Comparativa entre *pentest* manual i automatitzat: anàlisi i execució

Joel Orellana Fernández

Juny de 2020

Resum – En aquest món globalitzat, Internet és omnipresent i les empreses ara més que mai, confien i fan ús d'ell. Malauradament, l'abast d'aquest produeix que les organitzacions i els seus recursos siguin més vulnerables que mai i un bon objectiu pels cibercriminals. Com a conseqüència, la ciberseguretat ha desenvolupat un paper essencial per a qualsevol organització, essent, una de les màximes prioritats d'aquestes. El fet, però, és que la majoria d'empreses no acaben de comprendre com mantenir els seus recursos sota control. Un dels punts més vulnerables de les empreses són les seves aplicacions web, ja que es troben publicades a Internet i poden arribar a tractar amb dades confidencials, tant de clients com de l'organització en si. Amb la intenció de protegir aquest punt d'atac, contracten a empreses de ciberseguretat les quals els hi ofereixen realitzar auditories de seguretat web, executant tests de penetració manuals i/o automatitzats. Com les empreses les quals volen l'auditoria no solen saber quin tipus de tests són els més eficients, es decanten pel més econòmic. Aquest projecte pretén ser un exemple i demostrar quina seria la millor tècnica a l'hora de realitzar una auditoria de seguretat web.

Paraules clau – Aplicació web, Auditoria de seguretat, Automatització, Cibercriminal, Ciberseguretat, *Common Vulnerability Score System*, Escaneig de vulnerabilitats, *Exploit*, OWASP, *Pentest*, Risc, Test de penetració, Vulnerabilitat.

Abstract – In this globalized world, the Internet is omnipresent, and companies now more than ever, trust and make use of it. Unfortunately, this widespread phenomenon makes organizations and their resources more vulnerable than ever and a good target for cybercriminals. As a result, cybersecurity has played an essential role in any organization, being, today, one of the top priorities. The fact, however, is that most companies don't understand how to keep their resources under control. One of the most vulnerable points of a company is its web application, as it is on the Internet and deals with a lot of confidential data, both from customers and the organization itself. To protect this point of attack, they hire cybersecurity companies that offer them web security audits, performing a manual or automated penetration test. As the client company does not know which will provide better results, it decides to use the cheapest. That is why this project aims to be an example and demonstrate what would be the best technique when conducting a security audit for a web application.

Keywords – Automation, Common Vulnerability Score System, Cybercriminal, Cybersecurity, Exploit, OWASP, Penetration Test, Pentest, Risk, Security Audit, Vulnerability, Vulnerability Scanning, Web Application.

1 INTRODUCCIÓ

DES de fa anys, la ciberseguretat [1] és un aspecte fonamental en els processos duts a terme diàriament per qualsevol indústria, desenvolupant-se com a necessitat i convertint-se en unes de les màximes prioritats per a qualsevol organització. Els pirates informàtics [2], operant des d'arreu del món sense cap restricció, tenen com a objectiu a qualsevol organització que els pugui aportar benefici, ja sigui extraient dades confidencials amb intenció maliciosa o encriptant les mateixes per, posteriorment, demanar un rescat. Les organitzacions [3], per tant, han de focalitzar els seus esforços en protegir-se a si mateixos i als seus clients i associats.

La manera més comuna de protegir-se enfront una amenaça és realitzant, periòdicament, auditories de seguretat [4]. Aquestes consisteixen a analitzar sistemàticament el nivell de protecció de la informació d'una organització, tenint com a objectiu identificar, avaluar i, posteriorment, descriure les diferents vulnerabilitats que hi poden haver en l'anàlisi. Hi ha molts tipus d'auditories de seguretat [4], les quals es poden diferenciar segons el recurs de l'empresa al qual va dirigit, segons quin professional en el sector les portarà a terme o depenent de la quantitat d'informació que es tingui sobre l'entorn.

Les organitzacions no especialitzades en el món de la ciberseguretat o informàtica, normalment, no saben què necessiten per protegir les seves dades i les dels seus clients. Com a conseqüència contracten a diferents empreses especialitzades en seguretat informàtica perquè els hi realitzin les auditories. Com quasi totes les organitzacions tenen aplicacions o pàgines web, els ciberdelinqüents intenten perpetrar la seguretat d'aquestes per tal d'obtenir un profit econòmic o de reconeixement, convertint-se aquests *websites* en el taló d'Aquil·les de moltes empreses. Per aquest motiu, l'auditoria de seguretat més comuna és la de revisió web. Les organitzacions però, intenten estalviar tant diners com temps en aquest tipus d'auditories, volent la màxima eficiència possible en la identificació de vulnerabilitats del seu recurs. Per tant, necessiten saber

• *E-mail de contacte: joel.orellana@e-campus.uab.cat*
• *Menció realitzada: Tecnologies de la Informació*
• *Treball tutoritzat per: Oscar Martin Salas (DEIC)*
• *Curs 2019/2020*

amb certesa quina o quin conjunt de tècniques són les més eficients i que proporcionen millors resultats a l'hora de realitzar una auditoria de seguretat web.

Pels motius esmentats anteriorment, en aquest projecte s'ha dut a terme una auditoria de seguretat web [4], la qual es defineix com l'anàlisi extern de l'entorn, comprovant i analitzant les diferents vulnerabilitats que es poden trobar en aquest recurs. Concretament, s'ha realitzat l'execució d'un test de penetració (*pentest*) manual [5], el qual, mitjançant el coneixement teòric i la pròpia experiència, consisteix en utilitzar un *exploit* (codi que s'aprofita d'una vulnerabilitat de seguretat d'un sistema d'informació) per tal de descobrir i aprofitar activament les debilitats de l'entorn; i un *pentest* automatitzat [5], conegut també com a escaneig de vulnerabilitats. Aquest últim es basa en identificar vulnerabilitats conegudes sense explotar-les, mitjançant un algoritme de concordança per buscar un o més patrons predefinits. Si es troba una coincidència, l'escàner deduirà que el recurs és vulnerable. Finalment, s'ha dut a terme un estudi i comparativa sobre els resultats obtinguts en les diferents tècniques d'identificació de vulnerabilitats (*pentest* manual i automatitzat), tenint en compte certs factors que es descriuran en futurs apartats.

Durant el desenvolupament del projecte, per facilitar la identificació de vulnerabilitats de manera manual, s'ha fet ús de Burp Suite [6], una eina gràfica que ajuda als analistes a provar la seguretat del *website* i que s'explicarà amb més detall en els pròxims apartats.

A l'hora de fer la tria de l'aplicació web en la qual s'ha dut a terme l'auditoria, s'ha realitzat una recerca exhaustiva tenint com a objectiu què l'aplicació tingui una aparença i utilitat semblant a una botiga de comerç electrònic, ja que aquestes aplicacions, normalment, tracten diferents dades sensibles dels clients, com ara números de targetes bancàries, correus electrònics, adreces i telèfons d'interès, etc. Després de l'anàlisi previ de les diferents aplicacions web disponibles, s'ha identificat una que compleix tots els requisits esmentats amb anterioritat: **Juice Shop**. Juice Shop [7] és un projecte de codi obert que està allotjat pel Projecte de Seguretat d'Aplicacions Web Obertes (OWASP) [8] sense ànim de lucre i que està desenvolupat i mantingut per voluntaris.

Al llarg d'aquest document tècnic es detallarà el procés de desenvolupament que s'ha dut a terme en el projecte, les diferents tecnologies aplicades per a l'execució de l'auditoria, així com una classificació de les vulnerabilitats trobades en l'aplicació web Juice Shop, fent ús de les dues tècniques d'identificació de vulnerabilitats.

Aquest article pretén informar i mostrar una sèrie d'evidències sobre quina és la millor tècnica d'identificació de vulnerabilitats a l'hora de realitzar una auditoria de seguretat per a una aplicació web.

