

UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA MATEMATICĂ-INFORMATICĂ

LUCRARE DE LICENȚĂ

Web Application Vulnerability Scanner

COORDONATOR ȘTIINȚIFIC: ABSOLVENT:

Conf. Dr. RUXANDRA OLIMID NĂIBOIU TEODOR

BUCUREȘTI, IUNIE 2021

Cuprins

[Introducere 5](#_Toc73638267)

[Capitolul 1. Tehnologii și Instrumente 7](#_Toc73638268)

[1.1 Python 7](#_Toc73638269)

[1.2 Requests 7](#_Toc73638270)

[1.3 Urllib 8](#_Toc73638271)

[1.3.1 Urllib.parse 8](#_Toc73638272)

[1.3.2 Urllib.request 8](#_Toc73638273)

[1.4 BeautifulSoup4 8](#_Toc73638274)

[1.4.1 BeautifulSoup 9](#_Toc73638275)

[1.5 SSL 9](#_Toc73638276)

[1.6 Socket 9](#_Toc73638277)

[1.7 Subprocess 10](#_Toc73638278)

[1.8 Threading 10](#_Toc73638279)

[1.9 Tkinter 10](#_Toc73638280)

[1.10 OS 11](#_Toc73638281)

[1.11 Re 11](#_Toc73638282)

[1.12 ConfigParser 11](#_Toc73638283)

[1.13 Metasploitable 2 12](#_Toc73638284)

[1.13.1 DVWA 12](#_Toc73638285)

[1.14 Alte tehnologii utilizate 13](#_Toc73638286)

[Capitolul 2. Vulnerabilitățile aplicațiilor web 14](#_Toc73638287)

[2.1 Injections 14](#_Toc73638289)

[2.1.1 SQL Injections 15](#_Toc73638290)

[2.1.2 LDAP Injections 15](#_Toc73638291)

[2.1.3 OS Command Injection 15](#_Toc73638292)

[2.2 Broken Authentication 16](#_Toc73638293)

[2.2.1 Credential stuffing 16](#_Toc73638294)

[2.2.2 Session attacks 17](#_Toc73638295)

[2.3 Sensitive Data Exposure 18](#_Toc73638296)

[2.4 XML External Entities (XXE) 19](#_Toc73638297)

[2.5 Broken Access Control 20](#_Toc73638298)

[2.6 Security Misconfiguration 20](#_Toc73638299)

[2.7 Cross-Site Scripting (XSS) 21](#_Toc73638300)

[Bibliografie 23](#_Toc73638301)

**Abstract**

Proiectul care urmează să fie prezentat în această lucrare este o aplicație software de tipul *scanner* destinată dezvoltatorilor de siteuri web și persoanelor care se ocupă cu testarea și asigurarea securității și integrității acestora. *Scannerul* are ca scop indetificarea și raportarea breșelor de securitate găsite în procesul de testare automată numită și *scanare*. Utilizatorii aplicației pot interacționa diferit cu aceasta în funcție de sistemul de operare pe care aceștia aleg să execute scanarea. Pe sistemele Windows aplicația dispune de un *graphical user interface* (GUI) cu care utilizatorul poate interacționa, toate informațiile despre folosirea *scannerului* fiind puse la dispoziție de către interfață. Pe sistemele Linux aplicația trebuie folosită prin intermediul unui terminal, toată interacțiunea fiind realizată exclusiv prin linia de comandă. Odată ce utilizatorul introduce datele aplicației pe care acesta dorește să o supună testării, tot ce rămâne de făcut este să pornească scanarea și să aștepte raportul generat. La final aplicația printează informații cu privire la raport și ghidează utilizatorul către fișierele generate pentru vizualizarea rezultatelor detaliate.

Din cauza diversității metodelor de implementare ale unei aplicații web, acestea sunt predispuse la mici erori de proiectare. Aceste erori cauzează, în majoritatea cazurilor, apariția vulnerabilităților în sistemele care manageriază aceste aplicații. Este nevoie de astfel de unelte care să ajute la depistarea și rezolvarea acestor erori într-un interval cât mai scurt de timp.

# Introducere

Securitatea este, simplu spus, libertatea fata de risc sau pericol [1]. Fiind una dintre nevoile fundamentale umane, oamenii și-au dedicat o lungă perioadă din viață în căutarea a noi metode inovative și eficiente pentru a combate pericolele născute fie din natură, cum ar fi eventimentele meteorologice, animalele, virusurile și bacteriile, fie născocite chiar de aproapele lui. Astfel, cu timpul, am fost nevoiți să ne adăpostim în locuințe confecționate în asa fel încât acestea să asigure protecție față de orice pericol provenit din exteriorul acestora. Pe măsură ce evoluăm ca specie, această nevoie de protecție crește proporțional cu riscurile cauzate de avansul, atât tehnologic cât și evoluționar al omului.

Foarte asemănătoare nouă este și evoluția securității informației. Am observat cum imediat după apariția internetului a fost nevoie de metode nemaiîntâlnite de a ne proteja informațiile și datele personale, păstrate până în acel moment în seifuri sau în locații bine păzite. Apariția internetului ne-a deschis, deci, porțiile către o noua lume mai conectată și cu o mulțime de noi posibilități dar și noi pericole, noi riscuri asociate acestei invenții.

