Màster interuniversitari de seguretat de les tecnologies de la informació i de les comunicacions

Treball de Fi de Màster

Seguretat en Aplicacions Web

Generation of vulnerabilities and threat reports for web applications

Autor: Joan Caparrós Ramírez

Directors: Dr. Jordi Duch i Gavaldà, Dr. Robert Rallo Moya

RESUM

La quantitat d'informació personal i privada accessible a través d'un navegador d'Internet està augmentant exponencialment cada any, però les mesures de seguretat per protegir l'accés a aquests continguts no se sol considerar com una prioritat per als desenvolupadors web, generalment a causa de la complexitat de la seva identificació . L'objectiu d'aquest projecte és proporcionar un petit producte amb una interfície fàcil d'utilitzar que ajudarà als dissenyadors web i desenvolupadors a identificar els punts febles dels seus llocs web. El producte tindrà dues parts separades, primer es requerirà la identificació de les amenaces de seguretat que utilitzen una combinació d'anàlisi i eines d'exploració que es troben actualment al mercat, i en segon lloc el disseny d'un sistema per recollir els resultats de la identificació i generació d'informes que l'ajudarà a resoldre els problemes trobats.

ABSTRACT

The amount of personal and private information accessible through a web browser is increasing exponentially every year, but the security measures to protect the access to this content are not usually considered as a top priority for the web developers, usually due to the complexity of finding them. The aim of this project is to provide a small product with a user friendly interface that will help web designers and developers identify the weak points of their websites. The product will have two separate parts, first it will require the identification of the security threats using a combination of analysis and exploration tools that are currently in the market, and second the design of a system to collect the results of the identification and generate reports that will help solve the problems found.

Índex

Introducció

Descripció del projecte

El projecte de fi de Màster es presenta com un treball d'aprofundiment del conjunt de vulnerabilitats web que es troben definides en l'OWASP i en especial a les indicades dins del Testing Guide definit per la mateixa organització.

Objectius

L'objectiu final, i donat el llistat classificat de tipus de vulnerabilitats, es procedir a desenvolupar una eina web que permeti crear anàlisis de vulnerabilitats sobre diferents servidors webs i permeti llançar tipus d'anàlisis d'una vulnerabilitat concreta.
Aquest projecte introduirà:

- Un sistema amb una interfície amigable que permeti a algú no expert en seguretat llançar anàlisis i comprendre el seu resultat
- Un historial per tal de veure l'evolució de la seguretat dels servidors analitzats.
- Un panell de control que permeti veure simultàniament diferents màquines.

Planificació

El mètode executat per el desenvolupament del programari sol·licitat ha consistit en dos dos gran fases, cada una d'elles subdividides en les seves etapes:

- **Anàlisis**: Durant la fase d'anàlisis s'ha recaptat informació sobre tots els tipus d'atacs més comuns, així com les eines disponibles en el mercat encarregades d'executar escànners sobre aplicacions web, localitzant aquelles més utilitzades per la comunitat. Skipfish, Arachni, Nikto i Nmap van reunir els requisits sobre els escaneijos els quals l'aplicació resultant hauria de tenir.
  De la mateixa manera s'establirà el calendari que conclourà en data a finals de maig.

	Nom	Duració	Inici	Fi
1	Anàlisis escànners comercials	9d	10/02/2015	20/02/2015
2	Proves i disseny de l'aplicació	10d	23/02/2015	06/03/2015
3	Desenvolupament de la primera versió	10d	09/03/2015	20/03/2015
4	Comprovació dels directors	1d	23/03/2015	23/03/2015
5	Fase de millora 1	6d	24/03/2015	31/03/2015
6	Comprovació dels directors	1d	01/04/2015	01/04/2015
7	Fase de millora 2	21d	02/04/2015	30/04/2015
8	Elaboració de la memòria	70d	23/02/2015	29/05/2015

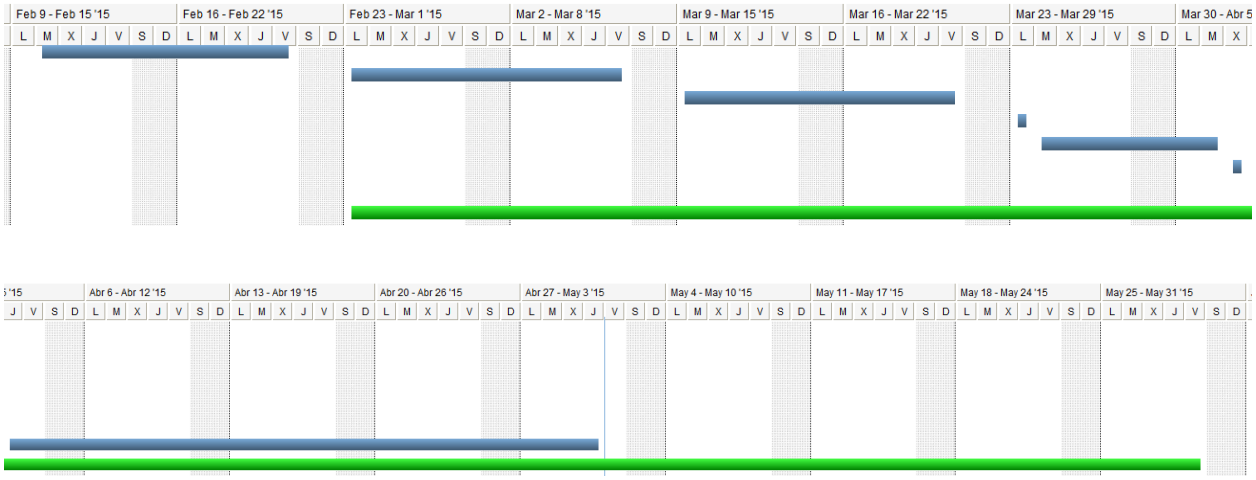
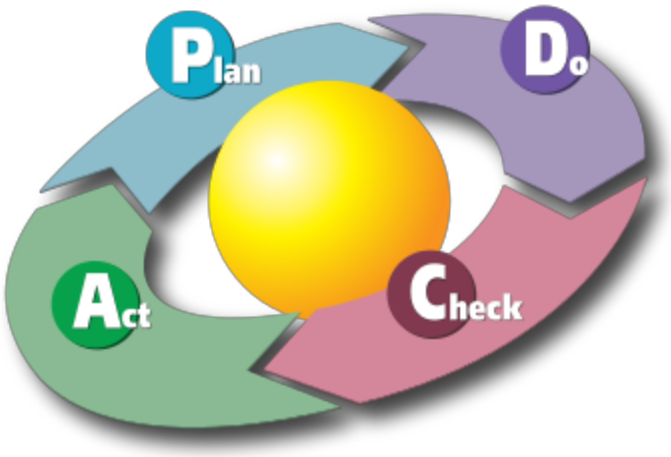


Diagrama de Gantt

- **Proves i disseny de l'aplicació**: De l'estudi de les eines existents es va procedir a provar cada una de les eines, anotant les comandes que caldria utilitzar i les possibilitats de sortida de cada un d'ells, una de les claus del projecte era com emmagatzemar totes aquestes sortides per després ser representades en forma d'informe de vulnerabilitats. Existien dues solucions, desar les sortides en un fitxer i accedir sota petició o emmagatzemar el resultat en la base de dades, es varen provar ambdues solucions i finalment veient que els resultats no excedeixen de la mida recomanable per a un registre estàndard es va triar l'opció de l'emmagatzemament en base de dades. Aquesta solució ens oferiria posteriorment una capa de cerca sobre el resultat si es volgués. L'altre gran decisió sobre la fase de disseny va ser el format en el que s'emmagatzemarien les dades, aquest havia de ser flexible però a la vegada estructurat per a llegir cada un dels resultats en forma d'objecte, entre les solucions més adequades tindríem la sortida en format XML o la que finalment es va utilitzar JSON, cada una amb els seus avantatges i inconvenients però igual d'útils.

Durant aquesta fase també s'establirà un primer esbós de l'interfície gràfica, es definirà el flux d'utilització i el model relacional de dades.

- **Desenvolupament i millora:** La fase de desenvolupament i millora defineix un cicle que s'executarà fins a l'acabament del projecte, així doncs aquesta constarà de quatre etapes: Planificació, Execució, Comprovació i Actuació, seguint el model definit per el cicle de Deming per a la millora continua del producte.



Els estudis i anàlisis inicials entraran en joc d'una manera activa i seran ja escollits en el seu format més adient s'establiran el temps a dedicar en cada una de les fases i realitzarà una previsió dels següents cicles corresponents al cicle de millora continua (Planificació).
Un cop la versió inicial estigui enllestida (Execució), aquesta es sotmetrà a diferents proves (Comprovació) d'on s'hauran d'extreure factors a corregir, canviar o millorar (Actuació), que alimentarà la següent fase del cicle.

Vulnerabilitats en aplicacions web

Les vulnerabilitats en aplicacions web són uns dels factors més importants que posen en greu perill la confidencialitat i integritat de la informació en els nostres servidors. Aquestes vulnerabilitats representen debilitats que poden ser explotades de formes diferents per aconseguir o ve comprometre les dades del servidor, introduint software maliciós per a nosaltres o per a futurs visitants de l'aplicació web o per a anular el servei total o parcial.

Tipus de vulnerabilitats

Per començar a parlar dels tipus de vulnerabilitats que existeixen caldria primer fer una petita menció a una de les entitats més importants en el món de la seguretat informàtica que elabora informes, estudis, eines i cobreixen tots els aspectes que recauen sobre aquest món. OWASP (The Open Web Application Security Project) es va establir com una organització caritativa sense ànim de lucre en els Estats Units el 21 d'abril del 2004, promovent i formant una comunitat oberta dedicada a les organitzacions que permeten concebre, desenvolupar, adquirir, operar i mantenir les aplicacions en que es poden confiar.
Així doncs s'ha de fer referència al llistat que estableixen cada any, mostrant aquelles vulnerabilitats que han representat un major impacte, sobre els quals els professionals de la seguretat web haurien de centrar els seus esforços per mitigar l'efecte de qualsevol atac.

Llistat top 10 OWASP 2013

- **Injection**: Correspon a les injecció de codi, sent les injeccions SQL una de les més comunes.
- **Broken Authentication and Session Management**: Correspon al mal maneig de les sessions en aquelles aplicacions que utilitzen autenticació.
- **Cross-Site Scripting (XSS)**: Ocorre quan hi validació pobre de la informació ingressada per l'atacant.
- **Insecure Direct Object References**: Pot derivar en un accés no autoritzat a informació crítica a causa d'errors en el disseny o desenvolupament.
- **Security Misconfiguration**: Correspon a configuracions no adequades que poden impactar en la seguretat de la pròpia aplicació.
- **Sensitive Data Exposure**: Es refereix a la protecció incorrecta de dades crítics com ara, per exemple, números de targetes de crèdit, contrasenyes, entre d'altres.

- **Missing Function Level Access Control**: Correspon a la manca de controls des del servidor, permetent un possible atacant accedir a funcions a les que no hauria.
- **Cross-Site Request Forgery (CSRF)**: Permet a un atacant generar peticions sobre una aplicació vulnerable a partir de la sessió de la víctima.
- **Using Known Vulnerable Components**: Correspon a l'explotació de llibreries, frameworks i altres components vulnerables per part d'un atacant per tal d'obtenir accés o combinar amb altres atacs.
- **Unvalidated Redirects and Forwards**: Els atacants aprofiten l'ús de redireccions de llocs web a altres llocs utilitzant informació no fiable (untrusted) per redirigir a les víctimes a llocs de pesca o que contenen malware.

Existeixen altres tipus de vulnerabilitats que no es presenten de forma tan freqüent com els anteriorment comentats, moltes d'aquestes vulnerabilitats fan referència a errors de disseny d'aplicacions o d'administració, de les pròpies aplicacions o dels servidors que els allotgen, que poden tenir conseqüències catastròfiques i posar en compromís el sistema davant un atac mal intencionat.

Detecció de vulnerabilitats

Escànners comercials i open source

Existeixen moltes eines en el mercat encarregades d'escanejar aplicacions web cercant vulnerabilitats, aquests incorporen eines automatitzades per simular tot tipus d'entrada i sol·licituds a l'aplicatiu, intentant localitzar els punts febles d'aquests.

Existeixen eines comercials i de codi obert, destinades als diferents sistemes operatius. Durant la fase d'estudi d'aquesta memòria s'han contemplat els següents escànners:

- **Nmap**

Nmap ("Network Mapper") és una eina de codi obert utilitzada per a la detecció de xarxes i per a l'auditoria de seguretat. Nmap utilitza paquets IP per determinar quins serveis estan disponibles dins d'una xarxa, quins sistemes operatius (i versions del sistema operatiu) que s'estan executant, quin tipus de filtres de paquets / tallafocs estan en ús, i dotzenes d'altres característiques. Nmap s'executa en tots els principals sistemes operatius i està disponible per a Linux, Windows i Mac OS X. Tot i disposar actualment d'una interfície gràfica, manté totes les seves eines disponibles per línia de comandes.

Enllaç de descarrega: https://nmap.org/download.html

- **Nikto**

Nikto és una eina de codi obert (GPL) definida com escànner de servidors webs que realitza proves exhaustives sobre diferents vulnerabilitats, incloent cerques sobre més de 6.700 arxius potencialment perillosos, controls de versions no actualitzades de més de 1.250 servidors, problemes específics de la versió sobre més de 270 servidors. També comprova si hi ha elements vulnerables de configuració del servidor, com ara la presència de múltiples arxius d'índexs, opcions de servidor HTTP, etc.

Nikto consta de diferents mòduls que permeten escanejar els hosts cercant vulnerabilitats de tot tipus, un exemple de crida seria el següent

~$ nikto -h <SERVER_IP> -p 80 -t 5 -T 2 -Format xml -o <OUTPUT_FILE>

on

- -p 80 indicaria el port on es vol realitzar l'escànner,
- -t 5 marcaria un timeout de 5 segons i
- -T 2 definiria un tipus concret d'escànner definit pels valors indicats dins de la següent llista:

0 - File Upload
1 - Interesting File / Seen in logs
2 - Misconfiguration / Default File
3 - Information Disclosure
4 - Injection (XSS/Script/HTML)
5 - Remote File Retrieval - Inside Web Root
6 - Denial of Service
7 - Remote File Retrieval - Server Wide
8 - Command Execution / Remote Shell
9 - SQL Injection
a - Authentication Bypass
b - Software Identification
c - Remote Source Inclusion
x - Reverse Tuning Options (i.e., include all except specified)

Enllaç de descarrega: https://github.com/sullo/nikto

- **Arachni**

Arachni és una eina de codi obert desenvolupada per proporcionar un entorn de proves de penetració. Aquesta eina pot detectar diverses vulnerabilitats de seguretat d'aplicacions web, és capaç de detectar diverses vulnerabilitats com injecció de SQL, XSS, la inclusió d'arxius locals, la inclusió d'arxius remots, redirecció no validades, entre moltes altres.

Arachni consta de diferents mòduls que permeten escanejar els hosts cercant vulnerabilitats de tot tipus, un exemple de crida seria el següent:

arachni --audit-forms --audit-links --audit-cookies
--checks=sql_injection_timing,sql_injection_differential,sql_injection --scope-auto-redundant=5
http://server.joancaparros.com/dvwa/vulnerabilities/sqli/ --scope-directory-depth-limit=50
--report-save-path=<fitxer informe>;

on :

- --audit-forms indicaria que volem testejar el correcte funcionament dels formularis de l'aplicació,
- --audit-links es testejaran tots els links de l'aplicació
- --audit-cookies les cookies generades es veuran sotmeses a testos
- --scope-auto-redundant=5 marcaria el nombre de rutes redundants que volem analitzar, és a dir de les següents adreces:

    http://test.com/?stuff=1
    http://test.com/?stuff=2
    http://test.com/?stuff=other-stuff
    http://test.com/?stuff=blah
    http://test.com/?stuff=blah&stuff2=1
    http://test.com/?stuff=blah&stuff2=2
    http://test.com/?stuff=blah2&stuff2=bloo
    http://test.com/path.php?stuff=blah&stuff2=1

    Només s'inclourien dins l'escaneig:

    http://test.com/?stuff=1
    http://test.com/?stuff=2
    http://test.com/?stuff=blah&stuff2=1
    http://test.com/?stuff=blah&stuff2=2
    http://test.com/path.php?stuff=blah&stuff2=1

- --checks=sql_injection_timing,sql_injection_differential indica quin tipus d'escànner en concret s'ha aplicat, aquests valors són configurables dins del llistat definit per l'aplicació, amb els següents valors:

interesting_responses, http_put, directory_listing, xst, origin_spoof_access_restriction_bypass, localstart_asp, captcha, http_only_cookies, unencrypted_password_forms, mixed_resource, password_autocomplete, credit_card, ssn, private_ip, emails, hsts, html_objects, cookie_set_for_parent_domain, cvs_svn_users, form_upload, insecure_cookies, webdav, allowed_methods, htaccess_limit, code_injection, unvalidated_redirect, code_injection_php_input_wrapper, rfi, sql_injection_timing, xss_dom_script_context, xss_event, csrf, os_cmd_injection, path_traversal, xss_dom, xss_path, os_cmd_injection_timing, xpath_injection, no_sql_injection_differential, xss, xss_script_context, source_code_disclosure, session_fixation, response_splitting, trainer, code_injection_timing, ldap_injection, sql_injection, no_sql_injection, xss_tag, file_inclusion, xss_dom_inputs, sql_injection_differential

Enllaç de descarrega: http://www.arachni-scanner.com/

- **Skipfish**

Skipfish és també una eina de codi obert destinada a la seguretat d'aplicacions web. Disposa de sistemes per comprovar dins de cada pàgina diverses amenaces preparant un informe final on contemplar tots els resultats. Aquesta eina va ser escrita en C, aquest fet fa que les seves accions estiguin altament optimitzades i que el seu consum de CPU sigui mínim. Aquesta eina també pretén oferir una alta qualitat i menys falsos positius i està disponible per a Linux, FreeBSD, Mac OS X i Windows.

Un exemple de crida seria el següent:
~$ skipfish -W /dev/null -LV -output2 URL

on se li indicarien diferents paràmetres relacionats amb el diccionari a emprar en el cas d'atacs amb força bruta, arachni genera un directori complet amb una sortida html, en aquest projecte es va considerar la utilització de 2 dels fitxers resultants child_index.js, samples.js que contenien tota la informació mostrada en el report final elaborat per skipfish.

Enllaç de descarrega: http://code.google.com/p/skipfish/

- **Wapiti**

Wapiti és una eina utilitzada per a l'escaneig de vulnerabilitats web molt emprada en les auditories. Realitza proves utilitzant el paradigma de black box no coneixent cap propietat de l'interior de la màquina escanejada. És compatible tant amb protocols GET i POST d'HTTP per a detectar múltiples vulnerabilitats com divulgació de fitxers, inclusió d'arxius, Cross Site Scripting (XSS), detecció Response splitting, XPath Injection configuracions .htaccess febles , revelació de còpies de seguretat entre altres.
Wapiti és una aplicació de línia de comandes.

Un exemple de crida, podria ser el següent:
~$ python wapiti http://test.com -n 10 -b folder -f html -o /tmp/scan_report

on:

- http://test.com serà la url escanejada
- -n 10 representarà el limit de urls escanejades amb el mateix patró
- -b folder indicarà a quina carpeta de la url escanejada es vol realitzar l'escanner
- -f definirà el format, en aquest cas mostrat com a html
- -o indicarà l'output on es desarà el report final

Enllaç de descarrega: http://wapiti.sourceforge.net/

Extracció d'informació del host

La base de l'estudi de qualsevol host començarà primerament amb un estudi de les propietats del propi servidor, en aquesta fase s'intentarà esbrinar el sistema operatiu de la màquina, executar una cerca de dns que coincideixi amb la IP entregada i el seu estat actual (si es troba operatiu o no).

NMAP fingerprint option

L'OS fingerprint és el procés de recopilació d'informació que permet identificar el sistema operatiu de la màquina escanejada. El procés es basa en el fet que cada sistema operatiu respon de forma diferent a una gran varietat de paquets malformats. D'aquesta manera, utilitzant eines que permeten comparar les respostes amb una base de dades amb referències conegudes, és possible identificar quin és el sistema operatiu. Nmap és una de les eines més utilitzades per dur a terme aquesta tasca

Utilització:
~$ sudo nmap -O -sS -oX - màquina_escanejada
o
~$ sudo nmap -p <port> -A -oX màquina_escanejada

La utilització en la nostre aplicació és centrarà en el port 80 del servidor ja que en tot moment tractarem les vulnerabilitats sobre serveis web,
~$ nmap -p 80 -T4 -A <SERVER_IP> -oX <OUTPUT_FILE>

```
<?xml version="1.0" ?>
<?xml-stylesheet href="file:///usr/share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 5.21 scan initiated Sun Apr 19 13:19:56 2015 as: nmap -p 80 -T4 -A -oX server.xml
server.joancaparros.com -->
```

```
<nmaprun scanner="nmap" args="nmap -p 80 -T4 -A -oX server.xml
server.joancaparros.com" start="1429442396" startstr="Sun Apr 19 13:19:56 2015"
version="5.21" xmloutputversion="1.03">
        <scaninfo type="connect"  protocol="tcp" numservices="1" services="80" />
        <verbose level="0" />
        <debugging level="0" />
        <host starttime="1429442396" endtime="1429442402">
                <status state="up" reason="conn-refused"/>
                <address addr="84.89.0.74" addrtype="ipv4" />
                <hostnames>
                        <hostname name="server.joancaparros.com" type="user"/>
                </hostnames>
                <ports>
                        <port protocol="tcp" portid="80">
                        <state state="open" reason="syn-ack" reason_ttl="0"/>
                        <service name="http" product="Apache httpd" version="2.2.22"
extrainfo="(Ubuntu)" method="probed" conf="10" />
                        <script id="robots.txt" output="has 1 disallowed entry &#xa;/" />
                        <script id="html-title" output="Site doesn&apos;t have a title (text/html)."
/>
                </port>
        </ports>
        <times srtt="12803" rttvar="7304" to="100000" />
</host>
<runstats>
        <finished time="1429442402" timestr="Sun Apr 19 13:20:02 2015" elapsed="6.18"/>
        <hosts up="1" down="0" total="1" />
        <!-- Nmap done at Sun Apr 19 13:20:02 2015; 1 IP address (1 host up) scanned in
6.18 seconds -->
</runstats>
</nmaprun>
```

El resultat ens deixa entreveure la IP del servidor estudiat, el tipus de servidor web utilitzat per oferir el web amb la seva versió instalada, i depenent del cas aproximarem quina versió de sistema operatiu està corrent.

dig DNS reverse lookup

Dig és una eina que ve en el paquet BIND (Berkeley Internet Name Domain) que és una implementació de protocols DNS i serveix per diagnosticar problemes amb els DNS. Aquesta eina ens permet consultar un servidor de noms i obtenir informació relacionada amb el domini o el host.

Utilització:
~$ dig IP_maquina_escanejada

Escànner de ports

L'escaneig de ports oberts en un servidor pot donar-nos una idea de quins serveis pot tenir actius, es tracte doncs d'un anàlisis de xarxa on es restrejaran els ports comuns amb la finalitat de detectar possibles vulnetabilitats associades a les aplicacions que estan operant a través d'ells.