L'organització de l'article es compon dels següents apartats:

- Introducció general i presentació del projecte
- Objectius
- Estat de l'art
- Metodologia
- Desenvolupament
- Resultats obtinguts
- Conclusions

2 OBJECTIUS

Tal com s'ha esmentat prèviament a la introducció, l'objectiu principal d'aquest projecte és definir quina tècnica d'identificació de vulnerabilitats d'una auditoria de seguretat web és més eficient i, amb la que s'obtenen uns millors resultats a l'hora d'identificar el màxim nombre de vulnerabilitats possible. Pel correcte desenvolupament del projecte, s'ha dividit la seva implementació en els següents objectius:

- Realitzar un *pentest* manual a l'aplicació web avaluada, el qual, tal com s'ha esmentat abans, consistirà en trobar qualsevol falla en l'aplicació web i aprofitar-la per identificar les debilitats de l'entorn. L'objectiu d'aquesta tècnica és identificar i analitzar les vulnerabilitats que puguin ser utilitzades en els processos comercials habituals del recurs. El *pentest* es centrarà principalment trobar vulnerabilitats en el *front-end* de l'aplicació, és a dir, en la interfície de l'usuari.
- Execució d'un test de penetració automatitzat a l'aplicació web avaluada Juice Shop. Aquest, a diferència de l'anterior, no es basa en el coneixement teòric, sinó que a partir de la seva execució, identificarà les falles conegudes mitjançant diferents patrons i comportaments predefinits. Per la realització del *pentest* automatitzat s'han fet ús de diferents escàners de vulnerabilitats que es detallaran en futurs apartats.
- Finalment, es durà a terme un estudi i comparativa de l'aplicació web sobre els resultats obtinguts en les diferents tècniques d'identificació de vulnerabilitats. En la comparativa es tindran en compte quatre factors:
    1. **Abast** el qual arriben les dues tècniques
    2. **Temps** dedicat
    3. **Cost** aproximat que suposaria per a l'empresa client
    4. **Grau de risc** de les vulnerabilitats identificades

En l'escaneig de vulnerabilitats, s'utilitzaran diferents escàners per a l'aplicatiu web, per tal d'abastar el màxim possible i obtenir així una identificació de vulnerabilitats major. A més, es realitzarà una breu avaluació del rendiment de detecció de vulnerabilitats dels escàners de vulnerabilitats d'aplicacions web, comparant quin dels seleccionats ha identificat més falles.

Per tal d'enfocar el projecte al màxim possible al món professional, en la comparativa es tindrà en compte el temps i cost que s'ha invertit en l'execució i avaluació de les dues tècniques utilitzades (test de penetració manual i escaneig de vulnerabilitats).

També es realitzarà una avaluació del risc de cada vulnerabilitat trobada per les dues tècniques, utilitzant el sistema *Common Vulnerability Score System* (CVSS) [9], el qual es defineix com una mètrica que permet avaluar la gravetat de les vulnerabilitats de seguretat del sistema informàtic, comparant quin dels dos mètodes ha pogut trobar vulnerabilitats d'un grau de risc més elevat.

Finalment, es mostraran les conclusions obtingudes de la comparativa, en les quals es detallarà quina de les dues tècniques és més eficient o si existeix una combinació de les dues amb les que s'obté un major nombre de vulnerabilitats identificades, tenint en compte el nivell de risc que proporcionen.

A més dels aspectes esmentats anteriorment, a continuació es llisten els objectius secundaris del projecte:

- Documentar cada vulnerabilitat trobada en les proves de penetració manuals i automatitzades (excloent-hi repeticions), explicant en què consisteix i afegint evidències de com s'ha realitzat l'explotació de la mateixa en el cas del *pentest* manual.
- Definir possibles conseqüències i mitigacions de cada vulnerabilitat trobada.

3 Estat de l'art

A mitjans dels anys 60 [10], la creixent popularitat dels sistemes informàtics que permetien executar diversos programes de forma simultània, va crear noves preocupacions entorn la seguretat d'aquests. Diversos dels principals experts en seguretat informàtica dels Estats Units d'Amèrica argumentaven que era molt senzill vulnerar les proteccions de seguretat dels sistemes i extreure informació sensible. Tenint com a objectiu l'estudi de la protecció dels sistemes, un grup de voluntaris va realitzar les primeres proves de seguretat, anomenades "test de penetració", les quals consistien en vulnerar els mecanismes de protecció que havien implantat. No va ser fins a la dècada dels 70 [10] que van aparèixer per primera vegada grups d'experts encarregats de realitzar diferents proves de penetració a un sistema, anomenant-los "tigre team".

En l'actualitat, la seguretat al voltant dels sistemes d'informació segueix sent una de les preocupacions clau per a tots els sectors i indústries. Encara que, la manera de dur a terme auditories de seguretat o tests de penetració s'ha anat desenvolupant i refinant cada cop més.

Les auditories de seguretat, tal com s'ha descrit anteriorment, consisteixen en l'avaluació del nivell de risc d'un sistema o aplicació en funció d'uns estàndards o bases establertes. Els estàndards de seguretat són normes obligatòries, mentre que les bases són el nivell mínim de seguretat acceptable per una organització.

El *pentesting* o *pentest* [10] consisteix a examinar i analitzar les vulnerabilitats de la infraestructura de l'organització, amb la intenció d'explotar-les d'una manera segura en un entorn controlat. El test de penetració pot ser executat de manera manual o automàtica. El manual [5] requereix un expert capacitat per a dur-lo a terme, diferents eines (interceptors, esnifadors, etc.) i, en general, sol ser un procés més lent i costós. En una situació crítica o si l'expert realitza *pentests* del mateix tipus amb freqüència, sol donar un millor resultat, ja que pot pensar com actuaria un atacant i, per tant, actuar en conseqüència. L'automàtic [5], requereix una única eina encarregada de realitzar els tests, la qual normalment fa el procés més ràpid, eficient, i fàcil d'utilitzar per a una persona que no té grans coneixements en l'àmbit. Encara que sempre serà necessària una posterior revisió manual dels resultats obtinguts.

4 Desenvolupament

4.1 Metodologia

A l'hora de plantejar una metodologia pel correcte desenvolupament del projecte, s'han tingut en compte els objectius esmentats prèviament, així com l'abast que hi haurà, essent un únic integrant en el projecte. També, com durant tota la realització del treball es duran a terme reunions de seguiment amb el tutor i entregues parcials amb el propòsit de mantenir la qualitat del projecte, s'ha proposat seguir una metodologia àgil [11], per tal de poder revisar periòdicament el correcte desenvolupament d'aquest. Concretament, s'ha decidit utilitzar la metodologia àgil *Scrum* [11], la qual és un marc de treball per a la gestió de projectes. El motiu d'aquesta elecció és l'experiència en la mateixa i, en realitzar entregues i reunions cada aproximadament 4-5 setmanes, comporta que es puguin modificar els objectius i planificació del projecte aportant flexibilitat i una millor productivitat. Finalment, durant tot el procés d'elaboració del treball s'ha disposat de l'assessorament de l'empresa de ciberseguretat de la qual l'autor d'aquest projecte forma part.

Juntament amb aquesta metodologia, s'ha establert una planificació general durant el desenvolupament del projecte que inclou les principals tasques i objectius definits a l'inici d'aquest article, les quals queden reflectides en el *product backlog* (llistat global de totes les tasques que es pretenen fer durant el desenvolupament del projecte) ordenades per prioritat i ordre d'execució:

Requisits generals

- Recaptació de requeriments i establiment dels objectius
- Definir l'abast, el marc de treball i una guia a seguir per al desenvolupament adequat del projecte
- Establiment del conjunt d'eines, escàners i l'aplicació web per realitzar els *pentests*

Pentest manual

- Recorre l'arbre *web* identificant l'estructura i les funcionalitats de l'aplicació
- Revisió de l'aplicatiu amb i sense usuari
- Revisió del marc d'autenticació i autorització
- Comprovació de la correcta gestió de les sessions d'usuari, errors de validació d'entrada i lògica del negoci

Pentest automatitzat

- Configuració dels escàners de vulnerabilitats
- Execució dels diferents escàners triats
- Eliminar dels resultats obtinguts els falsos positius

Anàlisi dels resultats

- Anàlisi dels resultats obtinguts amb l'execució dels dos tests de penetració
- Documentació de les vulnerabilitats identificades en el *pentest* manual i automatitzat
- Comparativa de resultats

La planificació s'ha anat revistant diàriament per tal de mantenir un seguiment actiu de la mateixa i un control real del temps i de les tasques que s'han hagut de realitzar. A més, en utilitzar *Scrum*, s'han definit *sprints* (cicles) setmanals, en els quals, al final de cada cicle, s'avaluava la planificació establerta per la pròxima setmana, per tal d'aportar flexibilitat i eficiència al projecte, aconseguint immediatesa en la resposta per adaptar el projecte i el seu desenvolupament a les circumstàncies específiques de l'entorn.

A causa de circumstàncies alienes, a mitjans del desenvolupament del projecte, ha sigut molt complicat complir amb les dates establertes en la planificació general. Encara que, a l'haver definit *Scrum* com a metodologia de treball per a la gestió del projecte, ha facilitat molt la reorganització de les tasques i dates, aconseguint així completar-lo adequadament.

