PROJECTE DE FINAL DE MÀSTER

CREACIÓ D'UNA EINA DE DETECCIÓ DE VULNERABILITATS WEB

Autor: Rafael Páez Jaime
Tutors: Jordi Duch i Robert Rallo
Gener de 2013

Agraeixo a totes aquelles persones i amics que m'han donat suport al llarg de l'elaboració d'aquest projecte.

Rafael Páez Jaime. 2013

Resum introductori

Internet és un món que està en constant avanç. Per aquest motiu cada cop és utilitzat per més persones i a la vegada s'ofereixen més serveis web, ja que amb aquests s'aconsegueix arribar a un nombre major de gent i així obtenir uns “beneficis” majors. Aquest fet, implica que hi hagin més pàgines web vulnerables i que el nombre d'atacants es vegi incrementat. Per aquest motiu, amb aquest projecte, s'espera realitzar un estudi sobre la seguretat que poden oferir aquests serveis en Internet, ja sigui per la seguretat dels propis serveis o la dels usuaris que els utilitzen.

En aquest document es realitzarà un estudi bastant complert sobre algunes de les vulnerabilitats que poden existir en els diferents serveis webs que utilitzen els usuaris i que poden ser aprofitades per cert tipus de persones, i com podem detectar aquestes per tal de poder corregir-les.

Per poder saber com protegir-nos davant d'aquest tipus de situacions, primerament farem un estudi exhaustiu de com poden ser explotades aquestes vulnerabilitats en els diferents escenaris que podem trobar i una vegada sabem com funcionen els atacs que es poden realitzar, ens centrarem en quines eines ens poden ajudar a detectar-les. Centrant-nos en les funcionalitats d'aquestes i en la forma de treball tenen per tal de poder comparar aquelles aplicacions que existeixen en Internet.

Ja que tots aquests aspectes són tractats d'una forma bastant teòrica, s'ha considerat important realitzar un sistema de detecció de vulnerabilitats web propi per poder analitzar amb més detall el comportament d'aquest tipus d'aplicacions i així poder adaptar aquestes eines a les nostres necessitats segons els requeriments de cada situació.

Aquest sistema es basarà en un eina que serà capaç de detectar certs tipus de vulnerabilitats web d'una manera el més automàtica possible, per tal de que l'usuari no hagi de tenir uns coneixements amplis sobre la detecció de vulnerabilitats web.

Gràcies a aquest projecte, s'ha aconseguit ampliar considerablement el coneixement sobre aquest tipus d'aplicacions, sabent com funcionen aquestes internament i com poden ser adaptades a les nostres necessitats.

Índex de continguts

Índex de taules

Índex de figures

Capítol 1: Introducció

Cada cop són més les empreses i companyies que ofereixen serveis a través de Internet. Aquest fet està remarcat per la quantitat de persones que la usen a diari, i encara més amb totes les facilitats que s'estan donant per connectar-se des de qualsevol lloc, com pot ser per exemple des del mòbil. Això fa que el número d'usuaris d'Internet hagi crescut considerablement, i per tant, els serveis oferts a Internet hagin guanyat una importància més que remarcable.

El fet de que cada cop hi hagin més usuaris connectats a Internet que utilitzen els mateixos serveis, implica que també hi ha un nombre molt gran de persones que potser no utilitzaran aquests serveis de la manera més "correcte" possible, i aquí és on entren en funció conceptes com privacitat i seguretat.

Com sabem, un mateix servei web pot ser utilitzat per diferents persones, però cadascuna d'aquestes espera que l'aplicació mantingui les seves dades únicament per al propi ús, sense que altres persones puguin accedir a elles, o fins i tot, existeixen serveis que han de ser accessibles només per a certes persones. Per poder realitzar això, és necessari identificar a aquestes persones/usuaris i per fer-ho, s'utilitzen mètodes d'autenticació i d'autorització, els quals, normalment, a través d'unes credencials (que solen ser un nom d'usuari i una contrasenya) permeten o deneguen l'accés a certs recursos o serveis webs. Permetent així que es mantingui la seguretat dels diferents usuaris. A pesar d'aquests sistemes de seguretat establerts, hi han certes persones que intenten evadir-los per poder accedir a recursos o informació a la que no estan permesos accedir.

En aquest document s'intentarà fer una recopilació i un anàlisi dels diferents atacs que es podrien realitzar contra els sistemes d'autenticació i d'autorització utilitzats per les aplicacions web, i com, aquelles persones encarregades de la seva realització, podrien comprovar si els seus sistemes són vulnerables a aquests atacs o no, i per tant, permetent l'accés a usuaris sense els suficients permisos.

1.1. Estructuració de la memòria

Per tal de que el lector pugui saber com està estructurada la memòria del projecte, es procedirà a realitzar una breu introducció de cadascun dels capítols que es poden trobar.

En el capítol número dos, s'explicaran les principals motivacions que han portat a l'elaboració d'aquest projecte i quins són els objectius que s'han establerts per al mateix

En el capítol tres podrem trobar l'estat de l'art, on es farà un estudi sobre els diferents tipus de vulnerabilitats referents a l'autorització i l'autenticació que existeixen en els serveis web que trobem en Internet. A més a més, també trobarem un anàlisis de les diferents eines que es poden utilitzar per fer proves de pentesting web[1] contra aquests serveis, com poden ser Burp o WebScarab per exemple.

Després de l'estat de l'art (capítol 4) s'establiran totes les especificacions que haurà de tenir el nostre sistema i que s'oferiran a l'usuari, indicant totes les característiques que contindrà i els requeriments existents.

En el cinquè capítol es realitza tot l'anàlisi de l'elaboració del projecte, començant amb les idees utilitzades i els requeriments establerts. Aquest, el trobarem dividit en uns altres tres subapartats on trobarem més detalladament l'explicació del procés de creació del projecte.

El primer d'aquests és l'anàlisi previ realitzat per poder estudiar correctament els diferents casos d'ús necessaris del projecte i veure clarament quines són les especificacions que aquest requereix.

En l'apartat del disseny es on, basant-nos en l'anàlisi previ dels requeriments, posem solució a les especificacions establertes.

Per últim, en el tercer apartat, es detalla tota la implementació realitzada durant el projecte, exposant com s'han resolt cada una de les especificacions i explicant cada una de les funcionalitats del projecte, des de la més mínima fins a la més complexa.

En el capítol sis, es on trobarem tot el conjunt de probes realitzades sobre l'aplicació. Analitzant tots els resultats obtinguts per comprovar si el resultat ha sigut l'esperat o no.

En el sèptim i últim capítol, es donaran les conclusions obtingudes al llarg de l'elaboració del projecte, detallant si s'han aconseguit els objectius desitjats o no, i els treballs futurs que existeixen per aquest.

En l'apartat de treballs futurs, trobarem les millores que es podrien realitzar en versions posteriors del nostre sistema de detecció de vulnerabilitats web, no només mencionat-les, sinó ampliant la informació indicant possibles formes de realitzar-les, ja que s'ha començat a investigar sobre el tema.

---

[1] Conjunt de proves que es realitzen sobre serveis web per tal d'avaluar el seu grau de seguretat.

Capítol 2: Motivació i objectius

En aquest capítol podrem observar quines han sigut les motivacions que ens han fet realitzar aquest projecte i quins són els principals objectius que s'esperen aconseguir amb la seva elaboració.

2.1. Motivació

Cada dia que passa, Internet es fa més i més important. El número de persones que es connecten a diari es major, i cada cop més les empreses depenen d'Internet per realitzar el seu treball. El que fa que aquest “món” sigui cada vegada més important.

A demés de totes aquelles persones que es connecten a la xarxa per poder parlar amb els seus amics i familiars, poder llegir diaris online i veure els resultats dels partits, existeixen altres persones que utilitzen Internet per al seu treball, i que realitzen diàriament tot tipus de gestions a través d'aquest medi, tramitant totes les transferències bancàries a través del seu propi ordinador o simplement intercanviant informació entre diferents punts de l'empresa, ja sigui entre equips d'una xarxa local o amb l'oficina d'una altra ciutat.

Aquests aspectes influeixen en que cada vegada més, Internet sigui el punt de mira de moltes persones, les quals no han de perquè tenir unes intencions ètiques, el que implica que totes aquests comunicacions poden veure's compromeses per una tercera persona.

Aquests fets, impliquen que cada cop el número de pàgines web existents en Internet sigui major, i que a la vegada el número de visitants s'incrementi, cosa que implica que la seguretat d'aquestes pugui ser vulnerada més fàcilment, i per tant tota la informació amb la que treballa. Sabent això, podem veure que el fet que un servei web tingui alguna vulnerabilitat o error de programació, pot suposar que tota la informació dels usuaris que accedeixen es vegi vulnerada, essent accessible i/o modificable per terceres persones.

Per poder evitar això, el que s'ha de fer es crear pàgines web segures, i per poder garantir

això el màxim possible, existeixen diferents programes i/o eines que utilitzen els webmasters per testejar les seves creacions o els pentesters per verificar la seguretat d'un espai web.

Tots aquests aspectes comentats, són els que han motivat a l'elaboració d'aquest projecte, per poder així comprendre millor com funcionen aquests aplicacions que proporcionaran la "seguretat" als futurs serveis webs utilitzats per milers de persones.

2.2. Objectius

Després d'haver fet un anàlisis sobre les diferents vulnerabilitats webs existents, hem pogut veure com el conjunt d'elles que afecten als sistemes d'autenticació i autorització posseeixen un percentatge bastant alt (podent arribar fins a un 26% si agrupem aquells que poden afectar a aquests mòduls) segons un dels últims estudis fet per ESET el juliol del 2012 [A], tal i com podem veure en l'imatge 1.

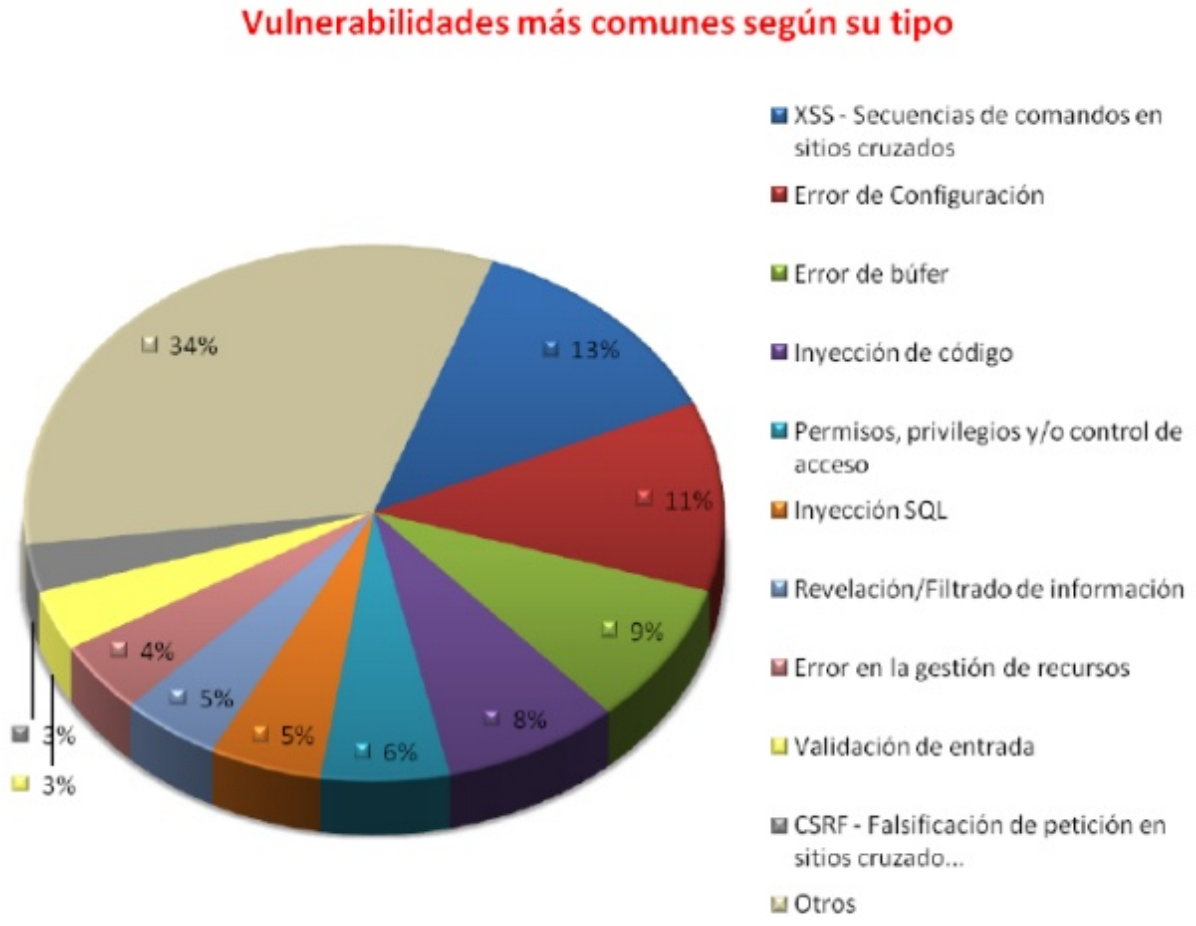

Figura 1. Estadístiques sobre les vulnerabilitats web per tipus segons un estudi realitzat per ESET

Gràcies a l'estudi realitzat prèviament, podem dir que l'objectiu principal d'aquest projecte serà realitzar un sistema que sigui capaç d'analitzar (de la forma més automàtica possible)

una web i detectar aquelles vulnerabilitats que estiguin relacionades amb l'autenticació i l'autorització, facilitant a l'usuari els resultats obtinguts de la manera més clara i concisa possible.

Per fer això, és realitzarà un estudi sobre les diferents eines existents en el mercat per poder saber com treballen aquestes i poder així, desenvolupar la nostra pròpia, arribant inclús a utilitzar algunes funcionalitats ja implementades per aquestes en cas de ser necessari, ja que moltes d'aquestes eines posseeixen una infinitat de configuracions, que permeten adaptar-les a gairebé qualsevol escenari.

Encara que l'usuari de l'aplicació haurà de tindre un mínim de coneixements creació de pàgines web, s'intentarà que el control d'aquesta sigui el més fàcil i intuïtiu possible, permetent així que qualsevol persona amb un mínims de coneixements pugui administrar l'aplicació. Per això, es crearà un entorn gràfic que permeti configurar les diferents funcionalitats del sistema sense necessitat de programar cap codi ni fer configuracions complexes.

Per poder mostrar els resultats, el nostre sistema haurà de ser capaç de recavar aquella informació i proves que siguin necessàries sobre la web a auditar i s'hauran de proporcionar a l'usuari de la manera més concisa possible, per tal de que així sigui ràpid poder veure quines són les vulnerabilitats en la web auditada.

Capítol 3: Estat de l'art

En aquest apartat realitzarem un estudi sobre els diferents vectors d'atac que poden existir per tal de vulnerar els sistemes d'autenticació i autorització dels diferents serveis webs i farem un recorregut per les diferents eines que ajuden a poder detectar i així poder evitar aquests tipus de vulnerabilitats.

3.1. Descripció de vulnerabilitats

Com s'ha comentat en l'apartat anterior, l'autenticació i l'autorització són dos aspectes molt importants alhora de dissenyar una aplicació web, ja que aquests seran els que permetin l'accés o no a segons quines aplicacions o funcions. Per aquest mateix motiu els atacants intenten saltar-se els sistemes de protecció existents per poder accedir a aquells apartats de la web on l'accés no els és autoritzat.

En aquest apartat intentarem descriure els principals atacs o vulnerabilitats existents en aquests processos per tal de poder saber com detectar-los correctament. Per fer-ho més fàcil, els categoritzarem en dos grups: els que intenten explotar el procés d'autenticació i els que intenten explotar el procés d'autorització.

3.1.1. Autenticació

En aquest primer apartat veurem quines són les principals vulnerabilitats i vectors d'atac que poden existir en els sistemes d'autenticació d'un servei web, per tal de poder veure com podem protegir-nos davant d'aquests.

3.1.1.1. Transmissió de credencials a través d'un canal xifrat

Gràcies a als protocols TLS i SSL[B] podem fer que les comunicacions que realitzem siguin xifrades, per tal de que si algú les intercepta no pugui ser capaç d'accedir a la informació enviada. Per aquest motiu, és important que les comunicacions que contenen informació sensible, com poden ser el nom d'usuari i el password, s'enviïn xifrades des del client cap al servidor.

A pesar d'existir el protocol HTTPS [2] i conèixer la importància de xifrar les comunicacions, hi han molts desenvolupadors els quals segueixen utilitzant el protocol HTTP per enviar les credencials d'usuari, ja que aquest és molt més simple de configurar, permetent que d'aquesta manera algú pugui accedir a aquestes dades abans d'arribar al servidor.

3.1.1.2. Enumeració d'usuaris

Normalment la llista d'usuaris d'una aplicació és informació confidencial (són pocs llocs on aquesta és pública) i són els propis usuaris els que han de conèixer el seu nom d'usuari. Tot i això, algunes aplicacions web donen certa informació quan s'introdueixen els valors de login, permetent conèixer si l'usuari entrat existeix en el sistema o no. Amb aquest tipus d'informació, es podria obtenir una llista de tots els usuaris existents, donant peu a possibles atacs de força bruta per obtenir l'accés a l'aplicació.

En l'imatge 2, podem trobar un exemple del login d'una web en el que es diu si un usuari existeix en la seva base de dades o no quan s'envien les dades al servidor.

Un exemple podria ser com la pàgina web de l'imatge 2, on al introduir un usuari no existent, rebem una alerta dient que aquest usuari no existeix a la base de dades, i en canvi, quan l'usuari sí que existeix però la contrasenya és incorrecta, rebem un missatge diferent, dient que la contrasenya facilitada no és correcte. Per tant, amb aquest tipus de prova, podríem saber si un usuari existeix o no en el sistema.

Un altre escenari on es podria obtindre una llista d'usuaris és el cas on els noms d'aquests segueixen un patró definit, com per exemple “comptabilitat1”, “comptabilitat2”, “comptabilitat3”, etc.

[2] Hypertext Transfer Protocol Secure. Protocol d'aplicació basat en HTTP que utilitza connexions SSL i TLS.

Figura 2. Exemple de dades incorrectes en un formulari web de login.

