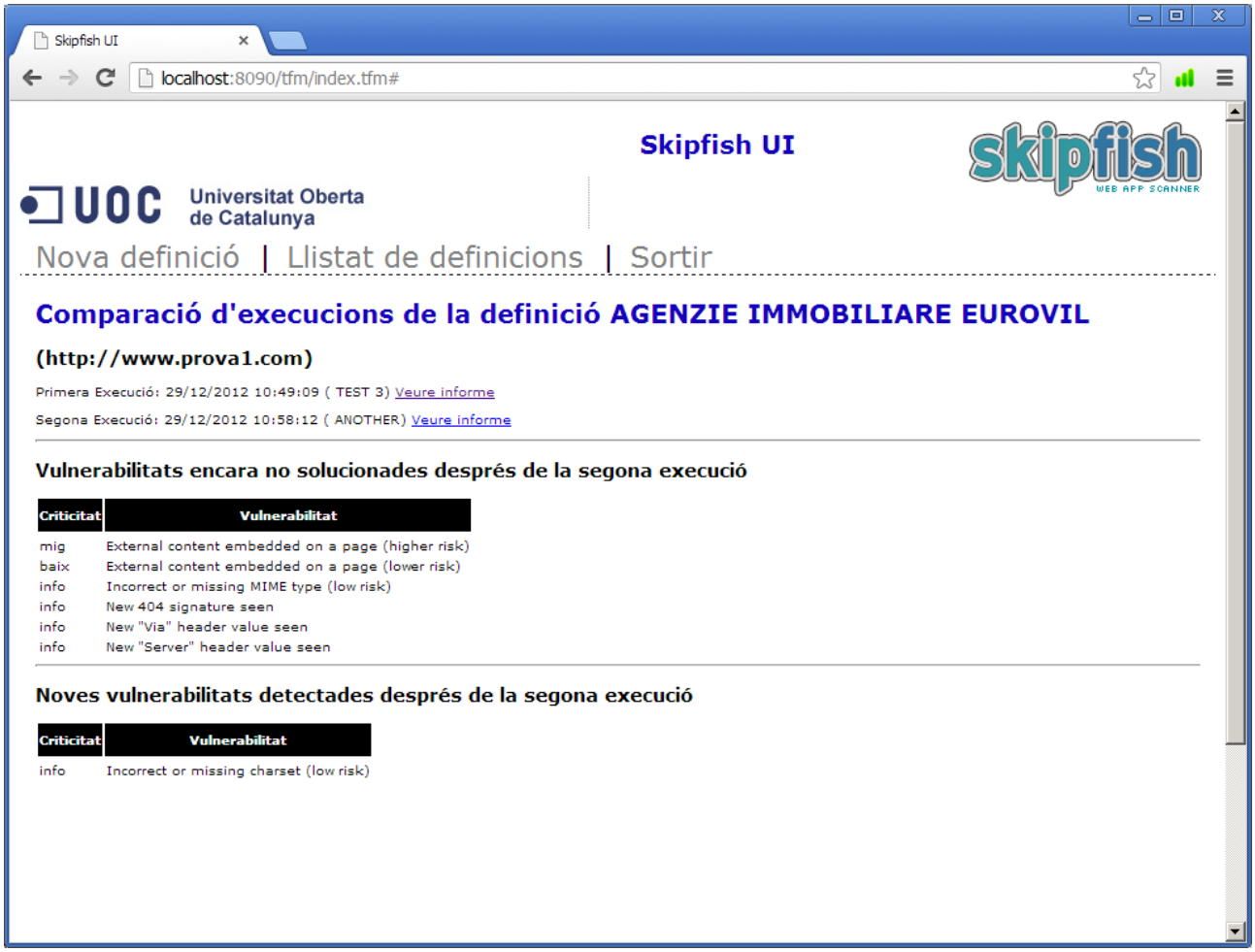
5.1.5.2 Comparar execucions triades

Aquesta opció només apareix quan s'han triat, mitjançant les *checkboxes* que les acompanyen, **dues** execucions:



Al fer *click* sobre el botó *Comparar execucions triades*, es demana confirmació





I en acceptar, es mostra l'informe comparatiu:

Aquest informe no pretén ser molt exhaustiu, només proporcionar de forma ràpida les diferències, en termes de vulnerabilitats, trobades entre dues execucions. Es tracta d'una eina de control de les correccions realitzades al lloc web.

En un primer terme es descriuen les execucions que intervenen. Cadascuna d'elles s'identificaran com a *Primera execució* i *Segona execució*, segons el moment en què es van sol·licitar. A cadascuna d'elles es mostra la data/hora de sol·licitud, la seva descripció i un *link* al seu informe (que es descriu més endavant).

Després es mostren 2 blocs:

- Vulnerabilitats encara no solucionades després de la segona execució

  Es mostra la llista de vulnerabilitats (la seva criticitat i la seva descripció) que encara no s'han solucionat després d'una segona execució. Es suposa que després de veure l'informe d'una primera execució, s'han dut a terme les mesures necessàries per evitar les vulnerabilitats trobades. Amb una segona execució posterior i la seva comparació es pot veure quantes s'han solucionat. Si es donés el cas de que s'han solucionat totes, sortiria el missatge:

  Després de la segona execució, les vulnerabilitats trobades en la primera s'han solucionat!

- Noves vulnerabilitats detectades després de la segona execució

Es mostra la llista de vulnerabilitats (la seva criticitat i la seva descripció) que han sortit noves després de les possibles accions realitzades per solucionar les vulnerabilitats trobades en la primera execució.

Si es donés el cas de que s'han solucionat totes, sortiria el missatge:
No s'han detectat vulnerabilitats noves després de la segona execució

5.1.5.3 Consulta (Informe)

L'informe final elaborat per Skipfish es limita a enumerar, de forma agrupada, les vulnerabilitats trobades. S'ha proposat un informe, a part d'enumerar aquestes vulnerabilitats, proporcioni més informació a l'usuari. Informació com ara descripció de la vulnerabilitat i/o suggeriments per a la resolució del problema.

Un informe d'exemple, consta de les següents seccions:

- Títol, indicant la descripció de l'execució i la URL d'inici:

Informe de l'execució de la definició Scan WebGoat ()

(http://localhost/WebGoat/attack)

- Resum. Hora d'inici i final, més el número de vulnerabilitats trobades per tipus:

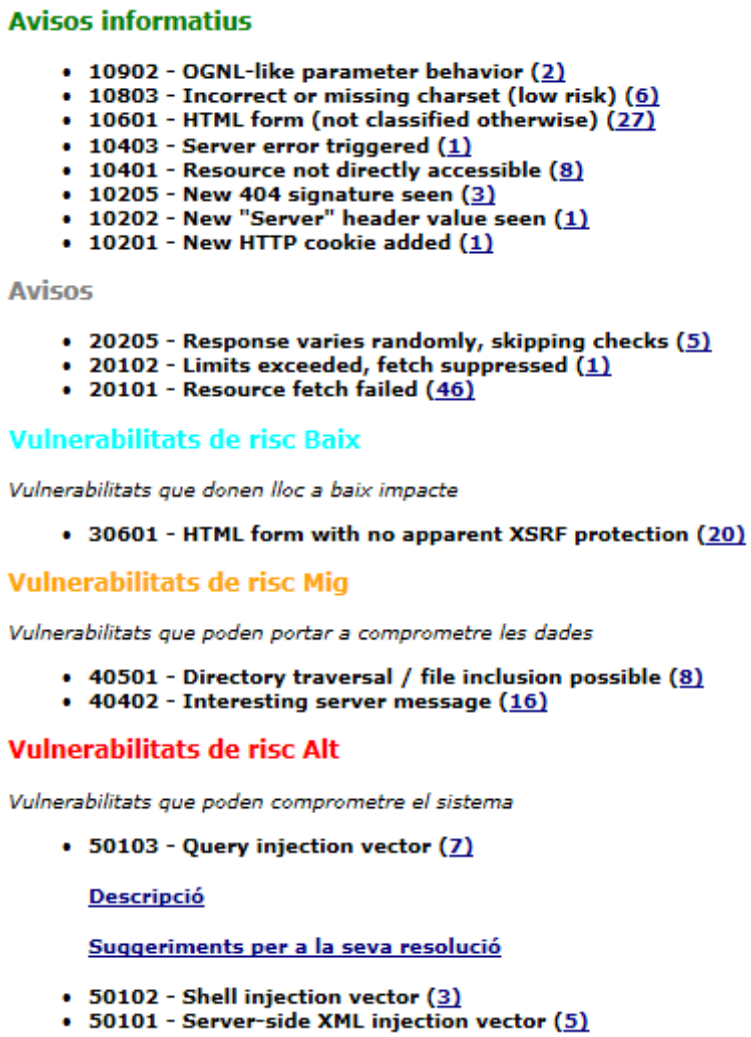
Resum de vulnerabilitats

Data d'inici: 27/12/2012 20:40:23

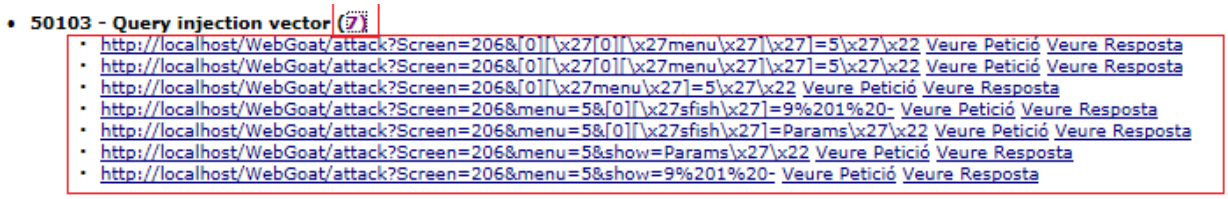
Data de finalització: 27/12/2012 21:09:35

- **Avisos informatius: 8**
- **Avisos: 3**
- **Severitat baixa: 1**
- **Severitat mitjana: 2**
- **Severitat alta: 3**

- Detall de les vulnerabilitats. Es mostren tants blocs com agrupacions de severitat es poden trobar (Avisos informatius, avisos, severitat baixa, severitat mitjana, severitat alta). Per cada bloc es pot especificar una petita descripció de la severitat que representa.

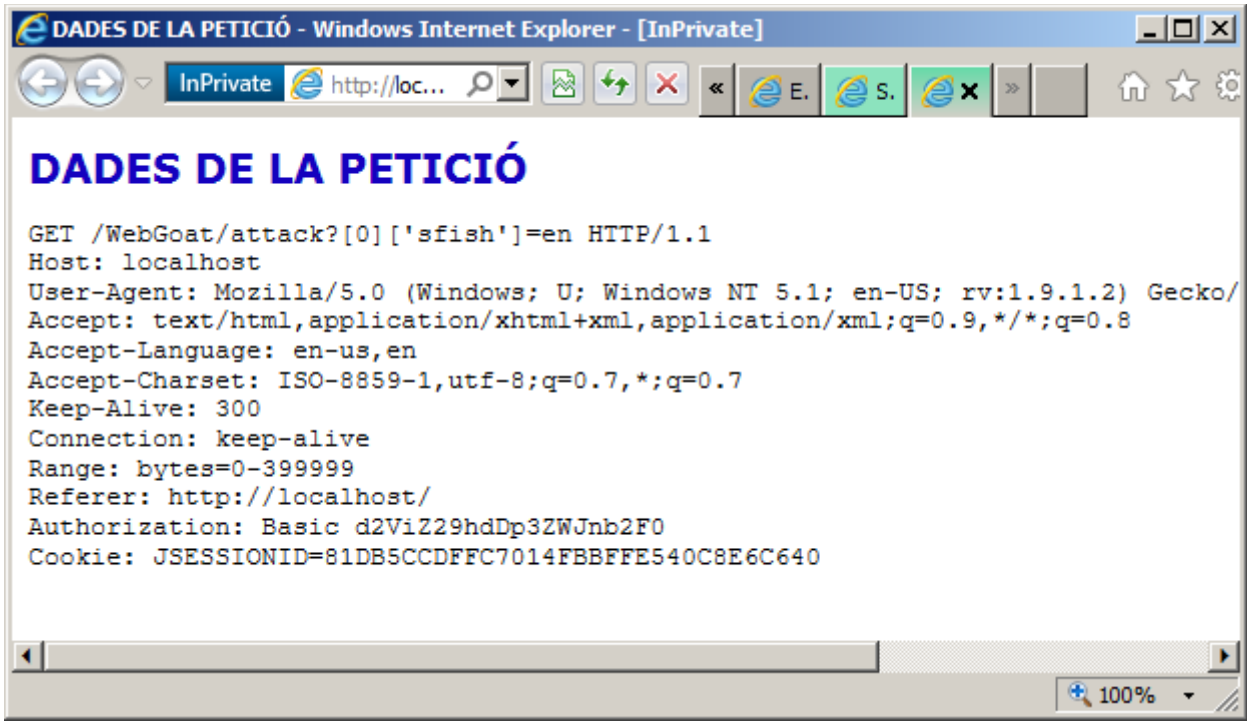


Cada vulnerabilitat està acompanyada d'un comptador. Aquest comptador és un *link* que permet plegar i desplegar les adreces on s'han trobat la vulnerabilitat descrita:



A la seva vegada, cada URL mostrada té un link per veure les dades de la petició i la seva resposta:

En fer *click* sobre aquests links, s'obre una finestra amb les dades de la petició o resposta:

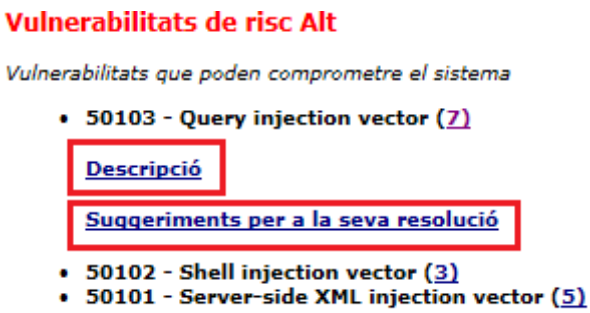

DADES DE LA PETICIÓ - Windows Internet Explorer - [InPrivate]
InPrivate
http://loc...
DADES DE LA PETICIÓ
GET /WebGoat/attack?[0]['sfish']=en HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.2) Gecko/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Range: bytes=0-399999
Referer: http://localhost/
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Cookie: JSESSIONID=81DB5CCDFFC7014FBBFFE540C8E6C640
100%


DADES DE LA RESPOSTA - Windows Internet Explorer - [InPrivate]
InPrivate
http://loc...
DADES DE LA RESPOSTA
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: private
Expires: Thu, 01 Jan 1970 01:00:00 CET
Content-Type: text/html;charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Thu, 27 Dec 2012 20:02:40 GMT
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
100%