Astăzi securitatea informației, sau securitatea cibernetică, joacă un rol extrem de important în buna funcționare a societății. Ea se definește ca diverse procese și practici ce au rolul de a proteja rețele, *device-uri*, programe și date în fața atacurilor, distrugerilor sau a accesului neautorizat [2].

Recent am fost martorii unui atac cibernetic de tipul *Ransomware* care a avut ca țintă “Colonial Pipeline”, o rețea de distrubuție de combustibil aflată în partea de Est a Statelor Unite. Atacul a cauzat o scădere a aprovizionării stațiilor de alimentare cu combustibil, având astfel un impact ridicat asupra economiei zonei afectate, producând daune de milioane de dolari și instaurând o panică generală în rândul cetățenilor [3]. Observăm astfel impactul dezastruos pe care o breșă de securitate îl poate avea atât pe plan politic și economic cât și psihologic.

Am văzut mai sus un singur tip de atac cibernetic, în realitate există o mulțime de astfel de atacuri de diferite forme și cu diferite intenții. Unele dintre ele sunt cauzate de conflicte politice, economice, iar altele din simplul motiv de a demonstra abilitățiile atacatorilor. O bună parte dintre aceste atacuri țintesc aplicațiile Web, cunoscute și sub numele de *siteuri web.* O aplicație web este un program care utilizează un *browser web* și metode *web* pentru a îndeplinii diverse atribuții pe internet [4]. Din cauza frecventelor interacțiuni umane cu aceste aplicații, riscul ca ele să devină ținta atacurilor cibernetice este unul foarte ridicat, astfel specialiștii în securitate cibernetică au investit timp și efort sporit în scopul securizării lor.

Securizarea aplicațiilor *web* este, pe zi ce trece, din ce în ce mai importantă. În fiecare zi zeci de milioane de persoane își încredințează atăt datele persoanale căt și confidențialitatea acestor aplicații, sarcina de a asigura această confidențialitate fiind pe umerii dezvoltatoriilor acestor *aplicații.* Totuși pe zi ce trece se descoperă noi vulnerabilități și noi atacuri din care un număr ridicat de oameni are de suferit. Așadar, dezvoltatorii și analiștii în securitate nu pot face față acestor atacuri fiind nevoie de un efort proactiv în securizarea aplicațiilor. Chiar și așa, din cauza numărului ridicat de metode intruzive, un dezvoltator este predispus să piardă din atenție un detaliu care se poate dovedi un punct critic în cazul realizării unui astfel de atac. Deci este nevoie de unelte din ce în ce mai performante și mai eficiente prin care astfel de catastrofe să fie evitate.

Acum că am definit conceptul de *aplicație web* este timpul să ne familiarizăm cu metodele prin care o astfel de aplicație este supusă testelor de siguranță înainte ca aceasta să fie disponibilă publicului. În mod normal, pentru a evita un atac, testerul se asigură că aplicația îndeplinește niște norme de securitate standard definite de OWASP [5].

Așadar în capitolele care urmează am să prezint o aplicație destinată în principal dezvolatorilor dar și testerilor, care are ca scop automatizarea proceselor de testare definite de OWASP și încapsularea lor într-o aplicație software. Interesul pentru această temă este datorat curiozității, cât și pasiunii mele pentru securitatea cibernetica, în special pentru securitatea *aplicațiilor web.* Astfel am decis să creez un program care să usureze procesul de securizare al aplicației și să minimizeze timpul petrecut practicând testele manual și erorile umane apărute pe parcursul acestora.

Consider că realizarea acestei lucrări a fost o provocare din care am reusit să-mi aprofundez cunoștiințele despre domeniul securitații și cred că rezultatul poate fi folosit de oricine dorește să dezvolte sau să testeze o *aplicațe web.*

# Capitolul 1. Tehnologii și Instrumente

Proiectul este realizat utilizând limbajul de programare Python. În cadrul acestui proiect am folosit diverse module. Un modul este un fișier Pyhon cu diverse definiții și instrucțiuni. Printre cele mai importante module folosite se numără modulul *requests,* folosit pentru a comunica cu *aplicația web*, modulul *urllib* folosit pentru manipularea adreselor *web*, modulul *BeautifulSoup4* pentru extragerea informațiilor, și alte module utile despre care urmează să vorbim în subcapitolele următoare.

## Python

Pyhon este un limbaj de programare util pentru aproape orice aplicație. Este un limbaj interpretat, orientat pe obiect cu o colecție vastă de module și o sintaxa atragătoare. Datorită asemănării lui cu limbajul *bash* și capacității sale de a executa o multitudine de operații des întâlnite în securitatea cibernetica, cât și a colecției impresionante de module special create pentru *Penetration Testing*, Pyhon a fost mereu limbajul preferat de inginerii de securitate și testeri.

Poate cel mai important avantaj al limbajului Pyhon în acest domeniu îl reprezintă rapiditatea cu care pot fi rezolvate problemele, implementarea codului și rezolvarea erorilor fiind mult mai rapidă în comparație cu alte limbaje de programare.