3.1.1.3. Proves de diccionari contra usuaris o comptes predeterminats

Moltes de les aplicacions web d'avui en dia, utilitzen software comú que ha estat dissenyat per a ser utilitzat per aplicacions terceres per tal de facilitar las tasca de la creació o manteniment dels llocs web, com poden ser per exemple els servidors de bases de dades, certs plugins, etc. El problema que això suposa és que moltes vegades la seguretat d'aquest software no és la més idònia i conté algunes bretxes de seguretat o configuracions per defecte que els administradors del lloc web desconeixen o "ignoren".

Dintre d'aquestes configuracions per defecte trobem les credencials d'usuari que s'utilitzen per la configuració del dispositiu, essent moltes d'aquestes iguals i establertes automàticament des del primer moment de la instal·lació, com per exemple "*admin/admin*" o "*user/1234*". Aquestes credencials es poden trobar a la documentació del dispositiu o software, i si no són canviades manualment, continuaran essent les mateixes, cosa que suposa que qualsevol persona que les conegui (o les busqui per Internet) pugui accedir-hi. A més d'això, el problema s'agreuja quan algun d'aquest software no permet eliminar ni modificar els comptes preestablerts.

Cal dir, que encara que hi han aplicacions que sí que han estat ben configurades, els noms d'usuaris i contrasenyes utilitzades són massa febles i/o fàcilment endevinables o el que és encara pitjor, camps de contrasenya en blanc. Això implica que utilitzant atacs de diccionari amb contrasenyes "típiques" es pugui obtindre accés a l'aplicació.

Una altra font d'informació per poder trobar credencials vàlides és el propi codi de les aplicacions, on sovint es poden trobar comentaris amb les contrasenyes utilitzades (que els desenvolupadors han oblidat esborrar) o porcions de codi que fan comparacions del tipus "*if user='admin' then page='/administracio' else page=''/default*'", indicant clarament el valor

de la variable per obtindre l'accés.

3.1.1.4. Proves de força bruta

Quan s'intenten obtindre els credencials d'una aplicació i s'han provat els mètodes descrits anteriorment sense èxit, l'única opció que ens queda es “endevinar” els valors necessaris, i un atac per força bruta ens pot ajudar a trobar-los.

Aquest tipus d'atac consisteix en anar provant totes les combinacions possibles que hi poden haver fins trobar aquella que permet l'accés a l'aplicació. Dit així, sembla molt fàcil trobar qualsevol parell d'usuari i contrasenya, però el principal problema que existeix és el temps requerit per poder provar totes les opcions possibles, ja que una contrasenya de longitud 8 en la que només s'utilitzen les lletres del diccionari en minúscules ja implica un total de més de 200 milions de contrasenyes possibles. Tot i així, si es té algun tipus d'informació sobre els caràcters permesos o longituds, aquest nombre de combinacions podria veure's reduït considerablement, a demés, que existeix la possibilitat de crear combinacions de contrasenyes a partir de patrons o petites modificacions del nom d'usuari (en el cas de que es conegués) com per exemple la contrasenya “r0b3rt” per a l'usuari “robert”.

3.1.1.5. Saltar-se el sistema d'autenticació

Una altra forma de poder ser autenticat, o com a mínim tenir accés a certs recursos restringits per a usuaris, és intentar saltar-se els sistemes d'autenticació que puguin existir, ja que d'aquesta manera no haurem de conèixer les credencials de cap usuari per poder obtindre accés. Per poder saltar-se aquests sistemes, existeixen diferents mètodes, els quals podríem categoritzar en quatre tipus.

El primer d'ells consistiria en intentar accedir directament a URLs que haurien de ser únicament accessibles quan un usuari ha estat prèviament autenticant. Això podria suposar saltar-se el pas de login de l'aplicació i si no hi han suficients controls de seguretat, accedir al recurs sol·licitat. Aquest problema és degut a que els desenvolupadors poden creure que a aquesta pàgina només es pot accedir una vegada s'ha autenticat un usuari, i no pensen en la possibilitat de que una altra persona pot conèixer aquesta URL.

Un altre error bastant comú, és utilitzar variables que indiquen si un usuari està autenticat o no, com podria ser la següent URL “*www.webvulnerable.com/index.php?auth*=1”, on la variable “*auth=1*” indica que l'usuari ha estat validat prèviament. Per tant, si un usuari no autenticat (el qual tindrà “*auth=0*”) modifica el valor de la variable per un “1”, serà capaç de saltar-se el control d'autenticació.

El valors ID de sessió utilitzats per l'aplicació també tenen un paper molt important en la seguretat de l'aplicació web, ja que encara que són valors “aleatoris” que el servidor genera a cada usuari, moltes vegades aquesta “aleatorietat” pot ser predictible (per exemple utilitzant un comptador que genera els valors ID d'una forma incremental), i per tant un atacant podria

modificar el valor ID de sessió enviat al servidor i utilitzar la sessió d'un altre usuari prèviament autenticat.

Per últim, les injeccions SQL[C] poden permetre a un atacant saltar-se el sistema d'autenticació injectant als camps del login una comanda SQL concreta. Com per exemple podria ser el típic cas de SQL injection en els formularis de logins, on s'introdueix "*' or 1=1* --" per tal de crear la següent consulta en el servidor al rebre les dades "*SELECT * FROM users WHERE username= ' or 1=1 -- ' AND password='' "* i poder saltar-se el sistema de login, ja que aquesta consulta sempre retornarà true.

3.1.1.6. Sistemes de recordatori/restauració de contrasenyes

Actualment, la majoria d'aplicacions web que treballen amb credencials d'usuari disposen d'un sistema de recuperació de contrasenya, que permet a un usuari que no recorda la seva poder ser autenticat en el sistema. Aquests sistemes poden estar implementats de moltes formes diferents, algunes d'elles més segures i unes altres més febles. Un exemple podria ser un sistema el qual envia un correu amb el nou password al correu que l'usuari va utilitzar en el moment de fer el registre, o un altre podria ser quan l'aplicació realitza una pregunta secreta a l'usuari que intenta accedir i si aquesta és correcte, li dóna l'opció d'introduir la nova contrasenya sense conèixer l'antiga. Com es pot veure, aquesta segona validació és més feble que la primera, ja que si un atacant intenta accedir al compte d'un usuari sense conèixer la seva contrasenya, en el primer cas haurà de tenir accés al seu correu per poder obtindre la nova, però en el segon cas únicament haurà d'endevinar la pregunta secreta que l'usuari va introduir, informació que depenent de la pregunta utilitzada es podria trobar per Internet o amb algun atac d'enginyeria social.

Deixant a una banda el tema de la restauració de contrasenyes, un altre fet que podria permetre a un atacant obtenir accés al compte d'un usuari és gràcies a l'opció de moltes aplicacions web que permeten emmagatzemar la contrasenya de la web i auto-completar-la quan s'accedeix a la mateixa, ja sigui guardant la contrasenya en el propi navegador o en una cookie. D'aquesta manera qualsevol persona que tingui accés a la màquina podria accedir al seu compte o fins i tot obtenir la contrasenya en clar.

3.1.1.7. Gestió de la caché i sortides de sessió

Com s'ha pogut comprovar amb tots els punts anteriors, l'autenticació d'un usuari és un aspecte a tenir molt en compte en les aplicacions web, però el fet de des-autenticar-lo també ho és, ja que si això no es fa correctament, la sessió pot ser utilitzada per una altra persona sense necessitat de conèixer les credencials. Per poder comprovar si aquest procés es fa correctament, existeixen alguns requeriments que s'haurien de complir per tal de que no es pogués aprofitar la sessió una vegada aquesta hagi sigut (o hauria de ser) tancada.

Un exemple de mala configuració que podria permetre a un atacant fer-se amb la sessió d'un altre usuari podria ser per exemple que quan es fa el logout de l'aplicació, aquest no es faci

correctament i no es tanqui la sessió al servidor, permetent que amb un reenviament de la cookie del client es pugui tornar a activar la sessió de l'usuari.

Un altre error en la creació de les aplicacions web és no establir un temps de tancament de sessió per un usuari inactiu, deixant sempre la sessió d'aquest oberta.

Per últim, comentar també que la funció de caché [3] dels navegadors pot jugar una mala passada als usuaris, ja que es podrien guardar dades en la caché d'aquests i si una altra persona té accés al navegador podria visualitzar aquestes dades simplement donant al botó "enrere".

3.1.1.8. Proves de Captcha

Per tal d'evitar atacs automàtics, moltes aplicacions webs utilitzen CAPTCHAs [D], els quals són petits sistemes que serveixen per a diferenciar màquines i humans a través d'un repte-resposta, com el que podem veure a l'imatge 3, on l'usuari haurà d'entrar les dos paraules que es troben "deformades" per demostrar que és realment una persona. A pesar d'això, existeixen alguns atacs contra aquests mètodes de protecció.

Figura 3. Exemple de CAPTCHA utilitzat en una pàgina web.

El principal atac és la creació de sistemes desxifradors de CAPTCHAs, els quals tenen un percentatge d'encert bastant alt ver certs tipus de CAPTCHAs. Amb aquest es podria llançar un atac automatitzat encara que el sistema estigués protegit per un CAPTCHA.

Després de l'aparició dels desxifradors, els creadors de CAPTCHAs van fer més difícil la codificació d'aquests per a les màquines, però tot i així, moltes aplicacions web poden ser vulnerables encara que el CAPTCHA generat sigui irrompible.

Alguns exemples d'aquests tipus de vulnerabilitats poden ser per exemple quan la solució del repte és enviada al servidor sense xifrar o amb un xifratge molt dèbil que pot ser trencat o quan l'ID del CAPTCHA té associada la seva resposta i a la part del servidor no es compara si l'ID enviat pel client és el mateix que el servidor va enviar, permetent utilitzar sempre el

---

[3] Mecanisme d'emmagatzematge temporal per a documents web per tal de reduir el temps de carga o l'ample de banda utilitzat.

mateix repte-resposta (el qual ja coneixem prèviament).

3.1.1.9. Autenticació de múltiples factors

L'autenticació de múltiples factors s'utilitza per no dependre només d'un únic valor per realitzar el login, i per tant així, incrementar la seguretat del mateix. Normalment aquest segon valor (encara que poden haver-hi més) és “alguna cosa que es té” o “alguna cosa que se sap”, d'aquesta manera, únicament aquella persona que posseeix aquest segon factor podrà accedir, encara que es coneguin les credencials.

Un dispositiu que s'utilitza molt en aquests sistemes és la clau RSA[E] (imatge 4), un dispositiu que genera una clau RSA que només és vàlida per a un cert temps, i que va canviant en cada un dels logins que es realitzen. Per tant, si algú volgués suplantar a un usuari, primer de tot hauria de conèixer les credencials de login i després hauria de posseir el dispositiu RSA per poder conèixer la clau generada en aquell precís moment.

Figura 4. Exemple de clau RSA.

3.1.1.10. Proves de condicions de carrera (situacions adverses)

Les condicions de carrera [F] són errors que poden arribar a produir-se quan el temps de realitzar una acció pot afectar a unes altres accions, causant problemes si es donen certes condicions o coincidències.

Un exemple podria ser una aplicació web en la que el primer pas del registre d'un nou usuari és comprovar si el nom triat està disponible o no i mentre l'aplicació està realitzant el processament de les altres dades, un altre usuari tria el mateix nom que l'usuari 1, i com que l'anterior acció (registre número 1) encara no ha acabat, l'usuari 2 pot escollir el mateix nom sense que el sistema se n'adoni de que ja està essent utilitzar.

Tot i que aquest tipus de situacions són molt difícils de detectar i explotar ja que depenen molt de diferents factors com poden ser el tràfic de xarxa, la carga del servidor, etc. existeixen atacs que podrien resultar exitosos en certes ocasions.

3.1.2. Autorització

Després d'haver parlat sobre els diferents problemes de seguretat que poden haver en els sistemes d'autenticació, parlarem sobre els que podem trobar en els mecanismes d'autorització.

3.1.2.1. Path traversal

Sovint, les aplicacions web disposen de mètodes o funcions que permeten l'accés a recursos o arxius dintre del propi sistema els quals després seran mostrats a l'usuari. A aquests arxius s'accedeix mitjançant la pròpia ruta del sistema, la qual és facilitada per l'entorn web o pels scripts que aquest pugui contindre.

Sabent que per accedir a l'arxiu "X" s'ha de facilitar la seva ruta, un atacant podria modificar aquesta ruta per tal d'accedir a l'arxiu "Y" en comptes del "X", el que provocaria que si no es verifica aquesta informació, l'atacant accedís a un recurs al qual no hauria de tindre permisos per poder accedir-hi. Depenent del tipus d'arxiu al que s'intenti accedir, es poden arribar a trobar contrasenyes del sistema, configuracions del mateix o arxius d'altres usuaris, motius més que suficients per intentar evitar-ho.

L'atac és molt simple, ja que s'aprofita el fet que la pròpia aplicació web mostra a l'usuari quin és l'arxiu o recurs al que es vol accedir, revelant en algunes ocasions el path del mateix (ja sigui complert o solament una part). Coneixent això, l'atacant intentarà "navegar" pels diferents directoris de l'aplicació (a través de "../") per tal de trobar aquells arxius que busca. Per tal d'accedir als diferents directoris del sistema, s'utilitzarà el "../" (o algunes de les seves variants) per tal de poder anar un directori "amunt".

Un exemple d'això podria ser accedir a l'arxiu "/etc/passwd" d'un sistema Unix a través d'una vulnerabilitat en el recurs de la web, on en la qual, el següent enllaç "www.webvulnerable.com/mostrarArxiu.php?arxiu=document.txt" mostrarà el contingut de l'arxiu "document" (valor passat a la variable "arxiu" pel mètode GET). Si es reemplaça "document.txt" per "../../../../../../../../../../etc/passwd" i no existeix una validació d'aquest camp, se'ns mostraria el contingut de "/etc/passwd" al nostre navegador en comptes del contingut de "document.txt".

En l'exemple de la imatge 5, podem veure com es pot accedir a l'arxiu "boot.ini" (dada) de Window aprofitant aquest tpipus d'error en una pàgina web.

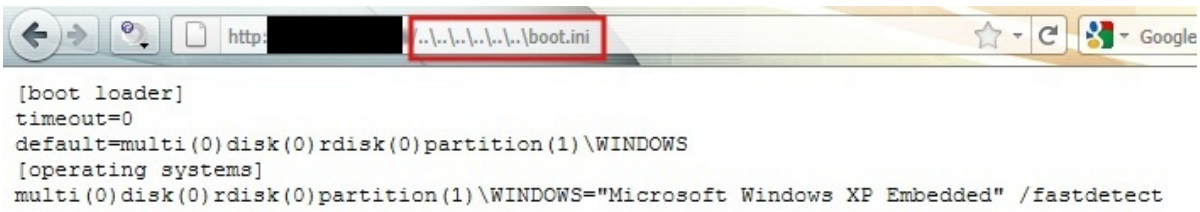

Figura 5. Exemple d'explotació de path traversal.

3.1.2.2. Saltar-se el sistema d'autorització

En algunes ocasions, quan es creen els sistemes d'autorització, no es tenen en compte totes les possibles opcions que es poden donar alhora de validar als usuaris. Això implica que donades certes condicions, un usuari sense els suficients permisos podria accedir a funcionalitats a les quals el seu accés hauria de ser denegat. Algunes d'aquests accions podrien ser les següents:

· La URL d'un recurs la qual hauria de ser accessible només pels administradors (ja que no es facilita cap link a aquest en la pàgina web a excepció de la pàgina de l'administrador). Si un atacant aconsegueix aquesta URI, podria accedir al recurs, ja que no s'han tingut en compte els nivells adequats de seguretat.

· Quan no es tanquen correctament les sessions que s'estan utilitzant i es permet accedir a un recurs propi d'un usuari una vegada la sessió d'aquest ha estat tancada (encara que l'usuari físic que intenta accedir sigui legítim).

· Situacions en les que un usuari pot accedir a recursos d'un altre usuari (als quals el seu accés hauria de ser denegat), ja sigui de l'administrador o qualsevol altra amb els mateixos permisos o privilegis.

· Recursos que poden ser accedits sense la necessitat de cap tipus d'autenticació on l'usuari hauria d'estar prèviament autenticat per poder accedir-hi.

3.1.2.3. Escalada de privilegis

Seguint amb el punt anterior, hi hauran uns altres casos on també es podrà accedir a funcions d'altres usuaris, però amb la diferència que en aquests tipus d'atacs, sí que haurem de fer modificacions en la consulta realitzada al servidor, com per exemple modificant el valor de les variables que s'envien. Per tant, s'intentarà poder realitzar accions pertinents a altres usuaris, les quals l'usuari autenticat no hauria de tenir suficients permisos per realitzar.

Cal remarcar que els casos on s'accedeix a recursos d'altres usuaris que tenen el mateix nivell de privilegis s'anomena “escalda de privilegis en horitzontal”, i quan els recursos accedits corresponen a un usuari amb més privilegis s'anomena “escalada de privilegis en vertical”.

Un exemple podria ser el de la següent URL “*www.webvulnerable.com/crearUsuari.php?rol=3*”, on la variable GET [4] “rol” indica el rol al que correspon l'usuari autenticat. Si un usuari canvia el “3” per un “1” (rol corresponent a l'administrador), podria arribar a executar l'acció si el control d'autenticació de la web no és apropiat.

[4] Mètode del protocol HTTP on la informació s'envia a través de la URL.

3.2. Eines de pentesting web

Com s'ha pogut observar en l'apartat, existeixen diferents tipus vectors d'atac que poden ser utilitzats per vulnerar la seguretat d'un servei web, podent realitzar accions no permeses o fins i tot modificacions que podrien comprometre la funcionalitat del servei.

Aquest és un fet que preocupa a molts dels usuaris que utilitzen aquests serveis, i per tant, els administradors d'aquests han de posar solució a les possibles bretxes de seguretat que poden comprometre la seva utilització. Per aquest mateix motiu, existeixen diferents eines i/o programes que permeten ajuden a aquelles persones que es preocupen per la seguretat d'un servei web.

Aquest conjunt d'eines existeixen principalment perquè hi han certes accions que no es poden realitzar directament amb les funcionalitats que un navegador normal ofereix, i per tant, aquestes no es poden testejar correctament. A més a més, l'existència d'aquestes eines faciliten molt la detecció d'errors de configuració o de vulnerabilitats web, ja que en molts casos s'ofereixen serveis que automatitzen aquestes tasques.