4.2 Mètriques de classificació de vulnerabilitats

En aquesta secció es defineixen les mètriques que s'han seguit per a la classificació de les vulnerabilitats identificades amb les dues tècniques de *pentesting*.

Hi han molts mètodes i maneres de definir les diferents vulnerabilitats d'una aplicació web. En aquest cas, es classificaran fent ús de les mètriques de classificació CVSS v3.1 que van del 0 al 10 [9] i segons l'impacte [12] que podrien ocasionar des del punt de vista tecnològic i de negoci:

Crític	Alt	Mitjà	Baix	Informatiu
Vulnerabilitats que poden permetre als atacants controlar completament les aplicacions i servidors web.	Violació parcial de la integritat, disponibilitat o confidencialitat dels actius els quals suposen un risc al creixement del negoci.	Vulnerabilitats que permeten a usuaris locals o remots accedir a informació per poder desenvolupar futurs atacs que de caràcter més greu.	Vulnerabilitats que proporcionen informació que un intrús podria utilitzar per realitzar futurs atacs.	Vulnerabilitats que revelen informació que no es pot utilitzar directament per atacar actius ni ser explotades en el moment actual.
9,0 - 10	7,0 - 8,9	4,0 - 6,9	0,1 - 3,9	0,0

Taula 1: Classificació de vulnerabilitats segons el CVSS i risc que representen

Tal com es pot observar en la Taula 1, la mètrica que s'ha seguit per classificar les vulnerabilitats identificades ha consistit en calcular el CVSS en funció de les característiques de la falla trobada en l'aplicació web i, si el resultat obtingut es troba entre els valors definits a la part inferior de la taula, se li assigna el nivell de risc que pot suposar per l'organització (Crític, Alt, Mitjà, Baix o Informatiu).

4.3 Arquitectura de l'aplicació web avaluada Juice Shop

En l'arquitectura [13], l'aplicació web Juice Shop està desenvolupada purament en JavaScript i Typescript. En la interfície s'utilitza el popular marc d'aplicacions web de codi obert anomenat Angular. A més, el disseny de la interfície d'usuari està implementat amb Material Design desenvolupat per Google.

El llenguatge basat en objectes JavaScript també s'empra en el *back-end* com exclusiu: el *framework* de Node.js, Express, és l'encarregat d'enviar el codi del client-side al navegador. Com a base de dades (BD) l'aplicació utilitza SQLite, una BD relacional que, a diferència d'altres bases de dades relacionals, no funciona com a un sistema gestió amb un paradigma client-servidor. Per tant, per la comunicació amb la Base de dades, es fa servir Sequelize i finale-rest com a capa d'abstracció. Això permet l'ús d'*endpoints* de l'API creats dinàmicament per a interaccions simples (operacions CRUD [14]) amb recursos de la BD i, a l'hora, l'execució de SQL personalitzat per a consultes més complexes.

Com a emmagatzemament de dades addicional l'aplicació web utilitza MarsDB, una base de dades lleugera NoSQL de MongoDB compatible amb la resta de components.

Per últim, l'aplicació ofereix un el protocol estàndard per a l'autorització OAuth 2.0 perquè els usuaris puguin iniciar la sessió a través del seu compte de Google.

La Figura 1 mostra els components esmentats prèviament i les vies de comunicació d'alt nivell entre les capes client, servidor i dades:

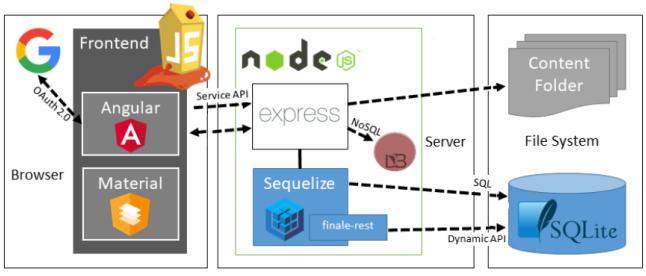

Fig. 1: Arquitectura de l'aplicació web avaluada Juice Shop

4.4 Happy Path

A l'hora d'analitzar una aplicació en recerca de vulnerabilitats de seguretat, és primordial entendre el funcionament i estructura d'aquesta abans de llançar qualsevol *exploit*. Sense aquesta comprensió primerenca del disseny de l'aplicació, és poc probable que el resultat del test de penetració manual sigui fructífer.

Una bona manera de comprendre l'aplicació és navegar per aquesta, de la mateixa manera en què un usuari ho podria fer, és a dir, explorant-la utilitzant les diferents funcionalitats que l'aplicació ofereix. Això, sovint, s'anomena *happy path testing*. El *happy path testing* [15] són un tipus de proves de programari que utilitza l'entrada coneguda i produeix una sortida esperada, és a dir, consisteixen a navegar per l'aplicació, executant les diferents funcionalitats que ofereix, mitjançant els diferents casos d'ús o el programari que la implementa.

Juice Shop, tal com s'ha esmentat anteriorment, és una aplicació de comerç electrònic que inclou els fluxos de treball típics d'una botiga web. Per tant, un exemple de *happy path testing* seria realitzar la compra d'un producte i que l'aplicació web reaccioni acord a què implicaria comprar-ne un (per exemple, mostrant a l'usuari un document amb el número de la comanda, el producte seleccionat, el preu, etc.). Aquestes proves s'han de repetir amb cada funcionalitat per tal de conèixer, exactament, com és i com reacciona l'aplicació enfront de l'execució habitual que un usuari podria fer dels diferents casos d'ús.

A continuació, en la Figura 2 es mostra la representació de les diferents seccions de l'aplicació i casos d'ús del *happy path testing*, per tal de comprendre millor el funcionament de l'aplicació web:

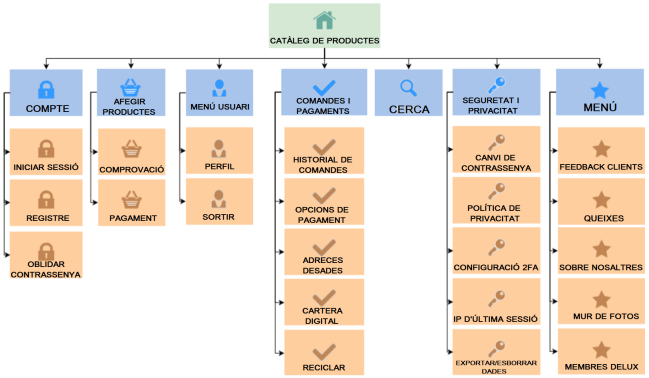

Fig. 2: *Sitemap* de l'aplicació web avaluada Juice Shop

4.5 Resum de les vulnerabilitats identificades amb el *pentest* manual

Per a l'execució del test de penetració manual, tal com s'ha esmentat en la introducció d'aquest article, s'ha fet ús de Burp Suite com a eina de reforç, el qual és una plataforma integrada per a la realització de proves de seguretat per aplicacions web. Aquesta, s'ha utilitzat com a *proxy*, permetent interceptar i modificar totes les sol·licituds i respostes entre l'aplicació objectiu (navegador) i el servidor. A més, en registrar detalls complets de totes les sol·licituds i respostes que passen bidireccionalment entre l'aplicació web avaluada i el servidor, ens permet conèixer amb exactitud com és la interacció entre aquests dos components. D'aquesta manera, s'ha pogut estudiar el comportament del servidor en enviar diferents sol·licituds malicioses amb la intenció de provar la seguretat del sistema.

També, s'ha fet ús de la metodologia OWASP per realitzar les proves de seguretat de l'aplicació web. El projecte *Web Security Testing Guide* (WSTG) [16] consisteix en una guia completa per provar la seguretat d'aplicacions web i serveis web. El WSTG, creat per diferents professionals de la ciberseguretat i voluntaris, proporciona un marc de bones pràctiques utilitzades pels encarregats de realitzar els tests de penetració i organitzacions de tot el món.