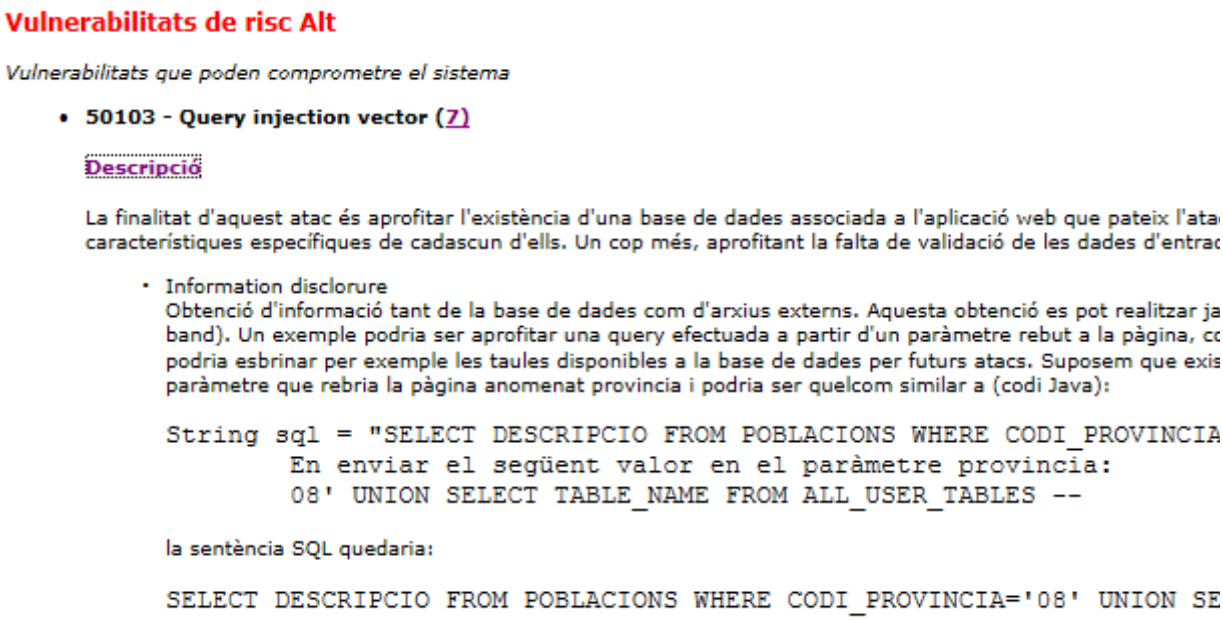
La característica diferenciadora d'aquest informe respecte al que ofereix *Skipfish* és la possibilitat d'afegir informació addicional sobre cada vulnerabilitat.

Cada vulnerabilitat té associat un codi, una descripció, informació en format HTML que descriu la vulnerabilitat i informació en format HTML que proposa solucions sobre la vulnerabilitat.

Aquesta informació està emmagatzemada en una base de dades, amb lo que es proporciona molta flexibilitat per dotar de més informació addicional als informes. Tant el codi com la descripció són obligatoris, però els altres dos atributs no. En cas de que aparegui una vulnerabilitat i tingui algun dels dos tipus d'informació, s'afegeix un link per a cada tipus acompanyant a la vulnerabilitat:



En aquesta imatge es pot veure que la vulnerabilitat *50103 – Query injection vector* està acompanyat per 2 *links*: *Descripció* i *Suggeriments per a la seva resolució*. Això és perquè en la fila corresponent a aquesta vulnerabilitat, les columnes relatives a aquestes dades estan informades. En fer *click* sobre aquests *links* es desplega a sota la informació que porten relacionada:



5.2 Implementació

5.2.1 Infraestructura utilitzada

Per desenvolupar una aplicació Web com és el cas que ocupa aquest treball, és necessari utilitzar una tecnologia que permeti la creació dinàmica de pàgines HTML. Actualment, existeixen diferents tecnologies que faciliten aquesta característica, entre les que trobem:

- CGIs escrits en algun llenguatge d'*scripting* com ara Perl o compilat com C/C++.
- Sistemes que interpreten un llenguatge que es permet incloure dintre del contingut d'una pàgina HTML, com seria PHP o l'antic ASP.
- Tecnologia propietària de Microsoft (.Net).
- Tecnologia Java (JEE, Java Enterprise Edition).

L'opció triada per a aquest projecte ha estat la última pels següents motius:

- Java és un llenguatge molt estès que, a diferència de les tecnologies Microsoft, no està lligat a cap plataforma concreta. Es poden executar aplicacions Java des d'un dispositiu mòbil fins a un Mainframe.
- Les eines disponibles per al desenvolupament Java són molt nombroses. També és cert que per altres casos, com el PHP, també hi ha una col·lecció notable d'eines i *frameworks.*
- Es la tecnologia en la que l'autor té més experiència professional.

Així doncs, havent triat l'opció JEE com a tecnologia de desenvolupament, s'ha procedit a buscar les eines necessàries per a dur-lo a terme. Cap de les eines triades tenen cap dependència del Sistema Operatiu. En el cas d'aquest TFM s'ha utilitzat Windows 7, però es podria haver desenvolupat amb les mateixes eines amb Linux, per exemple.

Els elements utilitzats han estat els següents:

- Contenidor WEB

  Atès que es tracta d'una aplicació Web que no requereix d'altres elements fora del propi contenidor Web, com ara contenidor d'EJBs, Apache Tomcat (versió 7.0.29[15]) era una opció suficientment vàlida.

  Apache Tomcat és una implementació *open source* de les tecnologies *Java Servlet* i *Java Servlet Pages*. Les especificacions d'aquestes tecnologies són desenvolupades sota el *Java Community Process*[19] (http://jcp.org/en/introduction/overview).

  Per a aquest projecte, s'ha utilitzat l'especificació Servlet 2.5 i la versió de Java 1.6 (http://www.oracle.com/technetwork/java/index.html).

- Sistema de Gestió de Base de Dades

  Qualsevol SGBD que suporti SQL i tipus de dades LOB (Large Objects), a més de disposar

d'un *driver* escrit en llenguatge Java, seria vàlid. De fet, no hi ha cap instrucció *propietària* fora de les instruccions de l'SQL estàndard que s'hagi utilitzat.

N'hi ha de diferents SGBD d'ús gratuït com PostgreSQL, Oracle Express Edition o MySQL. En aquest cas, s'ha utilitzat Oracle Express Edition (XE, versió 11g[16])

- Framework de desenvolupament

  Avui en dia, i gràcies als diferents *frameworks* de desenvolupament existents, no s'acostuma a desenvolupar una aplicació des de zero, sense cap eina. Aquests *frameworks* acostumen a implementar patrons de disseny coneguts que faciliten la feina, com ara el *Model View Controller*. La triada per a aquest projecte és Spring 3.0.

  Spring[14] és un *framework* escrit en Java que proporciona diferents mòduls. El que s'ha utilitzat per a aquest projecte és el pròpiament anomenat Spring Framework (Core) que incorpora:

  - Sistema de *Injecció de Dependències* (Dependency Injection) flexible basat tant en arxius XML i/o anotacions java.
  - Suport avançat per a Programació Orientada a Aspectes (*Aspect Oriented Programming*)
  - Suport per transaccions declaratives, sistemes de cau (*cache*) declaratius, validacions declaratives i sistemes de formateig declaratius.
  - Abstraccions per a l'ús de especificacions comunes de JEE com JDBC, JPA, JTA i JMS.
  - Suport per a l'ús d'altres *frameworks* que poden ser complementaris com ara Hibernate i Quartz.
  - Un *framework* molt flexible que implementa el patró MVC
  - Grans facilitats per al desenvolupament de tests.