Encara que existeixen diferents formes de categoritzar les vulnerabilitats web, en el punt anterior, hem vist que en aquest projecte ens centraríem en aquelles que afecten als mòduls d'autorització i d'autenticació, ja que són uns dels aspectes en els que més es centren els atacants.

3.2.1. Classificació

En un primer moment es va pensar en realitzar un anàlisi d'aquelles eines i productes que existeixen en l'actualitat i que ens ajuden a detectar vulnerabilitats web per tal de fer una classificació d'elles segons el tipus de vulnerabilitat que detectaven o explotaven, però es va veure que moltes d'aquestes eines abasten diferents aspectes, creant així programes molt complets i molt potents.

En aquesta secció, es farà una introducció a algunes d'aquestes eines, centrant-nos més en aquelles que afecten més directament als sistemes d'autorització i d'autenticació. Explicant les seves principals funcionalitats, per tal de poder comprovar com funcionen i poder realitzar la nostra aplicació.

Abans d'entrar directament en l'explicació un dels diferents productes existents, es farà una breu explicació dels diferents tipus d'eines web que existeixen, per així poder entendre millor les característiques de cada una d'elles.

3.2.1.1. Proxy Web

Quan es parla sobre informàtica i es fa referència a un *proxy*, s'esta parlant sobre aquell dispositiu o programa intermig que existeix entre dos punts diferents (imatge 6), pel qual passarà tota la informació que s'envia o es rep. Per tant, un *proxy web* el podríem veure com aquella màquina o programa que intermediari entre el navegador de l'usuari i el servidor web al que es vol accedir.

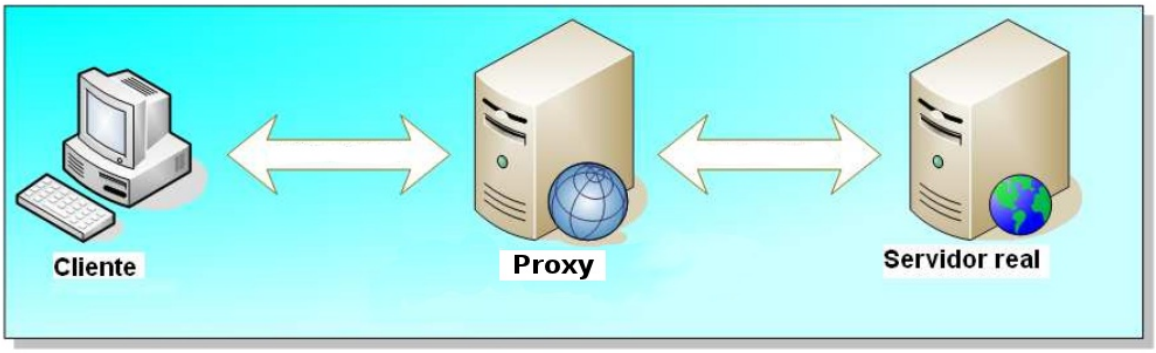

Figura 6. Representació d'un proxy en una comunicació web.

Aquests poden tenir diferents utilitats, però centrant-nos en la detecció de vulnerabilitats dels serveis web, direm que serveixen per poder analitzar i/o modificar tota la informació que s'intercanvia en les connexions amb el servidor web, podent així analitzar amb més detall les dades.

3.2.1.2. Plugins

Tot i que el conjunt d'eines que explicarem en aquest apartat l'hem decidit posar en un apartat diferent, la veritat és que es podria haver inclòs en l'anterior punt, però per fer-ho una mica més clar s'ha decidit categoritzar-los diferents.

Aquest conjunt de “petites” eines, anomenades *plugins*[5], els quals no tenen una finalitat concreta, s'afegeixen al navegador per tal d'aportar una nova funcionalitat a aquest. La seva funció bàsicament és la de donar a l'usuari més informació sobre la pàgina web que està visitant o la comunicació HTTP que s'està realitzant.

Hi ha diferents tipus de *plugins* per als navegadors d'avui dia, com poden ser per exemple els que s'encarreguen d'analitzar les capçaleres HTTP de la comunicació, els que permeten modificar les cookies o aquells que permeten una millor visualització del codi de la pàgina.

---

[5] Petites aplicacions que s'afegeixen a una altra (que actuarà com a principal) per poder afegir-hi a una funcionalitat nova i específica.

3.2.1.3. Web Spider

Un Web Spider (també anomenat *Web Crawler*) és un programa informàtic que s'encarrega de recórrer d'una forma metòdica i automàtica el contingut d'una pàgina web, analitzant tots els enllaços que apareixen en aquesta i anant recorrent-los recursivament. De tal forma que va descarregant el diferent contingut de servei web sense la interacció amb l'usuari, assegurant-se de que es visiten totes les parts de la web que estan enllaçades entre elles.

Existeixen diferents tipus de Web Spider, que fan que uns siguin més complets que altres. Alguns d'aquests permeten a l'usuari triar la profunditat fins a la que es vol arribar en la navegació per la pàgina web o també poden donar la possibilitat de definir la base de la URL sobre la que es vol fer la búsqueda.

Aquests normalment treballen emmagatzemant la llista d'enllaços que es van trobar (parts de la pàgina web) i els van visitant un a un. D'aquesta forma s'aconsegueix la llista d'enllaços abans que el propi contingut, ja que l'accés (i/o descàrrega) d'aquest suposa un increment de temps. A l'imatge 7 podem trobar un diagrama del funcionament d'un Web Spider.

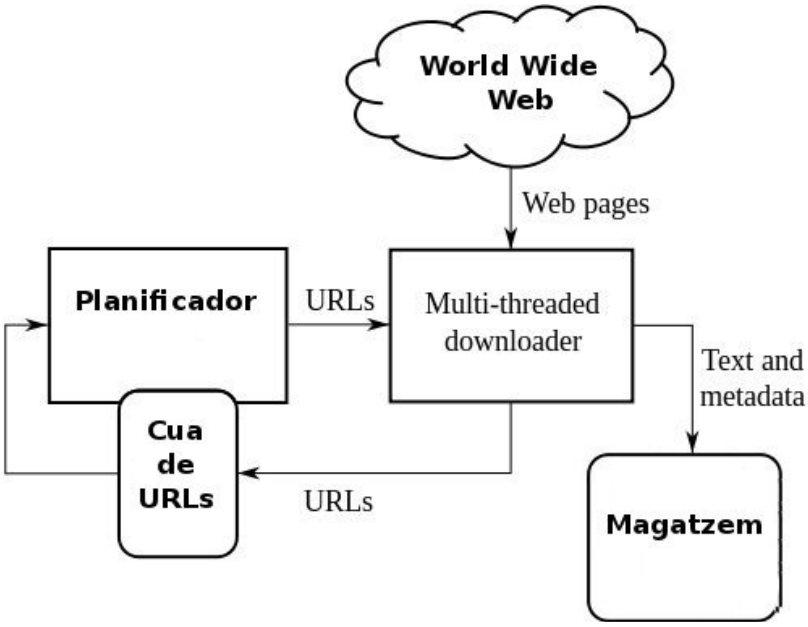

Figura 7. Diagrama del funcionament d'un Web Spider.

3.2.2. Productes existents

Una vegada s'ha fet una petita introducció d'algunes de les eines que ens ajuden en la detecció de vulnerabilitats web, realitzarem l'anàlisi d'alguns dels productes que podem trobar en Internet. Com s'ha comentat anteriorment, al existir un gran número d'aquests, només es parlarà d'aquells que estan més relacionats amb l'abast d'aquest projecte.

3.2.2.1. Burp Suite

Burp Suite és un producte multiplataforma (ja que està escrit en Java) creat per *portswigger*, el qual està a la vegada compost per diferents eines que s'encarregaran de testejar la seguretat de les aplicacions web. Tot i ser una *suite*[6], totes les eines i funcionalitats es troben integrades en una mateixa interfaç gràfica per tal de facilitar a l'usuari el seu ús. En l'imatge 8 podem veure l'aspecte general de la seva interfaç.

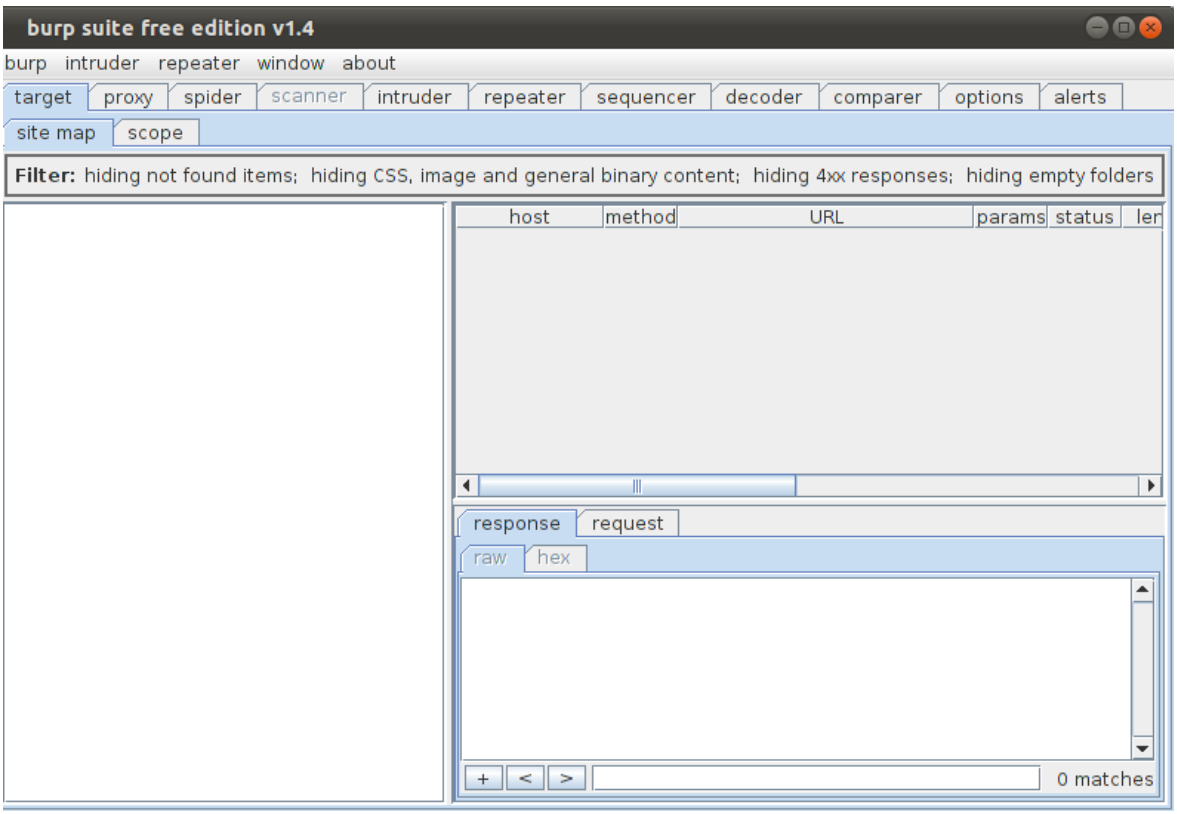

Figura 8. Interfaç de Burp Suite Free Edition

[6] Conjunt de programes que abasten una finalitat comuna.

Aquests diferents mòduls permeten realitzar tan atacs manuals com automàtics, i abasten gairebé qualsevol vulnerabilitat web existent. A més a més, els resultats de cadascuna de les proves es pot enllaçar amb una altra (encara que siguin atacs diferents) per poder així ampliar l'auditoria realitzada d'una manera més fàcil i ràpida.

A continuació es farà una breu descripció de cada un d'aquest mòduls per poder concretar una mica més la funcionalitats d'aquests:

- **Proxy:** Aquesta part, tal i com el seu nom indica i com s'ha explicat en l'apartat anterior, funciona com a proxy intermig entre el navegador i la web a auditar, permetent interceptar, analitzar i modificar el tràfic en ambdues direccions de qualsevol connexió HTTP i HTTPS existent.

- **Spider:** La seva funcionalitat és la d'un Web Spider (vist en l'apartat anterior) amb la peculiaritat de que posseeix diverses tècniques que permet realitzar un anàlisi més exacte i personalitzat.

- **Scanner:** Aquest mòdul, permetrà la detecció de vulnerabilitats web d'una manera totalment automàtica. Al ser automàtic, la detecció no pot ser màxima, ja que com s'ha estudiat, i han alguns tipus de vulnerabilitats en les que la interacció i la forma de pensar per part de l'usuari són completament obligatòries.

- **Intruder:** Eina que ajuda en els escàners manuals a automatitzar algunes accions per tal d'obtindre major rapidesa i efectivitat en les proves realitzades.

- **Repeater:** Mòdul que permet recollir dades d'altres mòduls per ajudar a la seva autimatització, modificant manualment aquella informació que ens interessa.

- **Sequencer:** Aquesta part permet l'anàlisi de la creació dels ID de sessions utilitzats per les pàgines web, o qualsevol element que hauria de ser aleatori. D'aquesta forma, podem comprovar el grau d'aleatorietat que proporciona el servei en la creació d'aquests.

- **Decoder:** La seva funció és la de transformar les dades entre diferents formats de codificació.

- **Comparer:** Aquesta última part, serà capaç de buscar les diferències entre dos objectes, permetent comparar d'una manera ràpida les respostes del servidor segons les dades enviades, per tal de veure si existeixen diferencies destacables entre elles.

Un aspecte molt important a destacar sobre Burp Suite, és que no es totalment gratuït, ja que el mòduls de l'*Scanner* requereix una llicencia per la que s'ha de pagar. Tot i així, és una eina molt molt completa encara que no s'obtenguin les parts de pagament.

3.2.2.2. WebScarab

WebScarab és un projecte crear per l'OWASP [7] amb la finalitat de poder avaluar la seguretat d'un servei web. La seva funció principal és la de funcionar com a proxy entre el navegador i el servidor web i així poder analitzar i/o modificar qualsevol informació que es transmet entre qualsevol comunicació HTTP o HTTPS. Cal dir, que al estar escrit en Java, permet ser executat en qualsevol plataforma que admeti a aquest.

Tot i ser un únic producte, el seu funcionament es basa en diferents mòduls enllaçats entre ells, en cada un té una funcionalitat diferent i complementa als altres.

A l'imatge 9 podem trobar una captura de la seva interfaç gràfica on podem veure les diferents funcionalitats que el composen.

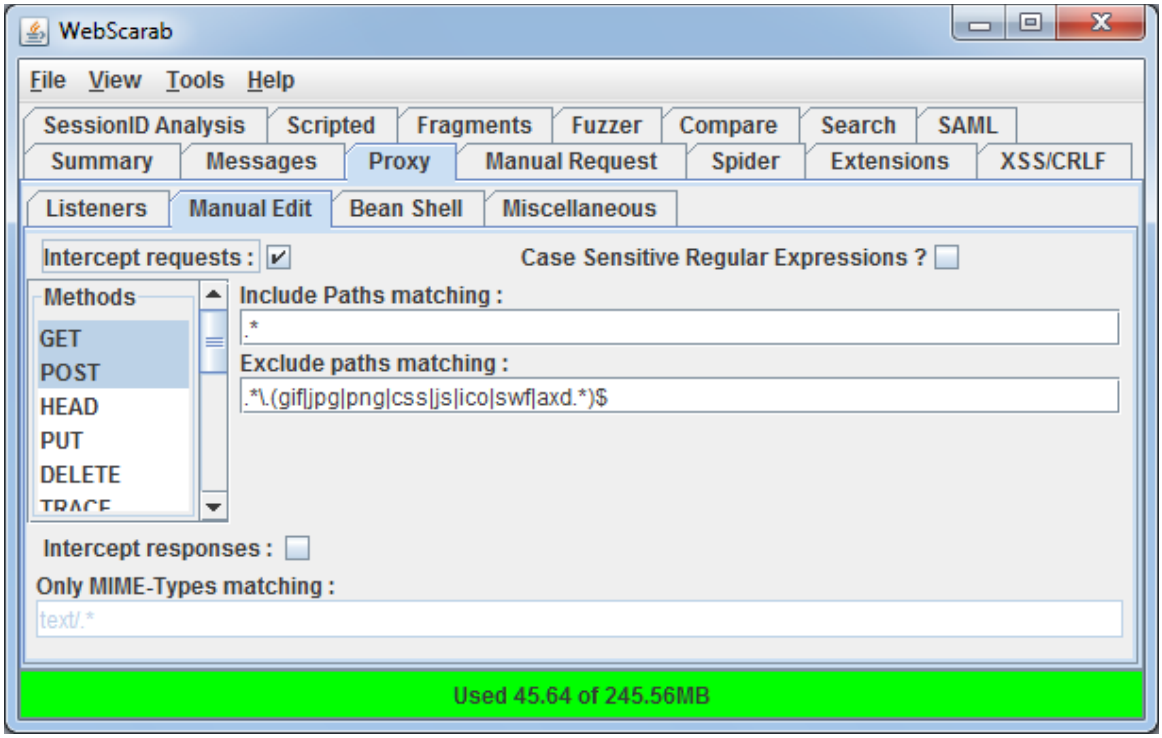

Figura 9. Interfaç gràfica de WebScarab.

Per centrar-nos una mica més en les diferents funcionalitats que WebScarab ofereix, es farà una breu descripció dels mòduls més importants que el conformen.

- **Proxy:** Aquest mòdul de l'aplicació funcionarà com a proxy entre en el navegador i el servidor web, permetent observar i/o modificar les dades abans de que s'enviïn al navegador. Cal destacar que gràcies a la negociació de SSL entre l'aplicació i el servidor web, es permet analitzar tràfic HTTP i HTTPS.

[7] Acrònim d'*Open Web Application Security Project*, el qual és un projecte de seguretat dedicat a estudiar i combatre les les causes que fan que el software sigui insegur.

· **Manual Requets:** Permet fer la modificació de les dades just abans de que s'enviïn o arribin.

· **Spider:** Funciona com un Spider, on a partir d'una URL es van buscant els diferents enllaços que aquesta conté per tal de poder obtenir totes aquelles parts de la web que es troben indexades.

· **Extensions:** Sistema de cerca de possibles arxius o directoris "oblidats" pel webmaster. S'encarregarà de buscar automàticament fitxers amb extensions ".bak", ".gzip", etc. per tal d'obtenir informació extra sobre la pàgina web auditada. Cal dir que dona a l'usuari la possibilitat d'editar els arxius que es volen buscar.