A continuació, en la Taula 2 es descriuen breument les vulnerabilitats [17] identificades a l'aplicació web avaluada, fent ús de la tècnica de penetració manual:

Nom	Descripció	Mitigació	Risc - CVSS
1. Injecció SQL	L'aplicació permet injectar codi SQL en el camp email del formulari de *login*, permetent iniciar sessió únicament amb el correu electrònic.	Utilitzar instruccions preparades i consultes parametritzades.	Crític - 10
2. Cross Site Scripting (XSS) - Stored	L'aplicació permet als usuaris injectar i emmagatzemar un codi potencialment perillós, que s'executarà als navegadors del visitant.	Validar totes les entrades dels usuaris per filtrar qualsevol mena d'injecció de codi.	Crític - 9,1
3. Afegir nous productes a través de l'API	És possible afegir i emmagatzemar nous productes a la base de dades del servidor amb un usuari registrat sense privilegis.	Configurar les llistes de control d'accés (ACL) assegurant un control adequat d'accés als recursos.	Crític - 9,1
4. XSS - Stored a través de l'API	L'aplicació permet als usuaris injectar i emmagatzemar un codi potencialment perillós, que s'executarà als navegadors del visitant.	Solucionar la vulnerabilitat 3. Afegir nous productes a través de l'API i validar les entrades dels usuaris aplicant-hi un filtre.	Crític - 9,1
5. Modificar la contrasenya de qualsevol usuari	L'aplicació permet modificar la contrasenya de qualsevol usuari, eliminant el paràmetre on s'especifica el *password* actual.	Validar els paràmetres enviats per l'usuari, per tal que no es pugui eliminar cap paràmetre.	Alt - 8,3
6. Cross Site Request Forgery (CSRF)	És possible enganyar els usuaris de l'aplicació fent que injectin codi JavaScript en el seu nom d'usuari, mitjançant peticions realitzades des de llocs web externs.	Utilitzar un token Anti-CSRF, fent que l'aplicació generi un token en qualsevol sol·licitud, evitant així aquests tipus d'atacs	Alt - 8,2
7. Identificar tots els usuaris de l'aplicació	L'aplicació permet identificar tots els usuaris registrats de l'aplicació web amb un usuari sense privilegis, realitzant una petició a un *path* concret localitzat al codi públic.	Restringir l'accés al path perquè només un administrador pugui accedir-hi.	Alt - 8,1
8. Sistema de cookies implementat incorrectament	L'aplicació permet interactuar amb els diferents usuaris o administradors de la pàgina suplantant les identitats d'altres usuaris existents (o no), modificant o eliminant les *cookies*.	Realitzar i implementar correctament una política de *cookies*, les quals identifiquin l'usuari i s'invalidin un cop l'usuari faci *logout* de l'aplicació.	Mitjà - 6,4
9. XSS - Reflected	L'aplicació permet als usuaris injectar codi a través d'un o més camps d'aplicació, que s'executaran al navegador de la víctima.	Valida l'entrada de tots els usuaris per filtrar qualsevol mena d'injecció de codi.	Mitjà - 6,1
10. Open Redirection	És possible evitar el sistema d'antiredirecció (*whitelist*) que té l'aplicació web amb la contaminació del paràmetre HTTP.	Descartar totes les peticions que incloguin un paràmetre repetit.	Mitjà - 6,1
11. Atacs de força bruta	L'aplicació no té cap mesura d'anti-força bruta o permet a un atacant realitzar aquests atacs.	Implementar un CAPTCHA o limitar el nombre d'intents d'inici de sessió	Mitjà - 5,3
12. Filtracions d'informació sensible	En l'aplicació existeixen diverses fuites d'informació a través d'errors no controlats o directoris confidencials oberts al públic.	Evitar l'accés a qualsevol part del *website* que pugui donar informació sobre el programari utilitzat.	Mitjà - 5,3
13. Afegir productes a la cistella d'altres usuaris	L'aplicació permet afegir productes a la cistella d'altres usuaris amb la contaminació del paràmetre HTTP.	Solucionar la vulnerabilitat de contaminació del paràmetre HTTP.	Mitjà - 4,9
14. Enumeració d'usuaris en la funcionalitat "forgot password"	És possible recopilar un conjunt de correus electrònics vàlids interactuant amb la funcionalitat de "*forgot password*".	L'aplicació hauria de respondre amb el mateix missatge d'error independentment si existeix o no l'email.	Baix - 3,7
15. Validació incorrecta de les dades	S'ha observat que certes operatives que, per exemple, sol·liciten la declaració d'un import monetari no comproven que aquest compleixi amb les restriccions que estableixen les polítiques del negoci.	Realitzar una correcta validació d'entrada tant a la banda del client com al servidor, evitant que s'incloguin valors no lògics i anul·lant el procés si aquest fos el cas.	Baix - 3,1
16. Unrestricted file upload	L'aplicació permet carregar fitxers en diferents formats incloent el ZIP, el qual no és recomanable, ja que poden incloure qualsevol mena d'arxiu, fent que la comprovació del tipus de fitxer sigui inútil.	Desactivar l'opció de penjar fitxers comprimits.	Informatiu - 0,0

Taula 2: Breu descripció, possible mitigació i risc de les vulnerabilitats identificades en el *pentest* manual

4.6 Resum de les vulnerabilitats identificades amb el *pentest* automatitzat

Per a les proves de penetració automatitzades (escaneig de vulnerabilitats) web, s'han executat diferents escàners amb la característica de ser tots ells un *software* de codi obert (*open source*) [18]. Aquesta condició ha estat imposada, ja que les empreses, cada vegada més, aposten per aquest tipus de *software*, implicant i animant als seus desenvolupadors a involucrar-se en les comunitats de codi obert de les quals depenen [19].

Els escàners de vulnerabilitats que s'han executat en l'aplicació web Juice Shop han sigut:

- *W3af* [20]: Web Application Attack and Audit Framework és un escàner de seguretat d'aplicacions web desenvolupant en Python, el qual és capaç d'identificar més de 200 tipus de vulnerabilitats.
- *Arachni* [21]: Arachni és un *framework* de Ruby, modular i d'alt rendiment que dóna suport als analistes de seguretat d'aplicacions web. A més, aquest és prou versàtil per abastar una gran quantitat de casos d'ús i proporciona una aplicació web d'alt rendiment altament configurable.
- *ZAP* [22]: Zed Attack Proxy és un dels escàners més populars de seguretat d'aplicacions web de codi obert i és un dels projectes OWASP més actius. A més, està pensat per ser utilitzat tant per aquelles noves persones en el món de la seguretat d'aplicacions web com pels professionals.
- *SkipFish* [23]: SkipFish és una eina activa de reconeixement de seguretat d'aplicacions web. Aquest, genera un informe final de resultats que serveix com a base per a la posterior avaluació manual de la seguretat de l'aplicació web.
- *Nikto* [24]: Nikto és un escàner de vulnerabilitats que explora els servidors web realitzant diverses proves per trobar vulnerabilitats. Aquest comprova versions obsoletes, elements de configuració del servidor, entre d'altres.

En aquest punt, he d'afegir que en un primer moment es van seleccionar els 3 primers escàners de vulnerabilitats mencionats anteriorment, els quals resultaven ser molt profitosos segons les diferents opinions de les empreses clients. Un cop els vaig executar, sorprenentment no vaig obtenir un nombre rellevant d'identificacions de vulnerabilitats en comparació al *pentest* manual. Com a primera solució, vaig optar per executar-los diverses vegades cadascun d'ells i com els resultats eren molt similars, vaig decidir afegir 2 escàners de vulnerabilitats més. Un cop més, els resultats obtinguts no van ser profitosos i, en buscar informació respecte al perquè d'aquests resultats, vaig trobar indicis [25] [26] [27] que esmentaven que aquests tipus d'escàners no tenen l'abast suficient per a les aplicacions web que inclouen una gran quantitat de directoris i nivells de profunditat. A més, com l'aplicació web avaluada Juice Shop utilitza *software* relativament nou, també complica la identificació de vulnerabilitats per aquests escàners. Finalment, vaig realitzar diferents configuracions per a cada escàner per tal d'intentar trobar una que em proporciones el nombre més gran de vulnerabilitats identificades, d'aquesta manera s'ha pogut obtenir un resultat similar a l'esperat.

També, a l'hora de configurar l'autenticació automatitzada dels escàners. Com que en l'aplicació avaluada apareixen diverses alertes i "pop-ups", va ser molt complicat crear un script que aconseguís accedir al menú de *login* i iniciar sessió amb les credencials corresponents. Al final, es va fer ús d'un altre *software* anomenat secureCode Box [28], el qual consisteix en una plataforma que inclou diversos escàners dels seleccionats i proporciona diverses plantilles per realitzar l'autenticació d'una manera senzilla.