## Requests

Requests este un modul Python care permite utilizatorului să interacționeze cu *request-urile* de tip *Hypertext Transfer Protocol* (HTTP) într-un mod simplu și rapid. Un *request* este o acțiune realizată asupra unei resurse aflate la o anumită adresa *web,* cu alte cuvinte, este o cale intermediară care transportă informații între client (sau browser) și server. Tipurile de *request-uri* variază în funcție de scopul acestora. În total există un număr de nouă tipuri de astfel de *request-uri* dar cele mai importante și cele pe care le-am folosit în dezvoltarea acestei lucrări sunt: GET, POST, PUT, DELETE [6].

GET este folosită pentru a obține date de la server.

DELETE șterge o anumită resursă specificată în *request*.

PUT este metoda folosită pentru a modifica reprezentăriile unei stări existente, cu o reprezentare nouă, definită în procesul de *request*.

POST este metoda specifică pentru a trimite date noi către server. Poate fi folosită pentru a actualiza și a crea noi date pe server.

## Urllib

Urllib este un modul care are în componența lui două module importante pe care le-am folosit.

### Urllib.parse

Modulul urllib.parse este folosit pentru manipularea și modificarea *Uniform Resource Locator* (URL). Cu ajutorul acestui modul, putem fragmenta, combina și extrage fragmente din URL-uri [7].

### Urllib.request

Modulul urllib.request definește instrucțiuni de cod utile pentru accesarea și interacțiunea cu metodele HTTP. Modulul ne furnizează un mod de comunicare puțin mai avansat decât modulul *requests* [8].

## BeautifulSoup4

Din componența modulului BeautifulSoup4 am folosit modulul cu acelasi nume, adică BeautifulSoup [9].

### BeautifulSoup

Modulul BeautifulSoup este cel mai folosit modul în Pyhon pentru preluarea informațiilor din paginile *web.* În principal am folosit acest modul drept metodă de extragere a diverselor părți componente din codul sursă al paginii *web.* Este eficient în preluarea datelor de tipul HTTP și *Extensible Markup Language* (XML).

## SSL

Modulul SSL (*Secure Sockets Layer)* este un modul care permite accesul la TLS (*Transport Layer Security).* Pentru ca datele noastre *online* să fie securizate este nevoie de protocoale criptografice care să codeze aceste date în secvențe de litere și cifre aleatoare și indescifrabile pentru om. Protocolul SSL folosește un sistem de criptare cu două chei, o cheie publică și o cheie privată. Pentru decodarea mesajului este nevoie de ambele chei, astfel doar persoanele autorizate pot să aibă acces la datele criptate în mesajul transmis.

Acest modul se foloseste pentru interacțiunea cu certificatul de securitate al aplicației *web*, cunoscut și sub numele de *SSL Certificate.* Un astfel de certificate asigură utilizatorii că transferul datelor dintre browserul lor și aplicația *web* se realizează printr-un canal criptat, astfel datele fiind protejate și confidențiale.

Folosit împreuna cu modulul *socket* acest modul poate obține informații detaliate despre certificatele SSL [10].

## Socket

Modulul socket definește o gama largă de procedee care au ca scop conectarea a două noduri de rețea pentru a favoriza comunicarea între ele.  
Procesul prin care două noduri de rețea realizează o conexiune este usor explicat daca ne imaginăm că un nod așteaptă un răspuns de la un nod care încearcă să comunice cu acesta. Acest proces simulează o conexiune între un client și un server.

Un socket definește un *endpoint* într-o rețea, astfel, socketul crează un canal bidirecțional de comunicare folosindu-se de adresa *Internet Protocol* (IP) și de un port deschis pe aplicație. Un *endpoint* este un *device* conectat la o rețea. [11].

## Subprocess

Un proces este definit ca un program care se află în execuție. Astfel de procese sunt controlate de *Central Processing Unit* (CPU) și reprezintă o mulțime de calcule efectuate de procesor (CPU) într-un interval foarte scurt de timp. Aceste procese se află într-o continuă execuție secvențială, astfel, un program execută mereu un singur proces.

Modulul Subprocess este folosit pentru a crea, a distruge sau a interacționa cu aceste procese. Așadar utilizăm acest modul pentru a interacționa direct cu diverse programe, prin intermediul unei altei aplicații, procesul nou creat fiind integrat în *firul* principal de execuție.

Definim un *fir* de execuție drept un set de astfel de procese care sunt executate pe rand și într-o ordine bine definită și determinată de CPU [12].

## Threading

Un *thread* este o modalitate prin care putem executa mai multe procese în același timp. Putem să asemănăm *thread-urile* cu mai multe programe care rulează concomitent și independent unele față de altele. În aplicațiile în care avem nevoie ca mai multe instrucțiuni să fie executate în același timp și în care există riscul ca execuția unei bucăți de cod să blocheze execuția alteia, creăm astfel de *fire de execuție* care fragmenează instrucțiunile programului și comunică cu CPU pentru a aloca memorie separată pentru *firul de execuție* folosit [13].

## Tkinter

Modulul Tkinter este o suită populară de unelte folosite pentru a crea interfețe grafice.