NMAP port scanning option

L'eina triada per realitzar l'anàlisis de ports es tracte de l'Nmap, una eina destinada a la exploració completa de xarxes, indicant els següents estats en els que es troben els ports:

- **Oberts** (open): ports que accepten connexions TCP/UDP o SCTP
- **Tancats** (closed): ports en els que tot i ser accessibles no tenen serveis associats en el servidor
- **Filtrats** (filtered): ports protegits per un tallafocs que no poden definir-se com a oberts o tancats.
- **Sense filtrar** (unfiltered): ports accessibles però que no poden definir-se com a oberts o tancats
- **No reconegut**: tot i que els anteriors estats haurien de definir la majoria de casos en els que els ports es troben, algunes vegades no es determinarà en quin estat es troben degut a que no es generarà una resposta valida.

D'aquesta manera també pot definir quins serveis utilitza la màquina escanejada, quin servei operatiu corre o fins i tot informació sobre el hardware associat implicat en el servidor.
Nmap realitza totes aquestes tasques comentades a través de l'enviament de paquets i realitzant

Utilització:
~$: nmap (opcions_escaneig) (màquina_a_escanejar)

Existeixen molts modes d'ús:

- **Escaneig TCP SYN** : Ràpid, no completa connexions TCP per a cada port.
  nmap -sS (màquina_a_escanejar)
- **Escaneig TCP**: Més lent, completa connexions TCP per a cada port.
  ~$: nmap -sT (màquina_a_escanejar)

  Exemple ambdós resultats:

```
Host is up (0.025s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
```

```
25/tcp open  smtp
80/tcp open  http
```

El següent cas UDP donat al seu temps d'execució es restringirà a una segona fase del projecte, ja que penalitzaria el rendiment i ús de l'aplicatiu:

- **Escaneig UDP**: Molt lent, obviat moltes vegades, però necessari per a completar una auditoria de ports completa.
  ~$: nmap -sU (màquina_a_escanejar)

  Exemple resultat:

```
Host is up (0.015s latency).
Not shown: 999 closed ports
PORT   STATE         SERVICE
68/udp open|filtered dhcpc

Nmap done: 1 IP address (1 host up) scanned in 1080.22 seconds
```

En aquest projecte, i per aprofitar tot el potencial de l'eina s'aprofitarà la capacitat de generar la sortida en xml del propi programari, es parsejarà i s'alimentarà la base de dades. On la comanda consistirà en:

~$ nmap -T4 -A -oX - (màquina a escaquejar)
-T4: Indicarà la velocitat (alta)
-A: Extreurà el sistema operatiu i propietats de la màquina escanejada
-oX: Especifica el format de sortida en xml

Donades les diferents possibilitats d'execució, el nostre programa executarà un escànner ràpid i simple, localitzant els serveis que en cada port obert localitzi per tenir una visió global dels serveis actius dins del servidor susceptibles a ser vulnerables.
La comanda consistirà en una crida TCP SYN amb velocitat ràpida i oferint el seu resultat en format xml en un fitxer:

~$ nmap -sS -T4 <SERVER_IP> -oX <OUTPUT_FILE>

Un resultat executat sobre un domini qualsevol tindria la següent forma:

```
<?xml version="1.0" ?>
<?xml-stylesheet href="file:///usr/share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 5.21 scan initiated Sun Apr 19 13:04:50 2015 as: nmap -sS -T4 -oX sjcapa.xml
server.joancaparros.com -->
<nmaprun scanner="nmap" args="nmap -sS -T4 -oX sjcapa.xml server.joancaparros.com"
start="1429441490" startstr="Sun Apr 19 13:04:50 2015" version="5.21"
```

```
xmloutputversion="1.03">
	<scaninfo type="syn"  protocol="tcp" numservices="1000" services="<<listat de ports>>" />
	<verbose level="0" />
	<debugging level="0" />
	<host starttime="1429441490" endtime="1429441490">
		<status state="up" reason="echo-reply"/>
	<address addr="84.89.0.74" addrtype="ipv4" />
	<hostnames>
		<hostname name="server.joancaparros.com" type="user"/>
	</hostnames>
	<ports>
		<extraports state="closed" count="998">
		<extrareasons reason="resets" count="998"/>
	</extraports>
	<port protocol="tcp" portid="25">
		<state state="open" reason="syn-ack" reason_ttl="57"/>
		<service name="smtp" method="table" conf="3" />
	</port>
	<port protocol="tcp" portid="80">
		<state state="open" reason="syn-ack" reason_ttl="57"/>
		<service name="http" method="table" conf="3" />
	</port>
</ports>
<times srtt="8466" rttvar="248" to="100000" />
</host>
<runstats>
	<finished time="1429441490" timestr="Sun Apr 19 13:04:50 2015" elapsed="0.36"/>
	<hosts up="1" down="0" total="1" />
</runstats>
</nmaprun>
```

Indicant que s'han trobat 2 ports oberts, corresponents a un servei web http ofert pel port 80 i un servei de correu smtp en el port 25. Tots aquests resultats donaran una visió global sobre el servidor estudiat i permetrà focalitzar quin tipus d'escanner cal aplicar en cada port.

Interesting Files and Folders

Denotarem com a fitxers o directoris interessants aquells que el servidor pot estar oferint amb o sense intenció prèvia de fer-los públics que podrien donar a conèixer propietats intrínsiques del servidor o sistema que podrien dur a atacs.
Les aplicacions web sovint estan compostes per diverses arxius i directoris.
És possible que amb el pas del temps alguns d'aquests arxius i directoris poden arribar deixar de tenir cap referència dins de l'aplicació, quedant sense ús i completament per oblidats per l'administrador / desenvolupador.

Donat que la majoria d'aplicacions es construeixen usant frameworks comuns, contenen una estructura que permet el descobriment d'aquests arxius independentment del servidor que s'usi.
Durant les etapes inicials de reconeixement d'un atac, els ciberdelinqüents intentaran localitzar arxius sense referències en l'esperança de que l'arxiu ajudi a comprometre l'aplicació web.
Aquest escànner conté un llistat de fitxers i directoris i intentarà trobar aquells fitxers que poden posar en perill la integritat de les dades i del servidor.

Existeixen diferents eines que analitzen el contingut dels servidors localitzant aquelles carpetes o fitxers que poden contenir informació sensible per explotar.

En el nostre projecte s'utilitzarà nikto, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

~$ nikto -h <SERVER_IP> -p 80 -t 5 -T 1 -Format xml -o <OUTPUT_FILE>

Que ens oferirà el resultat en format xml

```
<?xml version="1.0" ?>
<!DOCTYPE niktoscan SYSTEM "/usr/share/doc/nikto/nikto.dtd">
<niktoscan hoststest="0" options="-h server.joancaparros.com -p 80 -t 5 -T 1 -Format xml -o
t5nikto" version="2.1.4" scanstart="Sun Apr 19 15:11:37 2015" scanend="Thu Jan  1 01:00:00
1970" scanelapsed=" seconds" nxmlversion="1.1">

</niktoscan>
<?xml version="1.0" ?>
<!DOCTYPE niktoscan SYSTEM "/usr/share/doc/nikto/nikto.dtd">
<niktoscan hoststest="0" options="-h server.joancaparros.com -p 80 -t 5 -T 1 -Format xml -o
t5nikto" version="2.1.4" scanstart="Sun Apr 19 15:12:09 2015" scanend="Thu Jan  1 01:00:00
1970" scanelapsed=" seconds" nxmlversion="1.1">

</niktoscan>
<?xml version="1.0" ?>
<!DOCTYPE niktoscan SYSTEM "/usr/share/doc/nikto/nikto.dtd">
<niktoscan hoststest="0" options="-h http://server.joancaparros.com -p 80 -t 5 -T 1 -Format
xml -o t5nikto" version="2.1.4" scanstart="Sun Apr 19 15:12:35 2015" scanend="Thu Jan  1
01:00:00 1970" scanelapsed=" seconds" nxmlversion="1.1">
        <scandetails targetip="84.89.0.74" targethostname="server.joancaparros.com"
targetport="80" targetbanner="Apache/2.2.22 (Ubuntu)" starttime="2015-04-20 15:12:35"
sitename="http://server.joancaparros.com:80/" siteip="http://84.89.0.74:80/"
hostheader="server.joancaparros.com">
                <item id="999996" osvdbid="0" osvdblink="0_LINK" method="GET">
                        <description><![CDATA[robots.txt contains 1 entry which should
be manually viewed.]]></description>
```

```
                    <uri><![CDATA[/robots.txt]]></uri>

<namelink><![CDATA[http://server.joancaparros.com:80/robots.txt]]></namelink>
                    <iplink><![CDATA[http://84.89.0.74:80/robots.txt]]></iplink>
                </item>
        ...
                <statistics elapsed="6" itemsfound="5" itemstested="1691"
endtime="2015-04-20 15:12:41" />
        </scandetails>
</niktoscan>
```

On es localitzaran aquells fitxers els quals afebleixen el nostre servidor, informant a terceres persones informació rellevant sobre els recursos del nostre programari.

Misconfiguration - Default Files

La mala configuració del nostre servidor pot ser un focus de vulnerabilitats, els arxius predeterminats o mal configurats deixen entreveure informació sensible o fins i tot obrir una primera porta a atacs simples que afectarien a la integritat de tot el sistema.

En el nostre projecte s'utilitzarà nikto, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:
~$ nikto -h <SERVER_IP> -p 80 -t 5 -T 2 -Format xml -o <OUTPUT_FILE>

Que ens oferirà el resultat en format xml

```
<?xml version="1.0" ?>
<!DOCTYPE niktoscan SYSTEM "/usr/share/doc/nikto/nikto.dtd">
<niktoscan hoststest="0" options="-h http://server.joancaparros.com -p 80
-t 5 -T 2 -Format xml -o t2nikto" version="2.1.4" scanstart="Sun Apr 19
15:24:04 2015" scanend="Thu Jan  1 01:00:00 1970" scanelapsed=" seconds"
nxmlversion="1.1">
      <scandetails targetip="84.89.0.74"
targethostname="server.joancaparros.com" targetport="80"
targetbanner="Apache/2.2.22 (Ubuntu)" starttime="2015-04-20 15:24:04"
sitename="http://server.joancaparros.com:80/"
siteip="http://84.89.0.74:80/" hostheader="server.joancaparros.com">

            <item id="999996" osvdbid="0" osvdblink="0_LINK" method="GET">
                  <description><![CDATA[robots.txt contains 1 entry which
should be manually viewed.]]></description>
                  <uri><![CDATA[/robots.txt]]></uri>
```

```
<namelink><![CDATA[http://server.joancaparros.com:80/robots.txt]]></namel
ink>

<iplink><![CDATA[http://84.89.0.74:80/robots.txt]]></iplink>
            </item>

            <item id="999984" osvdbid="0" osvdblink="0_LINK" method="GET">
                  <description><![CDATA[ETag header found on server,
inode: 277305, size: 177, mtime: 0x50be4bf830454]]></description>
                  <uri><![CDATA[/]]></uri>

<namelink><![CDATA[http://server.joancaparros.com:80/]]></namelink>
                  <iplink><![CDATA[http://84.89.0.74:80/]]></iplink>
            </item>

            <statistics elapsed="2" itemsfound="3" itemstested="551"
endtime="2015-04-20 15:24:06" />
      </scandetails>
</niktoscan>
```

El resultat ens oferirà un llistat de possibles arxius, o metodes que són oferts per defecte que donarien a coneixer informació sensible del servidor, alguns d'aquests punts poden ser compartits amb els resultats de l'escàner sobe "fitxers interessants" ja que els arxius per defecte en molts cassos esdevenen directament fitxers que han de ser considerats a modificar per assegurar la seguretat dels nostres servidors.

Information Disclosure

La filtració d'informació (descobriment d'informació) per part d'un recurs pot revelar informació sobre l'objectiu. Aquest podria ser una ruta d'arxius o d'un compte, i donar indicacions a un possible atacant sobre com realitzar atacs sobre el servidor estudiat.

En el nostre projecte s'utilitzarà nikto, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

nikto -h <SERVER_IP> -p 80 -t 5 -T 3 -Format xml -o <OUTPUT_FILE>

Que ens oferirà el resultat en format xml

```
<?xml version="1.0" ?>
<!DOCTYPE niktoscan SYSTEM "/usr/share/doc/nikto/nikto.dtd">
```

```
<niktoscan hoststest="0" options="-h http://server.joancaparros.com -p 80
-t 5 -T 3 -Format xml -o t3nikto" version="2.1.4" scanstart="Sun Apr 19
16:08:51 2015" scanend="Thu Jan  1 01:00:00 1970" scanelapsed=" seconds"
nxmlversion="1.1">
      <scandetails targetip="84.89.0.74"
targethostname="server.joancaparros.com" targetport="80"
targetbanner="Apache/2.2.22 (Ubuntu)" starttime="2015-04-20 16:08:51"
sitename="http://server.joancaparros.com:80/"
siteip="http://84.89.0.74:80/" hostheader="server.joancaparros.com">


            <item id="001218" osvdbid="3092" osvdblink="3092_LINK"
method="GET">
                  <description><![CDATA[/sitemap.xml: This gives a nice
listing of the site content.]]></description>
                  <uri><![CDATA[/sitemap.xml]]></uri>
<namelink><![CDATA[http://server.joancaparros.com:80/sitemap.xml]]></name
link>

<iplink><![CDATA[http://84.89.0.74:80/sitemap.xml]]></iplink>
            </item>

            <statistics elapsed="3" itemsfound="4" itemstested="781"
endtime="2015-04-20 16:08:54" />
      </scandetails>
</niktoscan>
```

L'anàlisis dels arxius indicats revelaran l'estructura, metòdes o capceleres que el nostre servidor ofereix de forma oberta, que donarien indicis de com encarrilar un atac malintencionat.

Common Backdoors

Si un servidor ha estat compromesa anteriorment, hi ha una alta probabilitat que un ciber-criminal hagi instal·lat una porta del darrere per poder tornar fàcilment al servidor si fos necessari. Un mètode per aconseguir-ho és posar aquesta porta del darrere a l'arrel del servidor web o intentant amargar-ho dins de l'estructura de l'aplicació.
Durant les etapes inicials de reconeixement d'un atac, els ciberdelinqüents intentaran localitzar aquestes portes del darrera o web shells sol·licitant els noms dels scripts més comuns i coneguts.
Mitjançant l'anàlisi de la resposta, podrem determinar si existeic una porta del darrere o web shell.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --checks=backdoors --scope-auto-redundant=1
--report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE>
--report=json:outfile=<OUTPUT_FILE>.json

Common Backup Files and Folders

Una pràctica comuna en l'administració d'aplicacions web és crear una còpia / còpia de seguretat d'un arxiu o directori abans de fer qualsevol modificació a l'arxiu, canviant l'extensió o el nom del fitxer original per mostrar que es tracten de còpies de seguretat (per exemple '.bak','.orig','.backup',etc.).
Durant les etapes inicials de reconeixement d'un atac, els ciberdelinqüents intentaran localitzar els arxius que representen còpies de seguretat mitjançant l'addició d'extensions en arxius ja descoberts en el servidor web. Mitjançant l'anàlisi de les capçaleres de resposta des del servidor podem ser capaços determinar si existeixen fitxers de còpies de seguretat que posarien en compromís l'aplicació web.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --checks=backup_* --scope-auto-redundant=1
--report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE>
--report=json:outfile=<OUTPUT_FILE>.json

SQL injection

L'atac per injecció SQL ve donada per una o més vulnerabilitats en les aplicacions d'Internet que executen un filtrat incorrecte o insuficient sobre les dades entrades, permetent una execució de dades en forma de consultes SQL. Aquest atac està centrat en la capa de programació de sistemes complexos, de pàgines web dinàmiques i en general es deu a la falta d'experiència del desenvolupador.

La injecció SQL basa el seu objectiu en aprofitar la pròpia connexió de l'aplicació contra la bases de dades per a permetre a un atacant l'execució d'ordres directament sobre aquesta, posant en compromís la consistència de la base de dades i de la pròpia aplicació.

Realitzant aquest tipus d'atac es podran realitzar diferents accions, descrites a continuació:

• **Descobriment d'informació** (information disclosure) consistent en l'accés als registres de la base de dades i extreient informació no pública inicialment per els usuaris, com llistat d'usuaris, contrasenyes i altres dades emmagatzemades dins d'aquesta.

• **Suplantació d'identitat**, mitjançant el descobriment d'informació un atacant podia doncs pendre control de les dades d'un usuari registrat, i realitzar accions aparentment correctes però consentiment del usuari suplantat.

• **Elevació de privilegis,** tenint doncs el control de la base de dades l'atacant podría escalar els seus privilegis i realitzar accions les quals no està autoritzat.

• **Denegació de servei**. L'execució directa d'ordres SQL per part d'un possible atacant pot ocasionar la pèrdua del servei, ja sigui per esborrat de dades,a través d'ordres d'aturada o provocant un alt ús de comput que afectés negativament al temps de resposta de l'aplicació.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies
--checks=sql_injection_timing,sql_injection_differential,sql_injection --scope-auto-redundant=5
--report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE>
--report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json


```
{
  "version": "1.0.6",
  "options": {
    "scope": {
      "redundant_path_patterns": {
      },
      "dom_depth_limit": 10,
      "exclude_path_patterns": [
      ],
      "exclude_content_patterns": [
      ],
      "include_path_patterns": [
      ],
      "restrict_paths": [
      ],
      "extend_paths": [
      ],
      "url_rewrites": {
      },
      "auto_redundant_paths": 5
    },
    "datastore": {
      "report_path": "/tmp/d6b6d954b6ea931c1f9ad4a2e2a3a031"
    },
    "browser_cluster": {
```

```
    "pool_size": 6,
    "job_timeout": 120,
    "worker_time_to_live": 100,
    "ignore_images": false,
    "screen_width": 1600,
    "screen_height": 1200
  },
  "http": {
    "user_agent": "Arachni/v1.0.6",
    "request_timeout": 50000,
    "request_redirect_limit": 5,
    "request_concurrency": 20,
    "request_queue_size": 500,
    "request_headers": {
    },
    "cookies": {
    }
  },
  "input": {
    "values": {
    },
    "default_values": {
      "(?i-mx:name)": "arachni_name",
      "(?i-mx:user)": "arachni_user",
      "(?i-mx:usr)": "arachni_user",
      "(?i-mx:pass)": "5543!%arachni_secret",
      "(?i-mx:txt)": "arachni_text",
      "(?i-mx:num)": "132",
      "(?i-mx:amount)": "100",
      "(?i-mx:mail)": "arachni@email.gr",
      "(?i-mx:account)": "12",
      "(?i-mx:id)": "1"
    },
    "without_defaults": false,
    "force": false
  },
  "session": {
  },
  "audit": {
    "exclude_vector_patterns": [
    ],
    "include_vector_patterns": [
    ],
    "link_templates": [
    ],
    "links": true,
    "forms": true,
    "cookies": true
  },
  "checks": [
    "sql_injection_timing",
    "sql_injection_differential",
    "sql_injection"
```

```
    ],
    "platforms": [
    ],
    "plugins": {
    },
    "no_fingerprinting": false,
    "authorized_by": null,
    "url": "http://server.joancaparros.com/"
  },
  "sitemap": {
    "http://server.joancaparros.com/": 200
  },
  "start_datetime": "2015-05-01 14:27:53 +0200",
  "finish_datetime": "2015-05-01 14:27:56 +0200",
  "delta_time": "00:00:03",
  "issues": [
  ],
  "plugins": {
    "healthmap": {
      "name": "Health map",
      "description": "Generates a simple list of safe/unsafe URLs.",
      "author": "Tasos \"Zapotek\" Laskos <tasos.laskos@arachni-scanner.com>",
      "version": "0.1.5",
      "results": {
        "map": [
          {
            "without_issues": "http://server.joancaparros.com/"
          }
        ],
        "total": 1,
        "without_issues": 1,
        "with_issues": 0,
        "issue_percentage": 0
      }
    }
  }
}
```


NoSQL injection

Aquest tipus d'atac és similar al anterior, ambdues ataquen la capa de persistència de l'aplicatiu, en aquest cas s'intenta vulnerar Bases de dades NoSQL, aquestes proporcionen restriccions de consistència més fluixes que les bases de dades SQL tradicionals. Tot i que aquestes són més efectives en velocitat i escalat segueixen sent potencialment vulnerables a atacs d'injecció i conseqüentment a totes les accions descrites dins del atac d'injecció SQL

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=no_sql_injection --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

CSRF detection

Cross-Site Request Falsificació (CSRF) és un tipus d'exploit maliciós per atacar llocs web on l'objectiu final es el de forçar a un usuari final l'execució d'accions no desitjades en una aplicació web en la que es troba autenticat. L'atac per CSRF es centra específicament en les peticions webs, no al robatori de dades, ja que l'atacant no té manera de veure la resposta a la sol·licitud. Aquest tipus d'atac utilitza una part d'enginyeria social ja que s'aprofita de la propia confiança que un lloc té sobre un usuari en particular.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=csrf --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Code injection

La injecció de codi consisteix en l'explotació d'errors informàtics causats del tractament de dades no vàlides. L'objectiu de l'atac és el de presentar codi en un programa per fer canviar el curs de l'execució, modificant-los o aturant el seu funcionament. Dins d'aquesta categoria abarcaria la injecció de codi SQL, LDAP, XPath, NoSQL, OS commands i injecció de comandes a través d'un servei de correu IMAP/SMTP.

Aquest atac pot causar una denegació del servei o degut a una causa de corrupció de les dades una pèrdua de la informació del servidor.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=code_injection* --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

LDAP injection

L'Injecció LDAP és un atac utilitzat destinat a explotar les aplicacions web que integren sistemes d'autenticació LDAP perquè els seus usuaris puguin validar-se. Es tractaria d'un altre atac derivat d'una mala validació de l'entrada dels usuaris, aquesta es veu sotmesa a canvis per part del atacant modificant les declaracions LDAP fent servir un proxy local.

El resultat d'una correcte injecció podria derivar-se en:

- Execució d'ordres arbitràries, com la concessió de permisos per realitzar consultes no autoritzades.
- Modificació de contingut dins de l'arbre LDAP.

Les mateixes tècniques d'explotació avançades disponibles en SQL Injection es poden aplicar de manera similar en injecció LDAP.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=ldap_injection --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

File inclusion

La vulnerabilitat anomenada d'Inclusió d'arxius permet a un atacant incloure un arxiu dins del servidor atacat, normalment explotant uns mecanismes de "inclusió dinàmica d'arxius" implementats en l'aplicació de destinació, com a resultat l'atacant accedeix a fitxers que ja es troben en el propi servidor a través de l'explotació dels procediments d'inclusió vulnerables. Aquesta vulnerabilitat sorgeix a causa d'una mala depuració de les dades d'entrada.

Això pot conduir a alguna cosa com la sortida dels continguts de l'arxiu, però depenent de la gravetat, sinó que també pot conduir a:

L'execució de codi en un servidor web pot dur a :

- **Execució de codi** en el costat del client que pot conduir a altres atacs com cross site scripting.
- **Denegació de Servei**, afectant al servei deixant-lo invalidat durant un període de temps o de manera permanent.
- **Descobriment d'informació** emmagatzemada dins del servidor.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=file_inclusion --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Response splitting

HTTP Response splitting fa referència a un tipus de vulnerabilitat d'aplicacions web, resultat d'una manca de validacions dels camps d'entrada. Aquest es pot utilitzar per a realitzar atacs de cross-site scripting, defacement i enverinament de memòria cau web..