Un cop executats tots els escàners amb les configuracions corresponents, es van obtenir un conjunt de vulnerabilitats identificades per cadascun d'ells. D'aquest total, s'han eliminat les duplicitats i falsos positius [29]. Aquests últims es produeixen a causa de les febles comprovacions estàtiques que els escàners solen utilitzar per detectar vulnerabilitats. Quan un escàner intenta detectar una vulnerabilitat coneguda, utilitza un algoritme de concordança per buscar un o més patrons predefinits dins d'una resposta HTTP. Si es troba una coincidència, l'escàner deduirà que existeix la vulnerabilitat i informarà en conseqüència. Per tal d'eliminar els falsos positius, vaig comprovar cada vulnerabilitat identificada pels escàners, per així descartar que aquell recurs sigui o no realment vulnerable.

A continuació, en la Figura 3 es mostra el nombre de vulnerabilitats identificades per a cada escàner sense eliminar els falsos positius i el risc que representen cadascuna d'elles segons la Taula 1:

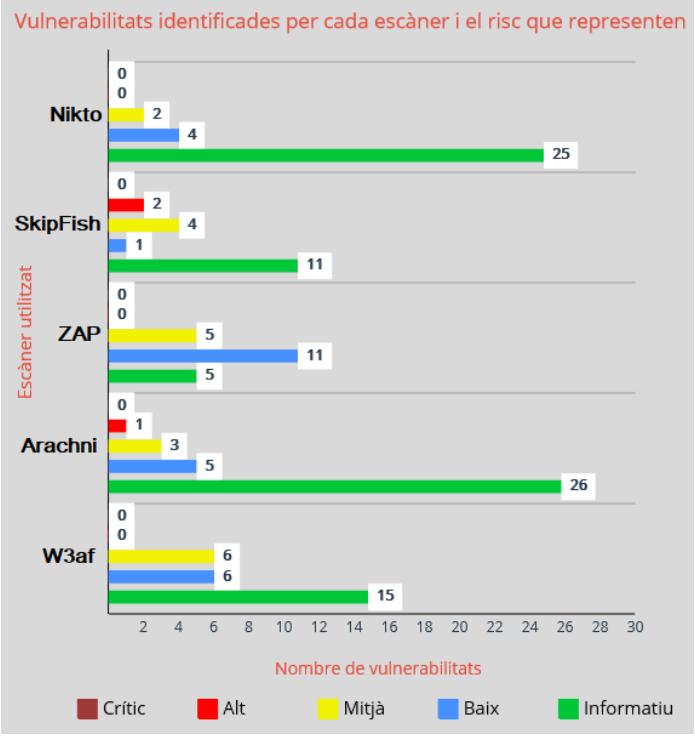

Fig. 3: Vulnerabilitats reportades per cada escàner i el risc que representen, sense eliminar falsos positius

La primera observació que es té a simple vista quan mirem a la Figura 3 és que els cinc escàners utilitzats identifiquen

un nombre similar de vulnerabilitats amb la diferència de què, Nikto, Arachni i W3af han identificat moltes més falles de caràcter informatiu. També, tant SkipFish com Arachni, han aconseguit identificar, de moment, diverses vulnerabilitats de risc alt. A més, he d'afegir que SkipFish ha identificat les mateixes vulnerabilitats en diferents recursos de la web.

Per exemple, l'única vulnerabilitat que ha aconseguit trobar SkipFish de risc baix, ha estat identificada 14 vegades en diferents recursos de l'aplicació web. Encara que en la Figura 3 no s'ha contemplat el nombre d'identificacions de la mateixa vulnerabilitat segons el recurs, s'ha d'esmentar que SkipFish en relació amb els altres escàners, ha sigut el que més vulnerabilitats ha identificat en cada recurs escanejat. Aquest resultat, en principi, és molt bo, ja que indica una bona eficiència per part de l'escàner, sempre que, un cop s'eliminin els falsos positius i duplicitats, els recursos identificats amb la mateixa vulnerabilitat siguin realment vulnerables.

Continuant amb la Figura 3, els escàners que han aconseguit trobar menys vulnerabilitats són l'escàner SkipFish amb un total de 18 identificacions, de les quals 2 són de risc alt i ZAP amb un total de 21. W3af i Nikto, tenen un nombre de vulnerabilitats identificades molt similar amb 27 i 29 respectivament i, finalment, Arachni, amb un total de 35 identificacions. Cal remarcar que tant SkipFish com Arachni han aconseguit identificar vulnerabilitats de risc alt, això pot demostrar la seva eficiència enfront dels altres 3 escàners de vulnerabilitats.

La següent Figura 4 mostra el nombre de vulnerabilitats identificades per a cada escàner executat en l'aplicació web avaluada i el risc que representen cadascuna d'elles segons la Taula 1, però aquest cop eliminant els falsos positius:

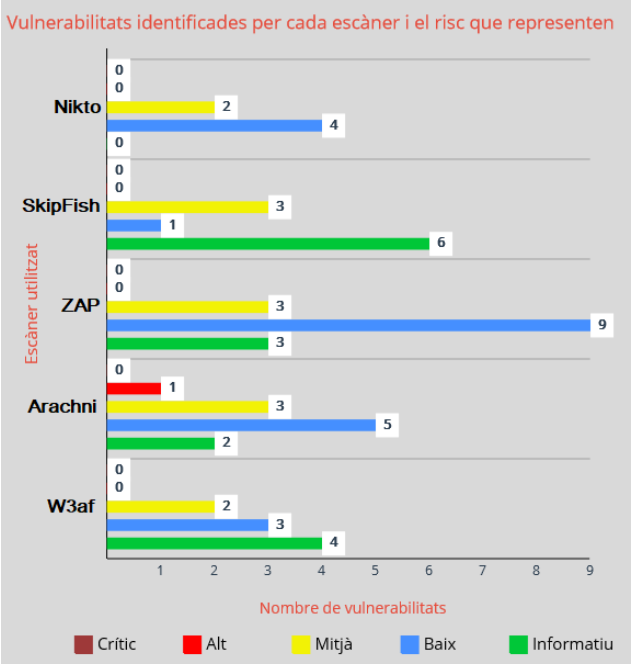

Fig. 4: Vulnerabilitats reportades per cada escàner i el risc que representen, eliminant falsos positius

Tal com es pot observar en la Figura 4, el nombre de vulnerabilitats identificades per cada escàner després d'eliminar els falsos positius i duplicitats s'ha reduït, en alguns casos, més d'un 65% (Nikto, W3af i Arachni). Zap i SkipFish han donat, en comparació als altres, uns bons resultats pel que fa al nombre real de vulnerabilitats identificades, passant de 21 i 18 fallades trobades a 15 i 10 respectivament, tenint un 71% i 55% d'encert.

Per una altra banda, els escàners W3af i Arachni han tingut un percentatge d'encert similar, obtenint un 33% en el cas de W3af i un 31% en el cas d'Arachni. En comparació als altres, aquest percentatge és molt baix, amb una troballa inicial de 27 i 35 vulnerabilitats respectivament, només 9 i 11 han estat correctament identificades. Finalment, l'escàner el menys eficient de tots, inicialment havia identificat un total de 29 falles de les quals només 6 han sigut reals, donant un percentatge del 20% d'encert.

És important destacar, però, que Arachni, encara que ha obtingut un percentatge molt elevat de falsos positius, ha aconseguit identificar correctament una vulnerabilitat de risc alt. Per l'altra banda, SkipFish, el qual té quasi un 20% més d'encert que l'escàner anterior, les dues vulnerabilitats de risc alt que havia identificat inicialment, han sigut falsos positius. Per tant, és cert que Arachni ha tingut un percentatge de falsos positius més elevat, però ha aconseguit trobar una vulnerabilitat més important que, pràcticament, qualsevol altre escàner. Com a conseqüència, s'ha considerat que Arachni, ZAP i SkipFish, en aquest ordre, han sigut els escàners de vulnerabilitats els quals han obtingut uns millors resultats.

Com s'ha esmentat anteriorment, SkipFish va identificar en un primer moment 18 vulnerabilitats, les quals afectaven molts recursos diferents. Després d'eliminar els falsos positius, aquest nombre s'ha reduït considerablement, deixant només 1 o 2 recursos vulnerables dels 14 o més indicats. Aquest ha sigut un dels principals motius pel qual SkipFish ha sigut el tercer i no el primer o segon en quant els escàners amb millors resultats.