· **SessionID Analysis:** Aquesta part intentarà analitzar el grau de aleatorietat dels ID de sessió que hi han emmagatzemats en les cookies del servidor.

· **Scripted:** Mòdul que permet a l'usuari realitzar *scripts* per tal d'automatitzar alguns processos.

· **Fragments:** S'encarrega d'extreure els scripts i comentaris del codi HTML.

· **Fuzzer:** Com el seu nom indica, es tracta d'un *fuzzer* [8] el qual s'utilitzarà per testejar certs paràmetres del servei web per tal de poder veure com reacciona el servidor davant de valors inesperats.

· **Compare:** S'encarrega de comparar diferents respostes per part del servidor per veure les diferències que hi han entre elles.

· **Search:** Permetrà a l'usuari realitzar certes búsquedes en les diverses comunicacions que s'han realitzat prèviament.

· **XSS/CRLF:** Aquest mòdul permetrà un anàlisi sobre el codi de la pàgina web per tal d'obtenir possibles vulnerabilitats *XSS* [9] o *CRLF* [10].

---

[8] Eina utilitzada per testejar de manera automàtica els diferents paràmetres d'una aplicació.

[9] Vulnerabilitat web en la que l'atacant injecta codi (normalment Javascript o HTML) per tal de que un usuari realitzi certes accions al visitar una web o un link.

[10] Tipus de vulnerabilitat web en la que s'aprofita la forma de tractament del salt de línia de diferents entre diferents sistemes operatius.

3.2.2.3. THC Hydra

THC Hydra és un software que s'utilitza per crackejar els sistemes de login de diversos serveis com HTTP, *FTP*, *TELNET*, etc. d'una manera molt fàcil i ràpida.

El programa està dissenyat per ser executat tant en sistemes Linux com Windows des de la línia de comandes, i a més a més en Linux es disposa d'una interfaç gràfica d'usuari.

Aquesta eina ha obtingut una reputació molt gran gràcies a la seva propietat de poder executar els atacs utilitzant *threads* [11], donant a l'usuari l'opció de treballar amb els que ell desitgi.

Per realitzar els atacs, el seu funcionament es basa en l'ús de diccionaris els quals contindran totes les possibles opcions que es volen provar, essent aquest completament necessaris per anar provar les diferents opcions i així obtindre les credencials d'usuari,

La seva execució bàsica des de la línia de comandes és molt senzilla, simplement haurem d'indicar el tipus de servei que volem atacar, els arxius que contindran els valors per a l'usuari i per al password, el host sobre el que es volen provar, el nom dels paràmetres d'aquests valors i la cadena que indicarà si la comanda ha tingut èxit o no. Un exemple podria ser la següent instrucció, la qual realitza un atac de força bruta sobre el formulari de login HTTP amb enviament de dades per POST[12] del host X.X.X.X, on el el nom de la variable per a l'usuari és “user” i per al password és “contrasenya”, s'utilitzarà l'arxiu “dic_usuaris.txt” per als possibles usuaris i l'arxiu “dicc_passwords” per a les contrasenyes i la paraula que indicarà si ha tingut èxit serà “Bienvenido”.

```
./hydra X.X.X.X http-post-form  \
"/login.php?user=^USER^&contrasenya=^PASS^:S=Bienvenido" \
-l "dic_usuaris.txt"  -p "dic_passwords.txt"
```

Cal remarcar que depenent del tipus de login que es vulgui crackejar, les opcions necessàries seran diferents, podent ampliar a més

Per últim comentar que aquest programa no requereix de cap llicencia per ser descarregat ni executat, per tant és complement gratuït.

---

[11] Seqüencia més petita d'instruccions que pot ser executada independentment. En sistemes multithread s'utilitzen per paral·lelitzar l'execució de codi i obtindre una velocitat major.

[12] Mètode del protocol HTTP on la informació s'envia en el codi de la consulta.

3.2.2.4. Brutus

Continuant amb l'enumeració d'usuaris, trobem Brutus, el qual és un programa que serveix pre crackejar contrasenyes de diferents serveis web com HTTP, POP3, SMB a través de l'ús de diccionaris o de la força bruta. Tot i que ofereix molts sistemes d'autenticació, no arribar a tenir tants com THC Hydra.

Aquest programa també disposa d'una interfaç gràfica per facilitar el seu ús a l'usuari, però un punt en contra és que només pot ser utilitzat en sistemes Windows, obligant a utilitzar aquesta plataforma per poder utilitzar-lo.

Tot i que l'oferta de protocols que ofereix no és molt gran, disposa de la possibilitat de poder importar mòduls nous que abastin altres tipus, i dóna la possibilitat de crear els teus propis segons els teus requeriments.

A l'imatge 10 podem veure una captura d'aquest programa.

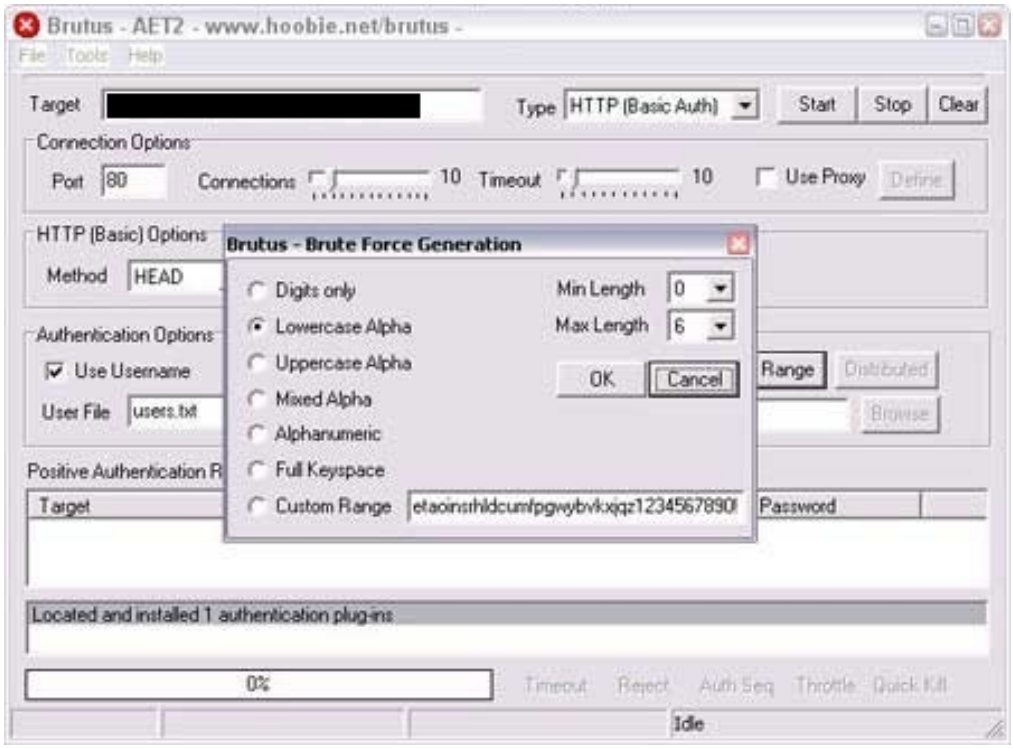

Figura 10. Captura de pantalla de Brutus.

3.2.2.5. Plugins per al navegador

Com hem vist anteriorment, els plugins son petits programes que complemente a un altres més gran per donar-li més funcionalitats. En aquest projecte com que està enfocat a la seguretat web, parlarem d'alguns dels plugins que existeixen per als navegadors web i que ens ajudaran a detectar vulnerabilitats en els serveis que s'ofereixen a través d'Internet.

Cal destacar que encara que en aquesta llista es parla de plugins concrets per a certs navegadors, en molts casos podem trobar símils per a la resta de navegadors que no són compatibles amb aquest en concret.

· **Live HTTP Headers:** Complement per al navegador Firefox que permetrà analitzar les capçaleres HTTP que s'intercanvien en els diferents consultes i respostes rebudes.

· **Tamper data:** Plugin que com el seu nom indica, serveix per poder modificar les dades que s'envien al servidor web des del navegador. Al activar-ho permet a l'usuari veure i modificar quins paràmetres POST s'envien, quines cookies, les capçaleres, etc.

· **Web Developer:** Complement que està format per un gran conjunt de funcionalitats que ens permeten analitzar amb més detall tot el contingut d'una web, donant l'opció a l'usuari de modificar certes parts del codi HTML, com per exemple per poder eliminar restriccions en els camps d'un formulari o alterar el contingut de les cookies.

· **Hack Bar:** Complement que incorpora un ampli ventall de mòduls que permeten realitzar diferents tipus d'atacs contra serveis web i permet modificar les dades POST que s'envien. Conté diversos codificadors/decodificadors per a les codificacions més usuals, així com un algunes opcions per ofuscar codi.

· **Firebug:** Plugin que té la finalitat de donar diverses opcions per analitzar amb més facilitat i detall el codi d'un servei web, permetent editar i debugar els CSS, Javascript i HTML en viu.

Capítol 4: Especificació

Aquest projecte consisteix en un anàlisis sobre algunes de les vulnerabilitats que es poden trobar en alguns serveis que s'ofereixen en Internet, principalment en les pàgines web. Per fer això, es crearà un sistema que sigui capaç de detectar-les i poder mostrar els resultats a l'usuari.

Primer de tot, s'ha fet un estudi sobre les diferents vulnerabilitats que existeixen, i per tal d'acotar el projecte, s'ha decidit focalitzar-lo únicament en aquelles que afecten a l'autenticació i a l'autorització, ja que s'ha considerat que aquestes juguen un paper molt important en les aplicacions d'avui dia.

En l'apartat anterior, hem pogut trobar una explicació dels principals vectors d'atac que es poden utilitzar per explotar-les, i s'ha fet un estudi sobre les diferents aplicacions que existeixen actualment per tal de poder crear la nostra pròpia, aprofitant (o millorant si és possible) la forma en la que treballen aquestes.

La nostra aplicació s'encarregarà de buscar de la forma el més automàticament possible aquelles vulnerabilitats (o errors de disseny) que poden existir en una pàgina web donada. L'usuari únicament haurà d'indicar quina web vol analitzar i quin tipus de proves vol realitzar. Cal remarcar, que tal i com s'ha comentat anteriorment, aquests proves únicament tindran en compte aquelles que afectin principalment als sistemes d'autorització i d'autenticació de la pròpia pàgina web.

La primera d'aquestes proves serà comprovar si les dades del formulari del login (credencials) s'envien per un canal segur utilitzant HTTPS. Per això, l'aplicació haurà de buscar aquelles parts de la web que continguin algun tipus de formulari de login i després identificarà aquelles que no utilitzin HTTPS per enviar les dades entrades per l'usuari.

Encara que aquesta prova anirà relacionada amb la de buscar les pàgines que contenen login, la segona prova es pot realitzar independentment de la primera, per si l'auditor té algun interès especial en detectar només aquelles parts de la web que contenen un formulari de login.

Una altra prova de la que disposarà la nostra aplicació és l'enumeració d'usuaris, en la qual

s'intentarà obtindre la llista d'usuaris del servei. Per això, serà necessari que la pàgina web tingui un error en el seu disseny (mostrant més informació a l'usuari de la veritablement necessària) i mostri un missatge diferent en el cas de que s'introdueixi un usuari vàlid i un de no vàlid (encara que la contrasenya facilitada no sigui correcta). En aquesta mateixa prova, també es donarà l'opció d'intentar obtenir les credencials d'alguns usuaris del servei utilitzant atacs per diccionari.

El fet que el propi navegador sigui capaç d'emmagatzemar certa informació considerada privada d'un usuari, pot suposar un risc per aquelles màquines que siguin compartides per diferents usuaris. Per evitar (o advertir) aquest fet, la nostra aplicació haurà de ser capaç de detectar aquelles pàgines de la web que tinguin la caché activada o el recordatori de contrasenyes disponible. Ja que si un usuari no tanca correctament el navegador, un altra podria accedir a les parts de la web que hi ha accedit anteriorment, encara que no conegui les credencials d'autenticació.

Els CAPTCHAs són uns sistemes que permeten evitar l'automatització de certes activitats per part dels usuaris dintre d'un servei web. Per tant, és molt important que certes funcionalitats de la pàgina web continguin un CAPTCHA per evitar aquesta acció. Per aquest motiu, la nostra aplicació permetrà detectar aquell contingut que posseeixi algun sistema de CAPTCHA en el seu codi.

Per poder realitzar totes aquestes proves, la nostra aplicació haurà d'analitzar les diferents respostes que fa el servidor, analitzant el codi HTML rebut. El sistema, permetrà emmagatzemar localment aquestes respostes, permetent realitzar les proves en qualsevol moment, sense necessitat de tenir accés a Internet (tret les proves d'enumeració d'usuaris, en les que es necessita la resposta directa del servidor en cada una de les peticions). Així que es facilitarà un web spider per tal d'obtindre automàticament cada un dels enllaços de la web a analitzar.

Per últim, l'usuari auditor, haurà de poder controlar-ho tot d'una manera fàcil i simple. Per aquest motiu s'ha decidit proporcionar un front-end [13] que pugui ser utilitzat a través del navegador. D'aquesta manera, si es vol oferir el servei de cerca de vulnerabilitats a través d'Internet (utilitzant algun servidor privat), només caldrà accedir-hi a través del navegador, sense necessitat d'instal·lar res en local, facilitant així la utilització de la mateixa.

A més a més, com que aquesta aplicació està pensada per a poder ser utilitzada sense amplis coneixements d'informàtica, la utilització d'aquesta a través d'Internet, facilita tot el procés d'instal·lació i configuració del sistema, i pot permetre una màquina (o connexió) més potent per poder realitzar totes les proves necessàries. Per tant, l'usuari només haurà de connectar-se a la web de l'aplicació, seleccionar les proves que vol realitzar i veure els resultats.

[13] Part del software que s'encarrega de la interacció amb l'usuari.

Capítol 5: Desenvolupament

En aquest apartat s'explicarà tot el procés de desenvolupament de la nostra aplicació. Començant amb l'anàlisi realitzat prèviament fins a arribar a la implementació final.

5.1. Anàlisi

Com hem pogut veure en els apartats anteriors, aquest projecte consisteix en la realització d'un sistema capaç de detectar certes vulnerabilitats web i poder presentar els resultats d'una manera clara i precisa.

Després d'haver realitzat un anàlisis exhaustiu sobre les diferents vulnerabilitats que es poden trobar en els serveis oferts en Internet, hem pogut comprovar que existeix un gran número d'eines que ajuden a la detecció d'aquestes, i que són utilitzades per qualsevol auditor web o administrador que vulgui analitzar la seguretat d'una pàgina web.

Aquestes eines, són completament imprescindibles per poder analitzar les diferents parts que existeixen en una pàgina web, ja que el navegador per si sol no ens permet realitzar certes operacions ni veure la informació completa del que es transmet en les comunicacions entre el client (navegador) i el servidor (pàgina web).

Una d'aquestes eines (la qual s'ha considerat de les més importants) són els proxies web, els quals són programes intermitjos entre el navegador i el servidor web que permeten analitzar i modificar tot el que s'intercanvia entre aquests. Per tant, aquest tipus de funcionalitat és el que nosaltres volem per a la nostra aplicació, ja que hem d'analitzar les respostes del servidor i poder modificar aquelles que enviem.

Tal i com s'ha dit en anteriors apartats, s'ha decidit crear una eina que sigui el més automàtica possible, i per aquest motiu, en la nostra aplicació s'ha decidit ometre l'ús del navegador directament amb la web a analitzar, ja que sinó l'usuari hauria d'interactuar amb aquesta, i això és precisament el que es vol evitar. D'aquesta forma, el que es crearà serà una aplicació

que l'usuari pugui administrar i que sigui aquesta qui s'encarregui de detectar les vulnerabilitats existents únicament amb la informació facilitada per l'usuari, sense que aquest hagi de navegar manualment per les diferents seccions de la pàgina web.

Dintre d'aquesta aplicació, hi hauran d'haver diferents mòduls, els quals tindran diferents funcionalitats entre ells i que permetran detectar les diferents vulnerabilitats existents que afecten als sistemes d'autorització i d'autenticació de la pàgina web.

Gràcies a tota la informació obtinguda en l'estudi realitzat prèviament, podem tenir una idea de com realitzar la nostra aplicació per realitzar aquestes tasques més específiques i que pugui ser accessible a la majoria d'usuaris "normals".

Per l'elaboració del nostre sistema, hem de fixar-nos en els criteris que s'han establert en els anteriors apartats per tal de poder realitzar un projecte atractiu i que compleixi tots els requeriments establerts:

- **Detecció de vulnerabilitats:** La principal característica que ha de tenir la nostra aplicació, és la de poder detectar les diferents vulnerabilitats existents en el servei web especificat. Per aquest motiu haurà d'analitzar les respostes obtingudes del servidor i poder modificar algunes de les que s'envien.

  Per tal de fer-ho d'una manera clara i concisa, s'haurien de separar les diferents proves per tal de que l'usuari pugui executar únicament aquelles que es necessiten, ja que algunes d'elles poden requerir molt temps per a la seva finalització.

- **Descarrega de resultats:** El nostre sistema, ha de poder mostrar els resultats obtinguts en les proves d'una manera precisa, sense que hi hagin dificultats per entendre'ls.

  A més a més, hauria de poder permetre als usuaris descarregar-se aquests resultats per poder-los consultar en qualsevol moment sense haver de tenir accés a Ineternet ni a la pròpia pàgina web. D'aquesta manera, una vegada realitzades les proves, l'usuari podria descarregar-se aquests resultats a la màquina local d'una manera ràpida i fàcil.

- **Interfaç gràfica:** Aquest sistema de detecció de vulnerabilitats està pensat per a que pugui ser utilitzat per persones sense un ampli coneixement informàtic. Per tant, s'ha de realitzar una interfaç que permeti controlar totes les característiques i funcionalitats que faciliten la detecció de vulnerabilitats en pàgines web sense la necessitat de conèixer com poder detectar aquests vectors d'atac manualment ni llargues comandes.

Per aquest motiu, s'ha de crear un servei web que permeti a l'usuari conèixer el grau de seguretat existent dels sistemes que afecten a l'autorització i autenticació d'un servei web, per poder comprovar si es compleixen els mínims desitjats.