En l'estàndard HTTP (RFC 2616), les capçaleres estan separades per una CRLF i les capçaleres de la resposta es separen del seu cos per dos. Per tant, el fet de no eliminar els CR i LF permet a l'atacant establir capçaleres arbitràries, perdent el control del cos, o trencant la resposta en dues o més respostes.

Aquest atac succeïx quan:

- Les dades entren a una aplicació web a través d'una font no fiable, emprant sol·licituds HTTP.
- S'inclouen dades en una capçalera de resposta HTTP enviades a un usuari de la web sense ser validades amb caràcters maliciosos.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=response_splitting --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

OS command injection

La injecció de comandes és un atac en el que l'objectiu és l'execució d'ordres arbitràries en el sistema operatiu host a través d'una aplicació vulnerable. Com molts altres atacs aquest ve causat per una falta de validació de les dades d'entrada de l'aplicació. Aquest tipus d'atac succeïx quan una aplicació passa dades subministrades pels usuaris i que són insegures a un shell del sistema. Aquestes comandes seran executades per el sistema operatiu atacat amb els privilegis de l'aplicació vulnerable.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=os_cmd_injection* --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Remote file inclusion / Remote File Retrieval

La inclusió d'arxius és un tipus d'atac que es basa en l'explotació dels procediments d'inclusió vulnerables dins de l'aplicació atacada. Aquesta vulnerabilitat es produeix quan un atacant introdueix una ruta d'accés a arxius per tal que sigui inclòs en la resposta. Aquest tipus d'atac es produeix a causa d'una mala neteja dels paràmetres d'entrada.

Aquest atac pot conduir a:

- Execució de codi al servidor web
- Execució de codi en el costat del client com JavaScript que poden conduir a altres atacs com cross site scripting
- Denegació de Servei
- Descobriment d'informació

En el nostre projecte s'utilitzarà arachni sota el títol Remote File Inclusion, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=rfi --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Donat que és un dels atacs on la revelació de dades es pot donar de manera més flagrant s'ha optat per incloure sota la denominació Remote File Retrieval l'escaneig que nikto fa, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

nikto -h <SERVER_IP> -p 80 -t 5 -T 7 -Format xml -o <OUTPUT_FILE>

Unvalidated redirects

La vulnerabilitat causada per redireccions no validades és produeix quan aplicació web accepta paràmetres d'entrada que poden suposar incloure peticions de redireccionament cap URLs de les que no tenim confiança. A través de la inclusió d'aquests paràmetres es forçaria una redirecció a un lloc maliciós, on executar atacs de phishing per tal d'aconseguir credencials legitimes dels usuaris del lloc atacat.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=unvalidated_redirect --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

XPath injection

Aquest atac és similar a la injecció de SQL, es produeixen quan un lloc web utilitza la informació subministrada per l'usuari per construir una consulta XPath per a dades XML, de la mateixa manera que el seu homòleg SQL ho feia utilitzant consultes SQL.
Un dels mètodes més habituals és utilitzar l'enviament d'informació intencionalment mal formada en el lloc web, d'aquesta manera un atacant pot esbrinar com s'estructura les dades XML, o accedir a dades que no sol tenir accés.

Aquest tipus d'atac permet:

- **Elevació de privilegis:** Si les dades que s'utilitzen per a l'antenticació estan basades en XML, un atacant pot ser capaç d'elevar els seus privilegis.

- **Descobriment d'informació**: Com hem comentat mitjançant consultes XML a través de XPath podem localitzar certs atributs i patrons de la informació subjacent. Una pàgina es considerarà plenament vulnerable si els paràmetres d'entrada no són tractats per comprovar que no s'hagin alterat les consultes

XML és un llenguatge estàndard la seva sintaxi és sempre independent de la implementació, el que significa que l'atac pot ser automatitzat.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=xpath_injection --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

XSS

Els atacs Cross-site scripting (XSS) són un tipus d'injecció, en el que l'atacant introdueix codi maliciós en l'aplicació web atacada. Aquest tipus d'atac succeïx quan a causa d'una mala neteja d'un paràmetre d'entrada de la web es poden introduir comandes que seran executades per l'aplicació, podent atacar a possibles usuaris futurs de l'aplicació si l'atac es realitza de forma permanent.

Cal tenir present que aquest atac presentarà dues formes de presentar-se, en forma persistent i en forma no persistent:

Atacs Persistents XSS

Un atac de cross-site scripting persistent succeïx quan l'atacant proporciona dades malintencionades a l'aplicació web i aquest les emmagatzema de forma permanent en una base de dades o alguna altra eina de persistència de dades. Les víctimes d'aquest atac corresponen a aquelles que accedeixes a pàgines on aquest codi maliciós és exhibit i executat per part del navegador del client.

Els resultats d'aquest tipus d'atac conduiran a possibles filtracions d'informació emmagatzemada en galetes dels usuaris, permetent segrestos de sessió i per tant la pèrdua del control de l'aplicació vulnerable

Atacs No persistents XSS

Un atac de cross-site scripting no persistent té un comportament similar l'anterior modalitat comentada, l'atacant introdueix mitjançant els paràmetres d'entrada de l'aplicació web comandes que seran processades en una suposada resposta a les dades rebudes, en aquest cas però, el codi no serà emmagatzemat així que l'atacant normalment haurà de combinar aquest atac amb enginyeria social per tal que la víctima executi una url maliciosa, que enviarà via correu tota la informació rellevant del client atacat.
Com en el cas anterior aquest atac permetrà a l'atacant rebre els continguts de les galetes dels clients i per tant oferint la possibilitat de segrestar la sessió de totes les aplicacions en que la víctima s'hagués autenticat.

Atacs vectorials avançats XSS

Actualment dins de les aplicacions web dinàmiques podem trobar molts objectes que poden acceptar dades d'entrada de diferents fonts, els atacs XSS avançats exploten aquesta capacitat i intenten interceptar aquests paràmetres per injectar codi maliciós que serà finalment executat en el costat del client.

Un exemple representatiu d'objectes susceptibles a ser vulnerables serien:

- **iframes**: <iframe src = "http://hacker.website/cross-site-scripting.html">
- **Taules**: <fons de la taula = "codi maliciós">
- **Elements d'enllaç**: <link rel = HREF = "codi maliciós" "stylesheet">
- **Imatges elements**: <img src = "codi malicous">
- **Elements Div**:

Aquest atac pot conduir a:

- Execució de codi al servidor web

- Execució de codi en el costat del client com JavaScript que poden conduir a altres atacs com cross site scripting
- Denegació de Servei
- Descobriment d'informació

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=xss* --scope-auto-redundant=5 --report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE> --report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Source code disclosure

Aquesta vulnerabilitat es produeix quan un atacant pot obtenir el codi font de les pàgines que ofereix el servidor d'aplicacions web, i que poden deixar al descobert dades sensibles, com cadenes de connexió a la base de dades, noms d'usuari i contrasenyes, juntament amb la lògica de negoci de l'aplicació.

Aquest tipus d'atac ens permetria realitzar diferents tipus d'operacions:

- Accedir a la base de dades o altres recursos de dades posant en perill la integritat d'aquestes, ja que disposaria de permisos de lectura, modificació o esborrat de dades de la base de dades.
- Control de l'aplicació al poder accedir a l'àrea d'administració de l'aplicació.
- Desenvolupar nous atacs derivats de l'estudi dels errors del codi font cercant problemes de validació de paràmetres d'entrada i vulnerabilitats lògiques.

En el nostre projecte s'utilitzarà arachni, descrit en l'apartat d'escàners de vulnerabilitats, utilitzant la comanda:

arachni <SERVER_IP> --audit-forms --audit-links --audit-cookies --checks=source_code_disclosure --scope-auto-redundant=5

--report-save-path=<OUTPUT_FILE>;arachni_reporter <OUTPUT_FILE>
--report=json:outfile=<OUTPUT_FILE>.json

Que ens oferirà el resultat en format json, mantenint l'estructura definida per l'aplicació Arachni.

Desenvolupament d'una eina web per a la detecció de vulnerabilitats i generació d'informes

---

Introducció

Un cop hem definit els punts a analitzar caldrà doncs començar a enfocar el treball en el desenvolupament de la pròpia aplicació.

L'eina permetrà efectuar tests de manera senzilla, donant la possibilitat a gent inexperta a analitzar pàgines web en un gran ventall de possibilitats, no caldrà cap procés de configuració dels paràmetres d'entrada de les eines emprades per part del servidor,s implificant la execució d'un test a un simple click de ratolí.

Els resultats informaran de manera clara si s'han produït errors, indicant la quantitat de vulnerabilitats trobades i un resum de les dades extretes del escànner, no obstant l'aplicació estarà enfocada a gent inexperta, l'usuari podrà accedir a les dades en brut obtingudes per l'eina intrínseca que ha efectuat l'escànner, deixant un registre enfocat a persones amb més experiència i d'on poder extreure més informació.

L'aplicació efectuarà els processos d'escaneig de forma independent a l'aplicació web, de forma asíncrona, no caldrà doncs esperar a que la tasca sol·licitada acabi per continuar fent servir la nostre eina. Es comptarà inicialment amb una cua de tasques que limitarà l'execució de les tasques, això ens permetrà controlar quantes tasques s'han sol·licitat, la quantitat que s'executarà simultàneament i parametritzar-ho en funció del maquinari del qual disposem.

En els pròxims apartats es mostraran diferents diagrames i gràfics de flux que ajudaran a entendre el funcionament de l'aplicació

Tecnologies i recursos emprats

L'aplicatiu requerirà d'un entorn que s'encarregarà d'allotjar, servir i executar les instruccions necessàries per tal de construir els informes sobre vulnerabilitats que els nostres usuaris hauran d'interpretar. Caldrà doncs definir un entorn adaptat a les necessitats de la tecnologia emprada durant el desenvolupament, en el nostre cas i com es descriu en el apartat d'objectius es desenvoluparà una eina web, amb capacitat d'emmagatzemament de resultats d'anàlisis realitzats per els diferents aplicatius existents en el mercat.
Així doncs definirem els següents recursos emprats, i necessaris per el correcte funcionament del programari desenvolupat en aquest projecte final de màster:

- **VPS Linux:** Servidor virtual privat amb sistema operatiu lliure Linux Ubuntu 12.04.5 LTS o superior, amb com a mínim 2 Gb de memòria RAM
  Es va iniciar el projecte amb una màquina VPS amb una capacitat de 512Mb de memòria RAM, aquest fet representava una limitació ja que algunes de les eines emprades en aquest projecte requereixen d'un gran consum de memòria i CPU.

- **Apache:** Servidor de pàgines web sota el protocol HTTP de codi obert Apache/2.2.22 o superior.
  L'aplicació havia de ser web, cosa que ens conduïa a utilitzar un servidor que pogués interpretar els llenguatges de programació i àmpliament suportat per la seva comunitat.

- **MySQL:** Sistema de gestió de bases de dades relacional que emmagatzemarà les dades i construirà les relacions de les diferents entitats de l'aplicatiu.
  Aquest defineix el nostre model de dades i establirà el lloc de persistència de tots els resultats de les eines emprades durant l'escaneig.

- **PHP:** Llenguatge de programació de pàgines web dinàmiques qualificat com a programari lliure en la seva versió 5.4.39-1.
  Aquest llenguatge interpretat dona lleugeresa i flexibilitat a l'aplicació, àmpliament utilitzat i amb molts anys d'ús per a desenvolupar llocs web.

- **Framework Symfony2:** Marc de treball orientat a objectes que permetrà el desenvolupament d'una aplicació escalable a través de mòduls i patrons de disseny definits en ell.
  En si representa un conjunt d'eines per a desenvolupar la nostre aplicació de manera organitzada, la gran aportació de funcionalitats afegides per la comunitat fan que el desenvolupament d'aplicacions sobre un marc de treball esdevingui més àgil.

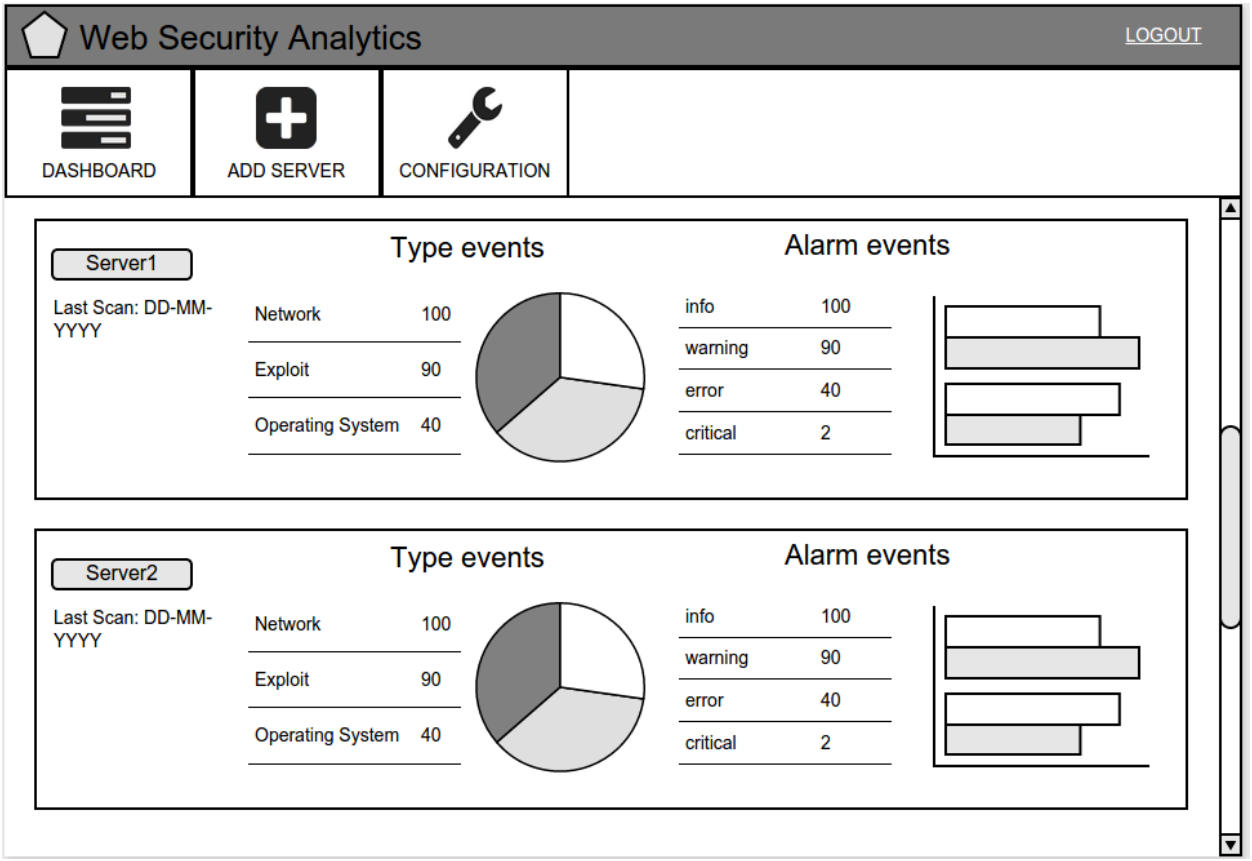
- **jQuery:** Marc de treball per a la tecnologia Javascript, per el tractament de l'aplicatiu web per part del client.
  No totes les funcionalitats seràn implementades per part del servidor, algunes d'elles seran executades pel navegador client, la carrega derivada és mínima i s'utilitza per afegir dinamisme.

- **RabbitMQ:** Gestor de cues de missatges encarregat del control d'execució de tasques que escalarà depenent de les capacitats dels maquinaris emprats.
  Aquest gestor de codi obert està disponible per a la gran majoria de llenguatges, molt lleuger i ideal per aplicacions on l'execució de tasques pot separar-se de l'execució en temps real de l'aplicació que l'allotja.

Mockup

El projecte requeria d'una implementació neta i clara d'un panell de control on poder observar tots els servidors estudiats (escanejats), la primera intenció del programador és sempre fer el màxim d'usable el contingut del qual es fa referència, és per això que abans de programar es realitzen varies proves de disseny.
En el nostre projecte també es varen desenvolupar centrats en la plana inicial, el resultat per això va diferir de la imatge resultant però només en termes de desplegament i ocultació de les gràfiques que fan referència al tipus de vulnerabilitat trobada.

S'inclou tot seguit el mockup inicial desenvolupat

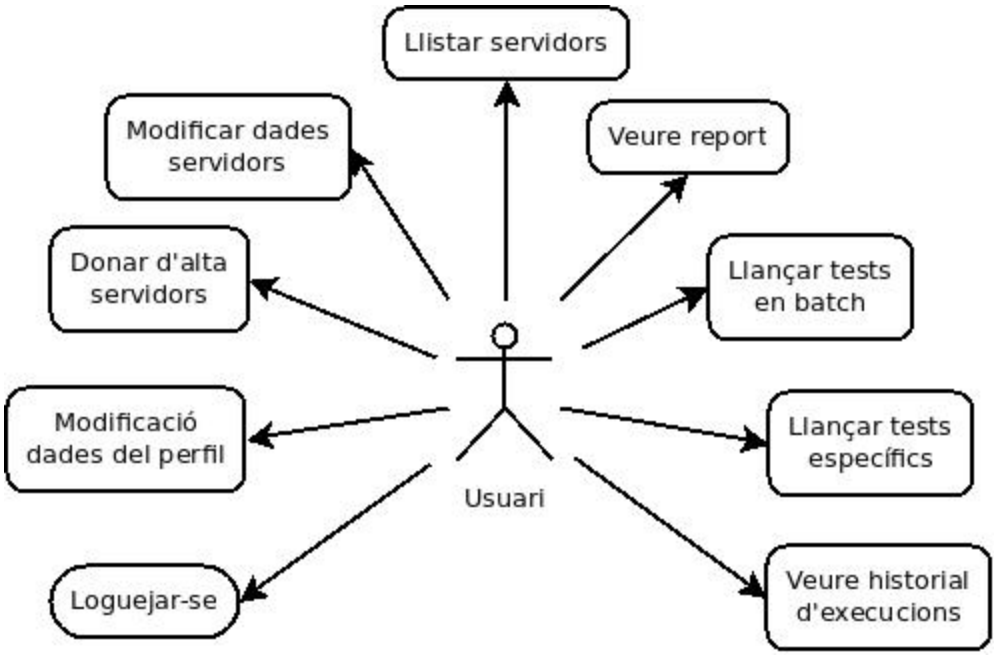


Cas d'ús

Donat que aquesta aplicació només consta d'un rol, i cal estar registrat per poder executar comandes el funcionamanet de l'aplicació podrà representar-se a través d'un diagrama de de casos d'ús molt senzill.

Primer de tot comentarem que els diagrames de casos d'ús són uns diagrames que defineixen el comportament de qualsevol aplicació, mitjançant una representació gràfica descriurem totes les crides als models.

En el següent diagrama mostrarem totes les accions que un usuari pot executar dins de l'aplicació desenvolupada:



Loguejar-se

És el punt d'entrada de l'aplicació, permet a els usuaris validar-se per tal d'accedir a l'aplicació, així doncs haurà d'identificant-se mitjançant un correu i una contrasenya, el sistema un cop validi aquestes dades atorgarà el rol USER al usuari, que el permetrà executar les accions que l'aplicació web ofereix.

Modificació de dades del perfil

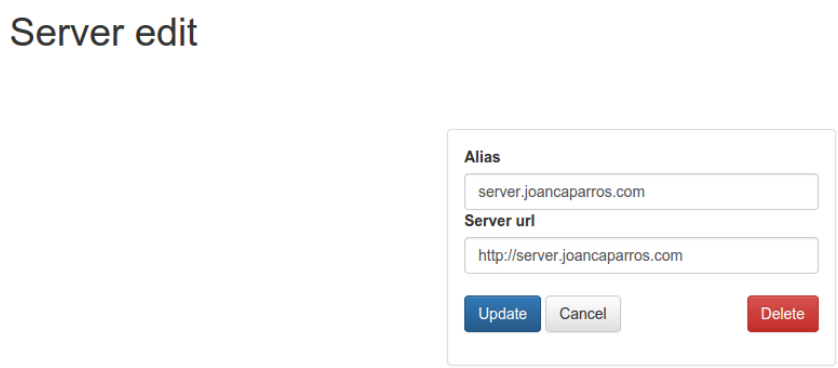
Les dades de l'usuari logat seran mostrades en tot moment dins de la barra superior de l'aplicació, en el cas que algun dels camps sigui incorrecte s'ha contemplat que tots els camps siguin modificables (Nom, Cognom, correu electrònic i contrasenya d'accés)

Donar d'alta servidors

La definició dels servidors a analitzar és senzilla, només necessitarem l'adreça de l'aplicació web, especificada en forma d'url valida (establint la cadena "http://") i un Alias, aquest darrer apareixerà en els nostres informes i facilitarà la lectura en el cas que la url de l'aplicació sigui molt llarga i confusa.