Tkinter a fost construit cu scopul de a oferi simplitate și diversitate în alcătuirea unei interfațe grafice cât mai atrăgătoare și intuitive.

Structura Tkinter este realizată dintr-o fereastră simplă, dar ușor customizabilă prin metode implicite, astfel adaugarea unui buton, unei liste de conținut sau mesajelor de ajutor poate fi realizată în doar câteva linii de cod. Asupra ferestrei simple, se aplica diverse modificări și funcționlități care, utilizând modului *threading,* rulează în paralel cu execuția din spatele interfeței grafice, oferind astfel o interacțiune placută între utilizator și aplicație.

Pe lângă modului Tkinter, sunt des întâlnite și modulele PyQt5, Kivy, wxPython, PyFrosm, utilitatea lor fiind dată de scopul aplicației [14].

## OS

Prin intermediul modulului OS putem interacționa eficient cu sistemul de operare pe care rulează aplicația. Acest modul este indispensabil în interacțiunea cu fișiere, căi sau pentru diversele modificări pe care putem să le aducem acestora. Avantajul folosirii acestui modul este dat de transparența față de sistemul de operare. Sintaxa modulului comunică eficient cu sistemul de operare indiferent de sistemul pe care acesta rulează. Spre exemplu sintaxa *os.stat(path)* returnează informații despre calea dată ca argument indiferent dacă aceasta este o cale a sistemului *Windows* sau a sistemului *Unix* [15]*.*

## Re

Numele modulului Re este un acronim pentru *Regular expression*. Re este un set de caractere speciale (*patttern-uri)* folosit pentru căuta și a potrivi diverse șiruri de caractere care îndeplinesc regula definită în expresia specială (*regular expression).*

Un exemplu de o astfel de regulă este *“ \d.\* ”* care,aplicată pe șirul *“Ana are 5 mere”,* găsește subșirul “*5 mere”.* Acest modul este util în extragerea informațiilor de orice formă din orice șir de caractere [16].

## ConfigParser

Este o practică standard ca majoritatea aplicațiilor să folosească un fișier de configurare pentru a oferi flexibilitate și libertate utilizatorului. Pentru a exista comunicare între un astfel de fișier și codul aplicației, este nevoie de o metodă prin care limbajul de programare să comunice cu limbajul din fișierul de configurare.

Modulul ConfigParser oferă posibilitatea de a accesa astfel de fișiere de forma *nume\_fișier.ini* și de a traduce datele astfel încât limbajul aplicației să le înțeleagă. În felul acesta datele variabile sunt ușor de configurat de oricine, crescând potențialul aplicației pentru o gamă mai largă de utilizări.

## Metasploitable 2

Metasploitable 2 este o versiune de Ubuntu Linux, alcătuită special din cod nesecurizat, fiind astfel o colecție de servicii vulnerabile din punctul de vedere al securitații. Principalul scop al acestui sistem este de a pune la dispoziție un mediu sigur, închis, care să faciliteze practicile comune de testare, de învățare și de practicare a diverselor metode sau ustensile folosite în procesul învățării conceptelor de securitate cibernetică.

Sistemul conține nenumărate vulnerabilițăți adaugate intenționat sau neintenționat pentru a putea susține aproape orice tip de abordare și pentru a oferi totodată un mediu legal în care profesioniștii dar și persoanele interesate pot să exerseze și să capete mai multe cunoștiințe.

Datorită faptului că aplicația are la baza un sistem Linux, regăsim pe aceasta un server *web,* care susține o mică suită de aplicații web vulnerabile, fiecare cu numele ei reprezentativ. Printre aplicațiile vulnerabile se află aplicația *Damn Vulnerable Web App* (DVWA) care conține o colecție variată de vulnerabilități comune și des întalnite pe *site-urile web* [17].

### DVWA

DVWA este aplicația preferată pentru testarea conceptelor securității *web* datorită numărului ridicat de vulnerabilități existente pe aceasta și interfeței orientative. Aplicația este folosită nu doar de profesioniști, ci și de studenți, profesori, persoane interesate de securizarea aplicațiilor web sau dezvoltatori care caută un loc în care să-și testeze aplicațiile software fără a le face publice sau a încălca legea.

Am ales această aplicație atât datorită numărului ridicat de vulnerabilități cât și tehnologiilor pe care le pune la dispoziție, reușind să mimeze foarte bine aplicațiile *web* des întâlnite zi de zi. Prezența unui număr ridicat de vulnerabilități ofera o arie mai mare de acoperire pentru testele care urmează să fie făcute pe situații reale, atunci când atât tipul cât si numărul de vulnerabilițăti prezente într-o aplicație o să difere.

## Alte tehnologii utilizate

Alte module care merită menționate și care au ajutat la implementarea acestui proiect sunt module auxiliare, cu scop utilitar, care fie eficientizează programul din punctul de vedere al complexității, fie îndeplinesc mici atribuții.

Modulul *queue* care este folositor pentru implementarea unei cozi de diverse evenimente. În cazul acestui proiect, modul *queue* a eficientizat procesul prin care se testa validitatea unui șir de caractere, astfel, mai multe șiruri de caractere citite dintr-un fișier au fost încărcate și reținute în memorie prin intermediul unei astfel de cozi.