5.2. Casos d'ús

Els casos d'ús són ideals per indicar-nos d'una forma molt clara qui i quines accions es podran realitzar en un projecte. Això es de vital importància, ja que especificaran quines seran les accions que hauran de ser implementades per portar a terme el correcte funcionament de l'aplicació.

En el nostre projecte, només es disposarà de dos rols diferenciats. El primer d'ells serà el d'usuari invitat i l'altra el d'usuari autenticat. L'usuari invitat, només podrà accedir al panell de login, en el qual haurà d'introduir les seves credencials per poder accedir al panell principal de l'aplicació.

Una vegada realitzat aquest procés, l'usuari passarà a ser un usuari autenticat i podrà accedir a les diferents funcionalitats del sistema, seleccionant l'opció desitjada i seguint les instruccions de la pròpia aplicació.

5.2.1. Diagrames de casos d'ús

En aquest apartat es realitza una descripció molt gràfica de les principals característiques que s'implementaran en el nostre projecte.

Per a això, són ideals els diagrames de casos d'ús, ja que permeten relacionar els diferents rols del nostre sistema amb les accions que podrà realitzar cadascun d'ells.

En el nostre cas, al tenir únicament dos rols, simplement s'han necessitat dos diagrames de casos d'ús (un per a cada un), el quals es poden observar en les figures 11 i 12.

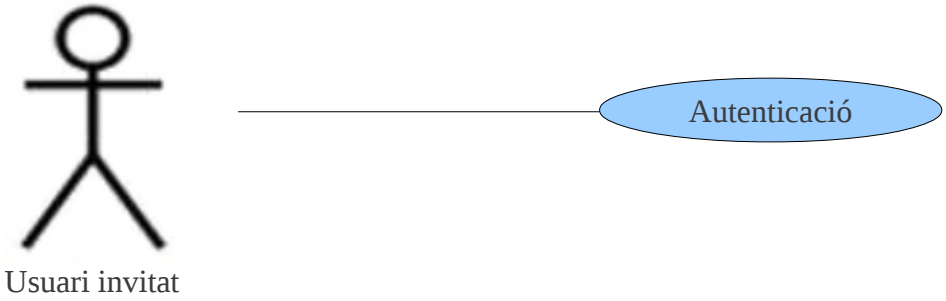

Figura 11. Diagrama de cas d'ús d'un usuari invitat.

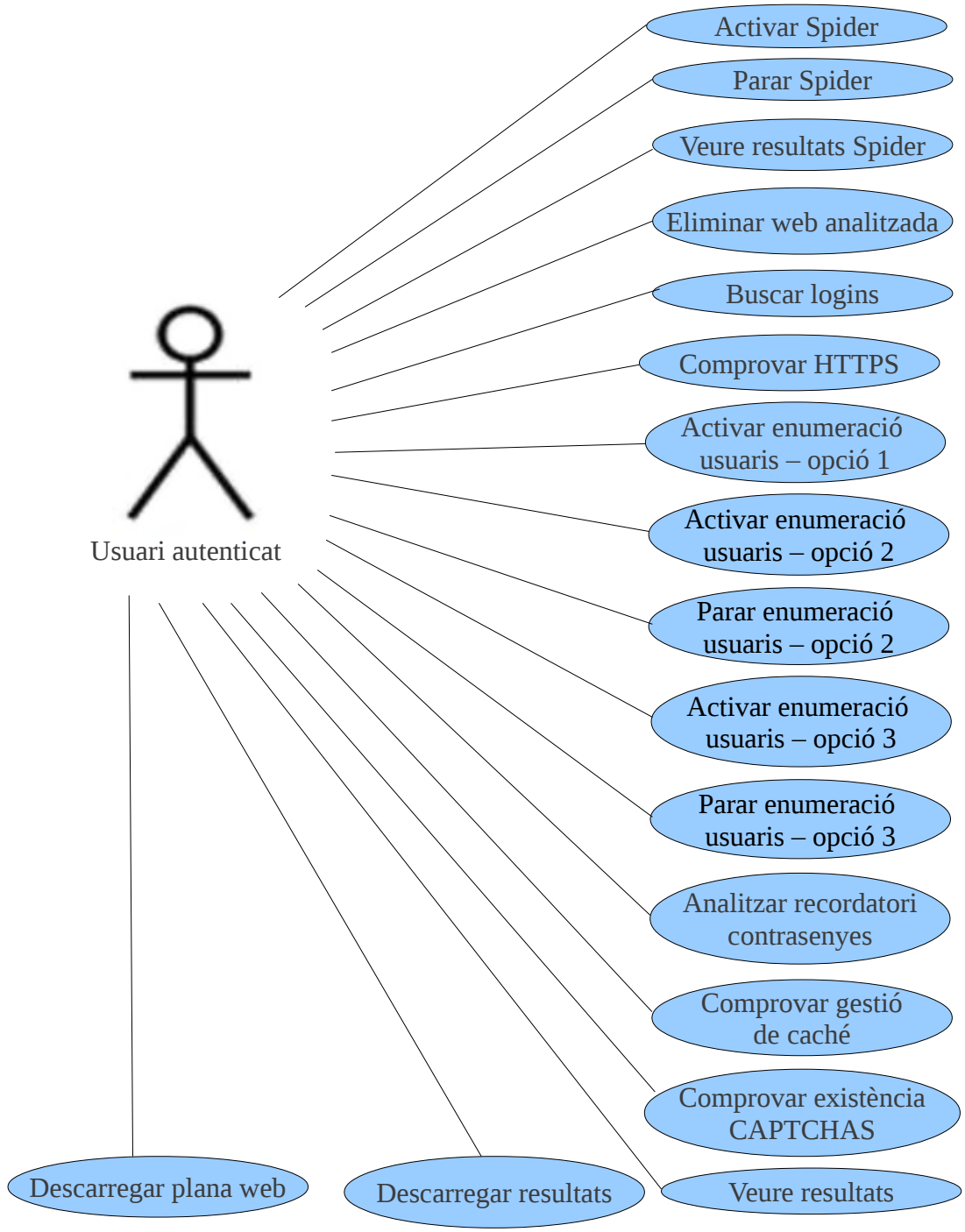

Figura 12. Diagrama de cas d'ús d'un usuari autenticat.

5.2.2. Casos d'ús estesos

Gràcies a la implementació de les següents taules, es pot observar d'una manera molt esquemàtica els detalls de cadascun dels casos d'ús descrits en l'apartat anterior. D'aquesta manera alhora de realitzar la implementació del nostre projecte serà molt més fàcil complir cadascun dels punts obtinguts en l'anàlisi de la nostra aplicació.

Cas d'ús	Autenticació
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet a un usuari autenticar-se en el sistema
Actors	Usuari invitat
Pre-condició	
Flux principal	1. L'usuari entra en la web de l'aplicació 2. L'interfaç motra el formulari de login del sistema 3. Es validen les dades de l'usuari 4. Es mostra l'interfaç principal de l'aplicació
Fluxos alternatius	3. Si hi ha un error en la validació de les dades s'avisarà a l'usuari
Post-condició	S'accedeix al panell principal de l'aplicació
Comentaris	

Taula 1. Cas d'ús estès de “Autenticació”

Cas d'ús	Activar Web Spider
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet activar el web spider per a una web
Actors	Usuari autenticat
Pre-condició	
Flux principal	1. L'usuari selecciona l'opció de l'spider 2. L'interfaç mostra el formulari per entrar el host desitjat 3. L'usuari entra el host desitjat per fer l'escaneig 4. L'interfaç valida les dades entrades per l'usuari 5. El sistema arranca l'spider i mostra un missatge d'informació a l'usuari
Fluxos alternatius	2. Si l'spider ja està en marxa es mostrarà un missatge advertint a l'usuari 4. Si hi ha un error en les dades s'informarà a l'usuari
Post-condició	El web spider s'ha arrancat correctament
Comentaris	

Taula 2. Cas d'ús estès de “Activar Web Spider”

Cas d'ús	Parar Web Spider
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet parar l'execució del web Spider
Actors	Usuari autenticat
Pre-condició	El web spider s'ha d'estar executant
Flux principal	1. L'usuari selecciona l'opció de l'spider 2. L'interfaç mostra el missatge advertint a l'usuari que està en marxa i li dona l'opció de parar-ho 3. L'usuari clica en l'opció de parar 4. El sistema para l'spider i comprova que efectivament s'ha parat 5. Es mostra un missatge dient que s'ha parat correctament
Fluxos alternatius	2. Si no està en marxa es mostrarà un missatge a l'usuari 4. Si no es pot parar es mostrarà un missatge advertint a l'usuari
Post-condició	El web spider s'ha parat
Comentaris	

Taula 3. Cas d'ús estès de “Para Web Spider”

Cas d'ús	Veure resultats de Web Spider
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet veure els resultats de l'spider sobre un host determinat
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat algun host anteriorment.
Flux principal	1. L'usuari selecciona l'opció de l'spider 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen veure els resultats 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web existents en el host i es dona l'opció de descarregar-se-les una a una, o de veure el llistat únicament de les planes que tenen login.
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge
Post-condició	Es mostra la llista de planes que conté el host seleccionat
Comentaris	

Taula 4. Cas d'ús estès de “Veure resultats de Web Spider”

Cas d'ús	Eliminar host del Web Spider
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet eliminar un host analitzat prèviament
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat el host prèviament
Flux principal	1. L'usuari selecciona l'opció de l'spider 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen eliminar els resultats 4. L'aplicació comprova les dades enviades per l'usuari i elimina els resultats d'aquest host
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades es mostrarà un missatge
Post-condició	S'eliminen els resultats del host seleccionat
Comentaris	

Taula 5. Cas d'ús estès de “Eliminar host del Web Spider”

Cas d'ús	Buscar logins
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet buscar totes les pàgines que tenen login
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat el host prèviament
Flux principal	1. L'usuari selecciona l'opció de l'spider 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen veure els resultats 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web existents en el host i es dona l'opció de descarregar-se-les una a una, o de veure el llistat únicament de les planes que tenen login 5. L'usuari selecciona l'opció de mostrar només aquelles planes que tenen login 6. L'interfaç fa la cerca i mostra les pagines amb login
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge 6. En el cas de no trobar-se cap es mostra un missatge a l'usuari
Post-condició	Es busquen i mostren les pàgines que contenen algun login
Comentaris	

Taula 6. Cas d'ús estès de “Buscar logins”

Cas d'ús	Detectar ús d'HTTPS
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet comprovar si els formularis amb login utilitzen una connexió segura
Actors	Usuari autenticat
Pre-condició	
Flux principal	1. L'usuari selecciona l'opció de comprovar HTTPS 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host que es vol comprovar 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web amb login existents en el host que no utilitzen HTTPS
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge
Post-condició	Es mostren aquells logins que no utilitzen HTTPS per enviar les dades
Comentaris	

Taula 7. Cas d'ús estès de “Detectar ús d'HTTPS”

Cas d'ús	Activar enumeració usuaris – Opció 1
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet detectar si el servidor respon diferent amb un usuari existent i un de no existent
Actors	Usuari autenticat
Pre-condició	
Flux principal	1. L'usuari selecciona l'opció d'enumeració d'usuaris – opció 1 2. L'interfaç mostra el formulari per executar la comprovació 3. L'usuari omple el formulari i l'envia a l'aplicació 4. L'interfaç validarà les dades entrades per l'usuari i executarà la prova amb les dades facilitades
Fluxos alternatius	4. Si hi ha algun error en les dades es mostrarà un missatge a l'usuari
Post-condició	Començarà l'enumeració d'usuaris – opció 1
Comentaris	

Taula 8. Cas d'ús estès de “Activar enumeració usuaris opció 1”

Cas d'ús	Activar enumeració usuaris – Opció 2
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet activar l'enumeració d'usuaris sense conèixer la contrasenya
Actors	Usuari autenticat
Pre-condició	
Flux principal	1. L'usuari selecciona l'opció d'enumeració d'usuaris – opció 2 2. L'interfaç mostra el formulari per executar la cerca d'usuaris 3. L'usuari omple el formulari i selecciona l'arxiu local que conté el llistat d'usuaris a comprovar 4. L'interfaç validarà les dades entrades per l'usuari i començarà l'enumeració d'usuaris amb les dades facilitades
Fluxos alternatius	4. Si hi ha algun error en les dades es mostrarà un missatge a l'usuari
Post-condició	Començarà l'enumeració d'usuaris – opció 2
Comentaris	

Taula 9. Cas d'ús estès de “ Activar enumeració usuaris opció 2”

Cas d'ús	Parar enumeració usuaris – Opció 2
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet parar l'enumeració d'usuaris
Actors	Usuari autenticat
Pre-condició	L'enumeració d'usuaris ha d'estar activa
Flux principal	1. L'usuari selecciona l'opció d'enumeració d'usuaris – opció 2 2. L'interfaç mostra l'opció per parar l'enumeració d'usuaris 3. L'usuari selecciona parar l'enumeració d'usuaris 4. L'interfaç detén l'enumeració d'usuaris
Fluxos alternatius	2. Si l'enumeració d'usuaris no està activa es mostrarà el formulari per activar-la 4. Si no es pot parar l'enumeració d'usuaris es mostrarà un missatge d'error a l'usuari.
Post-condició	Es detén l'enumeració d'usuaris
Comentaris	

Taula 10. Cas d'ús estès de “Parar enumeració usuaris opció 2”

Cas d'ús	Activar enumeració usuaris – Opció 3
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet buscar les credencials d'usuaris a través de diccionaris
Actors	Usuari autenticat
Pre-condició	
Flux principal	1. L'usuari selecciona l'opció d'enumeració d'usuaris – opció 2 2. L'interfaç mostra el formulari per executar la cerca d'usuaris 3. L'usuari omple el formulari i selecciona l'arxiu local que conté el llistat d'usuaris a comprovar 4. L'interfaç validarà les dades entrades per l'usuari i començarà l'enumeració d'usuaris amb les dades facilitades
Fluxos alternatius	4. Si hi ha algun error en les dades es mostrarà un missatge a l'usuari
Post-condició	Començarà l'enumeració d'usuaris – opció 3
Comentaris	

Taula 11. Cas d'ús estès de “Activar enumeració usuaris opció 3”

Cas d'ús	Parar enumeració usuaris – Opció 3
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet parar l'enumeració d'usuaris
Actors	Usuari autenticat
Pre-condició	L'enumeració d'usuaris ha d'estar activa
Flux principal	1. L'usuari selecciona l'opció d'enumeració d'usuaris – opció 3 2. L'interfaç mostra l'opció per parar l'enumeració d'usuaris 3. L'usuari selecciona parar l'enumeració d'usuaris 4. L'interfaç detén l'enumeració d'usuaris
Fluxos alternatius	2. Si l'enumeració d'usuaris no està activa es mostrarà el formulari per activar-la 4. Si no es pot parar l'enumeració d'usuaris es mostrarà un missatge d'error a l'usuari.
Post-condició	Es detén l'enumeració d'usuaris
Comentaris	

Taula 12. Cas d'ús estès de “Parar enumeració usuaris opció 3”

Cas d'ús	Comprovar recordatori de contrasenyes
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet comprovar si el recordatori de contrasenyes està desactivat
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat algun host prèviament
Flux principal	1. L'usuari selecciona l'opció de recordatori de contrasenyes 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen veure els resultats 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web existents en el host que tenen el recordatori de contrasenyes desactivat i es dona l'opció de descarregar-se-les una a una
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge d'error 4. En el cas de no trobar-se cap amb el recordatori de contrasenyes desactivat es mostra un missatge d'informació a l'usuari
Post-condició	Es mostrarà el llistat de planes web que contenen el recordatori de contrasenyes desactivat
Comentaris	

Taula 13. Cas d'ús estès de “Comprovar recordatori de contrasenyes”

Cas d'ús	Comprovar gestió de caché
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet comprovar si la caché del navegador està desactivada
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat algun host prèviament
Flux principal	1. L'usuari selecciona l'opció de gestió de caché 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen veure els resultats 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web existents en el host que tenen la caché desactivada i es dona l'opció de descarregar-se-les una a una
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge d'error 4. En el cas de no trobar-se cap amb la caché desactivada es mostrarà un missatge d'informació a l'usuari

Post-condició	Es mostrarà el llistat de planes web que contenen la caché del navegador desactivada
Comentaris	

Taula 14. Cas d'ús estès de “Comprovar gestió de caché”

Cas d'ús	Comprovar existència de CAPTCHAs
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet comprovar si s'utilitza algun CAPTCHA en un host determinat
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat algun host prèviament
Flux principal	1. L'usuari selecciona l'opció de CAPTCHAs 2. L'interfaç mostra els diferents hosts que s'han analitzat prèviament 3. L'usuari clica en el host del qual es volen veure els resultats 4. L'interfaç comprova el nom del host, mostra el llistat de pàgines web existents en el host que contenen algun CAPTCHA i es dona l'opció de descarregar-se-les una a una
Fluxos alternatius	2. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 4. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge d'error 4. En el cas de no trobar-se cap plana amb CAPTCHA es mostrarà un missatge d'informació a l'usuari
Post-condició	Es mostrarà el llistat de planes web que contenen algun CAPTCHA
Comentaris	

Taula 15. Cas d'ús estès de “Comprovar existència de CAPTCHA”

Cas d'ús	Veure resultats
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat alguna web anteriorment i s'han d'haver executat les proves pertinents sobre el host
Flux principal	1. L'usuari selecciona l'opció de mostrar resultats 2. L'interfaç mostra les diferents proves que es poden realitzar amb l'aplicació 3. L'usuari clica en aquella prova que vulgui veure els resultats 4. L'interfaç comprova l'opció i mostra el llistat de pàgines web

	analitzades anteriorment 5. L'usuari clica en el host del qual es volen veure els resultats 6. L'interfaç comprova el nom del host i mostra els resultats. També proporciona un enllaç per descarregar-se aquests resultats.
Fluxos alternatius	4. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 6. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge d'error 6. En el cas de no trobar-se cap resultat es mostrarà un missatge d'informació a l'usuari
Post-condició	Es mostraran els resultats de l'opció desitjada
Comentaris	

Taula 16. Cas d'ús estès de “Veure resultats”

Cas d'ús	Descarregar resultats
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat alguna web anteriorment i s'han d'haver executat les proves pertinents sobre el host
Flux principal	1. L'usuari selecciona l'opció de mostrar resultats 2. L'interfaç mostra les diferents proves que es poden realitzar amb l'aplicació 3. L'usuari clica en aquella prova que vulgui descarregar els resultats 4. L'interfaç comprova l'opció i mostra el llistat de pàgines web analitzades anteriorment 5. L'usuari clica en el host del qual es volen descarregar els resultats 6. L'interfaç comprova el nom del host i mostra els resultats. També proporciona un enllaç per descarregar-se aquests resultats. 7. L'usuari clica en l'opció de descarregar-se els resultats
Fluxos alternatius	4. Si no hi ha cap host analitzat prèviament es mostrarà un missatge a l'usuari 6. Si hi ha algun error amb les dades enviades per l'usuari es mostrarà un missatge d'error 6. En el cas de no trobar-se cap resultat es mostrarà un missatge d'informació a l'usuari
Post-condició	Es descarregaran els resultats de l'opció desitjada
Comentaris	

Taula 17. Cas d'ús estès de “descarregar resultats”

Cas d'ús	Descarregar plana web
Versió	1.0
Autor	Rafael Páez Jaime
Descripció	Permet descarregar-se un arxiu d'una plana web
Actors	Usuari autenticat
Pre-condició	S'ha d'haver analitzat algun host anteriorment i s'han de mostrar els resultats per pantalla
Flux principal	1. L'interfaç mostra el llistat de pàgines web disponibles juntament amb l'enllaç de descàrrega 2. L'usuari selecciona aquell arxiu que vol descarregar-se 3. L'interfaç comprova les dades facilitades per l'usuari i permet la descàrrega
Fluxos alternatius	1. Si no hi ha cap pàgina web disponible es mostrarà un missatge informatiu 3. Si hi ha algun error es mostrarà un missatge a l'usuari
Post-condició	Es descarregarà l'arxiu desitjat
Comentaris	

Taula 18. Cas d'ús estès de “Descarregar plana web”

5.3. Disseny i implementació

En aquest apartat s'explicaran totes les decisions que s'han pres per a la realització del projecte, i per a fer-ho d'una manera més clara i concisa, s'ha decidit separar el nostre sistema en 9 apartats diferents, els qual s'expliquen a continuació.