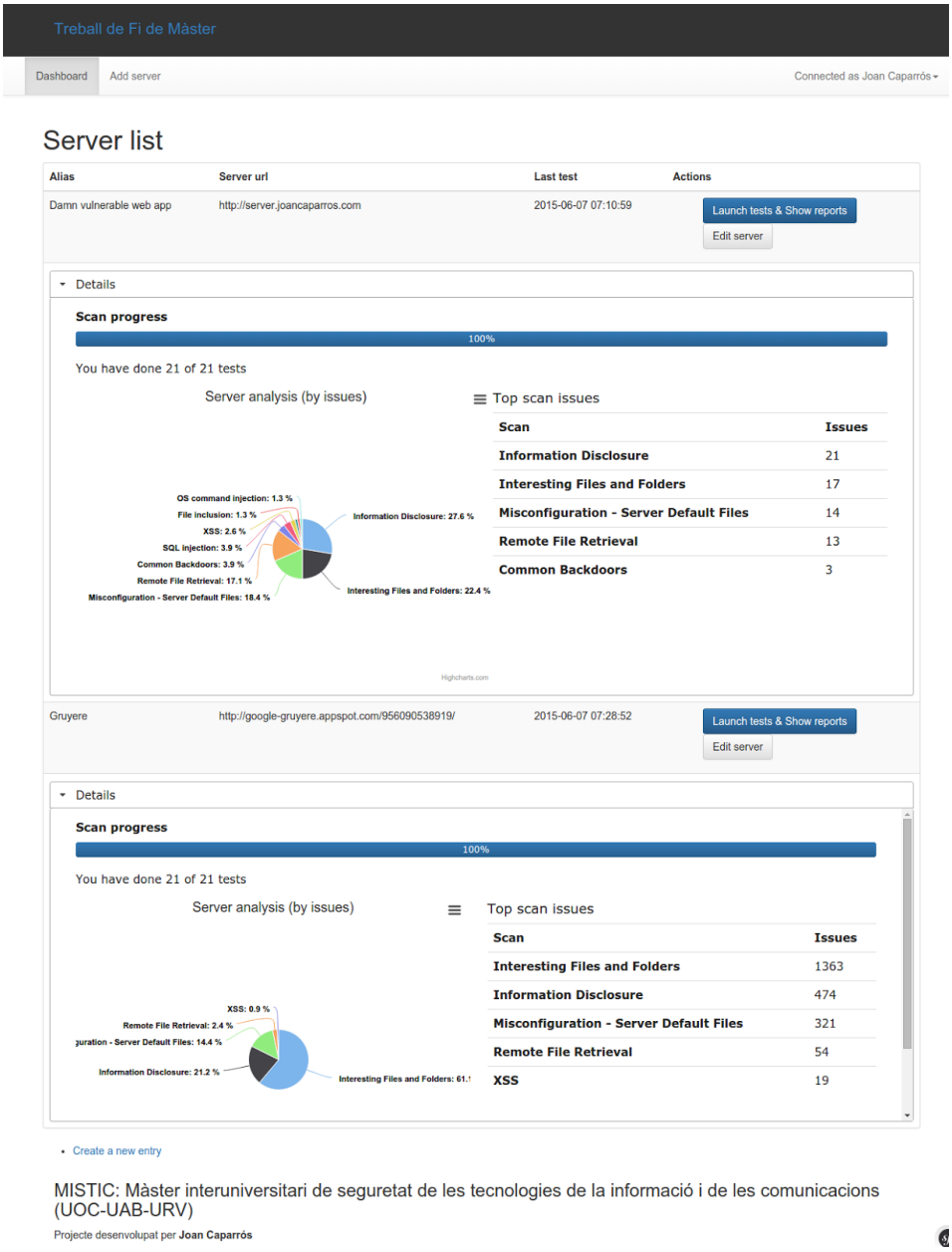
Modificar dades del servidor

Aquesta pàgina defineix un punt on poder modificar les dades entrades en un primer moment des de la plana d'alta dels servidors, permetent modificar el seu Alias o la ruta al recurs a analitzar. Tan mateix s'oferirà la possibilitat d'esborrar el servidor, si s'escull aquesta opció totes les proves que s'hagin realitzat sobre aquest quedaran de la mateixa manera eliminades del servidor.



Llistar servidors (Server List)

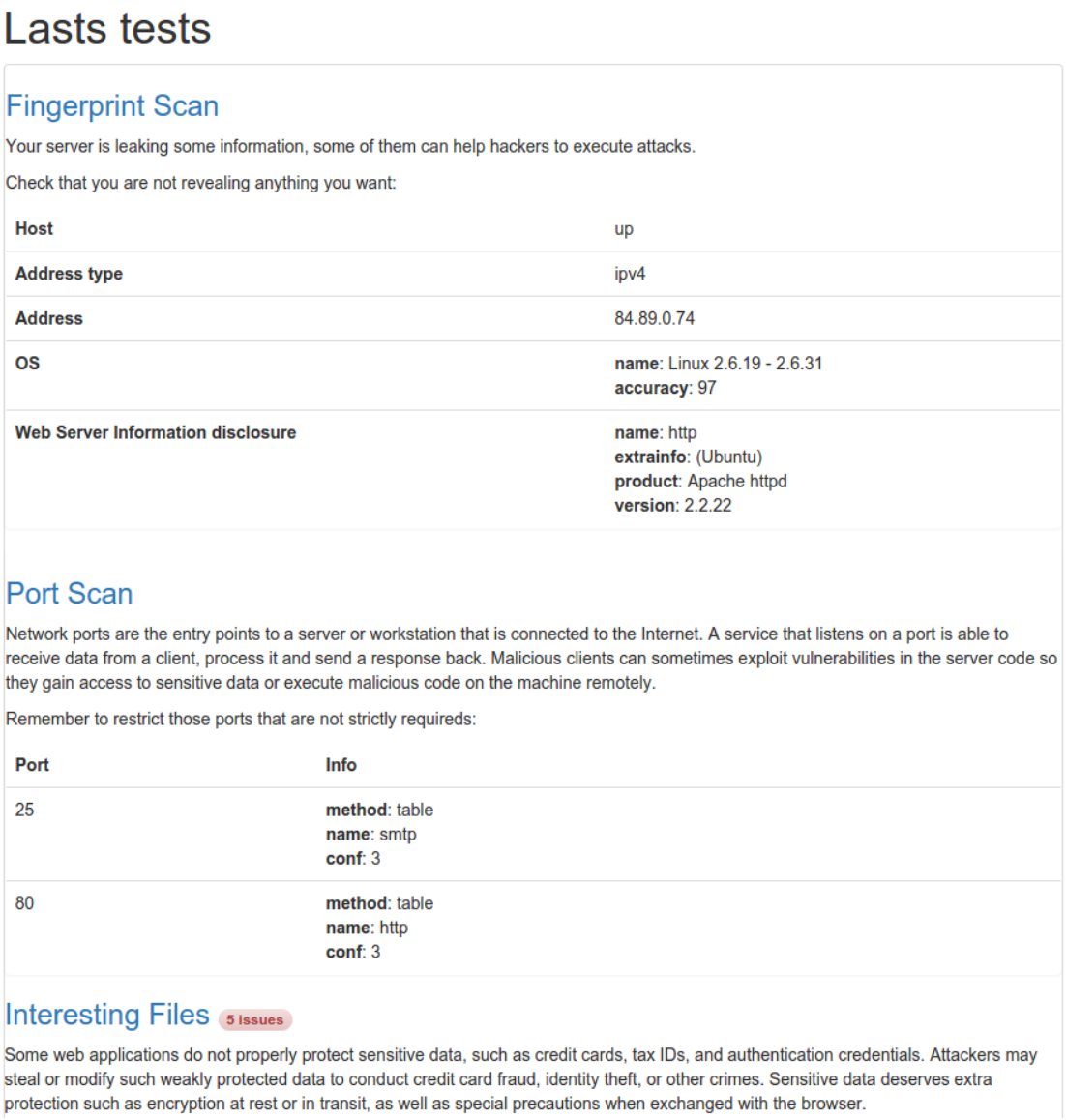
S'ofereix una vista on observar tots els servidors donats d'alta, amb la possibilitat d'accedir a els seus informes, modificar les seves propietats i en si, establir una pàgina d'inici on poder tenir un punt gloval del estat dels servidors contemplats en el sistema, oferint una idea general sobre l'estat d'aquest i quan s'ha executat per darrera vegada un escànner.
Aquesta pàgina era una de les específicacions del projecte i es requeria oferir una visió clara sobre l'estat dels servidors, és per això que a part d'una barra de progrés que ens indicarà quantes proves s'han efectuat també ens oferirà un resum de les vulnerabilitats més importants trobades en forma de gràfic i taula adjacent.



Veure report

La pàgina principal (Dashboard) d'un servidor mostra de manera simple les darreres proves efectuades sobre el servidor, formant un informe on poder veure l'estat actual de la màquina i sobre els apartats que el desenvolupador hauria de mostrar una especial atenció.
Com a primer indicador se'ns presentarà l'anàlisis abreujat que haurem vist durant la plana destinada a llistar els servidors i a continuació es mostraran en el ordre definit per l'aplicació cada un dels resultats de cada una de les proves realitzades, indicant la quantitat d'errors i oferint un enllaç directe on podrà dirigir-se en el cas de veure l'historial d'aquella prova i el resultat en cru (sense ser interpretat).

És en aquesta pàgina on l'usuari podrà imprimir l'informe general i tenir-la en un format pdf si s'escau.



Lasts tests

Fingerprint Scan

Your server is leaking some information, some of them can help hackers to execute attacks.

Check that you are not revealing anything you want:

Host	up
Address type	ipv4
Address	84.89.0.74
OS	**name**: Linux 2.6.19 - 2.6.31 **accuracy**: 97
Web Server Information disclosure	**name**: http **extrainfo**: (Ubuntu) **product**: Apache httpd **version**: 2.2.22

Port Scan

Network ports are the entry points to a server or workstation that is connected to the Internet. A service that listens on a port is able to receive data from a client, process it and send a response back. Malicious clients can sometimes exploit vulnerabilities in the server code so they gain access to sensitive data or execute malicious code on the machine remotely.

Remember to restrict those ports that are not strictly requireds:

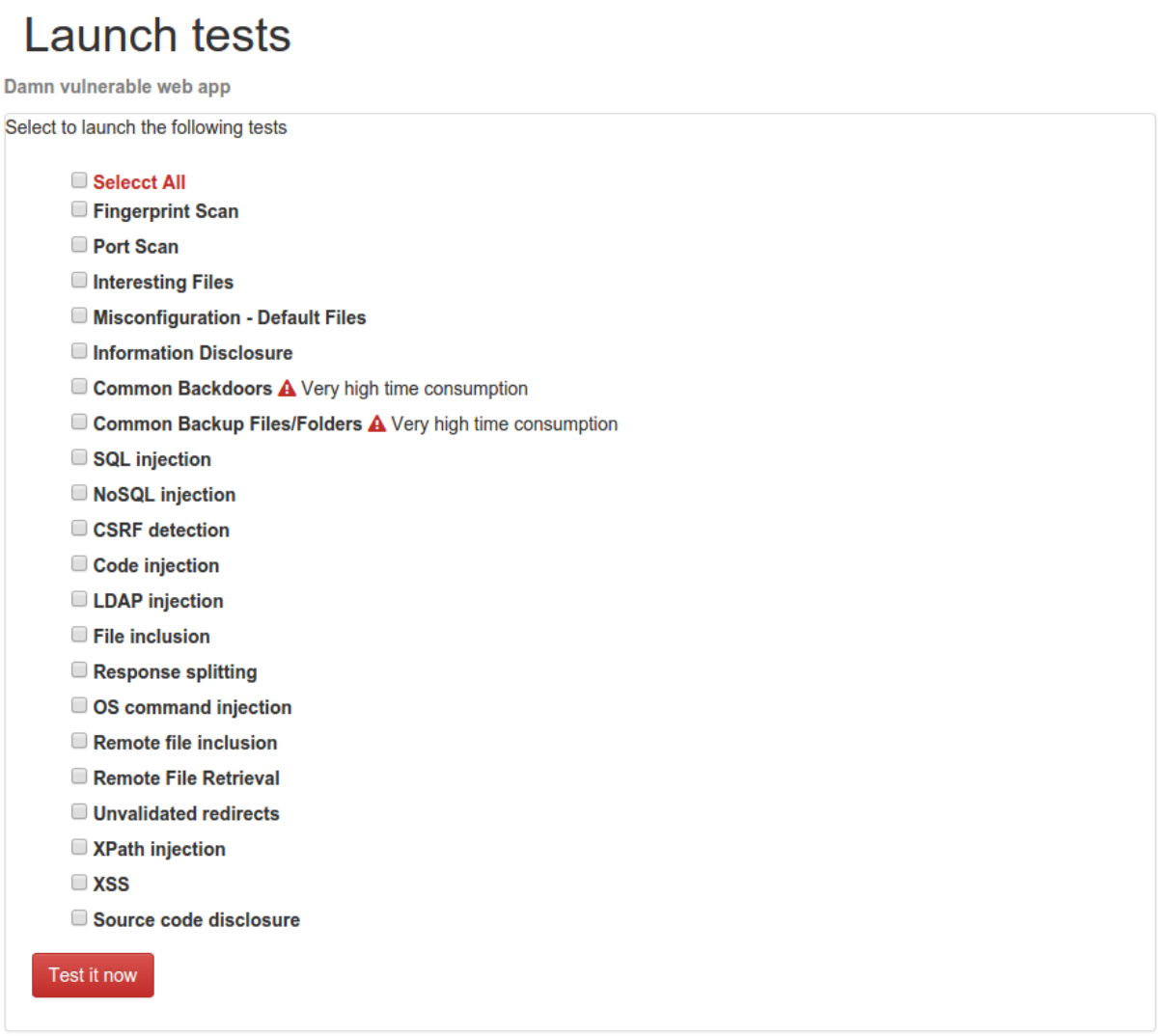
Port	Info
25	**method**: table **name**: smtp **conf**: 3
80	**method**: table **name**: http **conf**: 3

Interesting Files 5 issues

Some web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

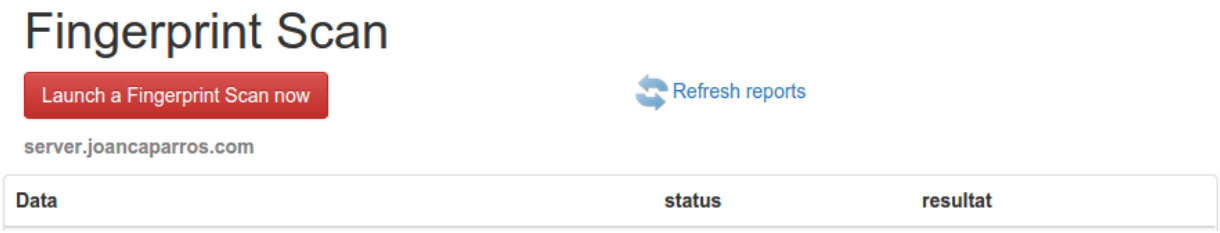
Llançar tests en bloc

La principal acció de la nostre aplicació serà la de llançar proves sobre els servidors donats d'alta, una de les opcions és la de seleccionar quines proves volem executar, podem escollir una en concret o totes elles, al clicar sobre "Test it now" totes aquestes accions seran afegides a la cua de tasques que s'executaran gradualment.



Llançar tests específics

Hem comentat que existeix la possibilitat d'enviar en bloc diferents accions que s'executaran sobre el servidor des de la pàgina de "Llançar tests en bloc", aquest no és l'únic punt on indicar que es pretén llançar una prova, cada una de les pàgines on es veurà l'historial de cada prova conté un botó on poder llançar la prova especifica de l'acció la qual s'està revisant.



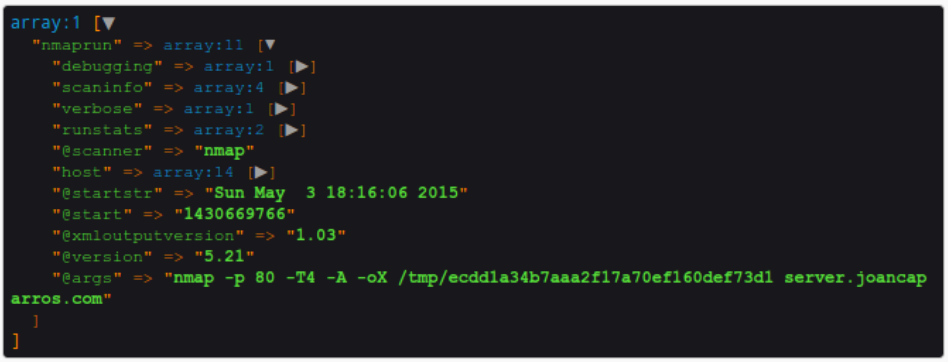
La tasca serà doncs tractada, i incorporada a la llista de tasques en cua amb l'estat QUEUED

Data	status	resultat
2015-05-03 18:07:14	QUEUED	

El gestor de tasques procedirà a executar-la quan disposi dels recursos i aquesta sigui la propera acció a la cua posant-la en estat PROCESSING

Data	status	resultat
2015-05-03 18:10:59	PROCESSING	

Finalment l'estat de la tasca prendrà el valor DONE, oferint el resultat en cru en forma d'array, ofert per a la comprovació dels resultats per gent especialitzada i a continuació una lectura amena interpretada dels resultats abstraient aquella informació més rellevant i oferta de forma amigable de cara a ser incorporada en un suposat informe

Data	status	resultat
2015-05-03 18:16:02	DONE	

Veure historial d'execucions

Fins al moment hem vist que podem veure l'informe, format per les darreres proves de cada tipus d'un servidor en concret i veure l'historial d'una prova en concret, però potser ens interessa tenir una visió més global del sistema per tal de veure quina de les accions han quedat en espera de ser executat, quines d'elles s'estan executant i quines ja han estat executades, així doncs es presenta una pàgina on contemplar totes les accions.

History

server.joancaparros.com

Refresh reports

Data	tipus	status	resultat
2015-05-01 14:37:25	Code injection	QUEUED	
2015-05-01 14:37:25	LDAP injection	QUEUED	
2015-05-01 14:37:25	File inclusion	QUEUED	
2015-05-01 14:37:25	Response splitting	QUEUED	
2015-05-01 14:37:25	OS command injection	QUEUED	
2015-05-01 14:37:25	Remote file inclusion	QUEUED	
2015-05-01 14:37:25	Remote File Retrieval	QUEUED	
2015-05-01 14:37:25	Unvalidated redirects	QUEUED	
2015-05-01 14:37:25	XPath injection	QUEUED	
2015-05-01 14:37:25	XSS	QUEUED	
2015-05-01 14:37:25	Source code disclosure	QUEUED	
2015-05-01 14:37:25	CSRF detection	PROCESSING	
2015-05-01 14:37:25	Port Scan	DONE	

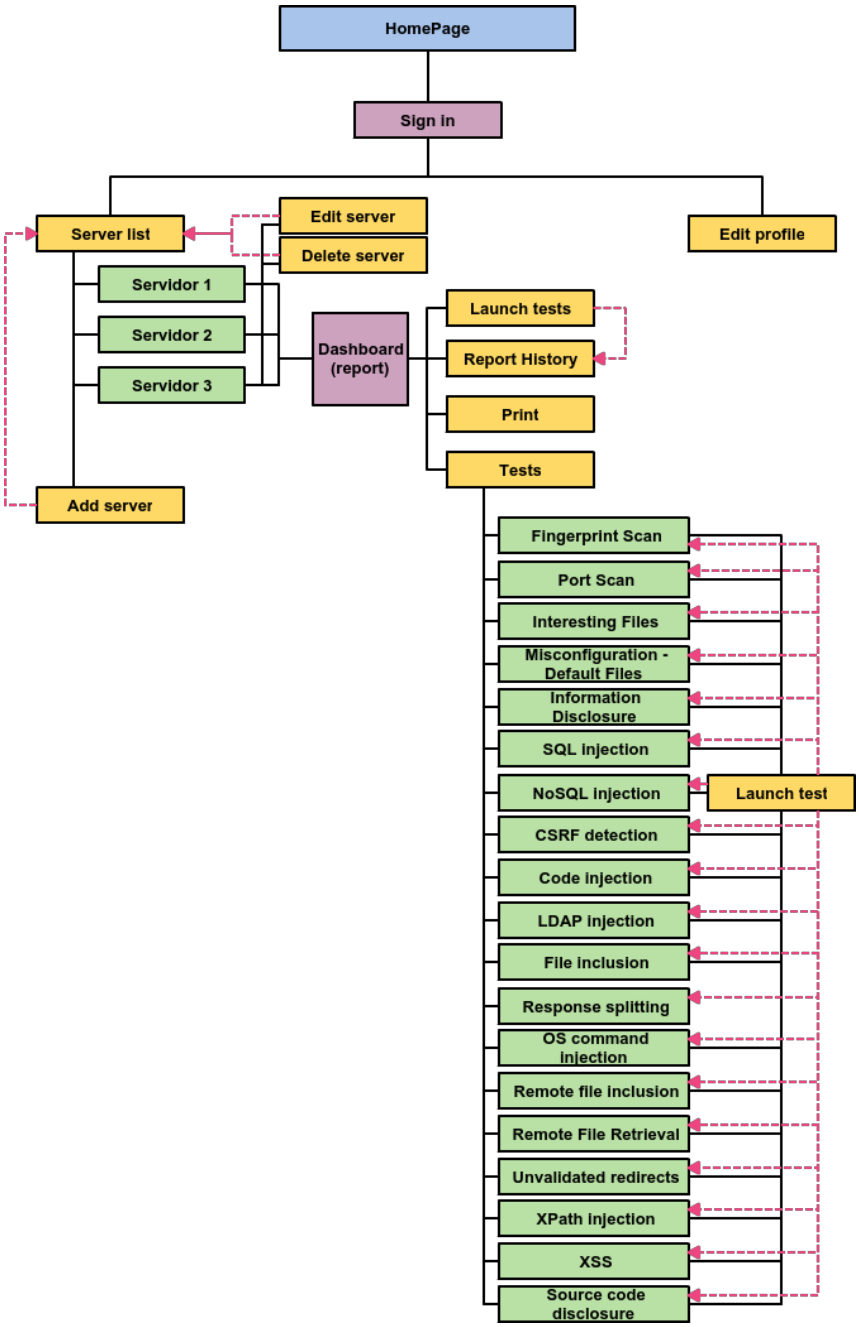
Port Scan

Network ports are the entry points to a server or workstation that is connected to the Internet. A service that listens on a port is able to receive data from a client, process it and send a response back. Malicious clients can sometimes exploit vulnerabilities in the server code so they gain access to sensitive data or execute malicious code on the machine remotely.