Modulul *datetime* ne oferă acces la datele calendaristice și timp. Cu ajutorul acestui modul am manipulat date de forma *ore:minute:secunde* și am reușit să aplic operații de scădere sau adunare a unor perioade de timp asupra datelor de forma specificată anterior.

Modulul *nmap* este o implementare a uneltei folosite pentru analizarea rețelelor care poartă același nume. *Network Mapper* (nmap) este folosit în special pentru maparea arhitecturii unei rețele și pentru detalierea servicilor și *host-urilor* care comunică pe aceasta. Un *host* reprezintă un calculator conectat la o rețea. Am folosit *nmap* pentru analizarea adresei IP a aplicației *web*.

Modulul *ctypes* este un modul care ne oferă tipuri de date compatibile cu limbajul C. Acest modul este folositor atunci când vrem să oprim execuția unui thread.

Modulul *shutil* oferă acces la operații de tip *high-level* pe fișiere. Aceste operații sunt ștergerea și copierea. Acest modul a fost implementat pentru a șterge fișierele temporare, create în timpul execuției.

Modulul *webbrowser* este un modul care interacționează și controlează *browserul* implicit al sistemului. Cu ajutorul acestui modul am implementat metode prin care utilizatorul poate vedea explicații detaliate disponibile pe internet cu privire la vulnerabilitățile descoperite și testate de *scanner*.

# Capitolul 2. Vulnerabilitățile aplicațiilor web

Înainte să vorbesc despre implementarea și arhitectura aplicației este nevoie să fixăm câteva noțiuni importante pe care le-am pus în practică în dezvoltarea acesteia. În acest capitol am să explic ce este o vulnerabilitate web, de câte tipuri sunt ele și cum afectează buna funcționare a unui *site web,* urmând ca în capitolul următor să explic detaliat abordarea pentru implementarea aplicației.

O vulnerabilitate web este o slăbiciune sau o configurare greșită a anumitor funcționalități pe care o aplicație web le îndeplinește. Uneori aceste vulnerabilități sunt abuzate de persoane rău intenționate care reușesc să preia controlul asupra aplicației pentru a obține date confidențiale în diverse scopuri personale. În funcție de modul în atacatorii reușesc să obțina acces la resursele securizate ale aplicației web, se disting diferite tipuri de astfel de vulnerabilități.

OWASP Top Ten este documentul oficial în care sunt discutate tipurile de riscuri la care *aplicațiile* *web* sunt supuse și tacticile de detecție și neutralizare ale acestora. Atât dezvoltatorii cât și *penetration* *testers* urmăresc îndrumările și tehnicile consemnate în document pentru a se asigura că arhitectura *aplicațiilor web* îndeplinește ultimele standarde de securitate. Acest proiect a fost realizat conform abordărilor și metodelor menționate și documentate în lista *Web Security Testing Guide* (WSTG) din cadrul proiectului OWASP [18].

În subcapitolele care urmează am să menționez trăsăturile fiecărui tip de vulnerabilități testate și integrate în acest proiect. Pentru fiecare categorie au fost implementate diferite abordări de testare, acestea urmează să fie detaliate în capitolul urmator.



## Injections

Tipul *Injections* reprezintă vulnerabilitățile care au la bază nesanitizarea *inputurilor* din aplicație. Acestea au loc atunci cand un șir de instrucțiuni provenit din exterior este interpretat ca parte din codul aplicație, astfel instrucțiunea în cauză execută acțiuni nedorite cu potențial dăunator și intrusiv. Putem găsi acest tip de vulnerabilitate în orice cămp care citește date de la utilizator.

Cele mai populare tipuri de *injection* sunt *SQL Injections*, *LDAP Injections, OS Command Injection.*

**

Figura 2.1. Exemplu SQL Injection

### SQL Injections

*Structured Query Language* (SQL) este un limbaj de programare specific bazelor de date. Astfel un *SQL Injection* este o vulnerabilitate care încearcă să manipuleze înregistrările unei baze de date cu scopul de a accesa *query-urile* din componența acesteia. O astfel de practică presupune introducerea instrucțiunilor de tip SQL în *inputurile* aplicației care sunt legate la o astfel de baza de date, și accesarea înregistrărilor, șirul introdus fiind, astfel, interpretat ca parte din limbajul SQL, permițând accesul la înregistrări la care utilizatorul nu este autorizat.

Pentru a preveni această vulnerabilitate soluția este găsirea unei metode care păstrează datele introduse, separat de comenzile și *query-urile* din baza de date.

### LDAP Injections

*Lightweight Directory Access Protocol* (LDAP) este un protocol software care permite accesul la diverse resurse, precum fișiere sau informații despre alti utilizatori, prin rețea. Protocolul poate fi folosit și pentru stocarea diferitelor date pe un sistem *sign-on* (SSO)

*Injection-ul* de tip LDAP funcționează pe același principiu ca și în cazul *SQL Injection*.

### OS Command Injection

Această vulnerabiliate este abuzată prin introducerea de comenzi care pot fi interpretate de sistemul de operare pe *serverul* pe care rulează *aplicația web* prin intermediul interfeței acesteia. Orice *input* din aplicație care nu este sanitizat cum trebuie, poate executa comenzi pe sistemul de operare.