Per poder veure d'una forma més esquemàtica el funcionament de la nostra aplicació, podem observar la figura 13, en la qual podem veure els 9 mòduls principals del nostre sistema de detecció de vulnerabilitats.

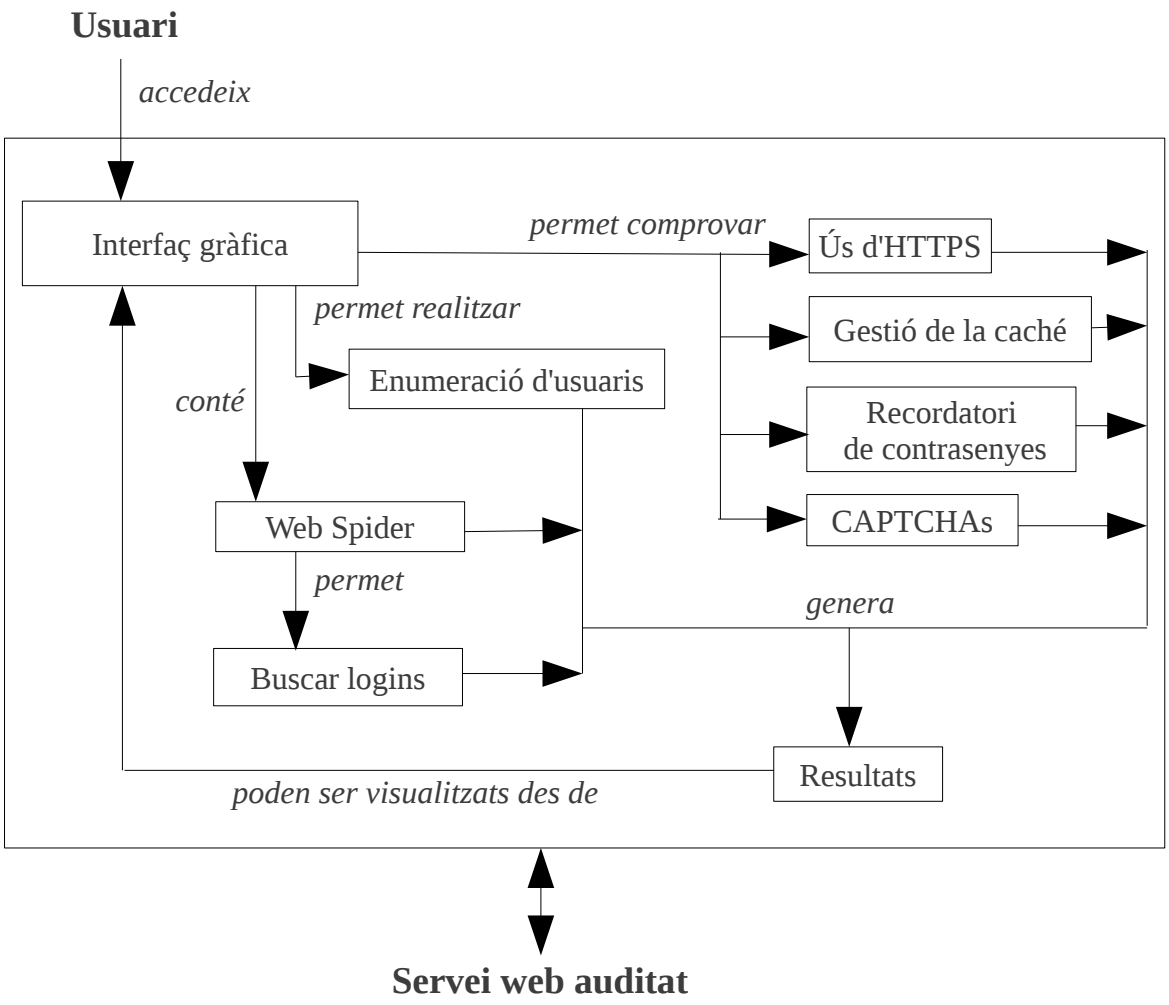

Figura 13. Esquema dels diferents mòduls existents en l'aplicació.

5.3.1. Web Spider

Uns dels requeriments principals de la nostra aplicació és que s'intenti crear un sistema de detecció de vulnerabilitats que requereixi la menor interacció per part de l'usuari. Per tant, després s'haver realitzat l'estudi sobre les diferents eines que existeixen, hem pogut observar que els Web Spider, són programes que recorren els diferents enllaços que es troben en una web, per tal d'intentar accedir (o conèixer) a cada un dels diferents apartats de la web que estan enllaçats a través d'ella. D'aquesta forma, únicament introduint la web principal, es podria arribar a una gran part del total de la pàgina web utilitzant únicament un programa.

Coneixent això, la nostra aplicació necessitarà una eina com aquesta, en la qual l'usuari només hagi d'introduir la web a auditar i que el sistema s'encarregui de fer la resta.

Per simular aquesta funcionalitat es va pensar en utilitzar alguna de les eines ja existents, però les funcionalitats d'aquestes no s'adequaven al cent per cent a les nostres, per tant es va decidir crear una eina pròpia.

Aquesta eina, utilitza la funcionalitat de wget [14], on jugant una mica amb les diferents opcions que ens facilita, podem anar accedint a les diferents parts de la web de manera automàtica i anar guardant les respostes que ens proporciona el servidor. Tal i com s'ha comentat anteriorment, al guardar en local les respostes del servidor, aquestes poden ser consultades en qualsevol moment sense la necessitat de tenir connexió a Internet, i d'aquesta forma algunes proves es podran fer directament sobre les respostes del servidor, creant menys soroll[15] als logs del mateix.

5.3.2. Cercador de logins

Aquest projecte es centra principalment en les vulnerabilitats que afecten a l'autenticació i a l'autorització dels serveis que s'ofereixen en Internet. Per aquest motiu, els formularis de login són un punt molt important a analitzar, ja que aquest són el principal punt per accedir com usuari identificat a l'aplicació.

La nostra aplicació disposarà d'un mòdul que es capaç de buscar entre totes les pàgines que conformen la web aquelles que contenen algun formulari de login, facilitant aquesta informació a l'usuari, sense necessitat de que hagi d'anar consultant cada una de les planes per buscar aquest tipus de formulari.

Per poder fer això, es va realitzar un anàlisis de com funcionen els formularis de logins per intentar automatitzar aquest procés de detecció.

En un primer moment s'havia pensat en utilitzar el nom dels camps d'usuari i password, però en aquesta anàlisi es va veure que l'única característica que tenien en comú era el camp password, on el tipus d'aquest sempre es “password”, ja que els altres encara que la majoria de vegades contenien dades característiques, hi havien un altres casos en que no es complia. Per tant, aquest mòdul és l'encarregat de buscar aquelles pàgines que contenen un camp d'un formulari amb *“type=”password”* ” [16], ja que aquest sempre correspondrà a un formulari pertinent a un login (per tal de que no es mostri per pantalla el password introduït per l'usuari legítim de l'aplicació).

A l'imatge número 14 podem veure un exemple d'aquest camp en el formulari d'una pàgina web.

Per poder implementar aquesta cerca dintre del codi HTML, s'ha utilitzat la comanda *grep*[17] de UNIX, la qual gràcies a les expressions regulars ens permetia crear un patró que contemplés els diferents casos possibles.

---

[14] Programa informàtic pertinent a GNU Project que s'encarrega d'obtenir el contingut de servidors web.

[15] Nota d'autor: Amb “soroll”, es fa referència a l'aparició de les accions realitzades en els logs.

[16] Atribut del llenguatge HTML que indica el tipus d'una etiqueta <input>.

[17] Eina basada en línia de comandes que va ser desenvolupada per a sistemes UNIX la qual es basa en la cerca de text pla en fitxers podent utilitzar expressions regulars.

Figura 14. Exemple d'un formulari de login en HTTP

5.3.3. Comprovació de HTTPS

Per garantir una comunicació segura entre el client i el servidor, s'hauria d'utilitzar un protocol que xifrés totes les dades que s'intercanvien aquests, ja que si alguna persona està *sniffant* [18] la comunicació podria veure tota aquesta informació.

En molts dels serveis d'Internet aquest fet no es té en compte, ja que la informació intercanviada no es de caràcter privat i per tant no suposa cap risc si alguna altra persona pot arribar a veure-la. Tot i això, molts dels webmasters no es percaten de que si no existeix un xifratge en la comunicació, les credencials de l'usuari que s'està logejant també s'envien en clar, i per tant poden ser emmagatzemades en qualsevol proxy intermedi que existeixi en la comunicació o vista per qualsevol altra persona. Per tant, encara que en la resta de la web no s'utilitzi una connexió segura (com per exemple HTTPS), en els formularis de login sí que s'hauria d'utilitzar per poder garantir la seguretat de les credencials de l'usuari.

La nostra aplicació s'encarregarà de comprovar si aquest tipus d'informació utilitza un canal segur al ser enviada o no.

Per fer-ho, es podria analitzar el port de destí del servei web, ja que si utilitza HTTPS hauria d'utilitzar el port 443 per a qualsevol comunicació amb el servidor. Però tal i com s'ha comentat abans, es possible que només s'utilitzi un canal segur en l'enviament de les dades del login.

---

[18] Acció que implica poder observar i capturar tot el tràfic de xarxa que passa per un punt determinat.

Per poder detectar això, s'ha pogut comprovar que qualsevol formulari web que utilitzi HTTPS hauria de redireccionar aquestes dades a una url on el protocol utilitzat sigui HTTPS, i per tant aquesta sempre començarà per “https://” en comptes de per “http://”.

A l'imatge 15 podem veure l'exemple del formulari web per al login del campus virtual de a UOC on s'utilitza HTTPS per enviar les credencials d'usuari.

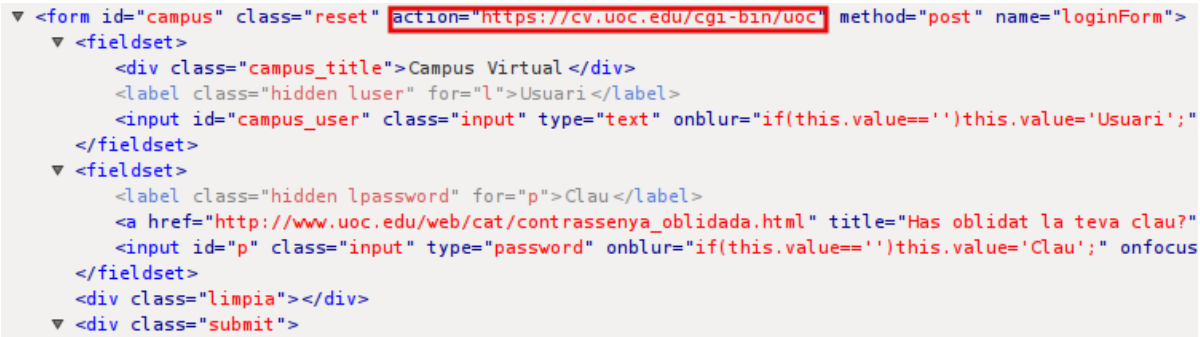

Figura 15. Exemple d'un formulari de login que utilitza HTTPS.

En aquest cas, igual que en el mòdul anterior, també s'ha utilitzat *grep* juntament amb les expressions regulars per poder implementar aquesta funcionalitat.

5.3.4. Enumeració d'usuaris

És molt habitual que els serveis que s'ofereixen a través d'Internet utilitzin diferents usuaris per poder oferir una configuració més adaptada a les seves necessitats o simplement per oferir diferents serveis. Normalment, això s'utilitza per definir diferents rols en els serveis, on cada un d'aquests tindrà diferents permisos segons el seu “status”, com per exemple poden ser l'usuari administrador que tindrà tots els permisos o l'usuari invitat que tindrà els privilegis mínims davant de l'aplicació.

Aquests comptes d'usuari han de ser de caràcter personal, i per aquest motiu aquests usen el parell usuari/contrasenya (anomenades credencials d'usuari) per poder accedir al servei i ser autenticats.

Les credencials d'usuari són un dels aspectes més importants de qualsevol servei d'Internet, ja que aquestes són utilitzades per poder identificar als diferents usuaris que existeixen i proporcionar aquelles funcionalitats o configuracions adaptades a ells. Per aquest motiu, aquest és un punt de vital importància per als atacants, ja que aquests intentaran obtindre aquestes credencials per poder accedir a funcions que no tenen permisos suficients i poder realitzar les accions que desitgin.

En molts casos, aquestes dades s'obtenen a través de programes espia[19] o virus[20] que s'introdueixen en les màquines dels usuaris, però en moltes altres ocasions, els atacants intentaran obtindre-les a través d'errors de configuració de la web o de l'administració de la mateixa. Per tant, aquest mòdul intentarà descobrir aquests errors en la pàgina a auditar, per tal de poder obtenir un llistat d'usuaris vàlids.

Per tal de diferenciar les diferents modalitats d'aquest atac, s'ha decidit realitzar un total de tres submòduls, on cada un d'ells s'encarregarà d'una funcionalitat diferent.

5.3.4.1. Obtenció d'usuaris

Una característica que facilita molt el treball, és si la pàgina web mostra diferents missatges en el cas d'entrar un usuari existent i un de no existent. Gràcies a això, es podria obtindre una llista d'usuaris vàlids simplement provant diferents noms i analitzant la resposta obtinguda pel servidor en busca del patró diferenciador.

Per fer que el nostre sistema sigui capaç de comprovar si la resposta del servidor és diferent per a un usuari vàlid i un de no vàlid, s'ha realitzat un programa en Java que simula l'operació que un navegador fa alhora de connectar-se amb un servei web.

El codi del programa el podríem representar d'una manera més visual amb els següents passos:

1. Obtenim els arguments
2. Creem el request
3. Executem el request
4. Obtenim la resposta del servidor
5. Guardem la resposta en un fitxer

Com sabem, en HTTP, les dades que l'usuari envia no sempre es fan pel mateix mètode, per tant, la nostra aplicació ha de ser capaç de distingir entre ells i donar a l'usuari l'opció d'enviar les dades necessàries pel mètode adequat. D'aquesta forma, l'usuari haurà de fer la crida al programa amb aquells valors que vol enviar al servidor indicant si ho vol fer com a GET, POST o com dades dintre de les cookies.

Una vegada el programa té les dades que vol enviar l'usuari (nom del paràmetre, valor i mètode) ha de crear el request per poder fer la consulta al servidor.

Per fer-ho, es crearà l'estructura d'una consulta HTTP tal i com es pot veure en l'imatge 16 amb les dades facilitades per l'usuari.

---

[19] Tipus específic de malware que té la finalitat d'analitzar i monitoritzar les accions que fa l'usuari en un ordinador sense el seu consentiment.

[20] Tipus específic de malware que té per objectiu alterar el comportament d'un ordinador reemplaçant arxius legítims per alters d'infectats.

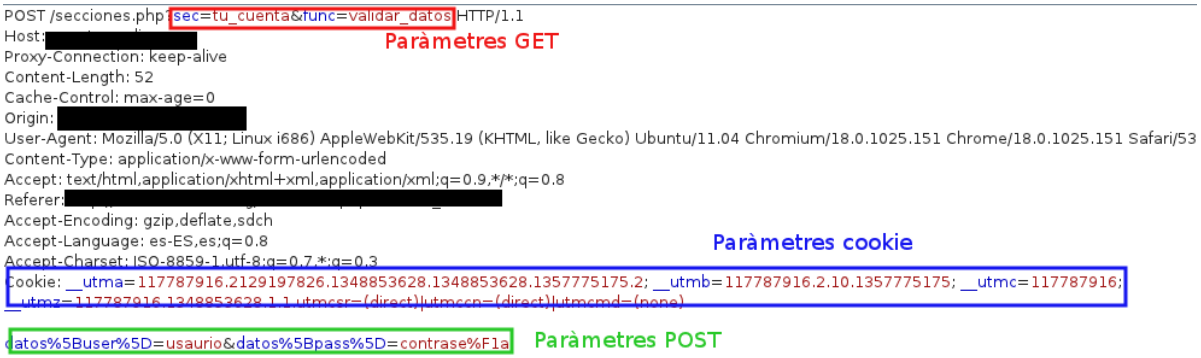

Figura 16. Exemple del request HTTP creat pel navegador al enviar un formulari de login.

