



Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea

Lucrare de licență

Absolvent

Alexandru Daniel Vid

Conducător

Ș.l. Dr. Ing. Ciprian Pavel Oprea

Februarie 2019



TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DEPARTAMENTUL CALCULATOARE

DECANUL FACULTĂȚII
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT
Prof. dr. ing. Rodica POTOLEA

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea

Lucrare de licență

1. Absolvent: Alexandru Daniel Vid
2. Conducător: Ș.l. Dr. Ing. Ciprian Pavel Opreșă
3. Conținutul lucrării: Pagina de prezentare, cuprins, listă de figuri, introducere, obiective și specificații, studiu bibliografic, fundamente teoretice, analiză și proiectare, detalii de implementare, teste și rezultate experimentale, manual utilizator, concluzii, bibliografie, anexe, CD.
4. Locul documentării: UTCN, Cluj-Napoca
5. Consultanți: Ș.l. Dr. Ing. Ciprian Pavel Opreșă
6. Data emiterii temei: 05.05.2018
7. Data predării: 18.02.2019

Semnătură Conducător
Ș.l. Dr. Ing. Ciprian Pavel Opreșă

Semnătură Absolvent
Alexandru Daniel Vid

Februarie 2019



Declarație pe proprie răspundere privind autenticitatea lucrării de licență

Subsemnatul *Alexandru Daniel Vid*, legitimat cu *CI* seria *XH* numărul *866549*, *CNP 1950417055056*, autorul lucrării *Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea* elaborată în vederea susținerii examenului de finalizare a studiilor de masterat la Facultatea de Automatică și Calculatoare, Departamentul Calculatoare, Specializarea *Calculatoare* din cadrul Universității Tehnice din Cluj-Napoca, sesiunea *Februarie* a anului universitar *2018/2019*, declar pe proprie răspundere, că această lucrare este rezultatul propriei mele activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate în textul lucrării și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență sau disertație.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Cluj-Napoca
12.02.2019

Semnătură
Alexandru Daniel Vid

Rezumat

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea încapsulează trăsăturile normale ale unui reverse proxy oferind în plus protecție împotriva posibilelor atacuri la nivel de rețea. Produsul oferă protecție prin implementarea a două tipuri de prevenire a atacurilor: listă neagră pentru adresele IP și blocarea unor request-uri pe baza URL-urilor prin analiza conținutului lor. Pentru demonstrarea fiecărei metode în parte, s-a folosit: blocarea adreselor IP utilizate de rețeaua Tor pentru lista de IP-uri "negre" și detectarea atacurilor de SQL injection pentru protecția împotriva URL-urilor cu conținut malițios. Detectia atacurilor de tipul SQL injection se realizează prin analiza URI-urilor trimise de către clienți, în relație cu un model antrenat anterior folosind machine learning. Ambele implementări asigură adăugarea cu ușurință de noi detecții, lista de ip-uri blocate fiind accesibila utilizatorului atât pentru vizualizare cât și pentru editare.

Cuprins

Listă de figuri	III
1 Introducere	1
1.1 Context	1
1.2 Motivație	2
1.3 Structura lucrării	3
2 Obiective și specificații	5
2.1 Obiective	5
2.2 Specificații	6
2.2.1 Specificații funcționale	7
2.2.2 Specificații non-funcționale	8
3 Studiu bibliografic	9
3.1 Abordări similare	9
3.2 Tehnici/Tehnologii/Surse folosite	14
4 Fundamente teoretice	17
4.1 Reverse proxy	17
4.2 Support vector machine	18
4.3 SQL injection	19
4.4 Adresele IP ale rețelei Tor	20
4.5 Sistem de prevenire a intruziunilor	21
5 Analiză și proiectare	25
5.1 Cerințele sistemului	25
5.2 Specificațiile cazurilor de utilizare	27
5.2.1 Actori, stakeholders si interese	27
5.2.2 Basic flow	27
5.2.3 Alternative flow	28
5.3 Arhitectura sistemului si modulele principale	29
5.4 Comportamentul sistemului. Diagrama de stare si de secventa	31
5.5 Dependințele sistemului	33