En general, que en un primer moment un escàner trobi una quantitat de vulnerabilitats superior respecte els altres (Arachni, Nikto o W3af), no vol dir necessàriament que sigui millor, ja que hi pot haver una gran quantitat de falsos positius i duplicitats. Per tant, el nombre d'identificacions no es relaciona directament amb l'eficiència de l'escàner, ja que és necessari el suport d'un humà per comprovar cada vulnerabilitat identificada per aquests de forma manual. Un cop es verifiquin les vulnerabilitats identificades per un escàner i aquest tingui un nombre molt més elevat d'identificacions, es podrà afirmar que és més eficient.

En aquest punt, s'esperava que les execucions dels escàners identifiquessin moltes més vulnerabilitats de qualsevol mena de risc i que no hi hagués tants falsos positius. Això, probablement és degut al fet que els escàners utilitzats no estan completament actualitzats [30]. Per una altra banda, Juice Shop pràcticament cada setmana és actualitzada amb noves llibreries amb l'última tecnologia, correcció de *bugs*, entre altres coses. Per tant, si els escàners inclouen patrons

de reconeixement de vulnerabilitats antics, és normal que moltes de les vulnerabilitats més noves no estiguin incorporades en les seves llibreries.

Finalment, cal afegir que els escàners han identificat, en alguns casos, les mateixes vulnerabilitats, fent que el resultat final del nombre de vulnerabilitats trobades de manera automatitzada sigui menor.

En la següent Taula 3 es descriuen breument les vulnerabilitats [17] identificades a l'aplicació web avaluada, fent ús de la tècnica de penetració automatitzada.

Nom	Descripció	Mitigació	Risc - CVSS
1. Injecció SQL	L'escàner ha identificat que l'aplicació permet injectar codi SQL en el camp de cerca de la pàgina web avaluada.	Utilitzar instruccions preparades i consultes parametritzades.	Alt - 7,2
2. Formulari de contrasenya no xifrat	S'ha identificat que el formulari de canvi de contrasenya envia les dades en clar al servidor.	Aplicar protocols d'encriptació segurs com ara el TLS o SSL.	Mitjà - 6,1
3. Open Redirection	És possible evitar el sistema d'antiredirecció (whitelist) que té l'aplicació web amb la contaminació del paràmetre HTTP.	Descartar totes les peticions que incloguin un paràmetre repetit.	Mitjà - 6,1
4. Identificació d'arxius i directoris sensibles	S'ha identificat un directori el qual inclou diversos arxius sensibles.	Eliminar de l'arrel web el directori i els arxius.	Mitjà - 5,3
5. Identificador de sessió en reescriptura URL	L'escàner ha identificat diverses peticions les quals revelaven la ID de sessió de l'usuari.	Introduir la ID de sessió com una cookie.	Mitjà - 5,3
6. Cross-Domain Misconfiguration	Cross-Origin Resource Sharing (CORS) és un mecanisme que permet als navegadors web realitzar sol·licituds entre dominis. S'ha identificat les capçaleres de CORS permetent l'accés als recursos de la web des de qualsevol domini extern.	Configurar les capçaleres HTTP de CORS a un conjunt de dominis més restrictiu o eliminar-les.	Mitjà - 5,3
7. Identificació del nom i la versió del servidor	S'ha identificat en una resposta del servidor, el nom i la seva versió utilitzada en l'aplicació web auditada.	Desactivar el camp "servidor"a totes les capçaleres HTTP.	Mitjà - 5,3
8. Informació sensible a l'URL	S'ha identificat una sol·licitud de canvi de contrasenya que conté informació sensible a l'URL.	Evitar enviar informació confidencial a través de l'URL.	Mitjà - 4,7
9. ClickJacking	S'ha identificat que el website no té protecció enfront a atacs de clickjacking.	Afegir la capçalera HTTP X-FRAME-OPTIONS.	Mitjà - 4,3
10. Divulgació d'informació sensible a través d'errors no controlats	S'ha identificat que el website revela informació de la lògica interna de l'aplicació i tipus de Base de dades que utilitza a través d'errors no controlats.	Enviar missatges d'errors generals, en comptes de revelar informació concreta sobre l'error.	Mitjà - 4,0
11. Fuites d'informació mitjançant la capçalera de HTTP "X-Powered-By"	S'ha identificat que el servidor de l'aplicació està filtrant informació a través de la capçalera de resposta HTTP "X-Powered-By".	Configurar el servidor de l'aplicació per suprimir aquest tipus de capçaleres, evitant així la fuita d'informació.	Baix - 3,7
12. Protecció XSS del navegador web no habilitada	S'ha identificat diverses respostes del servidor, en les quals la protecció XSS està desactivada permetent als agents maliciosos realitzar atacs de XSS.	Habilitar la capçalera XSS del servidor web, configurant la resposta del servidor a "1".	Baix - 3,7
13. Absència de tokens anti-CSRF	No s'han identificat tokens "Anti-CSRF", evitant que els atacants emetin sol·licituds fent-se passar per una víctima.	Utilitzar tokens "Anti-CSRF" com per exemple els OWASP CSRFGuard.	Baix - 3,7
14. Divulgació d'una IP privada	S'ha identificat una IP privada a diverses respostes enviades pel servidor de la web.	Eliminar l'adreça IP privada del cos de resposta HTTP.	Baix - 3,7
15. Absència de la capçalera X-Content-Type-Options	L'absència de la capçalera X-Content-Type-Options permet que el navegador interpreti i visualitzi informació no declarada pel website.	Establir aquesta capçalera de forma adequada per a totes les respostes del servidor.	Informatiu - 0,0
16. Inclusió de fitxers font de JavaScript de diversos dominis	S'ha detectat que l'aplicació web inclou diversos fitxers de script amb un domini de tercers.	Assegurar que els fitxers JavaScript provenen de fonts de confiança.	Informatiu - 0,0
17. Comentaris al codi	S'ha identificat diverses respostes del servidor amb comentaris, els quals podrien ajudar un atacant a entendre la lògica de l'aplicació.	Eliminar tots els comentaris que retornin informació.	Informatiu - 0,0
18. Identificació dels limits de la API en les capceleres	En diverses respostes del servidor, s'han identificat capçaleres que indiquen els límits de peticions que l'API del website pot suportar.	Eliminar les capçaleres que puguin donar aquesta informació a un atacant.	Informatiu - 0,0
19. Password field with autocomplete enabled	S'ha detectat habilitada la funció de recordar les credencials d'usuari. Aquesta pràctica no és recomanable, ja que un atacant podria capturar les credencials emmagatzemades i fer un mal ús d'aquestes.	Per evitar que els navegadors emmagatzemin aquestes credencials en els formularis HTML, desactivar el tag "autocomplete".	Informatiu - 0,0
20. Identificació d'una adreça de correu electrònic	S'ha identificat una adreça de correu electrònic en la web. Aquesta pràctica no és recomanable, ja que un atacant pot utilitzar-la per realitzar atacs de phishing.	Substituir les adreces de correu electrònic per un formulari de contacte.	Informatiu - 0,0

Taula 3: Breu descripció, possible mitigació i risc de les vulnerabilitats identificades en el *pentest* automatitzat

5 RESULTATS OBTINGUTS

En aquesta secció es mostren els resultats obtinguts durant el desenvolupament del projecte, és a dir, amb l'execució dels *pentest* manual i automatitzat.

Durant la revisió i anàlisi de l'aplicació web s'han detectat certes debilitats en controls relacionats amb la seguretat dels sistemes d'informació, la qual cosa podria afectar la confidencialitat, la integritat i la disponibilitat de la informació i els serveis que s'ofereixen en l'aplicació web.

Un cop descrites cadascuna de les vulnerabilitats identificades, conseqüències i mitigacions, tant en el test de penetració manual com a l'automatitzat, a continuació, es mostra un gràfic que recull els resultats obtinguts, indicant el percentatge de vulnerabilitats identificades en el *pentest* manual segons el risc representat en la Taula 1.

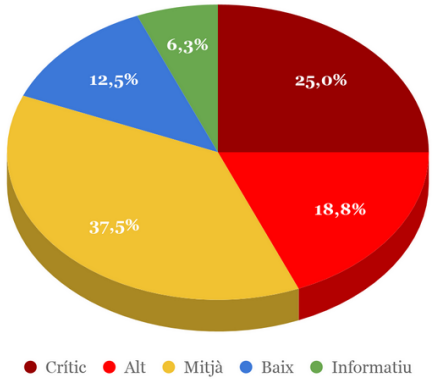

Fig. 5: Resum de les vulnerabilitats identificades en el *pentest* manual segons el risc representat en la Taula 1

La revisió de seguretat manual ha finalitzat amb la detecció de 16 vulnerabilitats. Com es mostra en la Figura 5, s'han detectat 4 vulnerabilitats de risc crític (25,0%), 3 vulnerabilitats de risc alt (18,8%) i 6 de risc mitjà (37,5% de l'import total). També s'han detectat 2 vulnerabilitats de risc baix (12,5%) i 1 d'informativa (6,3%).