Tenint el request creat, ara el que fa falta és enviar-ho al servidor utilitzant el protocol HTTP, com si es tractés d'un navegador normal. Per fer la implementació d'aquesta part s'utilitzarà un socket[21] per obrir la comunicació entre el client (nostre programa) i el servidor (servei web que es vol analitzar) i enviar el request i rebre la resposta per aquest canal.

Un cop rebuda la resposta, el nostre programa s'encarregarà d'escriure la resposta del servidor en un arxiu de text per poder-la consultar una vegada es tanca el socket, obtenint totes les capçaleres[22] utilitzades i el codi HTML.

Una vegada sabem com obtindre la resposta del servidor depenent de les dades enviades, únicament cal enviar un usuari conegut i un inexistent i comprovar si les respostes rebudes són les mateixes. En el cas de ser iguals, vol dir que el servei web no dona informació de si es tracta d'un usuari existent o no.

Després de saber com diferenciar entre usuaris existents i no existents, l'aplicació s'encarregarà d'obtindre una llista d'usuaris vàlids (sempre que la resposta del servidor sigui diferent per a un usuari vàlid i un de no vàlid) utilitzant un arxiu amb tots aquells noms que es volen comprovar. D'aquesta forma, el nostre mòdul podrà fer de manera automàtica la comprovació d'aquest.

Per realitzar-lo s'ha decidit utilitzar una de les eines que existeixen en Internet, concretament HTC-Hydra, ja que gràcies a l'estudi fet prèviament hem pogut observar que s'adequa perfectament a les nostres necessitats i es pot integrar en el nostre sistema sense gaires problemes.

En l'apartat de l'estat de l'art es pot trobar una explicació més detallada d'aquesta eina, però breument, podem dir que es tracta d'un programa que entrant les dades del formulari del login de la web desitjada, un fitxer amb diferents noms d'usuaris i un patró per diferenciar el cas d'èxit amb el de no èxit, és capaç d'automatitzar el procés d'enumeració d'usuaris i guardar els resultats en un log.

La nostra aplicació, el que farà és aprofitar el core d'aquesta eina per obtindre un llistat d'usuaris vàlid amb la informació que l'usuari de l'aplicació proporcionarà.

---

[21] Canal de comunicació virtual que estableix una comunicació entre dos punts d'una xarxa informàtica.

[22] Components del protocol HTTP que formen part en l'intercanvi de missatges i indiquen certa informació sobre els missatges.

5.3.4.2. Obtenció de credencials

Una altra possibilitat que existeix, és la d'obtindre les credencials d'usuari a través d'atacs per força bruta o de diccionari. Sabent això, la nostra aplicació també disposarà d'un mòdul capaç d'obtenir la parella usuari/contrasenya.

En aquest cas, igual que en l'anterior, s'ha decidit utilitzar HTC-Hydra, ja que també disposa d'una funcionalitat en la que es pot passar un fitxer que serà utilitzat per generar les contrasenyes que es provaran. Aquesta prova no explota cap vulnerabilitat en si mateixa, l'únic que fa és anar provant els diferents usuaris i contrasenyes que se l'hi passen per un arxiu i obtindre les respostes del servidor.

Tot i que per si sola no es tracta d'una vulnerabilitat, si que hi han moltes ocasions en les que els administradors dels propis serveis web no fan una correcta configuració d'aquests, i utilitzen comptes i contrasenyes per defecte, els quals poden ser coneguts per qualsevol persona en Internet simplement realitzant una cerca per la xarxa.

El nostre sistema, dona la possibilitat a l'usuari de seleccionar dos arxius que seran utilitzats per als noms d'usuari i contrasenya del servei web a analitzar, i retornarà una llista d'aquell parell de credencials que siguin vàlides, podent accedir d'aquesta forma al compte d'aquests usuaris sense el seu coneixement.

5.3.5. Recordatori de contrasenyes

En moltes ocasions, diferents usuaris utilitzen una mateixa màquina o sistema per accedir als serveis que s'ofereixen a Internet. Aquest, és un fet molt comú als ordinadors personals, ja que aquests solen ser utilitzats pels diferents membres d'una mateixa família.

Aquest fet, que aparentment no sembla suposar cap risc, es veu agreujat quan el navegador utilitzat per accedir a Internet, guarda informació privada dels usuaris, com pot ser per exemple les seves credencials. Això suposa un risc per a la privacitat de l'usuari, però sempre es deixa a mans de l'usuari el decidir si vol guardar aquesta contrasenya o no.

Tot i ser l'usuari de la pròpia màquina qui decideix si es vol guardar aquesta contrasenya o no, aquesta funcionalitat està creada per tal de facilitar l'entrada de les dades cada vegada que s'accedeix al recurs web, per tal d'evitar tornar a entrar-les. Com la gran majoria d'usuaris d'Internet no coneix gairebé res sobre seguretat informàtica, no són conscients del perill que això pot suposar, i per tant no presten atenció a aquest tipus de detalls i guarden informació privada de tal manera que pot ser accessible per qualsevol persona, ja que si per exemple l'ordinador es veu afectat per algun malware, aquest podria ser capaç d'accedir a aquesta informació.

Per aquest motiu, la nostra aplicació detectarà aquells formularis de login que tinguin activada l'opció de recordar la contrasenya, per tal de que sigui el propi administrador web el qui decideixi si habilitar aquesta funcionalitat o no.

Per fer-ho, es comprovarà si aquesta funcionalitat es troba desactivada en els camps password de cada un dels formularis de la pàgina web, utilitzant l'eina *grep* i expressions regulars.

5.3.6. Gestió de la caché

Continuant amb el tema d'emmagatzemar informació al navegador, ens trobem amb la caché d'aquest, la qual es capaç de "recordar" aquelles pàgines que s'han visitat. Això fa que en algunes ocasions sigui més ràpid l'accés a aquesta part de la web (ja que no fa falta consultar-la al servidor per que ja es troba en local) i que si hi ha algun error intentat fent la consulta al servidor es mostri la pàgina que hi ha guardada en caché.

Vist d'aquesta manera sembla una bona funcionalitat, ja que pot millorar la velocitat de càrrega en algunes ocasions, i preveu els possibles errors que puguin haver en la connexió amb el servidor. Però el problema apareix quan el mateix navegador és utilitzat per diverses persones, ja que aquesta informació pot ser visible per altres usuaris, accedint per exemple a un apartat web privat que hauria de requerir unes credencials d'usuari però que al estar guardada en caché, mostra la informació anterior sense necessitat de conèixer-les.

Per aquesta raó el nostre sistema de detecció de vulnerabilitats disposa d'un mòdul que es capaç de detectar, gràcies a les respostes del servidor, si la caché està activada o no. Ja que en el cas d'estar desactivada, estarà explícitament establert en el codi HTML de cada resposta del servidor, i el que es farà és recórrer el codi en busca d'aquests patrons gràcies a l'ús d'expressions regulars.

5.3.7. Captchas

Com s'ha vist en apartats anteriors, els CAPTCHAs permeten la distinció d'usuaris vers màquines, de tal forma que poden evitar l'automatització d'algunes funcionalitats gràcies al seu ús. Per aquest motiu, són molts els serveis d'Internet que els utilitzen per no ser susceptibles a atacs que utilitzin la força bruta.

Per Internet existeixen alguns serveis i eines que tracten d'evadir aquests sistemes, endevinant amb una probabilitat bastant alta el repte que es proporciona a l'usuari. Però el problema és que existeixen diferents tipus de CAPTCHAs, i per tant, cada una d'aquestes eines només serveix per a un cert tipus, essent ineficients per a CAPTCHAs que corresponguin a un altre tipus.

Aquest fet impedeix que es pugui realitzar un atac per intentar buscar vulnerabilitats automàticament sense la intervenció de l'usuari. Per aquest motiu, la nostra aplicació disposarà d'un mòdul capaç de detectar aquelles parts de la web que contenen algun tipus de CAPTCHA, per tal de que sigui l'usuari qui decideixi si vol intentar un atac manual o no (utilitzant algunes de les eines que existeixen).

5.3.8. Resultats

Una vegada es realitzen totes les proves en busca de les vulnerabilitats, s'han de mostrar els resultats d'una manera clara, i fer que estiguin disponibles en qualsevol moment, sense necessitat de tornar a executar cada una de les proves cada vegada que es vulguin consultar.

Per aquest motiu, el nostre sistema disposarà d'un mòdul que s'encarregarà d'emmagatzemar i mostrar els diferents resultats de les proves realitzades i oferirà l'opció de descarregar-se'ls a la màquina local per tal de poder visualitzar-los sempre que es vulgui.

5.3.9. Interfaç gràfica

Una vegada que es tenen tots els mòduls del nostre projecte, el que fa falta és alguna forma de poder enllaçar-los tots entre ells d'una manera còmoda i fàcil, per tal de no haver de recordar cada una de les instruccions i comandes necessàries per realitzar cada acció o veure els resultats.

L'interfaç gràfica serà la part del sistema que s'encarregarà de relacionar tots els components del mateix per a que puguin ser accessibles per l'usuari amb la major simplicitat possible. S'encarregarà de relacionar tots els components de la nostra aplicació per a que des d'un mateix lloc es puguin realitzar totes les accions i consultes necessàries sobre el servei web analitzat.

Per aquest motiu, s'ha decidit crear una interfaç web simple i intuïtiva, que sigui accessible des de qualsevol lloc, que eviti la confusió de l'usuari, que permeti realitzar absolutament totes les funcions de l'aplicació i que pugui ser utilitzada sense la necessitat de tenir amplis coneixements sobre sistemes informàtics o programació web.

El disseny de la web es basa en un menú des de el que l'usuari podrà seleccionar l'acció que desitgi realitzar en cada moment simplement accedint a ella. Per poder accedir a aquest menú, l'usuari haurà de fer una autenticació dintre de l'interfaç introduint el seu nom d'usuari i contrasenya.

Aquest control d'accés s'ha realitzat utilitzant les funcions que ofereix Apache[G] gràcies al seu arxiu .htpasswd, el qual conté una sèrie d'usuaris vàlids (emmagatzemats per parella usuari:password_xifrat) que el propi Apache s'encarrega de validar quan s'introdueixen. Com a nota, comentar que al estar emmagatzemats en un fitxer de text pla, aquestes contrasenyes apareixen xifrades per tal de que no puguin ser visibles a simple vista.

Un vegada l'usuari s'ha autenticat correctament, s'accedeix a la pàgina inicial, la qual conté un menú amb un total de 8 apartats (tal i com es pot veure en l'imatge 17), i l'usuari simplement haurà de clicar en l'opció desitjada i seguir els passos que es descriuen en cada cas.

Figura 17. Pàgina principal del panell d'administració.

5.3.9.1. Web Spider

Al accedir a aquest apartat, podrem veure la llista de les web que s'han analitzat anteriorment (en el cas d'existir alguna) i que per tant estan emmagatzemades en local, i també és dona la possibilitat d'introduir la direcció url de la nova web que es vol escanejar, tal i com es pot veure en l'imatge 18.

Figura 18. Panell d'administració del Web Spider.

Per fer això, es recorre el directori on es guarden les diferents webs emmagatzemades i es crea una taula amb cada una d'elles, juntament amb l'opció de visualitzar-les o d'eliminar-les, per tal de mantindre "ordenades" les pàgines web analitzades i poder eliminar aquelles que no necessitem més. D'aquesta manera, podrem tenir l'aplicació el més neta possible, per tal de que sigui més fàcil la seva administració.

En el cas de que l'spider estigui realitzant algun anàlisi en el moment que s'accedeix a aquesta secció, es mostrarà un missatge avisant a l'usuari i es donarà l'opció de parar-lo. Aquest funcionament s'ha realitzat utilitzant la instrucció *ps aux* [23], amb la qual trobarem l'estat actual d'aquest servei.

Des d'aquest mateix apartat, una vegada s'hagi realitzat un anàlisi amb l'Spider, es donarà l'oportunitat a l'usuari d'obtindre una llista d'aquelles pàgines que contenen algun login. Per a això, simplement haurà de visualitzar el resultat de l'spider i seleccionar l'opció de mostrar logins.

5.3.9.2. HTTPS

Aquest apartat serà l'encarregat de buscar quines són les pàgines que no envien les credencials d'usuari per un canal segur (HTTPS).

Al entrar, podrem veure una taula amb les diferents webs que ha analitzat l'Spider anteriorment i l'usuari únicament haurà de seleccionar "Analitzar" al costat del host que vol escanejar per tal d'obtindre la llista de logins que no utilitzen HTTPS. Comentar, que en el cas de no existir cap web analitzada anteriorment, es mostrarà un missatge informant a l'usuari.

Cal dir que des d'aquest apartat (igual que en l'anterior) també es poden eliminar les webs que s'han analitzat anteriorment en el cas de no ser necessàries.

5.3.9.3. Enumeració d'usuaris

Aquest apartat serà l'encarregat de realitzar l'enumeració d'usuaris segons els criteris explicats anteriorment. Quan l'usuari accedeixi, podrà seleccionar entre les tres opcions disponibles.

La primera s'encarregarà de comprovar si el servei web respon diferent davant un usuari existent i un de no existent. La segona intentarà buscar usuaris vàlids aprofitant el resultat afirmatiu de la prova número 1. Per últim, la tercera intentarà buscar credencials vàlides d'usuari a partir de l'ús de fitxers que contindran la llista de valors a provar.

---

[23] Instrucció que s'utilitza en UNIX per comprovar els processos que s'estan executant en una màquina.

5.3.9.3.1. Opció 1

Al entrar en la primera opció, veurem un formulari en el qual se'ns demanen les dades necessàries per poder saber si la resposta del servidor es diferent en cada cas. Tal i com podem veure en l'imatge 19.

Figura 19. Formulari de la prova número 1 d'enumeració d'usuaris

Aquestes dades són: el host sobre el que es volen fer les proves, el tipus d'enviament de dades (GET o POST), la url exacta del formulari de login, el nom dels paràmetres que corresponen a l'usuari i al password, un usuari existent i un de no existent i alguna capçalera o cookie en cas de ser necessària.

Una vegada entrades aquestes dades, l'aplicació utilitzarà el codi Java creat (explicat anteriorment) i mostrarà el resultat final, dient si la resposta del servidor és igual o diferent en ambdues casos.

5.3.9.3.2. Opció 2

La segona opció, és la que s'encarrega d'obtenir un llistat d'usuaris aprofitant la “vulnerabilitat” trobada en l'opció anterior.

Quan l'usuari accedeix a aquesta opció, es comprovarà l'estat actual d'aquest mòdul, i en el cas d'estar en funcionament, es mostrarà un missatge advertint a l'usuari que actualment està en ús i se li donarà l''opció de poder parar-ho.

En el cas de no estar en funcionament, se li mostrarà un formulari (molt semblant a l'anterior) que li demanarà les següents dades: host sobre el que es volen fer les proves, el tipus d'enviament de dades (GET/POST i HTTP/HTTPS), la url exacta del formulari de login, el nom dels paràmetres que corresponen a l'usuari i al password, capçalera o cookie a enviar en cas de ser necessària, la cadena amb la que es farà match per poder detectar si l'usuari existeix o no (indicant també si aquesta cadena apareixerà quan l'usuari sigui vàlid o quan no ho sigui) i per últim l'usuari haurà de seleccionar el fitxer local que contindrà tota la llista dels usuaris que voldrà provar (on cada línia del fitxer contindrà un nom d'usuari).

L'aplicació, al rebre aquestes dades, comprovarà que els valors són vàlids i executarà THC-Hydra amb les dades enviades per l'usuari. Com que HTC-Hydra espera un valor de password, l'aplicació utilitzarà un qualsevol, ja que aquest no el necessitem per a res, i Hydra "pensarà" que estem intentant obtenir les credencials utilitzant aquell password.

Com que Hydra guarda els resultats en el fitxer de logs establert, per llegir els resultats, el que farem serà accedir a aquest fitxer de log, obtenir els noms dels usuaris i crear un nou fitxer que contingui només aquesta informació, eliminant totes aquelles dades que genera el programa i que no ens interessen. Després s'esborrarà el fitxer de log creat per Hydra per tal de poder estalviar espai en disc i que no es vagi omplint el fitxer de log amb totes les proves realitzades.

5.3.9.3.2. Opció 3

La tercera i última opció del mòdul de l'enumeració d'usuaris serà exactament igual que la segona però amb la diferència que en aquesta es requeriran dos fitxers pujats per l'usuari. El primer d'ells serà el que contindrà la llista de noms d'usuari a provar, i el segon la llista de passwords. Comentar que quan es crida a Hydra, en aquesta ocasió sí que se li passarà per paràmetre el segon fitxer per poder comprovar les contrasenyes i no únicament els noms d'usuari.

A l'imatge 20, podem trobar un exemple de les credencials d'usuari obtingudes gràcies a la utilització de l'opció 3 d'aquest mòdul. Cal dir que en aquesta imatge s'ha ocutat el nom de la web analitzada per tal de no mostrar cap informació privada dels usuaris trobats.

Figura 20. Resultat de les credencials trobades utilitzant l'enumeració d'usuaris.

5.3.9.4. Recordatori de contrasenyes

Quan l'usuari selecciona aquesta opció, l'interfaç mostrarà la llista de les webs emmagatzemades en el servidor i l'usuari podrà seleccionar aquella en la que es volen fer les proves (o eliminar-la en cas de que es vulgui). En el cas de no existir cap, es mostrarà un missatge a l'usuari.

Per fer això, únicament haurà de clicar en “Analizar” al costat del host que es vol analitzar i l'interfaç mostrarà el llistat de les pàgines de la web que contenen el recordatori de contrasenyes desactivat o un missatge en el cas de no trobar cap.

5.3.9.5. Gestió de la caché

Aquest apartat, com el seu propi nom indica, s'encarregarà de comprovar com es gestiona la caché en el navegador quan s'accedeix a la web.

Com en els anteriors punts, l'interfaç mostrarà la llista amb els diferents hosts analitzats prèviament i donarà l'opció d'eliminar-los del servidor o d'analitzar-los en busca d'aquest tipus de vulnerabilitat.

Els resultats mostrats, seran la llista de totes aquelles web que contenen la caché desactivada, per tal de que l'usuari sigui capaç de decidir si alguna de la resta hauria d'estar desactivada

també (comparant els resultats d'aquest mòdul amb els de l'Spider). S'ha decidit mostrar els resultats d'aquesta forma, ja que el més usual és que la caché no estigui desactivada, i per tant, la llista podria ser massa llarga.