- IDE (Integrated Development Environment)

  L'IDE utilitzat és Eclipse Java EE IDE for Web Developers, version Juno Release[18]).

- Altres llibreries de suport/utilitats. S'ha utilitzat una sèrie de llibreries d'Apache Commons (http://commons.apache.org) com per exemple Commons Lang i una llibreria, Jackson JSON Processor (http://jackson.codehaus.org), per a la serialització/desserialització de Beans Java a/en JSON (JavaScript Object Notation).

5.2.2 Model de Dades

El model de dades d'aquesta aplicació és molt senzill. Només consta de 3 taules:

- ISSUES. Es tracta d'una taula de suport que s'utilitza per a l'elaboració dels informes. Es tracta d'una taula *mestra* força estàtica. Les columnes d'aquesta taula són:
  - ID. Numèric. Codi únic donat a una vulnerabilitat. És el codi amb el què treballa *Skipfish*.
  - NAME. Alfanumèric. Descripció de la vulnerabilitat. Extret de les especificacions de *Skipfish*.
  - INFO. Character Large Object (CLOB). Informació addicional que, d'estar informada, apareix en un *report* d'un test acompanyant a la vulnerabilitat.
  - SUGGESTIONS. Character Large Object (CLOB). Proposta de solució de la vulnerabilitat que, d'estar informada, apareix en un *report* d'un test acompanyant a la vulnerabilitat.

  La sentència DLL de creació d'aquesta taula és la següent:

```
CREATE TABLE issues (
  id          INTEGER       NOT NULL PRIMARY KEY,
  name        VARCHAR2(256) DEFAULT '' NULL,
  info        CLOB          NULL,
  suggestions CLOB          NULL
)
```

- SCAN. Taula de definicions de tests. Les columnes d'aquesta taula són:
  - ID. Alfanumèric. Codi únic donat a la definició. Es tracta d'un UUID (creat amb la mètode `java.util.UUID.randomUUID.toString()`). Gestionat pel sistema.
  - NAME. Alfanumèric. Descripció de la definició. No obligatori però desitjable.
  - CREATION_DATE. Datetime. Data de creació / modificació. Actualitzat pel sistema.
  - URL. Alfanumèric. URL inicial del test. Obligatori.
  - AUTH_USER. Alfanumèric. Usuari de l'aplicació a fer el test si és necessari. Opcional.
  - AUTH_PASS. Alfanumèric. Password de l'aplicació a fer el test si és necessari. Opcional.
  - URL_FORM. Alfanumèric. Formulari d'autenticació de l'usuari/password si existeix. Opcional.

- VERIFY_FORM. Alfanumèric. Formulari de comprovació de sessió vàlida. Opcional
- Com s'ha explicat en la part de descripció funcional de l'aplicació, la resta de columnes es correspon a la resta de paràmetres que contempla *Skipfish* i tenen la següent forma:

  PARM*C*_*O*

  On PARM és un prefix, *C* indica si el nom de l'opció és majúscula (U, d'*uppercase*) o minúscula (L, de *lowercase*) i *O* és el nom de l'opció. Per exemple, l'opció

  ```
  -Q              - completely suppress duplicate nodes in reports
  ```

  es correspondria a la columna PARMU_Q, mentre que l'opció

  ```
  -u              - be quiet, disable realtime progress stats
  ```

  es correspondria a la columna PARML_U.

La sentència DDL de creació d'aquesta taula és la següent:

```
CREATE TABLE scan (
  id            VARCHAR2(64)  NOT NULL,
  name          VARCHAR2(128) NOT NULL,
  creation_date DATE          DEFAULT SYSDATE NOT NULL,
  url           VARCHAR2(256) NOT NULL,
  auth_user     VARCHAR2(32)  NULL,
  auth_pass     VARCHAR2(32)  NULL,
  url_form      VARCHAR2(256) NULL,
  verify_form   VARCHAR2(256) NULL,
  parml_b       CHAR(1)       NULL,
  parml_c       INTEGER       NULL,
  parml_d       INTEGER       NULL,
  parml_e       CHAR(1)       NULL,
  parml_f       INTEGER       NULL,
  parml_g       INTEGER       NULL,
  parml_i       INTEGER       NULL,
  parml_k       VARCHAR2(10)  NULL,
  parml_l       INTEGER       NULL,
  parml_m       INTEGER       NULL,
  parml_p       INTEGER       NULL,
  parml_q       VARCHAR2(32)  NULL,
```

```
  parml_r       INTEGER       NULL,
  parml_s       INTEGER       NULL,
  parml_t       INTEGER       NULL,
  parml_w       INTEGER       NULL,
  parml_x       INTEGER       NULL,
  parml_z       VARCHAR2(256) NULL,
  parmu_b       VARCHAR2(256) NULL,
  parmu_c       VARCHAR2(256) NULL,
  parmu_d       VARCHAR2(256) NULL,
  parmu_e       CHAR(1)       NULL,
  parmu_f       VARCHAR2(256) NULL,
  parmu_g       INTEGER       NULL,
  parmu_h       VARCHAR2(256) NULL,
  parmu_i       VARCHAR2(256) NULL,
  parmu_k       VARCHAR2(256) NULL,
  parmu_l       CHAR(1)       NULL,
  parmu_m       CHAR(1)       NULL,
  parmu_n       CHAR(1)       NULL,
  parmu_o       CHAR(1)       NULL,
  parmu_p       CHAR(1)       NULL,
  parmu_q       CHAR(1)       NULL,
  parmu_r       INTEGER       NULL,
  parmu_s       VARCHAR2(256) NULL,
  parmu_t       VARCHAR2(256) NULL,
  parmu_u       CHAR(1)       NULL,
  parmu_w       VARCHAR2(256) NULL,
  parmu_x       VARCHAR2(256) NULL,
  parmu_y       CHAR(1)       NULL,
  parmu_z       CHAR(1)       NULL
)
```

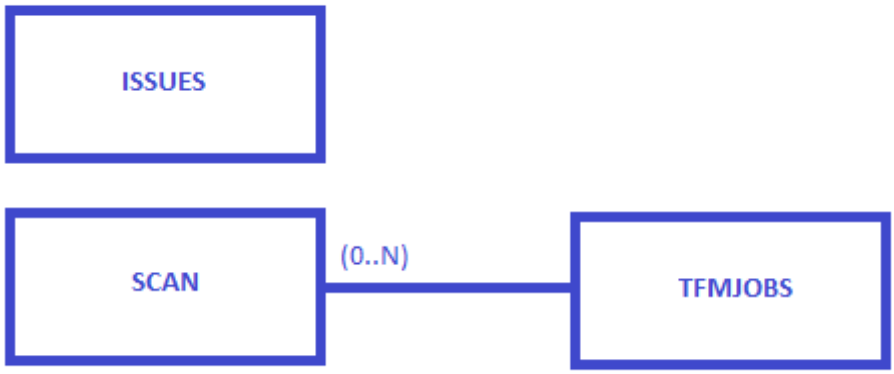
- TFMJOBS. Taula d'execucions de tests. Les columnes d'aquesta taula són:
  - ID. Alfanumèric. Codi únic donat a la definició. Es tracta d'un UUID (creat amb la mètode `java.util.UUID.randomUUID.toString()`). Gestionat pel sistema.
  - SCAN_ID. Alfanumèric. Codi (UUID) de la definició a la que pertany l'execució, relacionat amb el camp ID de la taula SCAN.
  - STARTED. Datetime. Data d'inici del test.