El gràfic següent, per l'altra banda, mostra els resultats obtinguts en la identificació de vulnerabilitats en el *pentest* automatitzat segons el risc representat en la Taula 1

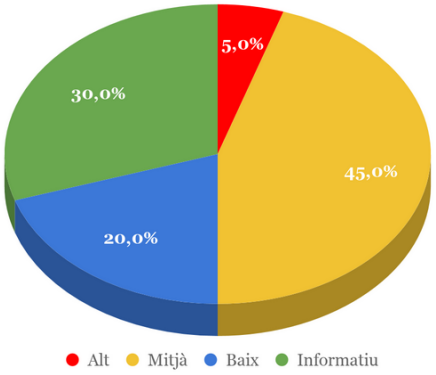

Fig. 6: Resum de les vulnerabilitats identificades en el *pentest* automatitzat segons el risc representat en la Taula 1

La revisió de seguretat automatitzada ha finalitzat amb la detecció de 20 vulnerabilitats. Com es mostra en la Figura 6, s'ha detectat 1 vulnerabilitat de risc alt (5%) i 9 vulnerabilitats de risc mitjà (45%). També s'han detectat 4 vulnerabilitats de risc baix (20%) i 6 d'informatives (30%).

Les vulnerabilitats identificades de risc crític, alt o mitjà amb les dues tècniques podrien permetre a un atacant obtenir accés a informació interna sensible i comprometre la integritat, confidencialitat i disponibilitat del sistema. A més, una combinació exitosa de dues o més vulnerabilitats que generin aquest nivell de risc, pot ser completament destructiva pel correcte funcionament del negoci.

Per una altra banda, les vulnerabilitats de risc baix o de caràcter informatiu proporcionen informació rellevant que un intrús podria utilitzar per realitzar futurs intents de compromís, denegació de servei o simplement revelen informació que no es pot utilitzar directament per realitzar un atac en aquell moment. Normalment, aquestes vulnerabilitats de risc menor, acaben creant un rastre cap a una altra de més greu, per aquest motiu no s'han d'infravalorar.

L'establiment del risc de les vulnerabilitats de les Figures 5 i 6 proporcionen una manera ràpida d'avaluar-les i determinar la seva gravetat potencial per a qualsevol organització. La forma d'agrupar les vulnerabilitats identificades és molt útil perquè els directius poden adaptar el model de resposta que més els hi convingui per mitigar-les, juntament amb altres bones pràctiques del sector i usos efectius de la tecnologia, per adaptar-se a les necessitats de la seva organització i gestionar aquest repte.

6 COMPARATIVA DELS RESULTATS

En aquesta secció es mostra la comparativa dels resultats obtinguts tenint en compte quatre factors esmentats prèviament en la introducció: Abast, Temps, Cost i Risc de les vulnerabilitats.

En l'abast es pretén comparar quin nivell de *coverage* s'ha obtingut amb el test de penetració manual i automatitzat, és a dir, quina de les dues tècniques ha pogut abastar més en la identificació de falles de l'aplicació web.

Pel que fa al temps, s'ha tingut en compte el desenvolupament sencer del *pentest* manual i automatitzat, des de la instal·lació i configuració de les eines i escàners, fins a tot el procés d'anàlisi i obtenció dels resultats indicat en la planificació del projecte.

Per a definir un cost per a les diferents tècniques, s'ha realitzat una cerca general dels preus actuals del mercat [31], els quals defineixen un interval de $4.000 fins a $100.000 per cada test de penetració. En general, el cost pot estar definit per les hores que s'invertiran en realitzar el test de penetració o per aplicació web (IP) analitzada. En aquest cas, el cost s'ha calculat segons el temps dedicat a l'hora d'executar els tests de penetració, tant automatitzat com manual, tenint en compte que només una persona l'ha dut a terme (quantes més persones, normalment, el preu s'incrementa exponencialment).

Finalment, s'ha realitzat una comparativa que mostra el nombre de vulnerabilitats identificades amb les dues tècniques i compara el risc de cadascuna.

	Pentest Manual	*Pentest* Automatitzat
Abast	En profunditat	En amplada
Temps	≈ 115 hores	≈ 110 hores
Cost	≈ 4025€	≈ 3850€
Nombre de Vuln.	16	20
Efectivitat	Excel·lent	Bona

Taula 4: Comparativa de les dues tècniques tenint en compte els 3 factors principals prèviament esmentats

Amb les proves de penetració manuals és pràcticament impossible abastar tota l'aplicació web, des de la A fins a la Z. Això és causa de raons òbvies com el temps i les habilitats de l'analista, mentre que en les automatitzades poden fer-ho amb l'ajut d'intervenció humana d'una manera més senzilla. L'avantatge de les proves manuals és la profunditat en la qual l'analista arriba amb la vulnerabilitat identificada, és a dir, mentre en la prova de penetració automatitzada s'analitza tota l'aplicació web, fent ús de la tècnica manual pots fer ús de diverses vulnerabilitats per trobar-ne d'altres generant un nivell de risc molt més elevat.

Les eines automatitzades funcionen d'una manera més ràpida. Durant un test de penetració, un enfocament manual complet sempre requerirà més temps del que es requereix en un automatitzat. Aquesta afirmació es compleix sempre que els escàners siguin els idonis per realitzar el test automatitzat en l'aplicació en concret, estiguin ja instal·lats i ben configurats. En el cas de l'execució dels diferents escàners de vulnerabilitats executats en l'aplicació Juice Shop, les 110 hores corresponen des de la instal·lació dels primers escàners, l'execució d'aquests i el posterior anàlisi dels resultats, fins a la instal·lació dels nous escàners, reconfiguració, execució i anàlisi final dels resultats. A l'haver tingut tants problemes inicialment amb aquesta part del projecte, les hores són pràcticament les mateixes que les dedicades en el test de penetració manual. Si es tornessin a executar les proves automatitzades, en tenir-ho tot enllestit, el temps es reduiria considerablement.

El cost, tal com s'ha esmentat, està directament relacionat amb el nombre d'analistes i les hores dedicades a realitzar les proves de seguretat. Els preus resultants són fruit de multiplicar el total d'hores per aproximadament 35 €/hora, que és una estimació de la retribució que cobra una empresa per realitzar una auditoria.

Finalment, l'efectivitat resultant és causada pel nombre de vulnerabilitats identificades i el grau de risc que suposen aquestes. En aquest cas, encara que les proves de

penetració automatitzades han trobat més vulnerabilitats, amb l'execució del *pentest* manual s'han pogut identificar un conjunt de vulnerabilitats que suposen un major risc pel correcte desenvolupament del negoci.

En la següent figura es mostra la comparativa final del nombre de vulnerabilitats trobades per les dues tècniques, segons el risc que implica cadascuna:

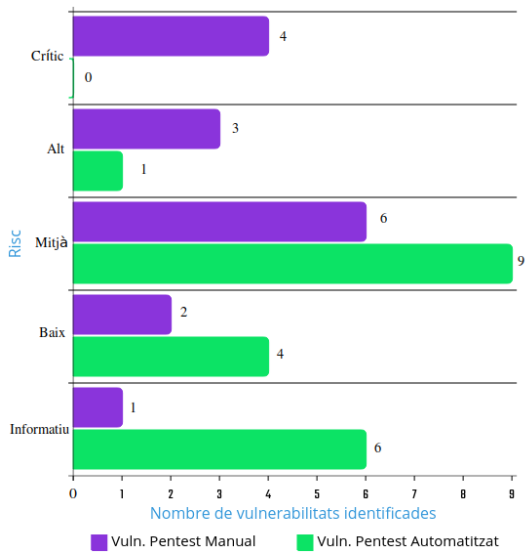

Fig. 7: Comparativa del nombre de vulnerabilitats identificades segons el risc, fent ús de les dues tècniques

Si s'observa la Figura 7, podem observar que el *pentest* automatitzat, en identificar un total de 20 vulnerabilitats, 4 més que el test de penetració manual, s'han obtingut uns millors resultats. Per una altra banda, en el *pentest* manual s'han identificat 4 vulnerabilitats crítiques i 3 de risc alt i en l'automatitzat, només 1 de risc alt. Aquestes vulnerabilitats poden comprometre d'una manera més severa la informació de l'organització i els seus clients.