Remember to restrict those ports that are not strictly requireds:

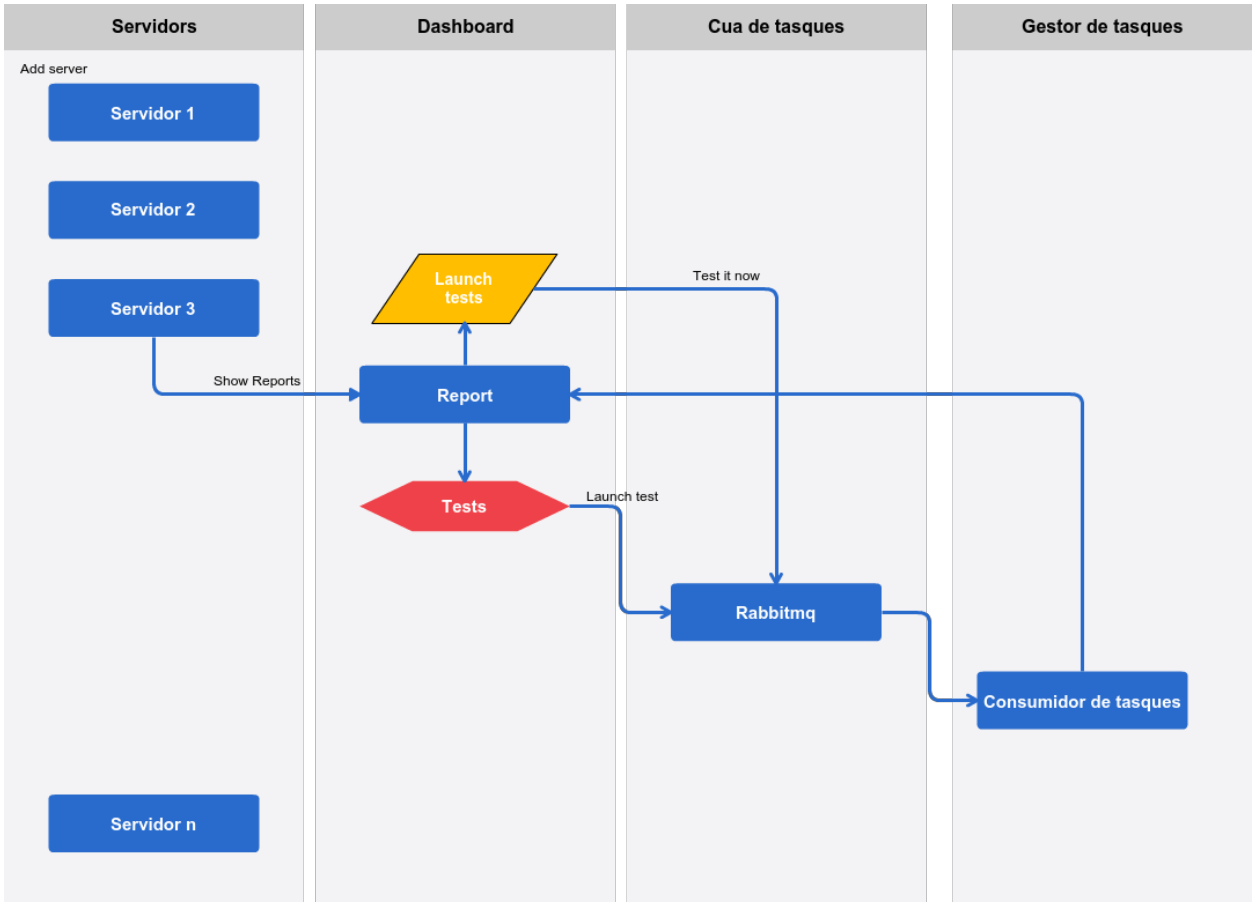
Port	Info
22	**method**: table **name**: ssh **conf**: 3
80	**method**: table **name**: http **conf**: 3
445	**method**: table **name**: microsoft-ds **conf**: 3

Mapa de l'aplicació web



*En rosa es marquen els redireccionaments automàtics

Diagrama de flux de les tasques



Implementació

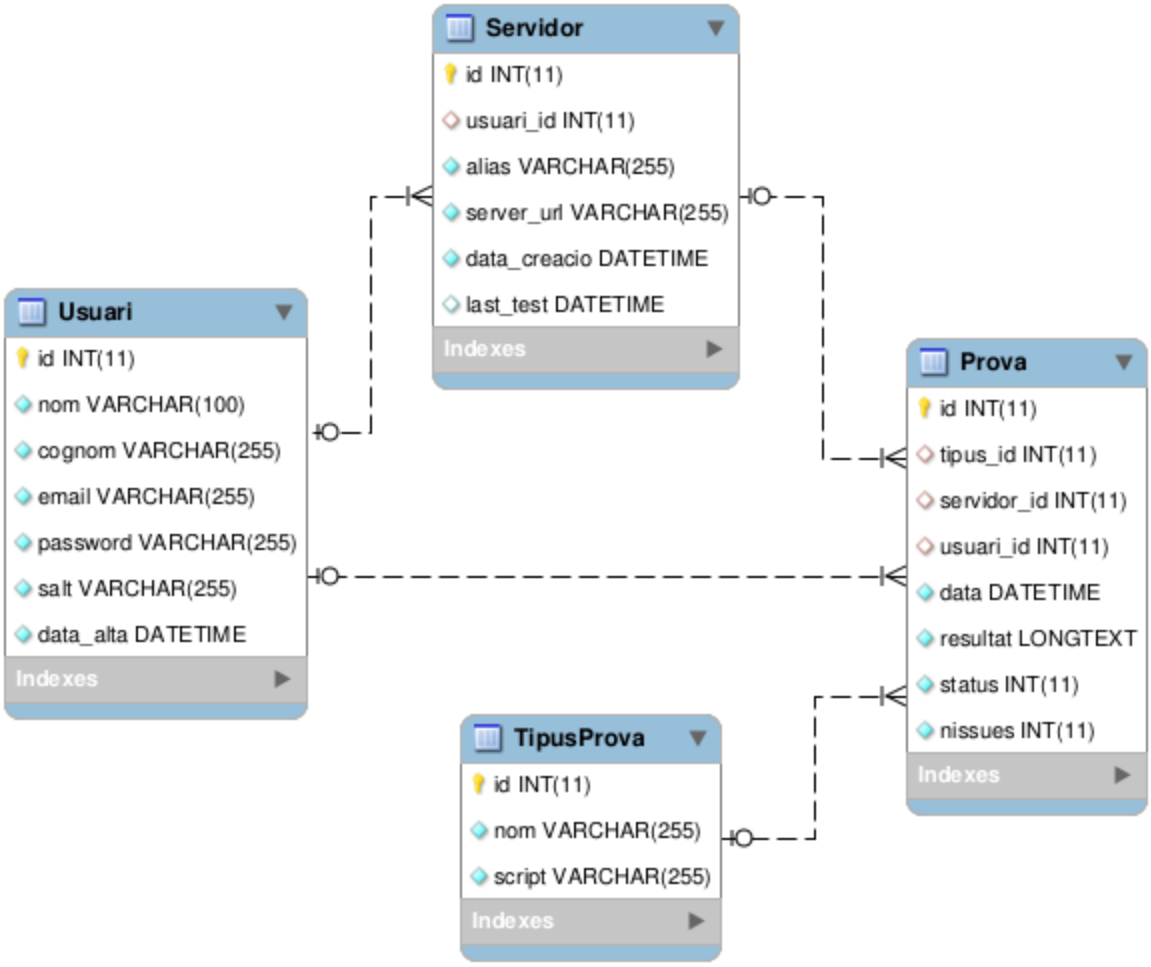
Model de dades

Definirem primer les entitats de la nostre aplicació, aquestes representaran models, objectes que es relacionaran entre si, aquests models i relacions marcaran l'estructura de la base de dades relacional.
Partirem doncs de les següents entitats:

- **Usuaris (Usuari):** Es definiran les propietats bàsiques per tal que les persones que puguin utilitzar l'aplicació desin la informació necessària per tal vincular les seves comptes amb els servidors analitzats.
- **Servidors a analitzar (Servidor):** Es desaran les dades de cada servidor a analitzar, com Alias i una adreça url en la que aniran dirigits tots els anàlisis realitzats per l'aplicació.
- **Proves definides dins de l'aplicació (TipusProva):** Els anàlisis de l'aplicació seran definits dins d'aquesta entitat, permetent un escalat gradual de l'aplicatiu adaptant-se a l'entorn del programari i a les proves requerides.

- **Resultats de les proves realitzades:** El resultat de tots els anàlisis (execucions d'instruccions) seran desats per complert, la interpretació i representació d'aquests definiran els Informes que l'usuari haurà de ser capaç de comprendre.

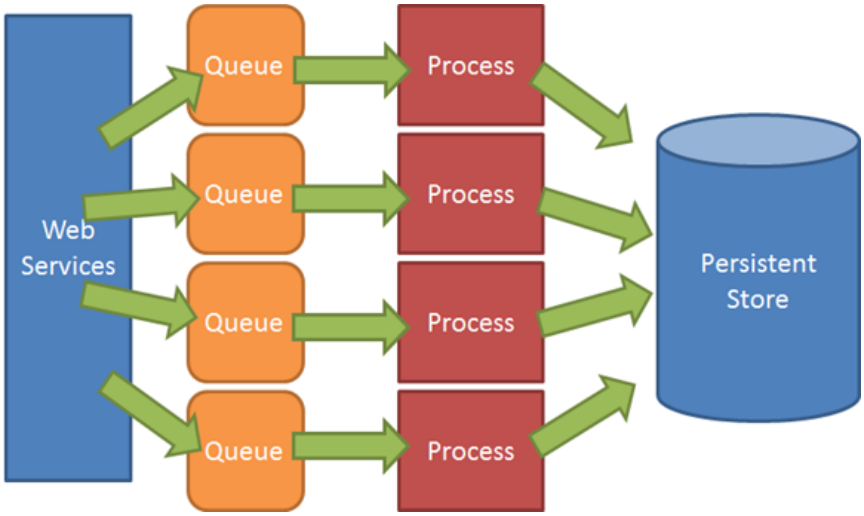
La representació dels diferents models definits dins d'un esquema de base de dades relacional, donaran com a resultat el diagrama EER:



Processament i anàlisis de resultats

En general les tasques d'escaneig requereixen d'una potència de càlcul i de memòria significativa, és per això que el processament de les peticions s'ha integrat dins d'un sistema de cues de missatges (rabbitmq - descrit dins de l'apartat de tecnologies).
Al executar una crida a una execució d'un escaneig d'un o més tipus d'escanners aquests s'incorporen en un llistat de peticions de comandes, aquests llistats són processats per un

numero de consumidors escalable, en el cas del nostre projecte el sistema constaria d'una simple llista de processament i un consumidor que s'encarregaria d'executar seqüèncialment cada una de les peticions, registrant els resultats obtinguts un cop finalitzades.

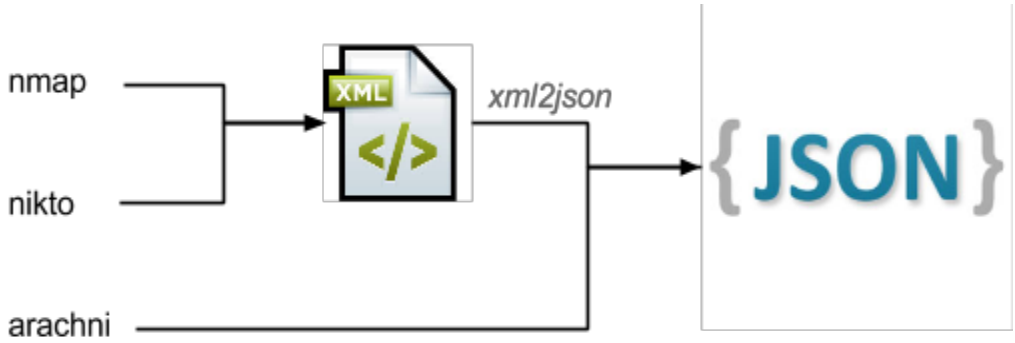


Que aporta un sistema controlat de cues al nostre projecte?
- Mitjançant el nombre de consumidors podem tenir un control sobre la quantitat de comandes que el maquinari pot suportar.
- En tot moment es té controlat l'estat de les tasques. Aquestes es classificaran podran pendre 3 estats diferents:
    - QUEUE: Quan la tasca ha entrat a la cua de missatges però no ha estat tractat per el consumidor
    - PROCESSING: El consumidor just ha pres posesió de l'execució del escaneig solicitat i es procedeix a la seva execució
    - DONE: El missatge ha estat tractat i el resultat
- Els consumidors poden ser executats per diferents màquines, incrementant el paral·lelisme de l'apliació.

Un cop tractat com es processen les tasques caldrà doncs remarcar com s'integren aquestes dins del propi sistema. El programa desenvolupat integra tres eines diferents, on cada una ofereix diferents possibilitats sobre com oferir els seus resultats. El repte de la integració romania en trobar un punt en comú, aquest punt, tot i divergir en el nombre de passos per obtenir el resultat en un mateix format era l'oferiment de la sortida com un objecte en forma JSON.
Un dels punts a favor de les eines triades eren que aquestes oferien el resultat en un sol fitxer, ja fos en format XML o directament en JSON. Així doncs i tenint el format resultant escollit només quedava per definir quins formats oferien les eines utilitzades i mitjançant convertidors intermitjos arribar a la solució esperada. *xml2json* (https://github.com/hay/xml2json) un projecte desenvolupat en python i de codi obert oferia la possibilitat de reconvertir els resultats entregats

en XML cap a un objecte de tipus JSON, així doncs ja teníem una via de conversió i d'uniformització dels resultats que s'emmagatzemarien posteriorment a la base de dades.

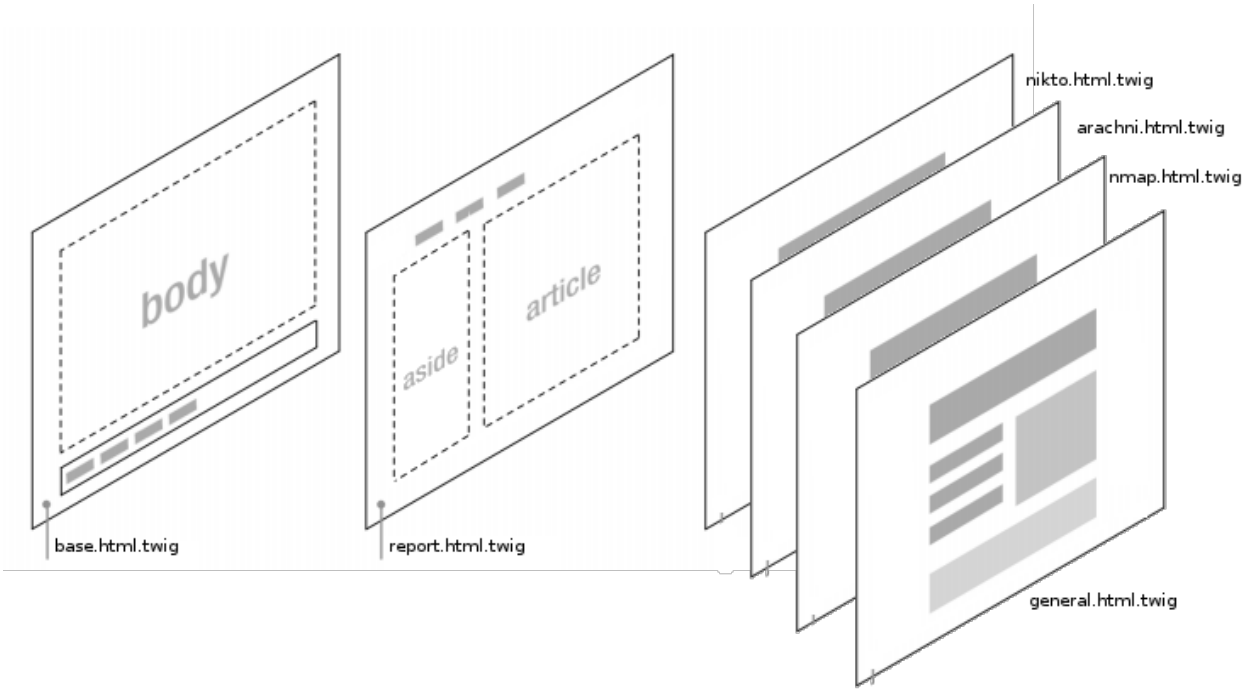
Passos per a la transformació dels resultats:



- comanda de conversió:
  ~$ xml2json -t xml2json -o arxiu.json arxiu.xml --strip_text

La unificació dels formats ofereix que cada resposta es presenta com un objecte resultat, al tractar-se d'objectes diferents cada un d'ells haurà de ser tractat d'una forma especifica. La interpretació dels resultats i la manera de mostrar-ho és un dels requisits essencials d'aquest projecte i cal entregar una lectura abreujada i clara de tot el contingut "en brut" que ofereixen les eines.
La traducció d'objectes cap a altres formats pot efectuar-se a través de capes intermitges o *crosswalks* que mapegen cada membre de l'objecte en cru cap a un nou objecte sortida, el nostre programari implementa un sistema de mappeix de dades a través de plantilles, cada tipus d'objecte resultat es renderitzat pel crosswalk o plantilla que li correspon, generant la vista "netejada" del objecte resultat anteriorment emmagatzemat en JSON.
Així doncs el programari consta inicialment de 3 plantilles que renderitzen els resultats segons si la sortida ha estat entregada de l'execució d'una instrucció nmap, si la sortida prové d'una crida al programari nikto, si aquesta prové de la crida al programari arachni o en el suposat cas que el resultat provingués d'un programari no contemplat oferiria una visualització amigable del objecte JSON emmagatzemat.
Aquest sistema de plantilles ha estat desenvolupat amb la clara intenció de poder ser escalable i afegir durant qualsevol etapa posterior de millora qualsevol altre template que es requerís, resultat d'afegir altres programaris a la aplicació.

Tot seguit es mostra en format gràfic l'estructura a tres nivells executat durant la renderització dels resultats dels anàlisis:



Instal·lació i desplegament de l'aplicació

Instal·lació de les dependències

Donada que aquesta eina està programada en PHP caldrà doncs tenir previament instal·lada una versió Apache amb el seu modul corresponent per a interpretar aquest llenguatge de desenvolupament de pàgines dinàmiques.
A continuació caldrà instal·lar *composer*, un gestor de dependències per a PHP que ens permetrà gestionar, declarar, descarregar i mantenir actualitzats tots els paquets de software que requereix el nostre projecte.
Un cop copiat el codi font de l'aplicació desenvolupada caldrà situar-nos dins de la carpeta contenidora i executar:

~$ composer install

amb aquest pas quedaran instal·lades totes les dependències, és llavors quan per assegurar la correcte instal·lació del paquet derivat de la gestió de cues caldrà executar

~$ composer update oldsound/rabbitmq-bundle

Inicialització de la base de dades

L'aplicació com s'ha comentat anteriorment requereix d'una capa de persistència que cal inicialitzar, és per això que el programari incorpora un fitxer amb totes l'esquema relacional que serà inclòs dins de la base de dades corresponent (per defecte aquesta s'ha anomenat Tfm), així doncs caldrà executar:

~$ php app/console doctrine:schema:create

En aquest moment contarem amb una base de dades buida, sense cap prova definida i sense usuaris predeterminats, és per això que es requereix d'un pas extra per incorporar tota la informació que formarà el conjunt de l'aplicació. Es procedirà a executar la següent instrucció:

~$ php app/console doctrine:fixtures:load

Instal·lació de les eines requerides pel programa

Com hem comentat l'aplicació web desenvolupada utilitza eines de tercers per tal de construir els seus informes, és per això que caldrà instal·lar de manera adequada aquestes eines i incloure-les dins del PATH per a una execució directa, sigui quina sigui la seva ubicació final. Recordem que caldrà doncs instal·lar les aplicacions Nmap, Nikto i Arachni, totes elles descrites dins de l'apartat d'escànners comercials.

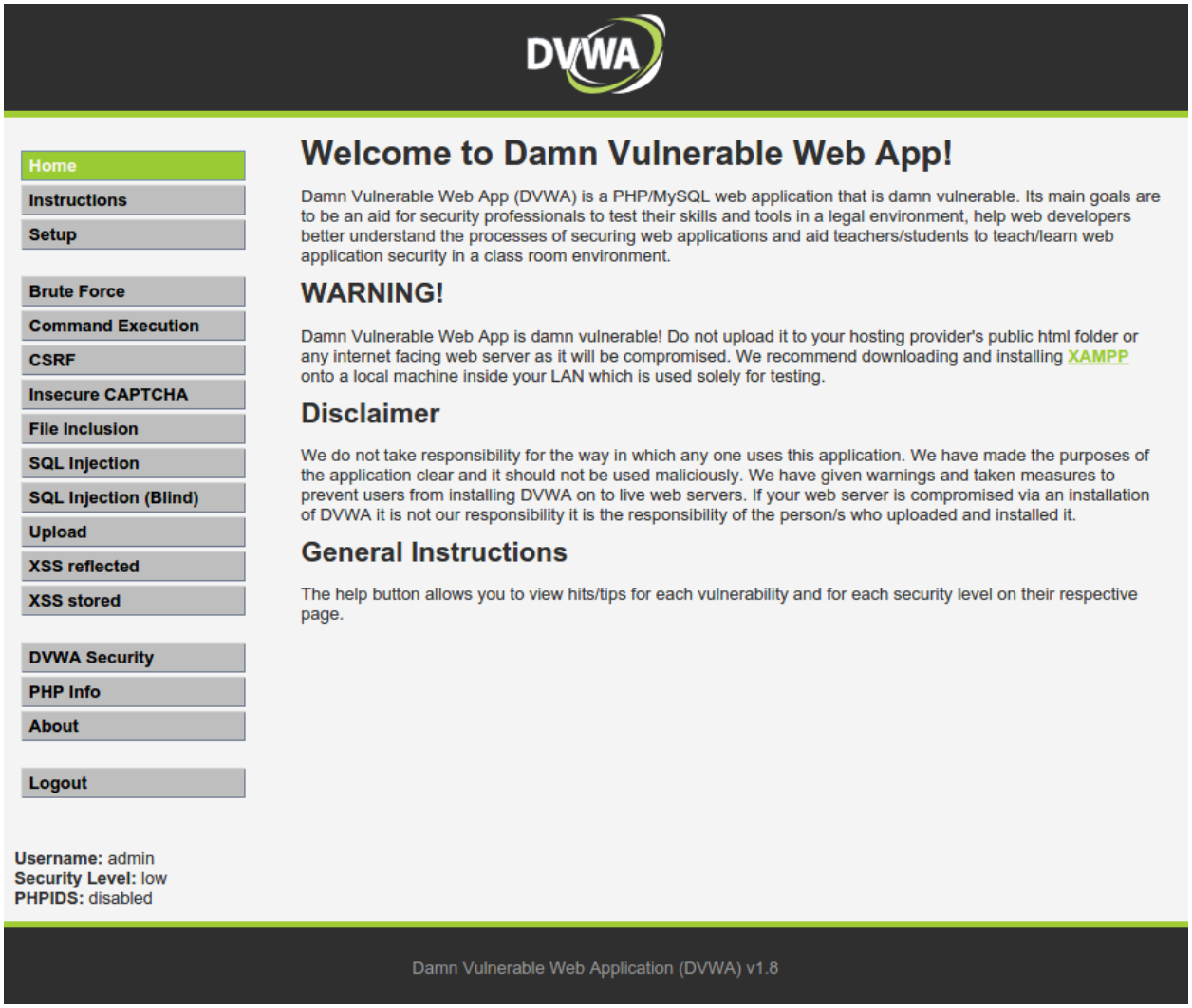
Proves de l'aplicació desenvolupada

Entorn de proves

Per a realitzar les proves s'ha utilitzat Damn Vulnerable Web App (DVWA), una aplicació destinada a la prova d'eines i de coneixements sobre les diferents vulnerabilitats existents, aquesta eina de codi lliure està desenvolupada en php i ens ofereix un marc complert on posar a prova el producte desenvolupat en aquest treball final de màster.

Primer de tot comentar que per comoditat en l'ús de l'escànner s'ha modificat el programa DVWA per anular el sistema de logeix i s'ha establert com a nivell de seguretat 'low', d'aquesta manera l'entorn de proves adquireix el grau de vulnerabilitat adequat per a començar a executar els diferents anàlisis de vulnerabilitats.

Així doncs la primera pantalla que es presentarà en l'entorn de proves serà la pàgina de benvinguda on es podrà escollir quin tipus de vulnerabilitat volem posar a prova.



Caldrà doncs comprovar que el nivell de seguretat **Security level** es troba en baix (**low**), observant el valor que apareix a la part inferior esquerra de la pàgina.