5.6	Algoritmi si metode	33
5.7	Diagrama de desfășurare	34
5.8	Justificarea design-ului	35
6	Detalii de implementare	37
6.1	Structura codului sursa	37
6.2	Algoritmi, metode si API-uri	42
6.2.1	Tor activity monitor	42
6.2.2	SQLi SVM	44
6.2.3	Interfață utilizator	46
7	Teste și rezultate experimentale	51
7.1	Teste de funcționalitate	51
7.2	Teste de performanță	53
8	Manual utilizator	55
8.1	Instalarea proiectului	55
8.2	Utilizare	55
9	Conluzii	59
9.1	Privire de ansamblu asupra sistemului	59
9.2	Dezvoltări ulterioare	59
	Bibliografie	61
A	Codul sursa(principalele fisiere)	65
A.1	Reverse proxy	65
A.2	Convertor de URI in trasaturi pentru modelul SVM	66
A.3	Monitorizarea adreselor IP ale rețelei Tor	68

Listă de figuri

1.1	Diagramă reverse proxy	1
2.1	Cutia neagră a sistemului.	7
3.1	Administrarea securității unei aplicații	10
3.2	Tipuri de trăsături ale limbajului SQL	13
3.3	Arhitectura unui sistem de clasificare a request-urilor HTTP	15
4.1	Folosirea unui reverse proxy în arhitectura unei aplicații.	17
4.2	Influențele aduse algoritmului de modificarea parametrului sigma în algoritmul de antrenare.	19
4.3	Exemplu de atac realizat prin SQL injection.	20
4.4	Exemplu de trafic realizat prin rețeaua Tor.	21
4.5	Integrarea unui sistem de prevenire a intruziunilor într-o rețea.	22
5.1	Basic flow pentru evenimente	27
5.2	Principalele module ale sistemului propus	29
5.3	Diagrama de stare a sistemului propus.	31
5.4	Diagrama de secvența a sistemului propus.	32
5.5	Diagrama de desfășurare a sistemului propus.	34
6.1	Modulul pentru monitorizarea activității rețelei Tor	37
6.2	Modulul pentru prevenirea atacurilor SQL injection	39
6.3	Interacțiunea dintre interfața grafică și celelalte module	41
6.4	Meniul "Home" in interfața grafică	48
6.5	Meniul "Configure" in interfața grafică	48
6.6	Meniul "Monitor" in interfața grafică	49
6.7	Raspunsul pentru SQL injection URL	49
6.8	Principalele module ale sistemului propus	50
7.1	Raportul dintre numărul IP-urilor de pe Blacklist și Whitelist dintr-o lună	52
7.2	Raportul dintre uptime-ul IP-urile de pe Blacklist și Whitelist din aceeași lună	52

8.1	Meniul "Home" în interfața grafică	55
8.2	Meniul "Configure" în interfața grafică	56
8.3	Meniul "Monitor" în interfața grafică	57

Capitolul 1

Introducere

1.1 Context

În general, tentativele de exploatare a vulnerabilităților unei aplicații vin sub formă de input către aceasta, input generat de către un atacator care intenționează să întrerupă activitatea sau să preia controlul aplicației. Un sistem de prevenire a intruziunilor (IPS) este poziționat între aplicație și clienții acesteia, și previne exploatarea unor astfel de vulnerabilități.

Prin folosirea unui reverse proxy pentru accesarea resurselor unui server de către clienți, poate să se aducă numeroase beneficii procesului de administrare a serverului [1]. Spre deosebire de un forward proxy care e un intermediar pentru o serie de clienți prestabiliți, pentru a accesa orice server, un reverse proxy e un intermediar pentru o serie de servere prestabilite pentru a fi accesate de orice client. Unul dintre avantajele folosirii unui reverse proxy este centralizarea întregului trafic al serverului/serverelor într-un singur punct de acces, aceasta fiind și principala caracteristică exploatată de acest proiect pentru filtrarea IP-urilor nedorite (în cazul nostru cele utilizate de rețeaua Tor) și verificarea conținutului URI-urilor pentru posibile atacuri (SQL injection).

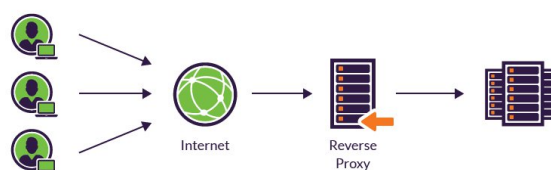


Figura 1.1: Diagramă reverse proxy

Figura 1.1 ilustrează modul de funcționare al unui reverse proxy în relație cu serverele aferente și posibili clienți.