5.3.9.6. Resultats

Aquest apartat de l'interfaç serà l'encarregat de mostrar d'una manera fàcil els resultats dels diferents mòduls de l'aplicació.

Quan l'usuari accedeixi podrà veure un llistat dels diferents mòduls de l'aplicació (tal i com es pot veure en l'imatge 21) i simplement haurà de seleccionar aquells resultats que vol consultar. Al seleccionar-ho, se'ns mostrarà una nova llista amb totes les webs que han sigut analitzades anteriorment, i al clicar en una d'elles es mostraran els resultats d'aquesta. Un exemple podria ser el que hem vist anteriorment a l'imatge 20, on podem observar la llista de credencials obtingudes en una web després d'haver executat l'opció 3 de l'enumeració d'usuaris.

Cal remarcar que aquest apartat no executa cap tipus d'anàlisi, únicament s'encarrega de recopilar els resultats dels altres mòduls accedint als fitxers que aquests creen cada cop que s'executen les proves d'una certa vulnerabilitat. Aquests resultats seran mostrats en format de llista per que així sigui més fàcil la seva visualització.

Figura 21. Menú amb els diferents resultats que es poden consultar.

Com s'ha comentat en altres apartats, es vol una aplicació que permeti accedir als resultats en qualsevol moment. Per aquest mateix motiu, aquest apartat disposa d'una funcionalitat per poder descarregar un fitxer de text amb els resultats de cada una de les proves.

El fet de que s'hagi decidit guardar els resultats en un fitxer pla és per la facilitat que suposa importar-los a altres eines, ja que aquest fitxer únicament emmagatzemarà aquella informació completament necessària.

Capítol 6: Proves

En aquest capítol és on s'explicaran tots aquells aspectes relacionats amb les proves realitzades i amb els resultats obtinguts en aquestes. Per fer-ho d'una manera més clara, el dividirem en dos subapartats, on en un d'ells s'explicaran les proves realitzades i en l'altre els resultats obtinguts.

6.1. Proves realitzades

Com s'ha vist en l'apartat de desenvolupament (capítol 5), el nostre projecte s'ha dividit en diferents mòduls que permeten realitzar les diverses funcions que especificaven en els requeriments d'aquest. D'aquesta forma, la seva elaboració i comprovació es feia més fàcil i més exacta.

Cada cop que es desenvolupava una funcionalitat de l'aplicació es testejava abans de passar-la a l'aplicació final per assegurar-nos que el seu funcionament era el correcte i que els possibles errors que es poguessin generar no afectessin als altres mòduls. Per tant, podem dir que abans de fer la integració de tots els mòduls, ens asseguràvem que aquests funcionaven correctament independentment dels altres.

El primer mòdul que es va desenvolupar va ser el Web Spider, ja que com s'ha dit anteriorment, aquest és el nucli principal de la nostra aplicació, ja que aquest és el que s'encarrega de recórrer la web a auditar en busca de tots els enllaços disponibles per tal de poder obtindre les respostes del servidor per a cada un d'ells i després poder analitzar-les en busca de vulnerabilitats.

En una primer moment no es va pensar en l'elaboració de la funcionalitat de parar el Web Spider, però quan es van fer les primeres proves, es va descobrir que algunes webs eren molt grans, i que el Web Spider podia arribar a trigar molt en recorre-les senceres. Per tant, es va pensar en l'idea de donar a l'usuari l'opció de parar-ho, ja que d'aquesta forma es podria parar aquest per poder analitzar una altra web en cas de ser necessari o simplement perquè la

informació recopilada era suficient.

Els següent mòduls en ser desenvolupats van ser els que s'encarregaven d'analitzar el codi HTML que es rebien per part del servidor, ja que la forma de treball d'aquests era molt similar. D'aquesta manera, varem comprovar com segons la pàgina web que s'analitzava hi havien alguns casos que no acabaven de funcionar be.

Investigant una mica més, varem veure que l'error era a causa de que alguns webmasters afegien “espais” en el codi i uns altres no, i també hi havien alguns que utilitzaven la cometa simple (') en comptes de la doble (“). Així que es van modificar les expressions regulars que s'utilitzaven en cada cas per poder contemplar tots els casos, obtenint així la funcionalitat desitjada.

Els últims mòduls en ser testejats independentment van ser els que feien referència a l'enumeració d'usuaris, ja que aquests eren bastant diferents a la resta.

Per comprovar que s'adequaven a les diferents pàgines existents, es van analitzar els request que el client enviava al servidor i les respostes que aquest proporcionava. D'aquesta manera es va veure que la nostra aplicació havia d'oferir l'opció a l'usuari de poder enviar dades amb capçaleres o cookies dintre del request creat. Així que una vegada es va implementar això, es va llançar l'atac contra diverses webs i vam poder comprovar com el funcionament era l'esperat.

La part de la integració amb l'interfaç gràfica, no va suposar molts problemes, ja que al ser testejats els mòduls independentment, sabíem que aquests funcionaven correctament.

Els problemes més greus que es van donar alhora de crear la interfície era el tema de tractar les diferents opcions que es podien donar, ja que si algú modificava el contingut de les variables que s'enviaven, l'aplicació podria respondre de manera incorrecte. Per tant, es van crear diferents filtres que asseguraven que les dades rebudes eren les correctes i si no era així es mostrava un missatge d'error a l'usuari.

6.2. Resultats obtinguts

Sabent que la nostra aplicació funcionava correctament amb tots els casos, es va decidir testejar-la completament, realitzant diferents anàlisis contra pàgines webs per tal d'obtindre una idea general de la seguretat dels sistemes d'autorització i d'autenticació que aquestes oferien, i així també comprovar l'efectivitat del nostre sistema de detecció de vulnerabilitats.

Per motius de privacitat, no es farà pública la llista de webs analitzades ni les vulnerabilitats (o informació) trobades, simplement es comentaran els aspectes més importants dels resultats obtinguts.

Un aspecte molt important a destacar, és que la majoria de les pàgines webs analitzades (tret de les de les grans organitzacions) no utilitzaven un medi segur per enviar les credencials dels usuaris. Per tant, si algun usuari rebia un atac de *man-in-the-midle* [24], l'atacant podria obtenir d'una forma molt fàcil les credencials d'usuari, ja que encara que aquestes eren

enviades per POST, al tenir accés al contingut complert, es podien obtenir sense cap problema.

Una altra dada molt important a destacar, és que gairebé el total de les webs analitzades, no mostrava més informació de la necessària en els logins, és a dir, que quan s'introduïa un usuari existent i un de no existent amb la contrasenya incorrecte, no es deia si l'usuari existia o no existia. Simplement es deia que el login era incorrecte i que comprovéssim l'usuari i la contrasenya entrada. Per tant, l'opció 2 del mòdul d'enumeració d'usuaris és de les que menys es va poder executar, ja que al ser l'opció 1 d'aquest negativa, no tenia sentit executar la segona.

L'opció 3 d'aquest mòdul, en canvi, ens va donar alguna informació molt interessant sobre les costums dels usuaris (i d'alguns administradors) alhora de crear els seus comptes, ja que moltes d'aquests credencials eren molt fàcils d'endevinar, com per exemple camps de contrasenya amb “asdasd” o “123456”.

Un fet que va cridar molt l'atenció, va ser quan es va provar l'opció 3 del sistema d'enumeració d'usuaris amb el mateix fitxer d'usuaris com a fitxer de contrasenyes, ja que en alguns casos, es va trobar que hi havien usuaris que tenien el mateix nom d'usuari com a contrasenya.

Altres resultats importants de destacar és que la majoria de pàgines web disposava de l'opció de recordar la contrasenya en els camps del password, deixant en mans de l'usuari final aquesta decisió. Això és degut a que l'usuari “típic” d'un servei web, busca la comoditat en aquest, i per tant, el fet de no haver d'omplir aquests camps cada cop que accedeix a la pàgina web, dóna “punts a favor” per a continuar usant aquest servei.

Per últim, comentar que la caché es troba per norma general activada, i que els sistemes de CAPTCHA no són molt usuals en els formularis web, tret d'aquelles web considerades més “series” o “importants”.

[24] Atac d'espionatge en xarxes que es basa en el posicionament d'una tercera persona aliena entre els dos punts legítims de la comunicació, poden visualitzar i/o modificar qualsevol informació que s'envii.

Capítol 7: Conclusions i treballs futurs

7.1. Conclusions

Al finalitzar l'elaboració d'aquest projecte de final de màster basat en la creació d'un sistema de detecció de vulnerabilitats, es pot dir que s'han aconseguit els coneixements esperats a l'inici de la creació del mateix.

S'ha realitzat un complert anàlisi sobre les diferents vulnerabilitats web existents i més especialment sobre aquelles que afecten als sistemes d'autorització i d'autenticació, estudiant com es poden realitzar els diferents vectors d'atac per explotar-les.

Un altre dels objectius del projecte era realitzar un estudi sobre com funcionaven les eines existents que tenen com a finalitat detectar les diferents vulnerabilitats existents en els serveis que s'ofereixen a través d'Internet, les quals són eines indispensables per a qualsevol auditor de seguretat web i per molts dels administradors o webmasters d'avui dia que es preocupen per la seguretat dels seus serveis.

A més a més del comentat anteriorment, s'ha pogut observar com es pot crear un nou producte per a aquelles persones que desitgin tenir una seguretat extra en els serveis que ofereixen o per ajudar a garantir la seguretat de molts dels portals que hi han en la xarxa gràcies a la utilització d'aquesta nova eina.

Tot el procés necessari d'elaboració d'una nova eina (o producte) s'ha treballat en aquest projecte, des d'una planificació inicial amb les diferents especificacions i requeriments, passant per un estudi de l'estat de l'art dels diferents productes existents en l'actualitat que tenen una funcionalitat igual i/o semblant, fins a realitzar un anàlisi complert de les necessitats i especificacions sol·licitades per l'usuari final de l'aplicació per poder arribar a implementar una correcta solució que compleixi cada una d'elles.

Per últim, s'ha comprovat que tot i haver finalitzat la creació del sistema desitjat, no es pot completar el mateix fins que no s'hagin realitzat totes les proves necessàries que indiquin que l'aplicació funciona tal i com s'esperava, realitzant diferents proves per veure com si els sistema respon correctament a cada una d'elles.

7.2. Treballs futurs

Com s'ha comentat al llarg de la memòria, aquest projecte pretenia realitzar una aplicació que sigués capaç de detectar certes vulnerabilitats web de la forma el més automàtica possible, i es volia que sigués un sistema fàcil d'administrar i executar.

Encara que s'han complert els objectius i les especificacions establertes, podem dir que aquesta podria ser la primera versió del nostre sistema de detecció de vulnerabilitats, i que hi han algunes funcionalitats i/o modificacions que es podrien afegir en posteriors versions, per tal de poder ampliar la capacitat d'anàlisi d'aquest.

En els següents punts podem trobar una sèrie de millores que s'han estudiat i que podrien ser implementades en posteriors versions de la nostra aplicació.

7.2.1. Afegir diccionaris al servidor

Una funcionalitat extra que es va pensar durant l'elaboració del nostre sistema de detecció de vulnerabilitats, va ser la d'implementar un mòdul que sigués capaç d'administrar diferents diccionaris per a les proves d'enumeració d'usuaris. Donant així la possibilitat a l'usuari de l'aplicació l'opció de triar alguns dels diccionaris existents en el servidor.

Per poder implementar això, s'haurien de modificar una mica les opcions 2 i 3 del mòdul d'enumeració d'usuaris, bàsicament la integració amb l'interfaç web.

Per fer-ho, s'afegiria una opció més en aquesta, la qual permetés la pujada de diccionaris al servidor amb una estructura definida XML[25] definida que contingués la informació d'aquest diccionari, com podria ser el tema de les paraules que conté, data de la seva creació, nom del creador, número de paraules, etc. D'aquesta manera, es facilitaria la feina a l'usuari de l'aplicació, perquè aquest no hauria de preocupar-se en buscar diccionaris si no en té cap i vol realitzar les proves. Simplement hauria d'entrar a l'apartat de diccionaris i visualitzar els que hi han disponibles.

Un exemple d'estructura XML per al diccionari podria ser el que trobem a la imatge 22 per exemple.

Per facilitar encara més la feina, es podria oferir una funcionalitat que permetés crear el diccionari on-line, és a dir, a través d'un formulari web oferit per la pròpia interfaç web. L'usuari podria crear un nou diccionari amb les paraules desitjades, i l'aplicació s'encarregaria de crear l'arxiu XML corresponent.

---

[25] eXtensivle Markul Language, Metallenguantje extensible d'etiquetes creat pel W3C molt utilitzat en el disseny web per a compartir informació d'una manera estructurada.

```xml
<?xml version='1.0' encoding='iso-8859-1' ?>
<diccionari>
        <titol>Jugadors de futbol</titol>
        <versio>1</versio>
        <autor>Rafa Paez</autor>
        <tematica>Noms de jugadors de futbol</tematica>
        <numero_paraules>325</numero_paraules>
        <data_creacio>28-12-2012</data_creacio>

        <llista_paraules>
                <paraula>beckham</paraula>
                <paraula>ronaldo</paraula>
                <paraula>cristiano</paraula>
                        ...
                <paraula>messi</paraula>
                <paraula>iniesta</paraula>
                <paraula>romario</paraula>
        </llista_paraules>
</diccionari>
```

Figura 22. Exemple de possible estructura d'un diccionari XML.

7.2.2. Atacs per força bruta

Continuant amb la millora de l'enumeració d'usuaris, es podria crear un mòdul que sigués capaç d'explotar aquesta vulnerabilitat utilitzant la força bruta en comptes d'atacs per diccionari.

Per implementar-ho s'hauria de crear un nou formulari web que donés l'opció a l'usuari de triar amb quin conjunt de caràcters vol treballar (per exemple “aA-zZ0-9”) i amb quina longitud de paraules (6 caràcters, 8 caràcters, etc), creant així un nou diccionari que contingués totes les possibles opcions segons l'elecció feta per l'usuari.

D'aquesta forma, el mòdul d'enumeració d'usuaris ampliaria el seu rang d'atac i podria obtindre així un nombre major de credencials o usuaris vàlids.

7.2.3. Atacs d'SQL injection

Una altra de les proves que hagués sigut interessant desenvolupar en aquest projecte, és la del mòdul de bypass d'autenticació, per poder accedir a aquelles parts de la web com un usuari autenticat però sense conèixer les credencials d'aquest.

Per poder realitzar aquest tipus d'accions, les injeccions SQL poden ser de molta utilitat, ja que gràcies a elles podríem aprofitar la no-validació de les dades entrades per l'usuari en el servei web per tal de crear una consulta a la base de dades que retorni un usuari i que ens

permeti accedir com a usuari registrat sense conèixer cap informació.

La implementació d'aquest mòdul es podria crear utilitzant l'ús de diccionaris i l'eina HTC-Hydra, on els diccionaris en comptes de contindre noms d'usuari, contindrien injeccions SQL que podrien arribar a permetre el bypass del sistema d'autenticació del servei web.

Per integrar-ho amb la nostra aplicació, es podrien crear diferents diccionaris, on cadascun d'ells contindria les instruccions per a cada una de les diferents bases de dades existents (Oracle, MySQL, MS-SQL, etc) i l'usauri a través d'un petit formulari web proporcionat per l'interfaç gràfica, seria capaç de seleccionar (en el cas de conèixer) quin tipus de base de dades es vol atacar. Si no conegués quin servidor de BBDD s'està utilitzant, podria marcar una opció per tal de que la pròpia aplicació intentés explotar la vulnerabilitat automàticament.

7.2.4. Integració amb la detecció d'altres tipus de vulnerabilitats

Com s'ha pogut comprovar en l'estudi previ a l'elaboració d'aquest projecte, existeixen diferents tipus de vulnerabilitats web, que es poden agrupar segons el punt que es vol atacar, però en l'elaboració d'aquest, s'ha optat per implementar una eina que només es centri en la part d'autorització i autenticació. Aquest fet suposa que no es tinguin en compte altres vulnerabilitats, com poden ser les injeccions de codi o la gestió de les sessions.

En la nostra aplicació s'ha decidit utilitzar diferents mòduls per realitzar cada una de les funcionalitats requerides, i una vegada es tenien totes, s'han anat integrant una a una en una interfaç gràfica per facilitar la seva execució. D'aquesta manera, és possible afegir més mòduls per tal d'ampliar l'abast de detecció d'aquesta eina.

Per fer això, s'haurien de crear els mòduls desitjats (podent-se crear independentment de la nostra aplicació) i després fer unes modificacions “mínimes” per poder-los integrar al nostre sistema de detecció de vulnerabilitats web, creant així un sistema més robust i escalable.

7.2.5. Importar reports d'auditories

Per últim, una altra idea que s'ha pensat per a properes versions d'aquest sistema, és la importació de reports de vulnerabilitats d'altres eines per tal de recollir-les totes en la nostra aplicació i poder visualitzar-les d'una manera més còmode.

Aquesta nova millora, requeriria un estudi sobre com creen els seus reports i logs les diferents eines existents, centrant-nos en la seva estructura, per tal de poder crear algun tipus de parser[26] que sigués capaç d'obtindre la informació d'aquests i poder crear l'estructura necessària per a la nostra aplicació.

---

[26] Eina que s'utilitza per analitza algun codi o text per poder representar-ho amb una altra estructura.

Referències bibliogràfiques

A) Fernando Catoira. *Vulnerabilidades, estadísticas e impactos (http://blogs.eset-la.com/laboratorio/2012/07/31/vulnerabilidades-estadisticas-e-impacto/)* Juliol del 2012. Accessible en la data 11-01-13

B) *RFC 4346 - The TLS Protocol, versión 1.1* (*http://tools.ietf.org/html/rfc4346*). Accessible en la data 11-01-13

C) Justin Clarke. *SQL Injection Attacks and Defense, Second Edition.* Juny del 2009

D) Henry S. Baird and Daniel P. Lopresti. *Human Interactive Proofs: Second International Workshop, HIP 2005.* Maig del 2005.

E) Richard A. Molin, *RSA and Public-Key Cryptography*. Novembre del 2002.

F) S. H. Unger. *"Hazards, Critical Races, and Metastability," Computers, IEEE Transactions.* En vol.44, no.6, pp.754-768. Junny del 1995

G) Ben Laurie, Peter Laurie. *Apache: The Definitive Guide*. Desembre del 2002.