- FINISHED. Datetime. Data de finalització del test.
- RESULT. Numèric. Return code de l'execució d'Skipfish.
- DESCRIPTION. Alfanumèric. Descripció identificativa de l'execució donada per l'usuari. Opcional.
- EMAILS. Alfanumèric. Llista d'adreces d'email per notificar la finalització de l'execució del test. Opcional.

La sentència DDL de creació d'aquesta taula és la següent:

```
CREATE TABLE tfmjobs (
  id          VARCHAR2(64)   NOT NULL PRIMARY KEY,
  scan_id     VARCHAR2(64)   NOT NULL,
  started     DATE           NULL,
  finished    DATE           NULL,
  result      INTEGER        NULL,
  description VARCHAR2(1024) DEFAULT '' NULL,
  emails      VARCHAR2(1024) DEFAULT '' NULL
)
```

El diagrama entitat-relació és el següent:



5.2.3 Definició de l'aplicació Web

En tractar-se d'una mòdul Web JEE, aquest té una estructura prefixada de directoris. Ha d'existir un directori `WEB-INF`, en el que ha d'existir, com a mínim, un descriptor anomenat `web.xml` on hi ha la informació necessària per tal que el contenidor (en aquest cas Apache Tomcat[15]) sàpiga com desplegar i tractar l'aplicació. En aquest directori, a més, poden existir dos subdirectoris (`lib` i `classes`) on resideixen les llibreries i les classes pròpies respectivament que utilitzarà l'aplicació.

El descriptor `web.xml` és el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
      id="WebApp_ID" version="2.5">
      <display-name>tfm</display-name>
      <servlet>
            <servlet-name>
                  mistic
            </servlet-name>
            <servlet-class>
                  org.springframework.web.servlet.DispatcherServlet
            </servlet-class>
            <load-on-startup>1</load-on-startup>
      </servlet>

      <servlet-mapping>
            <servlet-name>mistic</servlet-name>
            <url-pattern>*.tfm</url-pattern>
      </servlet-mapping>

      <welcome-file-list>
            <welcome-file>inici.tfm</welcome-file>
      </welcome-file-list>

      <resource-ref>
            <description>Oracle Datasource</description>
            <res-ref-name>jdbc/tfm</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Container</res-auth>
      </resource-ref>
</web-app>
```

Es tracta d'un descriptor molt senzill on s'ha de destacar principalment:

- La definició d'un servlet, anomenat `mistic`, i que representa una classe de Spring 3 que és `org.springframework.web.servlet.DispatcherServlet`. Aquest servlet és el centre del *framework* MVC de Spring. El nom donat, mistic, és important perquè depenent d'aquest nom ha d'existir el fitxer de definició del context d'aplicació d'Spring, que ha d'estar en el mateix directori i que rep el nom de `mistic-servlet.xml` (Spring permet altres alternatives, aquesta és la de defecte), que es descriu més endavant.

- La definició d'un *datasource* amb la definició de la connexió a la base de dades Oracle que s'utilitza a l'aplicació. Aquesta definició en Apache Tomcat està definida en l'arxiu de configuració del Tomcat anomenat *server.xml*:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Licensed to the Apache Software Foundation (ASF) under one or more contributor
      license agreements. See the NOTICE file distributed with this work for additional
      information regarding copyright ownership. The ASF licenses this file to
      You under the Apache License, Version 2.0 (the "License"); you may not use
      this file except in compliance with the License. You may obtain a copy of
      the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required
      by applicable law or agreed to in writing, software distributed under the
      License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
      OF ANY KIND, either express or implied. See the License for the specific
      language governing permissions and limitations under the License. --><!-- The
contents of this file will be loaded for each web application -->
<Context>

      <!-- Default set of monitored resources -->
      <WatchedResource>WEB-INF/web.xml</WatchedResource>

      <Resource name="jdbc/tfm" auth="Container" type="javax.sql.DataSource"
            driverClassName="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:@127.0.0.1:1521:xe"
            username="tfm" password="tfmpass" maxActive="20" maxIdle="10" maxWait="-1"
/>


</Context>
```

5.2.4 Configuració Spring

Un dels objectes principals d'Spring és l'anomenat *ApplicationContext*, on de forma declarativa es defineixen tots els objectes gestionats per aquest *framework.* Això proporciona independència al desenvolupador a l'hora de crear objectes. A més si es desenvolupa de forma adequada, basada en interfícies, una mateixa aplicació pot funcionar sense canvis estructurals en el seu codi però amb *implementacions* diferents. Per exemple, el mecanisme *natural* proporcionat pel llenguatge Java per crear objectes és mitjançant la instrucció `new`. Si es vol crear un nou objecte d'una classe, es pot fer de la següent forma:

```
NomClasse object = new NomClasse();
```

Això lliga molt a que dintre de l'aplicació sempre s'hagin d'utilitzar objectes de la classe `NomClasse`. Una primera aproximació per evitar aquest acoblament seria utilitzar interfícies:

```
public class NomClasseImpl implements NomInterficie {
…
```

I per utilitzar-la, es podria fer de la següent forma:

```
NomInterficie object = new NomClasseImpl();
```

Això ja és un pas endavant del problema del desacoblament Canviant totalment el codi de la classe NomClasseImpl seria suficient per no canviar la resta de l'aplicació.

Tot i això no acaba de ser una opció molt flexible. Potser interessa tenir diferents implementacions (per exemple una per un entorn de Test i una altra per un entorn de Producció) amb lo qual aquest sistema no seria del tot vàlid.

Ara bé, si d'alguna forma declarativa (com per exemple en un arxiu xml com és el cas d'Spring) es pot determinar quina implementació utilitzar, els canvis quedarien reduïts *únicament* al que serien fitxers de configuració.

Spring permet això mitjançant un arxiu xml on es fan les definicions dels *beans* utilitzats a l'aplicació. Si dintre del fitxer de configuració hi ha un *bean* definit, aquest pot representar una classe que implementa una interfície, per exemple

```
<bean name="definitionService"
class="edu.uoc.mistic.tfm.services.impl.DefinitionServiceImpl">
```

En aquest cas s'està definint un objecte (*bean*) al que es dóna un identificador (`definitionService`) d'una classe concreta (`edu.uoc.mistic.tfm.services.impl.DefinitionServiceImpl`).

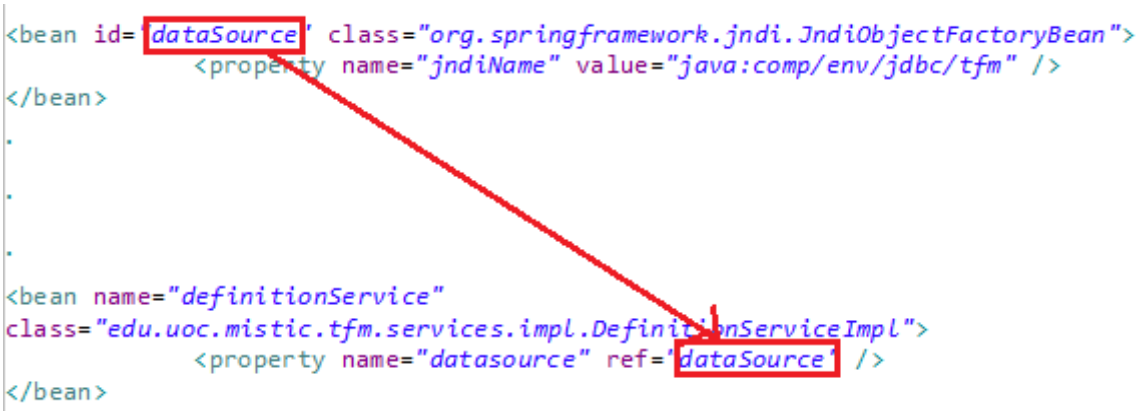
En l'exemple, aquesta classe implementa una interfície anomenada `edu.uoc.mistic.tfm.services.DefinitionService`.