Conform topului alcătuit de fundația OWASP cu cele mai mari 10 riscuri ale aplicațiilor în 2017 [2], SQL injection e considerat a fi cea mai mare vulnerabilitate a aplicațiilor web. Acest lucru se datorează faptului că mare parte din aceste aplicații se bazează pe procesarea conținutului furnizat de către utilizatori. Atacurile de tipul SQL injection constau în faptul că datele furnizate de către utilizator sunt introduse în interogări SQL, unde acestea sunt tratate ca și cod executabil [3]. Aplicațiile web vulnerabile la SQLi injection pot permite unor utilizatori neautorizați să facă interogări într-o bază de date asupra unor date la care nu ar avea acces în mod normal. Folosind acest tip de comportament neautorizat, un astfel de utilizator poate să obțină accesul la informații sensibile ale clienților, dar și a administratorilor aplicației, precum credențiale sau date personale. Această vulnerabilitate poate să ducă la furt de identitate sau fraudă.

În cazul rețelei Tor, aceasta le permite utilizatorilor să navigheze pe internet anonim. Anonimitatea online este importantă însă în multe cazuri aplicațiile web trebuie să știe cine se conectează la aceasta pentru a le putea determina intențiile. Numele de Tor vine de la "the onion router" care sugerează modul de operare al rețelei. Fiecare participant la rețea devine un nod de transfer, iar traficul rețelei traversează o serie de astfel noduri până să ajungă la nodul de ieșire ce creează conexiunea cu destinația dorită. Pachetele sunt criptate în mai multe "straturi", fiecare nod decriptând un singur strat de unde poate afla doar destinația nodului următor. Când pachetul ajunge la ultimul nod, acesta trimite conținutul la destinație fără să dezvăluie identitatea sursei. Această anonimitate ușurează desfășurarea atacurilor online. Conform datelor din rețeaua organizației CloudFlare 94% din traficul provenit din rețeaua Tor este alcătuit din request-uri automate de origine malițioasă [4].

1.2 Motivație

În piața actuală există multe sisteme de prevenire a intruziunilor ce oferă atât caracteristicile unui reverse proxy, cât și cele de securitate. Aceste caracteristici sunt oferite fie ca și produse individuale, fie ca și produse ce le încorporează pe ambele. Cu toate acestea, produsele de acest gen sunt în general scumpe, au o logică mascată de detectare a posibilelor probleme de securitate și sunt greu de înțeles și de configurat de către utilizator după propriile nevoi.

Prin oferirea utilizatorilor posibilitatea de a configura modul de funcționare a unui astfel de sistem, poate rezulta în sisteme mult mai eficiente și rapide, dedicate pentru preferințele și nevoile aplicației fiecărui utilizator în parte. Spre exemplu, un utilizator poate să decidă că nu are nevoie de funcționalitățile de detecție împotriva atacurilor de tip SQL injection pentru o anumită aplicație, întrucât aceasta nu prezintă vulnerabilități din acest punct de vedere, nefolosind baze de date. În cazul acesta prin eliminarea unui astfel de modul, se elimină și verificările aferente asupra request-urilor clienților, îmbunătățind astfel performanțele sistemului.

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea oferă

un sistem configurabil după preferințele utilizatorilor. Utilizatorul poate configura detecția bazată pe analiză request-ului primit de la client, acesta poate să aleagă care module să fie folosite pentru detecție, permițând și eventuala adăugare de noi module (cât timp acestea respectă anumite condiții de structură), meniurile din interfața utilizator fiind generate în mod dinamic în funcție de modelele prezente. Utilizatorul poate, de asemenea să configureze și lista IP-urilor blocate, permițându-i-se să șteargă din cele existente, respectiv să adauge unele noi, după bunul plac.

1.3 Structura lucrării

În această secțiune se prezintă structura lucrării pe capitole și o scurtă descriere a conținutului acestora:

Primul **capitol** 1, prezintă o scurtă introducere despre proiect, contextul acestuia și motivația pentru implementarea sistemului propus.

Capitolul 2 prezintă obiectivele lucrării, specificațiile sistemului, motivând deciziile luate în implementarea sistemului, cerințele funcționale și nonfunctionale necesare implementării sistemului.

Capitolul 3 descrie alte abordări similare ale problemelor tratate de proiectul propus, prin evidențierea asemănărilor și diferențelor dintre acestea și se explică tehnologiile și metodele folosite de proiect.