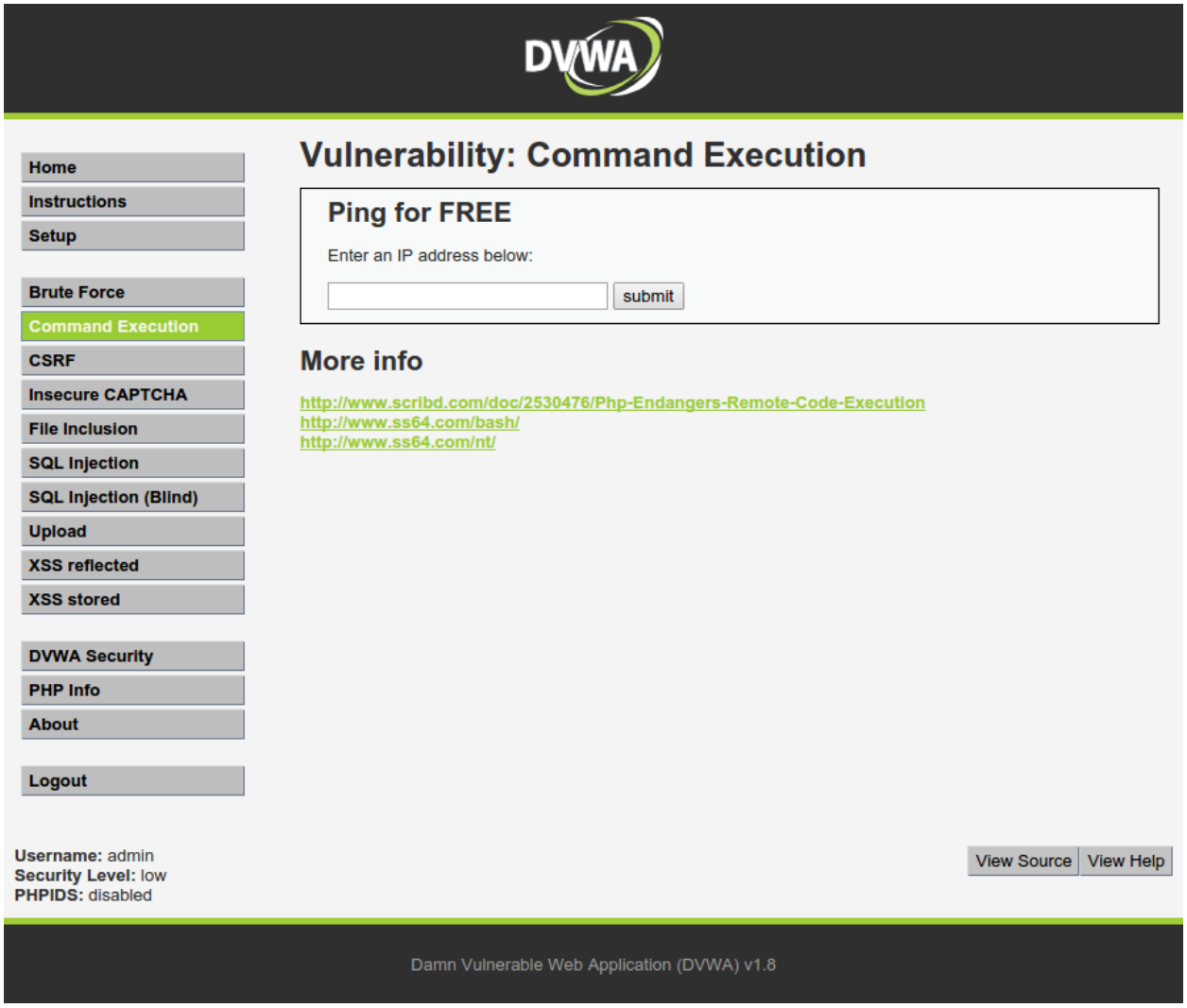
Es presentaran doncs diferents vulnerabilitats:

- **Brute Force** : tècnica obviada en aquesta memòria ja que no es tracte d'una vulnerabilitat en si, sinó d'un mètode de descobriment de contrasenyes utilitzant la força bruta de computació dels ordinadors actuals.

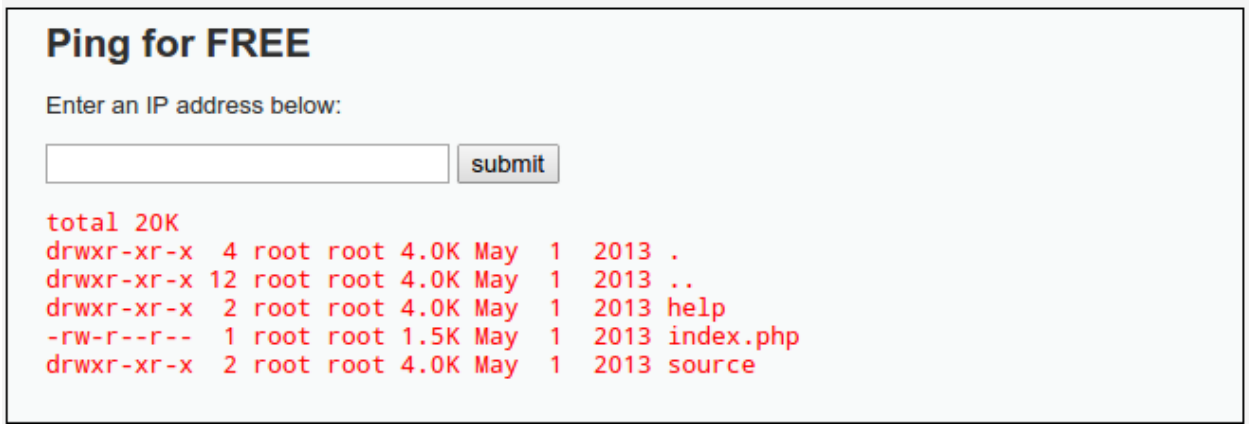
- **Command Execution** : en el següent apartat se'ns presentarà un input on la seva sortida és el resultat d'una execució d'una comanda shell. El fet de no netejar els

paràmetres d'entrada fan que aquests sigui vulnerable i que es puguin executar comandes en el servidor remot i observar la seva sortida:

```php
<?php
if( isset( $_POST[ 'submit' ] ) ) {
    $target = $_REQUEST[ 'ip' ];
    // Determine OS and execute the ping command.
    if (stristr(php_uname('s'), 'Windows NT')) {
        $cmd = shell_exec( 'ping  ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    } else {
        $cmd = shell_exec( 'ping  -c 3 ' . $target );
        echo '<pre>'.$cmd.'</pre>';
    }
}
?>
```



La comprovació de la vulnerabilitat és que si introduïm l'entrada '1;ls -lah' ls comanda posterior al punt i coma s'executarà i ens donarà el llistat de l'actual carpeta:

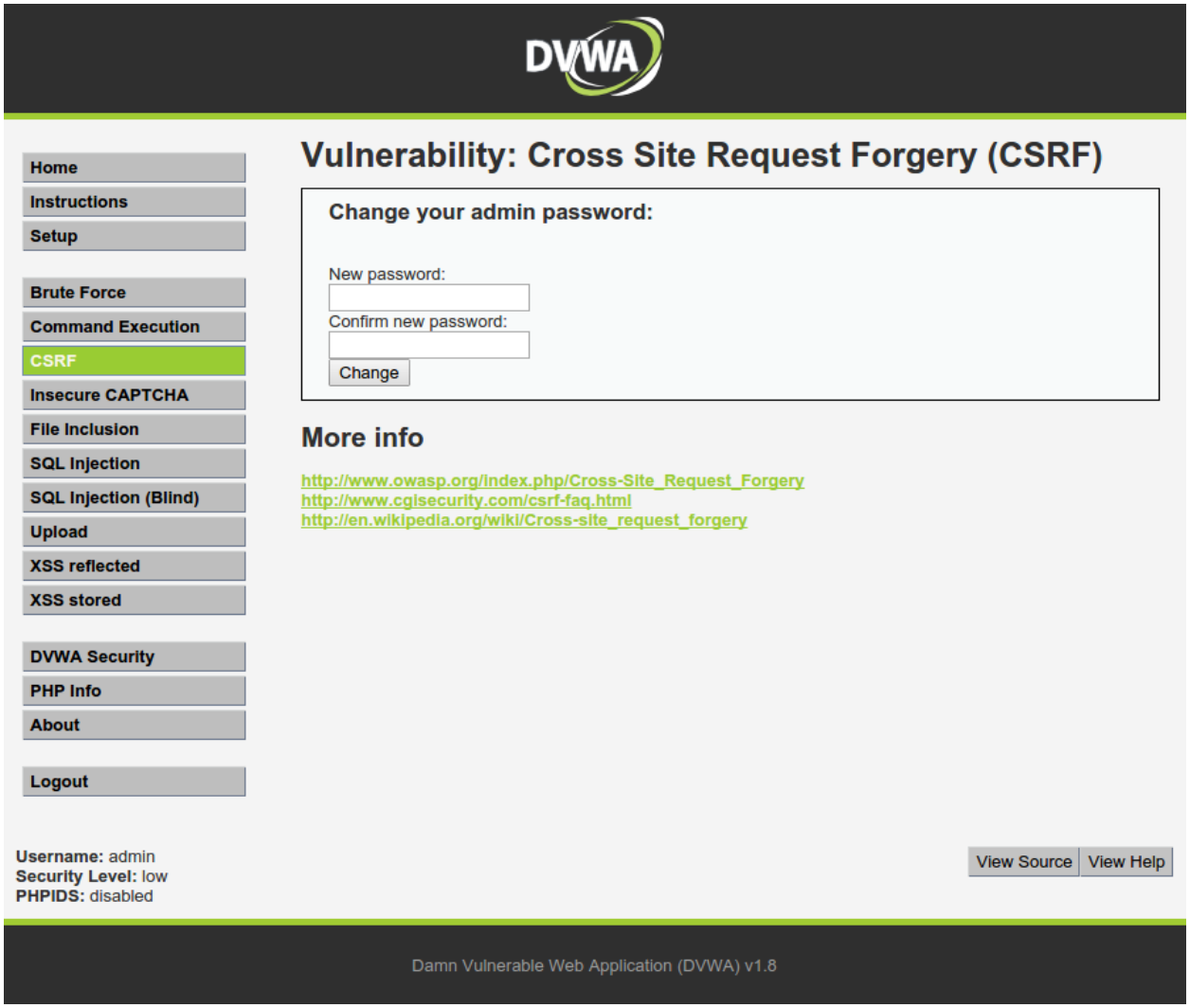


- **Cross Site Request Forgery (CSRF)** : CSRF és un atac que obliga a un usuari final executar accions no desitjades en una aplicació web en la qual s'està autenticat. Amb ajuda de l'enginyeria social (com l'enviament d'un enllaç per correu electrònic / xat), un atacant pot forçar a usuaris d'una aplicació web a executar accions sense el consentiment d'aquest. Una explotació amb èxit de CSRF pot comprometre les dades dels usuaris i el funcionament de l'aplicació web. En el cas que la víctima sigui l'administrador es posaria en perill tota l'aplicació.

  Tot seguit l'entorn de proves ens presenta dos inputs, corresponents a les entrades utilitzades per a canviar la contrasenya, el codi és senzill i en cap cas es comprova la procedència de les dades d'entrada, podent enviar des d'un altre servidor un formulari que compleixi amb els patrons del original i que efectuï una operació de canvi de contrasenya del usuari actual.

```
<?php
   if (isset($_GET['Change'])) {
      // Turn requests into variables
      $pass_new = $_GET['password_new'];
      $pass_conf = $_GET['password_conf'];
      if (($pass_new == $pass_conf)){
         $pass_new = mysql_real_escape_string($pass_new);
         $pass_new = md5($pass_new);
         $insert="UPDATE `users` SET password = '$pass_new' WHERE user =
'admin';";
         $result=mysql_query($insert) or die('<pre>' . mysql_error() . '</pre>' );
         echo "<pre> Password Changed </pre>";
         mysql_close();
      }
      else{
```

```
        echo "<pre> Passwords did not match. </pre>";
    }
  }
?>
```



En aquest cas construiríem un formulari en un altre servidor que referenciés al original, esperant que la víctima estigui logat de forma legítima:

```
<form action="http://server.joancaparros.com/dvwa/vulnerabilities/csrf/" method="GET">
New password:<br>
   <input type="password" autocomplete="off" name="password_new" value=”admin”><br>
   Confirm new password: <br>
   <input type="password" autocomplete="off" name="password_conf" value=”admin”>
   <br>
```

```
<input type="submit" value="Change" name="Change">
</form>
```

- **Insecure CAPTCHA**: vulnerabilitat obviada en aquest projecte ja que no es troba entre les més comunes definides per l'OWASP, però que seria fàcilment integrable utilitzant el modul *captcha* incorporat dins l'eina arachni.

- **File Inclusion:** Algunes aplicacions web permeten a l'usuari carregar arxius del propi servidor indicant una referència en la seva url. Si l'entrada no és netejada adeqüadament permetria a un atacant tenir accés a tot el sistema de fitxers del servidor vulnerable, deixant al descobert informació sensible i podent afectar a la seguretat de l'aplicació i del propi servidor. És normal també incloure adreces que apunten a altres pàgines web amb contingut maliciós que serien incorporades en la pàgina de retorn com a resultat

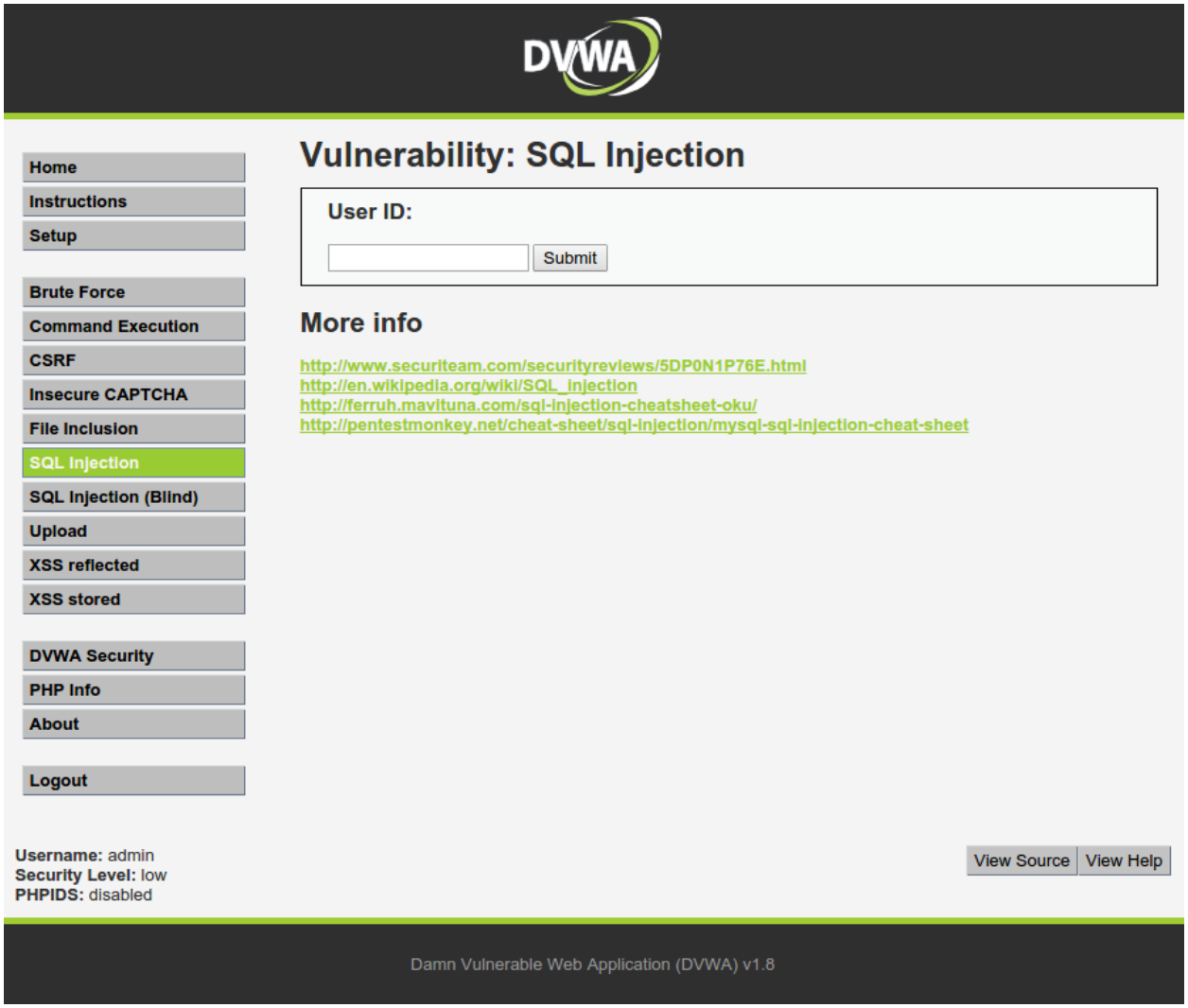
  Així doncs donat el codi:

```php
<?php
   $file = $_GET['page']; //The page we wish to display
?>
```

  L'execució de:
  http://server.joancaparros.com/dvwa/vulnerabilities/fi/?page=../../../../../../etc/hosts
  ens retornaria el contingut del fitxer hosts situat dins de la carpeta etc del servidor

- **SQL Injection:** Un atac d'injecció SQL consisteix en la inserció o la "injecció" d'una consulta SQL a través de les dades d'entrada del client dins de l'aplicació. La explotació amb èxit d'aquesta vulnerabilitat pot deixar al descobert dades sensibles de la base de dades, permetent al atacant inserir/modificar/suprimir les dades de bases de dades, executar operacions d'administració de la base de dades, recuperar el contingut d'un arxiu determinat present a l'arxiu de DBMS del sistema i en alguns casos permeten l'execució d'ordres al sistema operatiu. En aquest cas l'entrada no ha estat netejada i s'ha introduït directament dins de la consulta SQL, donant com a resultat un formulari vulnerable.

```php
<?php
if(isset($_GET['Submit'])){
    // Retrieve data
    $id = $_GET['id'];
    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid) or die('<pre>' . mysql_error() . '</pre>' );
    $num = mysql_numrows($result);
    $i = 0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");
        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';
        $i++;
    }
}
?>
```

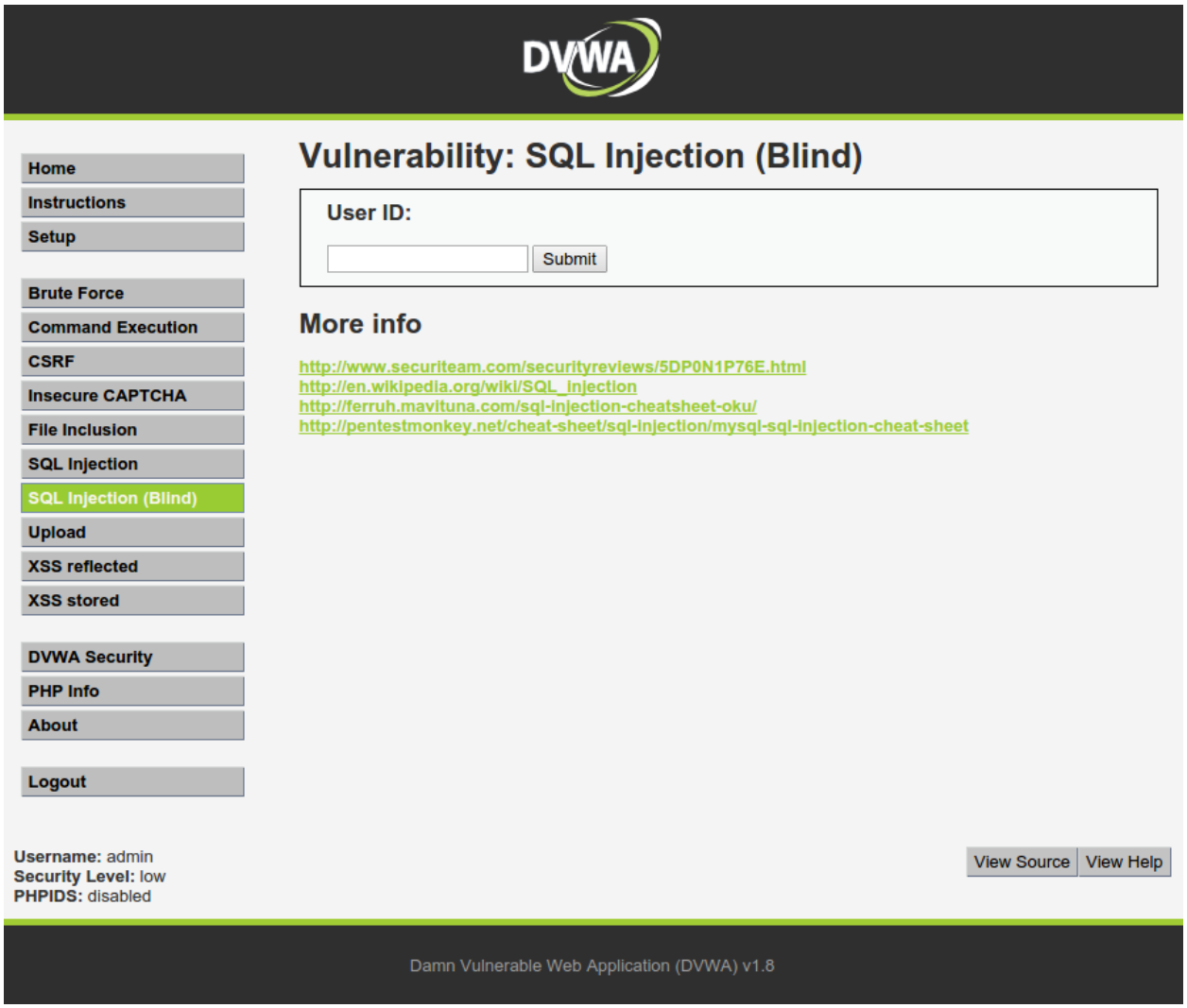


Permetent l'execució de comandes com

3' and 1=0 union select null,version()#

que revelarien en el camp Surname la versió del nostre servidor sql

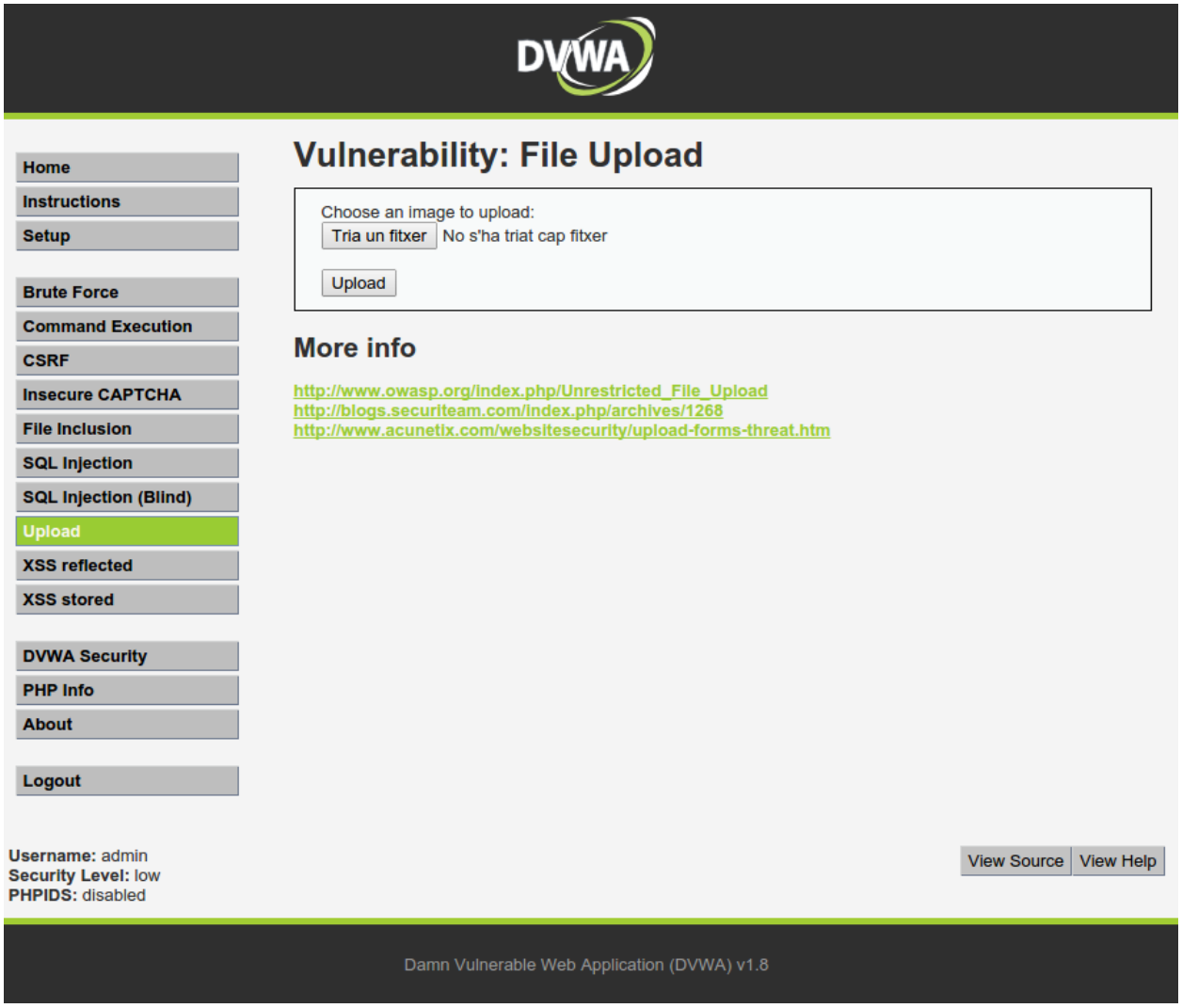
- **SQL Blind Injection:** Quan un atacant executa atacs d'injecció SQL, de vegades el servidor respon amb missatges d'error del servidor de base de dades exposant que la sintaxi de la consulta SQL és incorrecta.L'injecció SQL cega un atac idèntic al d'injecció SQL normal excepte que quan un atacant intenta explotar una aplicació, en lloc de rebre un missatge d'error útil, s'obté una pàgina genèrica del lloc web atacat.