En aquest cas, com els escàners utilitzats no tenen l'abast suficient per a les aplicacions web que inclouen una gran quantitat de directoris i nivells de profunditat, el resultat obtingut del *pentest* automatitzat no ha sigut tan profitós com s'esperava, identificant un conjunt de fallades molt inferior a l'esperat. La majoria de vulnerabilitats trobades per aquests eren falsos positius, havent de descartar pràcticament el 65% en alguns casos, a més de la quantitat de temps que ha requerit configurar cadascun dels escàners. Tot i això, no vol dir que una exploració de seguretat automatitzada no sigui útil, de fet, un primer anàlisi automatitzat hagués facilitat molt la fenia a l'hora de l'execució manual.

En general, però, les vulnerabilitats més complexes estan relacionades amb la lògica d'aplicació o el disseny de funcionalitats de seguretat, requerint una intervenció manual. Estudis de McAffe o BlackHat [32] han demostrat que, en el millor dels casos, les eines automatitzades només poden trobar el 45% de les vulnerabilitats. Per tant, el 55% restant requereix d'una intervenció manual.

Per tant, considero que amb l'objectiu d'obtenir els millors resultats, identificant el màxim nombre possible d'errades al sistema, estalviant temps i obtenint el millor rendiment i seguretat al sistema objectiu, s'ha de fer ús de les dues tècniques, fent que aquestes treballin en harmonia.

Finalment, però, com a conclusió de la comparativa tenint en compte tots els factors a valorar, puc afirmar que en els resultats obtinguts per l'execució de la prova de penetració manual s'ha obtingut una millor eficiència i rendiment. Encara que al *pentest* manual se li ha dedicat més temps, el cost és superior i el nombre de vulnerabilitats identificades és inferior, el nivell de risc de cada vulnerabilitat ha inclinat la balança dràsticament a favor seu.

7 Conclusions

Com a conclusió d'aquest projecte, podem afirmar que s'ha aconseguit efectuar una auditoria de seguretat en una aplicació web amb l'execució d'una prova de test manual i automatitzada. D'aquesta manera, s'ha pogut realitzar una comparativa tenint en compte diferents factors clau que defineixen quina de les dues tècniques ha donat millors resultats i, per tant, és millor en la identificació de vulnerabilitats.

Encara que, tal com s'ha comentat durant la comparativa, de vegades no és possible cobrir totes les incursions del sistema objectiu, ni realitzar auditories de seguretat web manualment, ja que hi pot haver una gran quantitat de càrregues útils. En aquests casos, podem fer servir una eina automatitzada per estalviar-nos molt esforç i temps, ja que en executar-la obtindrem informació general que ens serà de molta utilitat, abans d'iniciar la fase de descobriment de vulnerabilitats de forma manual. Per tant, la millor manera d'obtenir el màxim d'identificacions de vulnerabilitats en una aplicació web és fer ús de les dues tècniques, l'automatitzada per obtenir informació de manera ràpida i la manual per examinar i identificar categories de defectes específics i de l'aplicació dins del domini objectiu.

Com a possibles millores per a un futur es podria afegir una auditoria de seguretat a una aplicació mòbil i a un dispositiu IoT per tenir una comparativa més extensa i comprovar si els resultats obtinguts són els mateixos o similars.

Finalment, com a conclusió personal, considero que amb l'elaboració d'aquest projecte he obtingut nous coneixements i habilitats en l'execució i desenvolupament d'auditories de seguretat web, les quals em seran d'utilitat en l'actualitat i en un futur, en treballar per una empresa de seguretat informàtica on es realitzen aquest tipus d'auditories.

Agraïments

En primer lloc, agreixo al meu tutor de projecte Oscar Martin, per haver-me guiat i ajudat des de bon principi, aportant noves idees i diferents punts de vista, per poder així entregar un treball de qualitat. En segon lloc, m'agradaria agrair a la meva família, en especial a la meva parella Michelle Domínguez, per ajudar-me en tots aquells moments difícils que la realització d'aquest treball m'ha portat. Finalment, agraeixo als meus companys de feina i d'universitat, per haver-me aconsellat i animat durant tot aquest procés.

REFERÈNCIES

El següent enllaç porta a un document pujat a Dropbox amb les evidències del *pentest* manual explicades detalladament, així com els resultats obtinguts en les execucions dels escàners de vulnerabilitats: https://bit.ly/2Z3LWtN.

[1] Kaspersky. what is cyber security? https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security.

[2] Tthaicert. threat group. Disponible a: https://www.thaicert.or.th/downloads/files/A_Threat_Actor_Encyclopedia.pdf.

[3] David Ruiz. Malwarebytes Labs. Why all organizations must better protect sensitive data. Disponible a: https://blog.malwarebytes.com/business-2/2019/10/why-/.

[4] NGI. CIBERSEGURIDAD. Disponible a: https://www.ngi.es/ciberseguridad/.

[5] Tutorialspoint. Penetration Testing Manual&Automated. Disponible a: https://www.tutorialspoint.com/penetration_testing/penetration_testing_manual_automated.htm.

[6] Burp Suite. Application Security Testing SW. Disponible a: https://portswigger.net/burp.

[7] Bkimminich. Github. Juice-Shop. Disponible a: https://github.com/bkimminich/juice-shop.

[8] OWASP. Juice Shop. Disponible a: https://owasp.org/www-project-juice-shop.

[9] First.org. CVSS v3.1 Specification Document. Disponible a: https://www.first.org/cvss/v3.1/specification-document.

[10] Alpinesecurity. The History of Penetration Testing. Disponible a: https://alpinesecurity.com/blog/history-of-penetration-testing.

[11] Cprime. What is AGILE? - What is SCRUM? Disponible a: https://www.cprime.com/resources/what-is-agile-what-is-scrum.

[12] Atlassian. Severity Levels for Security Issues. Disponible a: https://www.atlassian.com/trust/security/security-severity-levels.

[13] Bjorn Kimminich. Owasp. architecture overview - juice shop. Disponible a: https://pwning.owasp-juice.shop/introduction/architecture.html.

[14] Martin Heller. InfoWorld. REST and CRUD: the Impedance Mismatch. Disponible a: https://www.infoworld.com/article/2640739/rest-and-crud--the-impedance-mismatch.html.

[15] Peter Bollen. core.ac.uk. BPMN: A Meta Model for the Happy Path. Disponible a: https://core.ac.uk/download/pdf/6767881.pdf.

[16] OWASP. WSTG - v4.1. Disponible a: https://owasp.org/www-project-web-security-testing-guide/.

[17] OWASP. OWASP Top Ten Web Application Security Risks. Disponible a: https://owasp.org/www-project-top-ten.

[18] Redhat. ¿Qué es el open source? Disponible a: https://www.redhat.com/es/topics/open-source/what-is-open-source.

[19] Matt Asay. Infoworld. Why open source has never been stronger. Disponible a: https://www.infoworld.com/article/3410758/.

[20] w3af. Disponible a: http://w3af.org.

[21] Arachni. Disponible a: https://www.arachni-scanner.com.

[22] OWASP ZAP. Disponible a: https://owasp.org/www-project-zap.

[23] Skipfish. Disponible a: https://tools.kali.org/web-applications/skipfish.

[24] CIRT. Nikto2. Disponible a: https://cirt.net/Nikto2.

[25] GitHub. Scanning Juice Shop. Disponible a: https://github.com/Arachni/arachni/issues/801.

[26] Robert Abela. Netsparker. Should you pay for a Web Application Security Scanner? Disponible a: https://www.netsparker.com/blog/web-security/comparison-commercial-non-commercial-web-application-security-scanner.

[27] Stackexchange. Vulnerability scanning of AngularJS. Disponible a: https://security.stackexchange.com/questions/108816/vulnerability.

[28] secureCodeBox. Disponible a: https://www.securecodebox.io.

[29] Whitehatsec. False-Positive. Disponible a: https://www.whitehatsec.com/glossary/content/false-positive.

[30] Arachni. Arachni is no longer maintained. Disponible a: https://www.arachni-scanner.com/blog/arachni-is-no-longer-maintained.

[31] Hacken. How much does Penetration Test Cost. Disponible a: https://hacken.io/research/education/how-much.

[32] Untrustednetwork. Automated tools detect only 45%. Disponible a: https://www.untrustednetwork.net/en/2019/10/19/do-automated-tools-really-detect-only-45-of-all-vulnerabilities.