În **capitolul** 4 sunt evidențiate și explicate pe scurt aspectele teoretice pe care se bazează proiectul.

Capitolul 5 descrie design-ul proiectului și cuprinde: cerințele sistemului, specificațiile cazurilor de utilizare, arhitectura sistemului, comportamentul sistemului, datele utilizate de sistem, dependențele sistemului, algoritmi esențiali și metodele folosite. Descrierea acestora se realizează prin asocierea cu diagramele aferente.

Capitolul 6 prezintă în amănunt detaliile de implementarea a proiectului. În acest capitol sunt abordate: organizarea codului sursă, descrierea principalelor clase și funcționalități ale proiectului și posibilitățile de desfășurare a execuției programului.

Capitolul 7 reprezintă metodele de validare a soluțiilor/sistemului descris în capitolele anterioare, scenariile de testare a corectitudinii funcționale, a utilizabilității și performanței.

Capitolul 8 descrie pașii de instalare și rulare a aplicației.

Capitolul 9 cuprinde ce s-a realizat, relativ la ce s-a propus, în ce constă experiența acumulată, care au fost punctele dificile atinse și o descriere a posibilelor dezvoltări și îmbunătățiri ulterioare.

Capitolul 2

Obiective și specificații

Acest capitol prezintă obiectivele lucrării, motivând deciziile luate în implementarea sistemului, specificațiile sistemului, cerințele funcționale și nonfuncționale necesare implementării sistemului.

2.1 Obiective

Elaborarea unor măsuri de securitate împotriva anumitor tipuri de atacuri sau dezvoltarea unei logici de filtrare a clienților serviți de către aplicație este posibilă și în partea de implementare a serverului, ceea ce ar putea oferi și o eventuală creștere în performanțele aplicației, eliminând astfel posibile interceptări suplimentare și validări ale traficului. Însă o astfel de implementare presupune o arhitectură mult mai complexă pentru partea de server și individual cunoștințe avansate despre securitate, posibilele vulnerabilități la care acesta poate să fie predispus, precum și modalități de combatere ale acestora.

O soluție mult mai simplă la această problemă este folosirea unui modul care să implementeze aceste funcționalități separat. Mare parte din produsele de pe piață, ce îndeplinesc aceste caracteristici sunt destul de scumpe și nu oferă foarte multă libertate din punctul de vedere al configurării securității dorite de către utilizator asupra produsului sau. În cazul IP-urilor blocate, multe aplicații nu oferă libertatea utilizatorilor de a edita listele de referință ale detecțiilor, acestea fiind actualizate automat conform unor date interne, iar eventualele abateri de la aceste reguli se realizează prin adăugarea de excepții. În cea ce privește validarea request-urilor primite de la clienți, mare parte din aceste sisteme nu oferă o protecție configurabilă. Cum s-a precizat mai sus, acest tip de sisteme pot să introducă mici întârzieri datorate validărilor suplimentare asupra request-urilor primite de la clienții produsului, însă în unele cazuri anumite validări sunt făcute inutil întrucât produsele respective neputând să prezinte vulnerabilități de acel fel.

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea are obiectivul de a oferi o componentă gratuită cât mai ușor de integrat și de configurat de

către utilizator după preferințele produsului sau, care să faciliteze o protecție cât mai eficientă cu suport pentru îmbunătățiri. Sistemul trebuie să fie ușor de instalat și de configurat, oferindu-i utilizatorului o interfață clară, sugestivă și ușor de folosit prin care să interacționeze cu acesta. Detecțiile sistemului trebuie să fie selectabile, utilizatorul putând alege în momentul pornirii sistemului ce vulnerabilități să fie tratate de acesta. Lista IP-urilor blocate trebuie să fie ușor de vizualizat și editabilă, permițând utilizatorului să își impună cu ușurință propriile reguli asupra modului de funcționare a sistemului. Pentru realizarea detecției împotriva atacurilor de tipul SQL injection se impune prelucrarea unor date reale pentru antrenarea modelului de machine learning. Prin folosirea unor date provenite din atacuri reușite sau tentative de atacuri reale, se poate crea o precizie mult mai bună pentru o clasificare cât mai precisă a posibilelor atacuri. Sistemul trebuie de asemenea să fie construit modular pentru a permite realizarea de modificări cu ușurință, iar încărcarea detecțiilor realizată dinamic, permițând astfel că adăugarea de noi detecții să fie cât mai simplă.