L'explotació en aquest cas seria la mateixa que la elaborada durant la injecció SQL normal, i només canviaria el mecanisme de detecció d'aquest.

- **File Upload:** Les càrregues de fitxers representen un risc significatiu per a les aplicacions. Les conseqüències de la càrrega de la carrega d'arxius poden ser diverses, l'atac més comú és el de carregar dins del servidor atacat codi maliciós per tal que aquest sigui executat per la pròpia víctima.

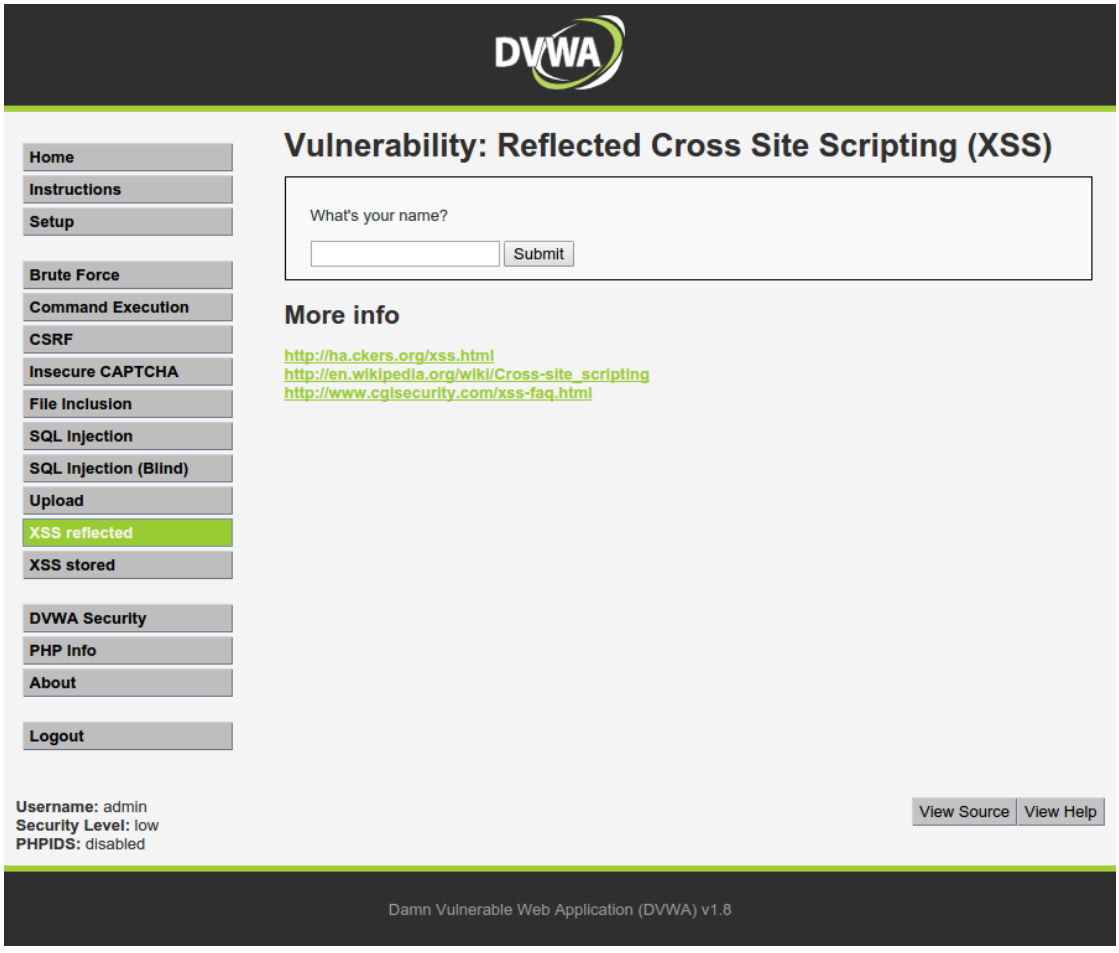
```php
<?php
   if (isset($_POST['Upload'])) {
          $target_path = DVWA_WEB_PAGE_TO_ROOT."hackable/uploads/";
          $target_path = $target_path . basename( $_FILES['uploaded']['name']);
          if(!move_uploaded_file($_FILES['uploaded']['tmp_name'], $target_path)) {
             echo '<pre>';
             echo 'Your image was not uploaded.';
             echo '</pre>';
           } else {
                      echo '<pre>';
             echo $target_path . ' succesfully uploaded!';
             echo '</pre>';
          }
      }
?>
```



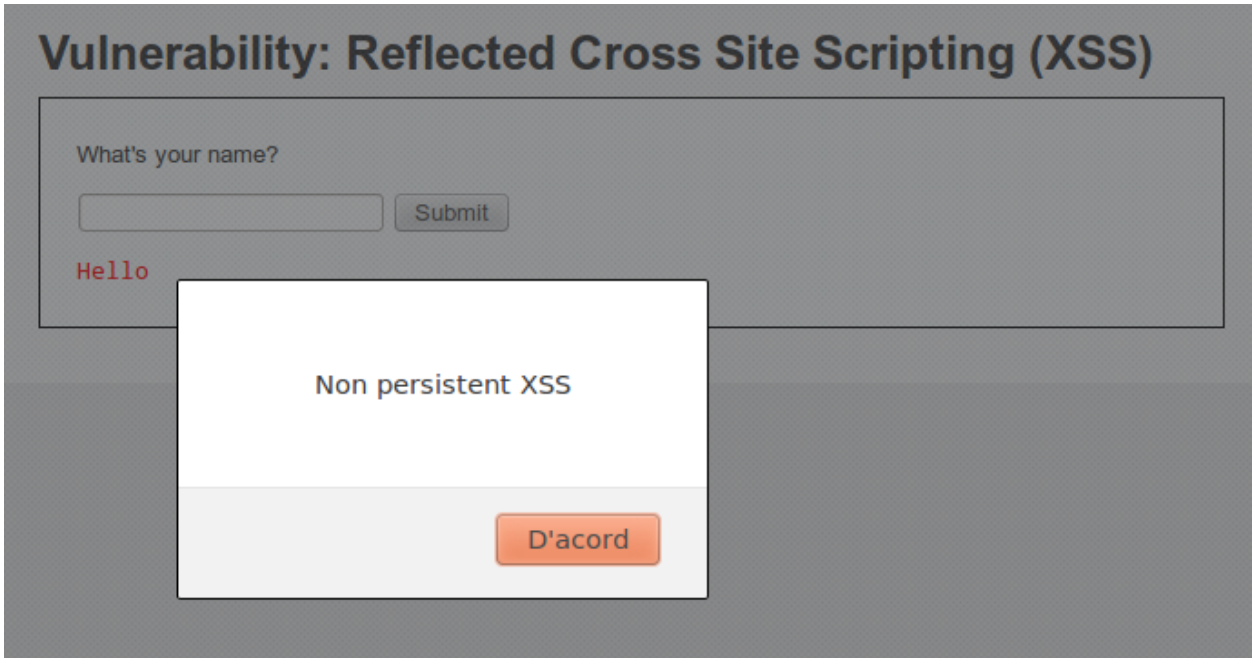
Un codi maliciós estàndard intentaria aprofitar la execució de shells integrats en una eina web, com podria ser el codi c99.php disponible a http://www.r57shell.net/.

- **Reflected Cross Site Scripting (XSS):** La definició d'aquest atac és el ja explicat atac no persistent de Cross Site Scripting, en aquest exemple un atacant podria injectar codi que seria executat durant la resposta de la pròpia pàgina. Aquest atac es degut a una manca de neteja de les dades d'entrada i permetria a un atacant enviar una url a una víctima per tal de forçar una execució codi maliciós per part del client, tenint com a conseqüència més habitual el robatori de cookies per tal de segrestar sessions legitimes dels usuaris atacats.

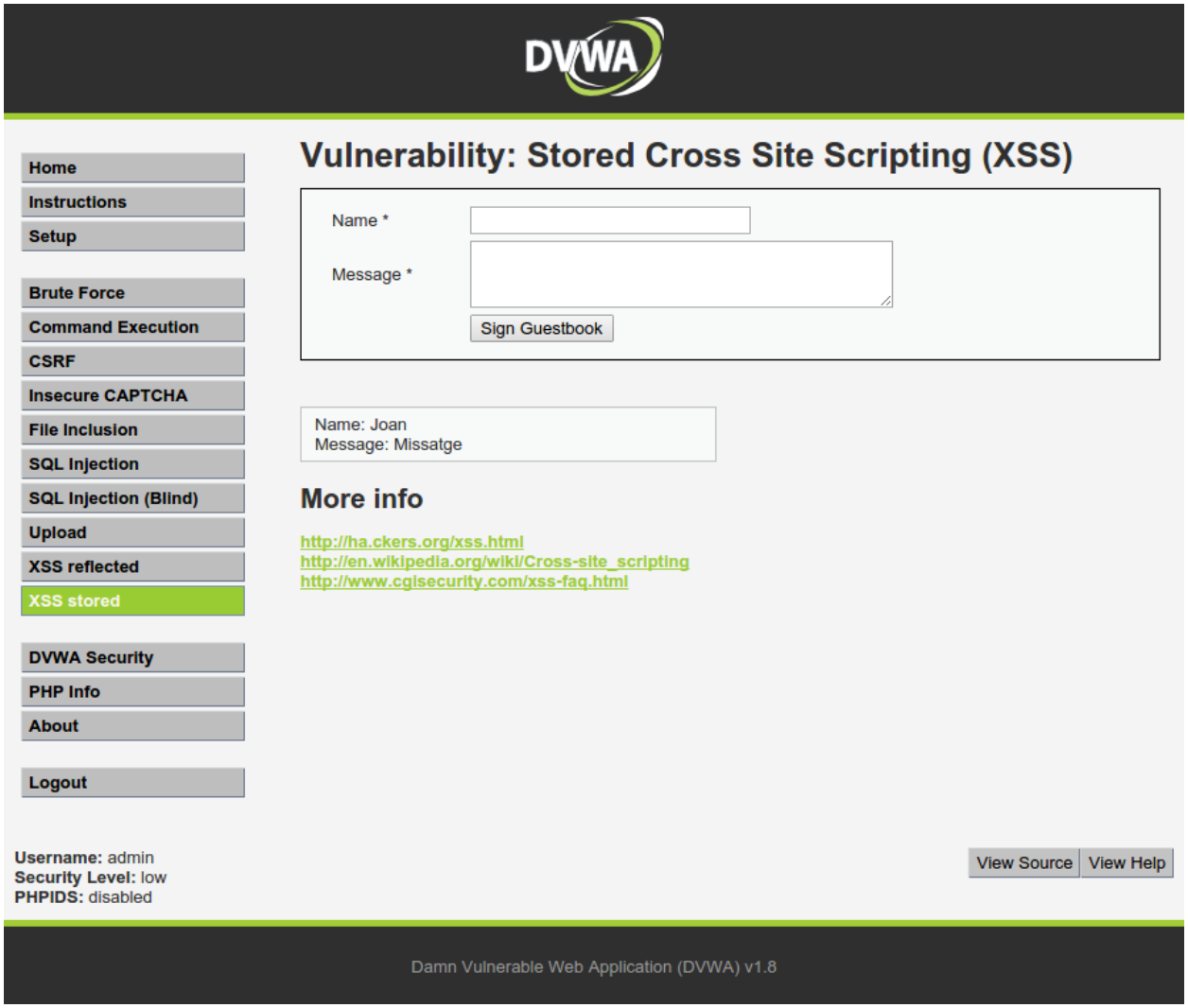
```php
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
 $isempty = true;
} else {
 echo 'Hello ' . $_GET['name'];
}
?>
```



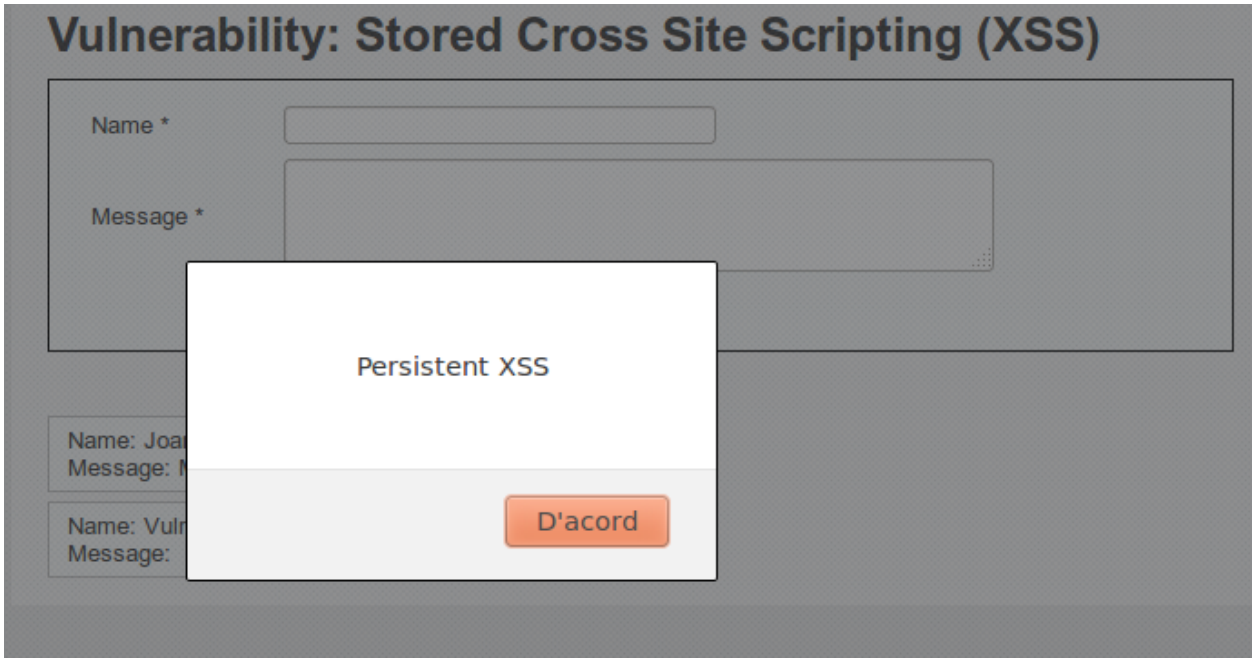
On l'execució de la següent url http://server.joancaparros.com/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28%22hola%22%29%3C%2Fscript%3E#, executaria el codi javascript alert("hola")



- **Stored Cross Site Scripting (XSS):** Similar a la vulnerabilitat anteriorment comentada aquesta es basa en la execució de codi maliciós introduït dins dels inputs d'entrada de l'aplicació, en aquest cas les comandes són emmagatzemades i per tant executades per tots els possibles usuaris de l'aplicació web, que es converteixen en víctimes del codi maliciós que l'atacant hagi introduït.



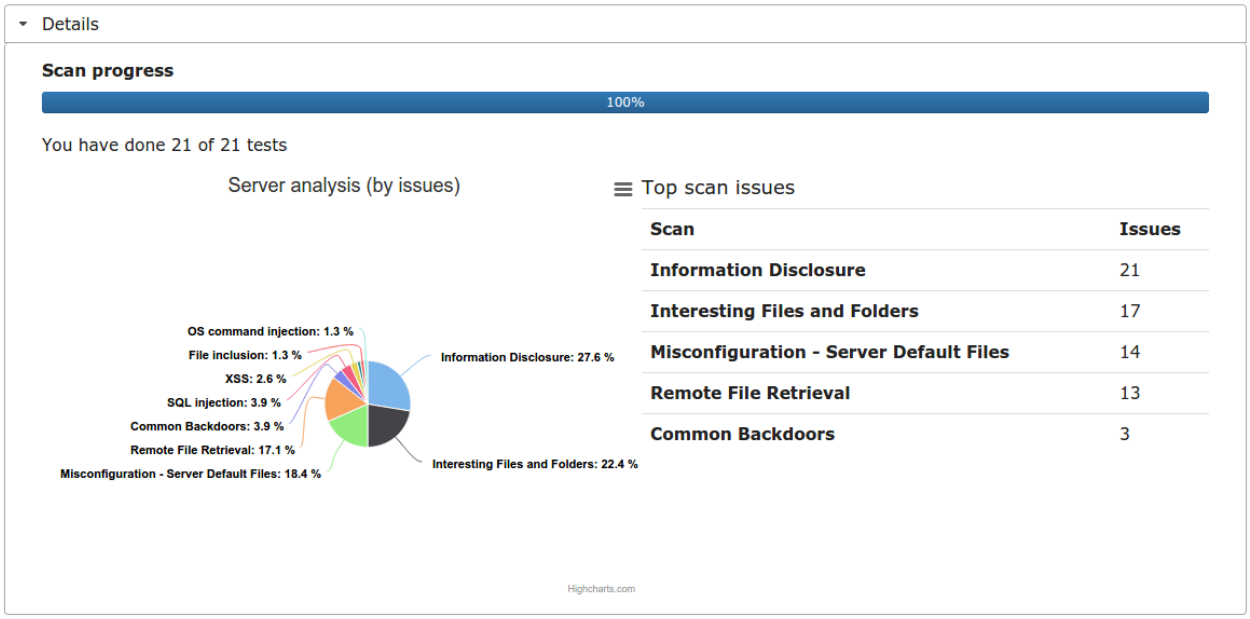
En aquest cas caldria introduir el codi que volguéssim executar dins del camp missatge (que no ha estat netejat de caràcters especials) i qualsevol visitant rebrà com a resultat la execució d'aquest.



Resultat de les proves

Un cop introduïda l'adreça on es troba ubicat el nostre entorn de proves es procedirà a llançar tota la bateria de testos, cabria doncs esperar que com a mínim es detectin les vulnerabilitats anteriorment comentades.

Tot seguit s'exposa el resultat de l'informe executant les proves sobre el servidor server.joancaparros.com que conté el programari DVWA a l'arrel del servidor :



- **Fingerprint scan:** Informació revelada pel servidor

Your server is leaking some information, some of them can help hackers to execute attacks.

Check that you are not revealing anything you want:

Host	up
Address type	ipv4
Address	84.89.0.74
OS	name: Linux 2.6.19 - 2.6.31 accuracy: 97
Web Server Information disclosure	name: http extrainfo: (Ubuntu) product: Apache httpd cpe: cpe:/a:apache:http_server:2.2.22 version: 2.2.22

- **Port scan:** ports oberts en el servidor escanejat

Network ports are the entry points to a server or workstation that is connected to the Internet. A service that listens on a port is able to receive data from a client, process it and send a response back. Malicious clients can sometimes exploit vulnerabilities in the server code so they gain access to sensitive data or execute malicious code on the machine remotely.

Remember to restrict those ports that are not strictly requireds:

Port	Info
22	method: table name: ssh conf: 3
80	method: table name: http conf: 3
443	method: table name: https conf: 3
3306	method: table name: mysql conf: 3

- **Interesting files:** El servidor revela informació sensible o mètodes interessants de ser estudiats de cara a possibles atacs.

Some web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

We have found some interesting files that you should check:

Affected URL: #TEMPL_ITEM_NAME_LINK#

- **Description**: #TEMPL_MSG#

  Method: #TEMPL_HTTP_METHOD#

Affected URL: http://server.joancaparros.com:80/robots.txt

- **Description**: Server leaks inodes via ETags, header found with file /robots.txt, inode: 292357, size: 26, mtime: Wed May 1 09:46:46 2013

  Method: GET

- **Description**: "robots.txt" contains 1 entry which should be manually viewed.

  Method: GET

Affected URL: http://server.joancaparros.com:80/login.php

- **Description**: /login.php: Admin login page/section found.

  Method: GET

Affected URL: http://server.joancaparros.com:80/phpmyadmin/

- **Description**: Uncommon header 'x-ob_mode' found, with contents: 0

  Method: GET

- **Description**: /phpmyadmin/: phpMyAdmin directory found

  Method: GET

Affected URL: http://server.joancaparros.com:80/index

- **Description**: Uncommon header 'tcn' found, with contents: list

  Method: GET

- **Description**: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See

http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php

Method: GET

Affected URL: http://server.joancaparros.com:80/

- **Description**: Cookie PHPSESSID created without the httponly flag

  Method: GET

- **Description**: Cookie security created without the httponly flag

  Method: GET

- **Description**: Retrieved x-powered-by header: PHP/5.4.38-1+deb.sury.org~precise+2

  Method: GET

- **Description**: The anti-clickjacking X-Frame-Options header is not present.

  Method: GET

- **Description**: The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

  Method: GET

- **Description**: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

  Method: GET

- **Description**: Web Server returns a valid response with junk HTTP methods, this may cause false positives.

  Method: UXSYQPEW

Affected URL: http://server.joancaparros.com:80/login/

- **Description**: /login/: This might be interesting...

  Method: GET

  References of this vulnerability: http://osvdb.org/show/osvdb/3092

Affected URL: http://server.joancaparros.com:80/security/web_access.html

➢ **Description**: /security/web_access.html: This might be interesting... has been seen in web logs from an unknown scanner.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/3093

- **Misconfiguration - Default Files:** El servidor posseïx fitxers mal configurats o per defecte que deixen al descobert informació del servidor.