Pentru a preveni această vulnerabilitate este nevoie ca *aplicația web* să ruleze sub niște restricții stricte care să nu permita rularea comenzilor pe *serverul* aplicației.

## Broken Authentication

Acest tip definește vulnerabilitațile aparute în urma configurării greșite a funcțiilor de autentificare și manageriere a unei sesiuni. O sesiune este o perioadă de timp definită de dezvoltatorul aplicației în care datele utilizatorului sunt rețiune pe parcursul navigării pe *aplicația web.* Acestă funcție permite navigarea între paginile unui *site* fără a fi nevoie ca utilizatorul să se autentifice de fiecare dată.

Acest tip este foarte răspândit în lumea *aplicațiilor web* și foarte complex. Este foarte ușor pentru atacatori să se folosească de liste de parole și nume de utlizatori, folosind unelte automate, pentru a ghici datele de autentificare ale altor utilizatori. Este nevoie doar de un singur cont cu privilegii administrative pentru a compromite o intreagă aplicație.

Așa cum am menționat mai sus, există două tipuri importante de astfel de atacuri.

Atacuri care se folosesc de vulnerabilități în implementarea funcțiilor care gestionează sesiunile *session management*, și atacuri care vizeză datele de autentificare, cunoscute sub numele de *credential stuffing*.

### Credential stuffing

Un atac de tipul *credential stuffing* constă în încercarea de a introduce date valide în câmpul de autentificare cu speranța că atacatorul nimerește, din întâmplare, un cont valid. Atacul se realizează cu ajutorul uneltelor automatizate care dispun de liste mari ce conțin date de autentificare obținute din scurgerile de date confidențiale din trecut. Aceste liste, numite *dicționare* conțin și parole comune, de obicei implicite pe baza că dezvoltatorii si administratorii unei aplicații uită să modifice o astfel de parolă.

Această practică este des folosită sub forma unui atac provenit din mai multe zone geografice. Un atacator se află în controlul mai multor calculatoare aflate în diferite zone geografice. Acesta lansează un atac simulan de pe toate aceste calculatoare cu scopul de a păcăli sistemele de siguranță ale unei aplicații. Acest atac este cunoscut sub numele de *Botnet*.

Acest tip de atac poate fi prevenit în multe feluri. Câteva dintre ele presupun ca dezvoltatorul să se asigure că nu a lăsat nici o parola implicită în componența aplicației, că a implementat un sistem care să nu-i permită unui utilizator să încerce să forțeze o autentificare într-un interval scurt de timp și că a implementat o condiție la înregistrarea unui cont ca parola setată să aibă o complexitate ridicată.



Figura 2.2. Exemplu credential stuffing

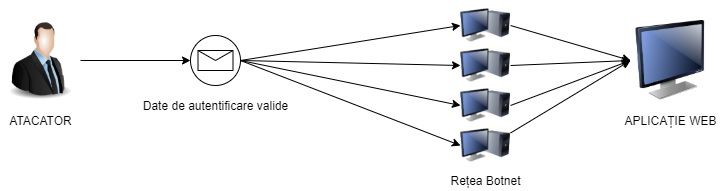


Figura 2.3. Exemplu atac Botnet

Observăm în figurile 2.2 si 2.3 diferența uriașă pe care o are impactul unui

atac simplu, local, fată de un atac folosind o rețea Botnet. Șansele ca atacul să aibă succes sunt exponențial mai mari în cazul figurii 2.3.

### Session attacks

Un alt tip de atac care intră în componența *Broken Authentication* este reprezentat de atacurile care vizeaza impersonarea unui alt utilizator. Acest lucru se realizeză atunci când un atacator fură *ID-ul* sesiunii, prin diverse metode, al unui utilizator autentificat. Acest lucru este realizabil din cauza unei configurari greșite a sistemului care gestionează sesiunile. Este posibil ca *ID-ul* să nu fie transportat sau stocat într-un mod text criptat, astfel un atacator poate să copieze pur și simplu datele din *ID*. Alte practici care trebuie evitate sunt autentificarea fara un *two factor login,* afișarea *ID* în URL, păstrarea aceleași sesiuni pe toată durata procesului de autentificare, aplicația nu invalidează sesiunea atunci când un utilizator se deloghează de la aceasta sau este inactiv o perioadă mai lungă de timp.

Toate aceste obiceiuri duc la o aplicație nesigură și cu risc ridicat de compromitere. Pentru a evita astfel de evenimente, dezvoltatorul trebuie să se asigure că folosește un sistem de gestionare al sesiunii care rulează pe server și care este destul de bine configurat pentru a genera și a stoca *ID-ul* în siguranță.