El codi *client* que ha d'utilitzar aquesta classe, s'ha de limitar a *cridar* a Spring mitjançant l'*ApplicationContext* per obtenir una instància:

```
import edu.uoc.mistic.tfm.services.DefinitionService
.
.
DefinitionService service = ApplicationContext.getBean("definitionService",
DefinitionService.class)
```

Si en algun moment es vol canviar la implementació, a part de desenvolupar-la, l'únic canvi a fer seria en el fitxer de definició d'Spring. El codi *client* quedaria igual, sense canvis.

Spring permet, a més, que els *beans* definits puguin ser referenciats entre ells. En l'exemple exposat, el *bean* requereix d'un `javax.sql.DataSource` com a propietat. Aleshores, es defineix un *bean* que sigui el `DataSource` que es necessita i es pot passar com a referència:



Així doncs, la definició d'Spring per a aquesta aplicació és la següent:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">
       <context:component-scan base-package="edu.uoc.mistic.tfm.controller" />
       <bean
class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"
/>
       <bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" /
>
       <bean id="scanpath" class="java.lang.String">
              <constructor-arg value="d:\\temp\\scans\\" />
       </bean>
       <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
              <property name="jndiName" value="java:comp/env/jdbc/tfm" />
       </bean>

       <bean id="scanThread" class="edu.uoc.mistic.tfm.tasks.ScanThread"
              scope="prototype">
              <property name="datasource" ref="dataSource" />
              <property name="path" ref="scanpath" />
              <property name="command" value="d:\\prog\\skipfish\\skipfish.cmd" />
       </bean>

       <bean name="execution" class="edu.uoc.mistic.tfm.controller.ExecutionController">

              <property name="path" ref="scanpath" />
       </bean>

       <bean name="index" class="edu.uoc.mistic.tfm.controller.IndexController">

       </bean>

       <bean name="definitionController"
class="edu.uoc.mistic.tfm.controller.DefinitionController">
```

```
	</bean>

	<bean name="definitionService"
class="edu.uoc.mistic.tfm.services.impl.DefinitionServiceImpl">
		<property name="datasource" ref="dataSource" />
	</bean>

	<bean name="executionService"
class="edu.uoc.mistic.tfm.services.impl.ExecutionServiceImpl">
		<property name="datasource" ref="dataSource" />
		<property name="path" ref="scanpath" />
	</bean>

	<bean id="taskExecutor"
		class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
		<property name="corePoolSize" value="5" />
		<property name="maxPoolSize" value="10" />
		<property name="queueCapacity" value="25" />
	</bean>
</beans>
```

A continuació es descriuen els diferents *beans* en base a la seva funcionalitat:

- Definicions relacionades amb Spring. Spring permet utilitzar *beans* en funció d'unes anotacions especials. Per poder utilitzar aquesta característica, s'inclouen les següents definicions:

  ```
  <context:component-scan base-package="edu.uoc.mistic.tfm.controller" />
  <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping" />
  <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />
  ```

  Això permet que l'*ApplicationContext* consideri *beans* de tipus *Controller* aquelles classes que pertanyin al paquet i als seus fills i que estiguin *marcats* amb l'anotació `@Controller`.

- Elements amb finalitat de configuració específica. Es tracta de *beans* que defineixen aspectes específics de la configuració de l'aplicació:
  - *scanpath*. Indica la carpeta arrel a partir d'on s'emmagatzemaran els tests executats (aquest *path*, juntament amb l'identificador de cada execució, serà el valor que rebrà el paràmetre **`-o`** d'Skipfish.
  - *dataSource*. Es tracta del `javax.sql.DataSource` obtingut via JNDI que serà utilitzat en tota l'aplicació.
- *Thread* d'execució d'Skipfish. Atès que aquest *thread* utilitza els paràmetres *scanpath i dataSource*, es defineix aquí per tal que Spring s'encarregui de la creació de l'objecte i de l'assignació d'aquestes propietats.
- *Controladors*. Són classes d'Spring gestionades pel Servlet definit a l'aplicació. Aquestes classes estan *marcades* amb l'anotació `@Controller`. Els mètodes que es corresponen a accions mapejades directament amb URLs (que són les que finalment són cridades) també estan *marcats* per l'anotació `@RequestMapping`. Aquesta anotació, entre d'altres, té 2 paràmetres: un per assignar la seva URI i una altra (opcional) per assignar el mètode HTTP. En aquesta aplicació només s'ha utilitzat el mètode HTTP GET.

  Tots els controladors hereten de la classe `edu.uoc.mistic.tfm.controller.BaseController`

que és abstracta i a la seva vegada implementa la interfície `org.springframework.context.ApplicationContextAware`. Aquesta interfície requereix implementar el mètode *setApplicationContext(ApplicationContext context)*, cosa que assegura que qualsevol classe que implementi aquesta interfície rep l'ApplicationContext de Spring. Addicionalment la classe abstracta incorpora un *logger* (`org.apache.commons.logging.Log`).

Els controladors definits són els següents:

- *index*. Només serveix per presentar la pàgina d'inici (URI: `/index.tfm`) mitjançant el mètode *index*. No fa més que mostrar la JSP /WEB-INF/jsp/index.jsp. Aquesta és la pàgina inicial de l'aplicació.
- *definitionController*. Aquest *controller* incorpora els mètodes (mapejats amb les seves corresponents URIs) per a la gestió de definicions. Aquests són:
  - *list* (URI: */definitionList.tfm*). Presenta la JSP `/WEB-INF/jsp/definition/definitionList.jsp` amb la llista de definicions disponibles. Prèviament ha calculat la llista que utilitza la JSP.
  - *create* (URI: */definitionCreate.tfm*). Presenta la JSP /WEB-INF/jsp/definition/definitionForm.jsp per a l'alta d'una nova definició.
  - *edit* (URI: */definitionEdit.tfm*). Presenta la JSP /WEB-INF/jsp/definition/definitionForm.jsp per al manteniment d'una definició. Prèviament ha buscat la definició a editar.
  - *delete* (URI: /definitionDelete.tfm). Rep per paràmetre l'identificador de la definició i esborra la definició de la taula. Un cop finalitzada l'operació calcula la llista actual i presenta la JSP /WEB-INF/jsp/definition/definitionList.jsp amb la llista de definicions disponibles.
  - *update* (URI: */definitionUpdate.tfm*). Rep per paràmetre l'identificador de la definició i la resta de dades del formulari i actualitza la definició de la taula. Un cop finalitzada l'operació calcula la llista actual i presenta la JSP /WEB-INF/jsp/definition/definitionList.jsp amb la llista de definicions disponibles.
  - *insert* (URI: */definitionInsert.tfm*). Rep les dades del formulari, calcula un nou identificador i insereix la nova la definició de la taula. Un cop finalitzada l'operació calcula la llista actual i presenta la JSP /WEB-INF/jsp/definition/definitionList.jsp amb la llista de definicions disponibles.
- *executionController*. Aquest *controller* incorpora els mètodes (mapejats amb les seves corresponents URIs) per a la gestió d'execucions. Aquests són:
  - *list* (URI: */executionList.tfm*). Rep l'identificador de la definició i presenta la JSP `/WEB-INF/jsp/execution/executionList.jsp` amb la llista d'execucions disponibles de la definició. Prèviament ha calculat la llista que utilitza la JSP.
  - *executionRequest* (URI: */executionRequest.tfm*). Presenta la JSP /WEB-INF/jsp/execution/executionForm.jsp per a sol·licitar una nova execució. En aquest formulari es recullen la descripció de l'execució i els emails on s'han d'enviar les notificacions.
  - *executionScan* (URI: */executionScan.tfm*). Rep les dades de l'execució a realitzar

(identificador de definició, descripció de l'execució i adreces d'email), llença el *thread* que crida a Skipfish i presenta la JSP /WEB-INF/jsp/execution/executionRequestesd.jsp.

- *executionDelete* (URI: /executionDelete.tfm). Rep per paràmetre l'identificador de la definició i de l'execució i esborra l'execució de la taula. Un cop finalitzada l'operació calcula la llista actual i presenta la JSP /WEB-INF/jsp/execution/executionList.jsp amb la llista d'execucions disponibles.
- *executionReport* (URI: */executionReport.tfm*). Rep per paràmetre l'identificador de la definició i de l'execució, busca les dades d'aquesta (tant a base de dades com a *filesystem*) i presenta la JSP /WEB-INF/jsp/execution/executionReport.jsp amb l'informe de l'execució.
- *executionCompare* (URI: /*executionCompare.tfm*). Rep per paràmetre l'identificador de la definició i de dues execucions, fa la tasca de comparació entre les dues i presenta la JSP /WEB-INF/jsp/execution/executionCompare.jsp amb l'informe de l'execució.
- *viewFile* (URI: /*viewFile.tfm*). Rep per paràmetre l'identificador de la definició, de l'execució i el path de la petició per recuperar o bé la petició o bé la resposta presentant la JSP /WEB-INF/jsp/execution/viewFile.jsp.

- *Serveis*. Són les classes que proporcionen la gestió dels diferents ítems que formen el model de negoci dins una arquitectura MVC. Aquests serveis són utilitats pels *controllers* per efectuar les operacions sobre les definicions i les execucions. De forma general implementen mètodes *CRUD* per al manteniment de les definicions i execucions, entre d'altres mètodes relacionats amb cada entitat.

  Els serveis estan basats en interfícies que indiquen quins mètodes s'han d'implementar. A banda, s'ha de desenvolupar com a mínim una implementació de cada interfície per tal que l'aplicació funcioni correctament.

  Tots els serveis hereten de la classe edu.uoc.mistic.tfm.services.BaseServices que és abstracta i a la seva vegada implementa la interfície org.springframework.context.ApplicationContextAware. Aquesta interfície requereix implementar el mètode *setApplicationContext(ApplicationContext context)*, cosa que assegura que qualsevol classe que implementi aquesta interfície rebi l'ApplicationContext de Spring. Addicionalment la classe abstracta incorpora un *logger* (org.apache.commons.logging.Log) i un objecte org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate, que és una facilitat d'Spring per utilitzar JDBC de forma molt fàcil. Aquest objecte es crea a partir del *DataSource* declarat.

  Els serveis definits són els següents:

  - *definitionService*. Implementació (*edu.uoc.mistic.tfm.services.impl.DefinitionServiceImpl*) de la interfície *edu.uoc.mistic.tfm.services.DefinitionService*. Es tracta del servei que gestiona les definicions. Els mètodes que proporciona aquest servei són:

    - `public DefinitionBean find(String id) throws Exception;`

      Retorna un *bean* de la classe *edu.uoc.mistic.tfm.beans.DefinitionBean* a partir d'un identificador. Si no es troba, retorna *null*. En la implementació el que fa és fer una *query* a la taula SCAN de la base de dades. Si la *query* retorna una fila, es mapegen les columnes amb el *bean*. Aquest mapeig es fa mitjançant la classe *edu.uoc.mistic.tfm.beans.DefinitionRowMapper* que implementa la interfície *org.springframework.jdbc.core.RowMapper*, que és una facilitat que proporcions Spring per realitzar aquestes tasques.

    - `public boolean delete(String id, boolean results) throws Exception;`
      Elimina una definició amb identificador *id*. A més, intenta esborrar també les execucions que pugui tenir la definició si el paràmetre `results` té valor `true`. Si l'operació ha aconseguit eliminar la definició, el mètode torna `true`, en cas contrari false. En la implementació el que fa és fer una instrucció *delete* a la taula SCAN de la base de dades, així com opcionalment una instrucció *delete* a la taula TFM_JOBS on la columna *ID_SCAN* coincideixi amb el valor *id*.
    - `public boolean delete(String id) throws Exception;`
      Es tracta d'una sobre escriptura del mètode anterior, assumint per defecte el valor *true* en el paràmetre `results`. La implementació proporcionada simplement fa la següent crida:

      ```
      return delete(id, true);
      ```

    - `public List<DefinitionBean> list() throws Exception;`
      Retorna una llista (una implementació de *java.util.List*) de *beans* de la classe