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

It seems that there are some missconfigured or default files, please check the next list:

Affected URL: #TEMPL_ITEM_NAME_LINK#

➢ **Description**: #TEMPL_MSG#

Method: #TEMPL_HTTP_METHOD#

Affected URL: http://server.joancaparros.com:80/robots.txt

➢ **Description**: Server leaks inodes via ETags, header found with file /robots.txt, inode: 292357, size: 26, mtime: Wed May 1 09:46:46 2013

Method: GET

➢ **Description**: "robots.txt" contains 1 entry which should be manually viewed.

Method: GET

Affected URL: http://server.joancaparros.com:80/index

➢ **Description**: Uncommon header 'tcn' found, with contents: list

Method: GET

- **Description**: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php

  Method: GET

Affected URL: http://server.joancaparros.com:80/

- **Description**: Cookie PHPSESSID created without the httponly flag

  Method: GET

- **Description**: Cookie security created without the httponly flag

  Method: GET

- **Description**: Retrieved x-powered-by header: PHP/5.4.38-1+deb.sury.org~precise+2

  Method: GET

- **Description**: The anti-clickjacking X-Frame-Options header is not present.

  Method: GET

- **Description**: The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

  Method: GET

- **Description**: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

  Method: GET

- **Description**: Web Server returns a valid response with junk HTTP methods, this may cause false positives.

  Method: YXKGAQLJ

Affected URL: http://server.joancaparros.com:80/config/

- **Description**: /config/: Directory indexing found.

  Method: GET

  References of this vulnerability: http://osvdb.org/show/osvdb/3268

Affected URL: http://server.joancaparros.com:80/docs/

➢ **Description**: /docs/: Directory indexing found.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/3268

- **Information Disclosure:** Es filtra informació degut a problemes de programació o configuració del propi servidor.

Information disclosure is considered to be a serious threat, wherein an application reveals too much sensitive information, such as mechanical details of the environment, web application, or user-specific data. Subtle data may be used by an attacker to exploit the target hosting network, web application, or its users. Therefore, leakage of sensitive data should be limited or prevented whenever possible.

Please check the next list of files that are exposing some sensitive information:

Affected URL: #TEMPL_ITEM_NAME_LINK#

➢ **Description**: #TEMPL_MSG#

Method: #TEMPL_HTTP_METHOD#

Affected URL: http://server.joancaparros.com:80/robots.txt

➢ **Description**: Server leaks inodes via ETags, header found with file /robots.txt, inode: 292357, size: 26, mtime: Wed May 1 09:46:46 2013

Method: GET

➢ **Description**: "robots.txt" contains 1 entry which should be manually viewed.

Method: GET

Affected URL: http://server.joancaparros.com:80/config/

➢ **Description**: /config/: Directory indexing found.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/3268

- **Description**: /config/: Configuration information may be available remotely.

  Method: GET

Affected URL: http://server.joancaparros.com:80/phpinfo.php

- **Description**: /phpinfo.php: Output from the phpinfo() function was found.

  Method: GET

- **Description**: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.

  Method: GET

- **References of this vulnerability**: http://osvdb.org/show/osvdb/3233

Affected URL: http://server.joancaparros.com:80/index

- **Description**: Uncommon header 'tcn' found, with contents: list

  Method: GET

- **Description**: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php

  Method: GET

Affected URL: http://server.joancaparros.com:80/

- **Description**: Cookie PHPSESSID created without the httponly flag

  Method: GET

- **Description**: Cookie security created without the httponly flag

  Method: GET

- **Description**: Retrieved x-powered-by header: PHP/5.4.38-1+deb.sury.org~precise+2

Method: GET

- **Description**: The anti-clickjacking X-Frame-Options header is not present.

  Method: GET

- **Description**: The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

  Method: GET

- **Description**: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

  Method: GET

- **Description**: Web Server returns a valid response with junk HTTP methods, this may cause false positives.

  Method: VTMNBTQD

Affected URL:
http://server.joancaparros.com:80/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

- **Description**: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.

  Method: GET

  References of this vulnerability: http://osvdb.org/show/osvdb/12184

Affected URL:
http://server.joancaparros.com:80/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42

- **Description**: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.

  Method: GET

  References of this vulnerability: http://osvdb.org/show/osvdb/12184

Affected URL:
http://server.joancaparros.com:80/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42

➢ **Description**: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/12184

Affected URL:
http://server.joancaparros.com:80/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42

➢ **Description**: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/12184

Affected URL: http://server.joancaparros.com:80/docs/

➢ **Description**: /docs/: Directory indexing found.

Method: GET

References of this vulnerability: http://osvdb.org/show/osvdb/3268

- **Common Backdoors:** El resultat d'aquesta prova és positiva, arachni interpreta la vulnerabilitat per sql injection es pot considerar en si una porta d'accés a la informació del nostre servidor, així doncs els resultats coincidiran amb el resultat del tipus de vulnerabilitat esmentat.

- **Common Backup Files/Folders:** Resultat negatiu, no es presenten arxius de backup dins de l'aplicació.
- **SQL injection:** Primera prova on l'escanner hauria d'obtenir resultats, el resultat és **positiu** trobant no només les pàgines web destinades a provar les vulnerabilitats per sql injection i sql blind injection, sinó que ens indica que el formulari inicialment creat per dur a terme atacs de força bruta també és vulnerable davant d'aquests tipus d'atac.

An SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

Affected URL: http://server.joancaparros.com/vulnerabilities/sqli_blind/

- **Description**: Due to the requirement for dynamic content of today's web applications, many rely on a database backend to store data that will be called upon and processed by the web application (or other programs). Web applications retrieve data from the database by using Structured Query Language (SQL) queries. To meet demands of many developers, database servers (such as MSSQL, MySQL, Oracle etc.) have additional built-in functionality that can allow extensive control of the database and interaction with the host operating system itself. An SQL injection occurs when a value originating from the client's request is used within a SQL query without prior sanitisation. This could allow cyber-criminals to execute arbitrary SQL code and steal data or use the additional functionality of the database server to take control of more server components. The successful exploitation of a SQL injection can be devastating to an organisation and is one of the most commonly exploited web application vulnerabilities. This injection was detected as Arachni was able to inject specific SQL queries, that if vulnerable, result in the responses for each request being delayed before being sent by the server. This is known as a time-based blind SQL injection vulnerability.
- **Severity**: high
    - **Form element affected**
    - **Action**: http://server.joancaparros.com/dvwa/vulnerabilities/sqli_blind/
        - **Method**: get
        - **Affected input name**: id
        - **Affected inputs values:**
            - "id": "1' and sleep(16)='"
            - "Submit": "Submit"

- **References of this vulnerability**: OWASP: http://www.owasp.org/index.php/Blind_SQL_Injection MITRE - CAPEC:http://capec.mitre.org/data/definitions/7.html WASC:

---

http://projects.webappsec.org/w/page/13246963/SQL%20Injection W3 Schools: http://www.w3schools.com/sql/sql_injection.asp

---

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/sqli/

- **Description**: Due to the requirement for dynamic content of today's web applications, many rely on a database backend to store data that will be called upon and processed by the web application (or other programs). Web applications retrieve data from the database by using Structured Query Language (SQL) queries. To meet demands of many developers, database servers (such as MSSQL, MySQL, Oracle etc.) have additional built-in functionality that can allow extensive control of the database and interaction with the host operating system itself. An SQL injection occurs when a value originating from the client's request is used within a SQL query without prior sanitisation. This could allow cyber-criminals to execute arbitrary SQL code and steal data or use the additional functionality of the database server to take control of more server components. The successful exploitation of a SQL injection can be devastating to an organisation and is one of the most commonly exploited web application vulnerabilities. This injection was detected as Arachni was able to inject specific SQL queries, that if vulnerable, result in the responses for each request being delayed before being sent by the server. This is known as a time-based blind SQL injection vulnerability.
- **Severity**: high
    - **Form element affected**
    - **Action**: http://server.joancaparros.com/dvwa/vulnerabilities/sqli/
        - **Method**: get
        - **Affected input name**: id
        - **Affected inputs values:**
            - "id": "1' and sleep(16)='"
            - "Submit": "Submit"
- **References of this vulnerability**: OWASP: http://www.owasp.org/index.php/Blind_SQL_Injection MITRE - CAPEC:http://capec.mitre.org/data/definitions/7.html WASC: http://projects.webappsec.org/w/page/13246963/SQL%20Injection W3 Schools: http://www.w3schools.com/sql/sql_injection.asp

---

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/brute/

- **Description**: Due to the requirement for dynamic content of today's web applications, many rely on a database backend to store data that will be called upon and processed by the web application (or other programs). Web applications retrieve data from the database by using Structured Query Language (SQL) queries. To meet demands of many developers, database servers (such as MSSQL, MySQL, Oracle etc.) have

additional built-in functionality that can allow extensive control of the database and interaction with the host operating system itself. An SQL injection occurs when a value originating from the client's request is used within a SQL query without prior sanitisation. This could allow cyber-criminals to execute arbitrary SQL code and steal data or use the additional functionality of the database server to take control of more server components. The successful exploitation of a SQL injection can be devastating to an organisation and is one of the most commonly exploited web application vulnerabilities. This injection was detected as Arachni was able to inject specific SQL queries, that if vulnerable, result in the responses for each request being delayed before being sent by the server. This is known as a time-based blind SQL injection vulnerability.

- ➢ **Severity**: high
  - **Form element affected**
  - **Action**: http://server.joancaparros.com/dvwa/vulnerabilities/brute/
    - **Method**: get
    - **Affected input name**: username
    - **Affected inputs values:**
      - "username": "any_name' or sleep(16) # "
      - "password": "5543!%any_secret"
      - "Login": "Login"
- ➢ **References of this vulnerability**: OWASP: http://www.owasp.org/index.php/Blind_SQL_Injection MITRE - CAPEC:http://capec.mitre.org/data/definitions/7.html WASC: http://projects.webappsec.org/w/page/13246963/SQL%20Injection W3 Schools: http://www.w3schools.com/sql/sql_injection.asp

- **NoSQL injection**: Resultat negatiu, com cabria esperar no s'han trobat vulnerabilitats relacionades amb aquest tipus d'atac.

- **CSRF detection**: El resultat d'aquest anàlisis és positiu, és estrany veue que la pàgina indicada és la prova referent al captcha i no es fa referència a la prova especifica de CSRF. Semblaria que Arachni com la resta d'escanners comercials tenen un dels seus punts febles en la detecció d'aquesta vulnerabilitat i és per això que no d'indiquen la totalitat de les vulnerabilitats que hauria d'indicar.

CSRF is an attack that forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email or chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and

operation, when it targets a normal user. If the targeted end user is the administrator account, a CSRF attack can compromise the entire web application.

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/captcha/

- **Description**: In the majority of today's web applications, clients are required to submit forms which can perform sensitive operations. An example of such a form being used would be when an administrator wishes to create a new user for the application. In the simplest version of the form, the administrator would fill-in: * Name * Password * Role (level of access) Continuing with this example, Cross Site Request Forgery (CSRF) would occur when the administrator is tricked into clicking on a link, which if logged into the application, would automatically submit the form without any further interaction. Cyber-criminals will look for sites where sensitive functions are performed in this manner and then craft malicious requests that will be used against clients via a social engineering attack. There are 3 things that are required for a CSRF attack to occur: 1. The form must perform some sort of sensitive action. 2. The victim (the administrator the example above) must have an active session. 3. Most importantly, all parameter values must be **known** or **guessable**. Arachni discovered that all parameters within the form were known or predictable and therefore the form could be vulnerable to CSRF. _Manual verification may be required to check whether the submission will then perform a sensitive action, such as reset a password, modify user profiles, post content on a forum, etc._
- **Severity**: high
  - **Form element affected**
  - **Action**: http://server.joancaparros.com/dvwa/vulnerabilities/captcha/
    - **Method**: post
    - **Affected input name:**
    - **Affected inputs values**:
      - "step": "1"
      - "password_new": ""
      - "password_conf": ""
      - "Change": "Change"
- **References of this vulnerability**: Wikipedia: http://en.wikipedia.org/wiki/Cross-site_request_forgery OWASP:http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF) CGI Security: http://www.cgisecurity.com/csrf-faq.html

- **Code injection**: Cap problema detectat, resultat correcte.

- **LDAP injection**: Cap problema detectat, resultat correcte.

- **File inclusion**: L'escanner detecta una vulnerabilitat d'inclusió de fitxers, modificant un dels links ens permet accedir a fitxers del propi servidor, posant en risc la entegritat del d'aquest, ja que deixa exposats per exemple els noms d'usuaris que tenen accés a la màquina.

The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a "dynamic file inclusion" mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

Affected URL: http://server.joancaparros.com/dvwa/

- ➢ **Description**: Web applications occasionally use parameter values to store the location of a file which will later be required by the server. An example of this is often seen in error pages, where the actual file path for the error page is stored in a parameter value -- for example `example.com/error.php?page=404.php`. A file inclusion occurs when the parameter value (ie. path to file) can be substituted with the path of another resource on the same server, effectively allowing the displaying of arbitrary, and possibly restricted/sensitive, files. Arachni discovered that it was possible to substitute a parameter value with another resource and have the server return the contents of the resource to the client within the response.
- ➢ **Severity**: high
  - ○ **Link element affected**
  - ○ **Method**: get
    - ■ **Affected inputs values:**
      "page": "/etc/passwd"
      Full affected request
      GET /dvwa/vulnerabilities/fi/?page=%2Fetc%2Fpasswd HTTP/1.1
      Host: server.joancaparros.com Accept-Encoding: gzip, deflate
      User-Agent: Arachni/v1.0.6 Accept:
      text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Cookie:
      PHPSESSID=l3h9omrkmtm11lskuhgksq8684;security=low
- ➢ **References of this vulnerability**: OWASP:
  https://www.owasp.org/index.php/PHP_File_Inclusion

- **Response splitting**: Cap problema detectat, resultat correcte.

- **OS command injection**: El resultat d'aquesta prova és positiva, Arachni detecta correctament la pàgina destinada a aquest tipus en concret de vulnerabilitat i ens indica les referències cap a les webs més populars sobre seguretat per tal de poder corregir de la manera més adequada aquest tipus d'error en la programació.

OS command injection is a technique used via a web interface in order to execute OS commands on a web server. The user supplies operating system commands through a web interface in order to execute OS commands. Any web interface that is not properly sanitized is subject to this exploit. With the ability to execute OS commands, the user can upload malicious programs or even obtain passwords. OS command injection is preventable when security is emphasized during the design and development of applications.

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/exec/

- ➢ **Description**: To perform specific actions from within a web application, it is occasionally required to run Operating System commands and have the output of these commands captured by the web application and returned to the client. OS command injection occurs when user supplied input is inserted into one of these commands without proper sanitisation and is then executed by the server. Cyber-criminals will abuse this weakness to perform their own arbitrary commands on the server. This can include everything from simple `ping` commands to map the internal network, to obtaining full control of the server. Arachni was able to inject specific Operating System commands and have the output from that command contained within the server response. This indicates that proper input sanitisation is not occurring.
- ➢ **Severity**: high
  - **Form element affected**
  - **Action**: http://server.joancaparros.com/dvwa/vulnerabilities/exec/
    - **Method**: post
    - **Affected input name**: ip
    - **Affected inputs values**:
      - "ip": "1 ; /bin/cat /etc/passwd ; "
      - "submit": "submit"
- ➢ **References of this vulnerability**: OWASP: http://www.owasp.org/index.php/OS_Command_Injection WASC:http://projects.webappsec.org/w/page/13246950/OS%20Commanding

- **Remote file inclusion**: Arachni no és capaç de detectar cap anomalia derivada de l'explotació d'aquesta vulnerabilitat.

- **Remote File Retrieval:** Els resultats de Nikto són pobres, indicant arxius que ja varem descobrir utilitzant tècniques de revelació de fitxers interessants o de descobriment de dades.

Inside Web Root. Resource allows remote users to retrieve unauthorized files from within the web server's root directory.

Affected URL: #TEMPL_ITEM_NAME_LINK#

➢ **Description**: #TEMPL_MSG#

Method: #TEMPL_HTTP_METHOD#

Affected URL: http://server.joancaparros.com:80/dvwa/robots.txt

➢ **Description**: Server leaks inodes via ETags, header found with file /dvwa/robots.txt, inode: 292357, size: 26, mtime: Wed May 1 09:46:46 2013

Method: GET

➢ **Description**: "robots.txt" contains 1 entry which should be manually viewed.

Method: GET

Affected URL: http://server.joancaparros.com:80/dvwa/

➢ **Description**: Cookie PHPSESSID created without the httponly flag

Method: GET

➢ **Description**: Cookie security created without the httponly flag

Method: GET

➢ **Description**: Retrieved x-powered-by header: PHP/5.4.38-1+deb.sury.org~precise+2

Method: GET

- **Description**: The anti-clickjacking X-Frame-Options header is not present.

  Method: GET

- **Description**: The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS

  Method: GET

- **Description**: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type

  Method: GET

- **Description**: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS

  Method: OPTIONS

- **Description**: Web Server returns a valid response with junk HTTP methods, this may cause false positives.

  Method: OANLFHDY

Affected URL: http://server.joancaparros.com:80/dvwa/index

- **Description**: Uncommon header 'tcn' found, with contents: list

  Method: GET

- **Description**: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php

  Method: GET

- **Unvalidated redirects**: Cap problema detectat, resultat correcte.

- **XPath injection**: Cap problema detectat, resultat correcte.

- **XSS**: El resultat d'aquesta prova és positiu, les pàgines afectades no corresponen a les destinades a la explotació del tipus de vulnerabilitat sinó que moltes altres estarien patint aquest tipus de vulnerabilitats i caldria aplicar els metòdes de depuració adeqüats per tal de mitigar els resultats d'un possible atac.

Cross-site Scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user in the output it generates without validating or encoding it.

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/upload/

- ➢ **Description**: Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction. Cross Site Scripting (XSS) allows clients to inject scripts into a request and have the server return the script to the client in the response. This occurs because the application is taking untrusted data (in this example, from the client) and reusing it without performing any validation or sanitisation. If the injected script is returned immediately this is known as reflected XSS. If the injected script is stored by the server and returned to any client visiting the affected page, then this is known as persistent XSS (also stored XSS). Arachni has discovered that it is possible to insert script content directly into an HTML event attribute. For example `<div toponmouseover="x=INJECTION_HERE"</div>`, where `INJECTION_HERE` represents the location where the Arachni payload was detected.
- ➢ **Severity**: high
- ➢ **References of this vulnerability**: ha.ckers: http://ha.ckers.org/xss.html Secunia: http://secunia.com/advisories/9716/ WASC:http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting OWASP:www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

---

Affected URL: http://server.joancaparros.com/dvwa/vulnerabilities/fi/?page=include.php

- ➢ **Description**: Client-side scripts are used extensively by modern web applications. They perform from simple functions (such as the formatting of text) up to full manipulation of client-side data and Operating System interaction. Cross Site Scripting (XSS) allows clients to inject scripts into a request and have the server return the script to the client in the response. This occurs because the application is taking untrusted data (in this example, from the client) and reusing it without performing any validation or sanitisation. If the injected script is returned immediately this is known as reflected XSS. If the injected script is stored by the server and returned to any client visiting the affected page, then this is known as persistent XSS (also stored XSS). Arachni has discovered that it is possible to insert script content directly into an HTML event attribute. For example `<div onmouseover="x=INJECTION_HERE"</div>`, where `INJECTION_HERE` represents the location where the Arachni payload was detected.
- ➢ **Severity**: high
- ➢ **References of this vulnerability**: ha.ckers: http://ha.ckers.org/xss.html Secunia: http://secunia.com/advisories/9716/ WASC:http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting OWASP:www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

- **Source code disclosure**: Cap problema detectat, resultat correcte.

Conclusions

Aquest treball final de Màster ha estat orientat a la creació d'un projecte de construcció d'una eina per a escanejar aplicacions webs, posant en pràctica tots els coneixements adquirits durant les assignatures del Màster interuniversitari de seguretat de les tecnologies de la informació i de les comunicacions.

La principal dificultat d'aquest projecte en particular, radica en l'estudi de les eines disponibles en el mercat i escollir quines d'elles serien les encarregades de dur a terme cada una de les proves destinades a avaluar cada una de les vulnerabilitats més comunes descrites per l'OWASP, donat que el meu camp de treball laboral és el desenvolupament web s'ha intentat aprofundir en tecnologies que no estava habituat, com les cues de tasques i la gestió d'aquestes com a tasques en paral·lel de la pròpia aplicació web.
També destacar que s'han utilitzat repositoris de control de versions per permetre un control de canvis durant la fase de disseny ja que es varen provar varies alternatives de codi. Durant la realització de l'aplicació va fer falta d'un canvi d'entorn VPS degut al consum de memòria i CPU que alguna de les eines requeria, així doncs i gràcies a la facilitat d'instal·lació no va suposar una gran complicació.

Tot i que la primera versió va ser realitzada molt ràpidament les petites peculiaritats de cada eina integrada va fer que es requerís d'una fase de millora continua llarga per tal d'adaptar la base que s'havia creat inicialment.

Com a conclusió final aquest projecte permet valorar els diferents escànners que podem trobar en el mercat, les seves formes d'execució, les vulnerabiltats més comunes reportades per l'entitat més important en aquest tema (OWASP) i com a punt final la importància de mostrar totes les dades d'una forma amigable per tal de veure a simple vista en quins punts cal requerir d'una especial atenció per corregir i assegurar els nostres servidors.

Treballs futurs o punts de millora

El sistema utilitza com un dels seus punts estrella un sistema de gestió de tasques que treballa en paral·lel a la pròpia aplicació, les tasques podrien executar-se en una o més màquines independents, aquesta idea ens podria donar com a resultat ideal una aplicació on cada usuari tingués un consumidor de tasques independents i on sempre pogués tenir només una tasca en execució, està clar que aquesta versió requeriria d'una gran capacitat de maquinari que pogués suportar com a màxim una execució concurrent igual al nombre d'usuaris registrats dins de l'aplicació.

Un altre treball futur i que podria ubicar-se en el apartat de settings inclòs dins dels primers mockups és el de configuració avançada de les crides que s'executen, permetent que els propis usuaris defineixin les seves pròpies comandes utilitzant les eines de les que disposen, incĺús poder triar si l'escaneig d'una prova en concret volen que s'executi amb una eina o amb una altre si les dues estan definides dins de l'aplicació.

Bibliografia

[1] *OWASP The Open Web Application Security Project*
https://www.owasp.org/index.php/Main_Page
[2] OWAP Top 10 - 2013
https://www.owasp.org/index.php/Top_10_2013-Top_10
[3] Penetration Testing
ftp://ftp.informatik.uni-stuttgart.de/pub/library/medoc.ustuttgart_fi/.FACH-0137/FACH-0137.pdf
[4] Arachni Web Application Security Scanner
http://www.tic.udc.es/~nino/blog/psi/2012/arachni.pdf
[5] Informe de Amenazas CCN-CERT IA-06/14 |Recomendaciones generales ante ataques a servicios web
[6] *OWASP Testing Guide v4. OWASP Testing Project.*
https://www.owasp.org/index.php/OWASP_Testing_Project
[7] Arachni Command line user interface
https://github.com/Arachni/arachni/wiki/Command-line-user-interface