## Sensitive Data Exposure

Așa cum reiese și din titlu, acest tip de vulnerabilitate reprezintă expunerea, într-o modalitate care să permită citirea, datelor sensibile, confidențiale, publicului. Atacurile practicate implică interceptarea datelor aflate în tranzit, între victimă și *aplicația web.* Datele personale cum ar fi parolele, secretele de serviciu, mesajele confidențiale, adresele, numerele de telefon, datele bancare, au nevoie de cea mai mare protecție. Acest atac presupune inerceptarea datelor, metodă cunoscută sub numele de *man in the middle* în timp ce acestea tranzitează internetul. Dezvoltatorul aplicației trebuie să se asigure că aplicația folosește protocoale criptografice actualizate și că este configurată în asa fel încât datele să fie transmise doar prin protocolul HTTPS. HTTPS este protocolul HTTP la care se adaugă un stratul TLS/SSL cu rol de criptare, astfel toate datele aflate în tranzit între *browser* și server sunt transportate într-un tunel criptat.

Alte vulnerabilități care intră în această categorie sunt date de interacțiunea dezvoltatorului cu *aplicația*. Acesta trebuie să se asigure că pe parcursul dezvoltării nu au fost lăsate, din greșeală mesaje sensibile la vedere, cum ar fi, în comentarii sau undeva în pagina web. Mai mult, aplicația trebuie să conțină un certificat SSL valid emis de un *Certificate Authority* (CA) de încredere și autorizat pentru a asigura utilizatorul că datele sunt criptate de o sursa de încredere.

Un atac frecvent întâlnit în astfel de cazuri este realizat de un atacator care se preface a fi un CA de încredere pentru a emite un certificat fals, care este apoi folosit de o *aplicație web* cu scopul de a păcăli utilizatorii pentru ca aceștia să-și introducă datele personale fără să suspecteze nici o tentativă de fraudă.

O astfel de vulnerabilitate poate fi evitată dacă dezvoltatorul evită să stocheze inutil date confidențiale, clasifică datele în funcție de confidențialiatea lor, respectând termenii de confidențialitate, se asigură că toate datele sensibile sunt criptate și foloseste parole puternice pentru securizarea punctelor de acces administrativ.

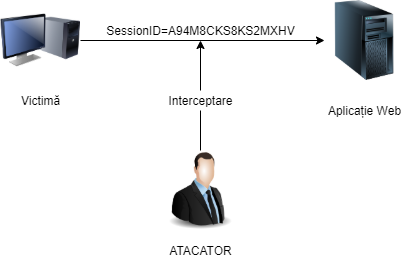


Figura 2.4. Un atacator interceptează o sesiune

## XML External Entities (XXE)

Foarte asemănător cu tipul *Injections*, vulnerabilitățile de tip XEE au la bază exploatarea modului în care este interpretat un *input* XML. Dacă aplicația acceptă *input* XML în mod direct, în special de la surse care nu sunt de încredere sau dacă aplicația foloseste *document type definitions* (DTD) există riscul ca aplicația să fie vulnerabila. Vulnerabilitatea poate fi abuzată pentru a extrage date, executa diverse comenzi pe server sau pentru a-l supraîncărca, atac cunoscut sub numele de *denial-of-service* (DOS).

Fiind asemănătoare cu *Injections*, vulnerabilitatea poate fi evitată dacă *input-urile* sunt sanitizate, tehnologiile de interpretare a XML sunt actualizate și dacă se folosesc structuri de date mai puțin complexe de forma *JavaScript Object Notation* (JSON). JSON este un format de date ușor de interpretat de către aplicații și ușor de citit de către oameni.

## Broken Access Control

Funcția de *Access Control* gestionează privilegiile pe care utilizatorii unei aplicații le au. Cu alte cuvinte, funcția controlează ce date sunt accesibile și cui, la un moment dat. În cazul în care funcția nu este bine implementată, există riscul ca un simplu utilizator să aibă acces la resurse care, în mod normal, ar trebui să fie accesabile doar de un administrator. Atacatorii pot abuza de aceste scăpări și pot prelua controlul aplicației cu un minim efort.

Obținerea unui astfel de acces poate fi realizată în multe feluri și pentru scopuri diferite. Un atacator poate încerca să obțină acces de citire sau modificare a unor date de pe server, poate obține acces administrativ, sau poate să evite verificările de privilegii în cazul în care dorește să acceseze o resursă restricționtă.

Acestă vulnerabiliate poate fi prevenită dacă dezvoltatorul implementeză metodele de control pe server. Acest lucru rezolvă situația în care un atacator poate să modifice diverse instrucțiuni din metode pentru a-și crea avantaje.

## Security Misconfiguration

Configurările greșite sunt unele dintre cele mai răspândite vulnerabilități de pe o *aplicație web.* Acestea pot apărea în orice strat al aplicației, fie că este vorba de rețeaua pe care rulează serverul, fie, de *browserul* de pe care aceasta este accesată. Vulnerabilitatea apare atunci când există prea multe funcționalități inutile pe aplicație, setările implicite nu sunt modificate, când nu se actualizează sistemele de securitate ( Antivirus, Firewall), când există erori care furnizează mesaje sensibile pe care atacatorii le pot abuza pentru a obține informații sensibile sau când se folosesc funcționalități despre care se cunosc deja vulnerabilități.