edu.uoc.mistic.tfm.beans.DefinitionBean. En la implementació el que fa és fer una *query* a la taula SCAN de la base de dades.

- `public boolean insert(DefinitionBean bean) throws Exception;`
  Insereix una nova fila a partir del contingut del *bean* de la classe *edu.uoc.mistic.tfm.beans.DefinitionBean* que rep per paràmetre. En la implementació el que fa és fer un *insert* a la taula SCAN de la base de dades.
- `public boolean update(DefinitionBean bean) throws Exception;`
  Actualitza una definició a partir del contingut del *bean* de la classe *edu.uoc.mistic.tfm.beans.DefinitionBean* que rep per paràmetre. En la implementació el que fa és fer un *update* a la taula SCAN de la base de dades.

- *executionService*. Implementació (*edu.uoc.mistic.tfm.services.impl.ExecutionServiceImpl*) de la interfície *edu.uoc.mistic.tfm.services.ExecutionService*. Es tracta del servei que gestiona les execucions. Els mètodes que proporciona aquest servei són:

  - `public DefinitionBean find(String definitionId, String id) throws Exception;`

    Retorna un *bean* de la classe *edu.uoc.mistic.tfm.beans.ExecutionBean* a partir d'un identificador de definició i un identificador d'execució. Si no es troba, retorna *null*. En la implementació el que fa és fer una *query* a la taula TFM_JOBS de la base de dades. Si la *query* retorna una fila, es mapegen les columnes amb el *bean*. Aquest mapeig es fa mitjançant la classe *edu.uoc.mistic.tfm.beans.ExecutionRowMapper* que implementa la interfície *org.springframework.jdbc.core.RowMapper*, que és una facilitat que proporcions Spring per realitzar aquestes tasques.

  - `public boolean delete(String id) throws Exception;`
    Elimina una execució amb identificador *id*. Si l'operació ha aconseguit eliminar la definició, el mètode torna `true`, en cas contrari false. En la implementació el que fa és fer una instrucció *delete* a la taula TFM_JOBS de la base de dades.
  - `public List<DefinitionBean> list(String id) throws Exception;`
    Retorna una llista (una implementació de *java.util.List*) de *beans* de la classe *edu.uoc.mistic.tfm.beans.ExecutionBean* d'una definició concreta especificada en el paràmetre *id*. En la implementació el que fa és fer una *query* a la taula TFM_JOBS de la base de dades comparant la columna SCAN_ID amb el paràmetre *id*.
  - `public boolean fillIssueData(List<Issue> issues) throws Exception;`
    Mètode d'utilitat per omplir amb les dades complementàries la llista de vulnerabilitats trobades en una execució. La llista és una implementació de *java.util.List* on cada element és un *bean* de la classe *edu.uoc.mistic.tfm.skipfish.beans.Issue*. La implementació utilitza la taula ISSUES per buscar aquestes dades complementàries.

5.2.5 Altres elements

Dintre de l'aplicació hi ha d'altres elements, no relacionats amb Spring:

- Beans

  Sobre aquests ja s'ha fet esment en la definició dels serveis. Es tracta de classes POJO (Plain Old Java Object) que bàsicament contenen propietats i els corresponents mètodes *getter* i *setter*.

  Els *beans* s'han organitzat en 2 paquets:

  - `edu.uoc.mistic.tfm.beans`

    Conté els beans propis de l'aplicació:

    - `DefinitionBean`, amb la informació relativa a una definició de test.
    - `ExecutionBean`, amb la informació relativa a una execució.

    També conté classes d'utilitat basades en Spring per a fer el mapeig entre base de dades i bean, que implementen la interfície *`org.springframework.jdbc.core.RowMapper`*:

    - `DefinitionRowMapper`.
    - `ExecutionRowmapper`.

  - edu.uoc.mistic.skipfish.beans

    Conté els beans que reconeix *Skipfish* a partir dels objectes JSON que genera:

    - `Issue`, amb la informació relativa a una vulnerabilitat
    - `Mime`, amb la informació relativa als diferents mime/types trobats en l'execució
    - `Report`, es tracta del propi informe generat per *Skipfish*. Bàsicament conté una llista de `Issue` i `Mime`.
    - `Sample`, element que contempla un `Issue o un Mime`.

5.3 Conclusions sobre l'eina

Aquesta eina tal i com està concebuda en aquest estadi es pot considerar una *prova de concepte*. Per convertir-se en una eina *presentable,* ja sigui com a codi lliure, ja sigui comercial, requeriria des del meu punt de vista una sèrie de millores i ampliacions.

S'ha de considerar com un punt de partida que pot fer-se créixer, aprofitant la potència d'*Skipfish.*

Aquestes millores i ampliacions podrien contemplar:

- Finalitzar detalls d'implementació, com ara l'enviament de correus, un control exhaustiu d'errors i excepcions i una gestió de *logs* adient.
- Securització de l'aplicació. Definició d'usuaris i *rols* per a utilitzar l'aplicació, tenint així que autenticar-se per accedir a aquesta.
- Millora en la interfície gràfica.
- Generació d'informes d'acord amb les especificacions funcionals *Software Assurance Tools: Web Application Scanner Functional Specification 1.0*[3] el projecte *SAMATE* (*Software Assurance Metrics and Tools Evaluation*) del NIST (*National Institute of Standards and Technology*. En aquest projecte es fa una primera aproximació com ara informes delta i aportació d'informació i referències addicionals, però faltarien d'altres tipus d'informe, així com diferents formats de sortida (PDF, Excel, ...).
- Sistema de planificació de les execucions. Poder planificar l'execució d'un test a una hora determinada per evitar problemes de contenció en el lloc a examinar (per exemple, a la matinada sense necessitat que ningú en persona tingui que estar present per executar el test).
- Millores en la gestió de les definicions i de les execucions. Poder aplicar filtres de cerca, oferir paginacions,etc...
- Eines per a la gestió de la informació complementària. Poder editar la informació relacionada i els suggeriments de solució per a les vulnerabilitats, amb eines que permetin la introducció WYSIWYG però que el contingut final sigui en format HTML per poder-lo presentar formatejat correctament en els informes.
- Finalment, assegurar-se que aquesta eina no és vulnerable, com a mínim pel que fa a la part de dades d'entrada.

6 BIBLIOGRAFIA / REFERÈNCIES

[1] OWASP, Category: Attack. The Open Web Application Security Project [online]. Disponible: https://www.owasp.org/index.php/Category:Attack

[2] OWASP Testing Guide v3. OWASP Testing Project. The Open Web Application Security Project [online] Disponible: https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

[3] Software Assurance Tools: Web Application Security Scanner Functional Specification, National Institute of Standards and Technology Std., Rev. 1.0. [online]. Disponible: http://samate.nist.gov/docs/webapp_scanner_spec_sp500-269.pdf

[4] Web Application Security Scanner Evaluation Criteria Version 1.0. Web Application Security Consortium 2009. [online]. Disponible: http://projects.webappsec.org/w/page/13246986/Web%20Application%20Security%20Scanner%20Evaluation%20Criteria

[5] Web Application Scanners Accuracy Assessment. Freeware & Open Source Scanner. Secury Tools Benchmarking. Shay-Chen. [online]. Disponible: http://sectooladdict.blogspot.com.es/2010/12/web-application-scanner-benchmark.html

[6] Web Application Scanner RXSS Detection Accuracy Scanner RXSS-GET RXSS-POST RXSS-TOTAL RXSS-FalsePositive. Secury Tools Benchmarking. Shay-Chen. [online]. Disponible: http://wavsep.googlecode.com/files/Web%20Application%20Scanner%20RXSS%20Detection%20Accuracy%20v1.0.pdf

[7] Web Application Scanner SQLi Detection Accuracy. Secury Tools Benchmarking. Shay-Chen. [online]. Disponible:http://wavsep.googlecode.com/files/Web%20Application%20Scanner%20SQLi%20Detection%20Accuracy%20v1.0.pdf

[8] Web Application Scanner SQLi Detection Accuracy Blind Exposure Identification. Secury Tools Benchmarking. Shay-Chen. [online]. Disponible:http://wavsep.googlecode.com/files/Web%20Application%20Scanner%20SQLi%20Detection%20Accuracy%20v1.0.pdf

[9] Web Application Scanner SQLi Detection Accuracy Error-Dependent Exposure Identification. Secury Tools Benchmarking. Shay-Chen. [online]. Disponible: http://wavsep.googlecode.com/files/Web%20Application%20Scanner%20SQLi%20Detection%20Accuracy%20-%20ErrorBased%20v1.0.pdf

[10] State of the Art: Automated Black-Box Web Application Vulnerability Testing. Jason Bau, Elie Bursztein, Divij Gupta, John Mitchell. Stanford University [online]. Disponible: http://theory.stanford.edu/~jcm/papers/pci_oakland10.pdf

[11] A Scale for Crawler Effectiveness on the Client-Side Hidden Web. Víctor M. Prieto, Manuel Álvarez, Rafael López-García, and Fidel Cacheda1. Comunication and Information Technologies Department - University of A Coruña [online]. Disponible: http://www.doiserbia.nb.rs/img/doi/1820-0214/2012/1820-02141200015P.pdf

[12] OWASP Top Ten Project [online] Disponible: https://www.owasp.org/index.php/OWASP_Top_Ten_Project

[13] Skipfish: http://code.google.com/p/skipfish/

[14] Spring Framework: http://www.springsource.org/

[15] Apache Tomcat: http://tomcat.apache.org

[16] Oracle Express Edition: http://www.oracle.com/technetwork/products/express-edition/overview/index.html

[17] Apache Commons: http://tomcat.apache.org

[18] Eclipse IDE Framework: http://www.eclipse.org

[19] Java Community Process: http://jcp.org/en/introduction/overview