Această vulnerabilitate crește riscul ca o aplicație să fie predispusă unui atac mai mult decât orice altă vulnerabilitate. Datoriă diversității de locuri în care această vulnerabilitate apare, este dificil pentru dezvoltatori și testeri să combată eficient orice greșeala. Deseori acesta este punctul de start, deci este vital ca toate configurările de securitate să fie implementate corect pentru a descuraja orice tentativă de încurajare la un astfel de atac.

## Cross-Site Scripting (XSS)

Cea mai răspândită formă de vulnerabilitate este *Cross-Site Scripting*. Conform OWASP, două din trei aplicații au prezentat o astfel de vulnerabilitate. Această vulnerabilitate folosește *browserul* victimei pentru a executa cod malicios ca parte din HML sau JavaScript. *Există* trei tipuri de XSS.

### Reflected XSS

Acest atac presupune introducerea de cod extern în browser folosind doar un singur *request* HTTP. Un astfel de atac are loc atunci când un browser interpretează datele introduse de atacator prin *input-uri* nevalidate. Este o practică comuna ca un atacator să trimită cod JavaScript pentru a efectua diferite acțiuni asupra paginii web. Atacatorii pot instala *key loggers*, un timp de software care monitorizează tastele introduse de utilizator, pot modifica *linkuri* sau pot extrage *cookies* din browserul victimei. Acest tip de atac este realizabil o singură dată, deoarece se bazează pe sistemul *request-response*, adică trimite o cerere serverului, iar acesta răspunde, în funcție de tipul *request-ului,* cu datele solicitate.

### Stored XSS

Este același lucru cu tipul precedent, doar că, de data aceasta, aplicația reține codul introdus. Asta cauzează codul să fie executat permanent pe browser, astfel fiind posibil ca atacatorul să poată prelua controlul browserului victimei, să captureze informații sensibile și să scaneze rețeaua pe care aplicația rulează. Această vulnerabilitate este una dintre cele mai periculoase din cauza modului în care afectează utilizatorii. Un utilizator poate fi victima unui astfel de atac pur și simplu vizitând o pagină web în care a fost introdus cod malicios. Acest atac este și mai periculos în zonele des frecventate de administratorii paginii. Datele conturilor acestora pot fi furate prin astfel de metode, atacatorii preluând astfel controlul aplicației.

### DOM XSS

Este foarte asemănător cu Stored XSS doar că este practicat îndeosebi pe aplicațiile de tip single-page.

Toate aceste tipuri de vulnerabilități se ramifică în practici meticuloase pe care urmează să le prezint odată cu detalierea testelor implementate de *aplicația scanner.* Deci în capitolul următor voi prezenta arhitectura aplicației, împreună cu implementarea testelor de detecție pentru cele mai importante și răspândite vulnerabilițăți, și voi detalia modul de utilizare și modul de raportare al *aplicației scanner.*

# Capitolul 3. Aplicația

# Bibliografie

|  |  |
| --- | --- |
| [1] | T. B. Azad, Introduction to Security, 2008. |
| [2] | J. D. GROOT, What is Cyber Security? Definition, Best Practices & More, Data Insider, 2020. |
| [3] | G. S. M. E. Zachary Cohen, What we know about the pipeline ransomware attack: How it happened, who is responsible and more, CNN, 2021. |
| [4] | R. Gibb, What is a Web Application?, 2016. |
| [5] | OWASP, „Web Application Security Testing,” [Interactiv]. Available: https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/. [Accesat 01 Iunie 2021]. |
| [6] | „Requests: HTTP for Humans,” [Interactiv]. Available: https://docs.python-requests.org/en/master/. [Accesat 02 Iunie 2021]. |
| [7] | „urllib.parse — Parse URLs into components,” [Interactiv]. Available: https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse. [Accesat 02 Iunie 2021]. |
| [8] | „urllib.request — Extensible library for opening URLs,” [Interactiv]. Available: https://docs.python.org/3/library/urllib.request.html#module-urllib.request. [Accesat 02 Iunie 2021]. |
| [9] | [Interactiv]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accesat 02 Iunie 2021]. |
| [10] | [Interactiv]. Available: https://docs.python.org/3/library/ssl.html. [Accesat 02 Iunie 2021]. |
| [11] | [Interactiv]. Available: https://docs.python.org/3/library/socket.html. [Accesat 02 Iunie 2021]. |
| [12] | [Interactiv]. Available: https://docs.python.org/3/library/subprocess.html. [Accesat 02 Iunie 2021]. |
| [13] | Randy. [Interactiv]. Available: https://whatsabyte.com/blog/processor-threads/. [Accesat 02 Iunie 2021]. |
| [14] | [Interactiv]. Available: https://docs.python.org/3/library/tkinter.html. [Accesat 02 Iunie 2021]. |
| [15] | [Interactiv]. Available: https://docs.python.org/3/library/os.html. [Accesat 02 Iunie 2021]. |
| [16] | [Interactiv]. Available: https://docs.python.org/3/library/re.html. [Accesat 02 Iunie 2021]. |
| [17] | „Metasploitable 2,” [Interactiv]. Available: https://docs.rapid7.com/metasploit/metasploitable-2-exploitability-guide/. [Accesat 02 Iunie 2021]. |
| [18] | [Interactiv]. Available: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\_Application\_Security\_Testing/. [Accesat 02 Iunie 2021]. |