2.2 Specificații

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea trebuie să fie capabil să servească ca și un reverse proxy pentru un server, să blocheze atacurile de tip SQL injection asupra lui și să nu permită conectarea clienților cu IP-uri conținute în lista neagră.

Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea va procura utilizatorului o interfață grafică prietenoasă, ușor de folosit, prin intermediul căreia, acesta va putea să seteze mediul de rulare al sistemului. Interfața va permite setarea specificațiilor server-ului, adresa și portul pe care acesta acceptă conexiuni, dar și a interfețelor prin intermediul cărora se pot realiza conexiuni la server.

În interfața grafică se vor afișa și eventualele detecții realizate de produs. Într-o fereastră separată utilizatorul trebuie să aibă posibilitatea să vizualizeze toate deciziile sistemului și motivele din spatele deciziilor, permițând astfel acestuia să înțeleagă modul de funcționare, respectiv să raporteze sau să modifice sistemul (în cazurile în care i se oferă această posibilitate) când comportamentul acestuia nu se află în conformitate cu nevoile sau cerințele sale.

În momentul configurării modului de rulare al sistemului, utilizatorul trebuie să aibă și posibilitatea de a impune ce module de securitate să fie folosite de acesta în timpul rulării. Pentru a eficientiza cât mai mult sistemul, utilizatorul poate să aleagă care sunt modulele de interes pentru propria aplicare, evitând astfel validarea unor evenimente ce nu prezintă interes pentru acesta.

În timpul rulării sistemul va asculta pe interfețele setate de către utilizator pentru posibile cereri de conexiuni la server-ul setat. În funcție de modulele alese în momentul pornirii, acesta va verifica sau nu adresa clientului validând astfel conexiunea. În cazul în care adresa clientului se află pe lista neagră de adrese, conexiunea acestuia este refuzată,

iar aplicația înregistrează această decizie în fereastra de evenimente vizibilă utilizatorului. După realizarea conexiunii la server, fiecare request trimis de clienți către acesta va fi evaluat conform modulelor configurate. Dacă request-urile sunt considerate că fiind "curate" acestea sunt trimise mai departe la server. În caz contrar, clientului i se întoarce un cod de eroare, iar request-ul nu va mai fi trimis mai departe către server, de asemenea înregistrând evenimentul în fereastra de evenimente vizibilă utilizatorului.

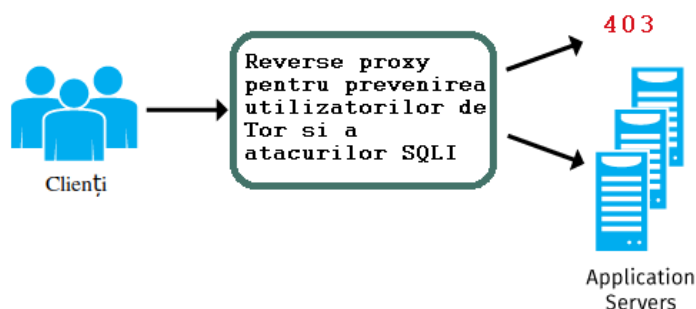


Figura 2.1: Cutia neagră a sistemului.

Figura 2.1 prezintă cutia neagră a sistemului propus.

2.2.1 Specificații funcționale

Sistemul trebuie să prezinte o interfață grafică ușor de folosit de către utilizator și să fie capabil să redirectioneze traficul interceptat către un anumit server, clasificând și filtrând traficul malițios. Pentru a atinge obiectivele proiectului, următoarele cerințe funcționale trebuie îndeplinite:

- Să realizeze conexiunea la un server HTTP/HTTPS și să redirectioneze traficul primit către acesta.
- Să intercepteze traficul venit pe o anumită interfață și port prestabilit.
- Să prelucreze request-urile primite de la clienți într-un format specific clasificatorului de SQL injection.
- Să nu redirectioneze request-urile clasificate ca și SQL injection.
- Să blocheze conectarea clienților ce folosesc IP-uri clasificate ca IP-uri de Tor.
- Să permită utilizatorului să editeze și să vizualizeze lista IP-urilor de Tor.
- Să prezinte în interfața grafică toate intervențiile realizate asupra traficului (blocări de conexiuni sau de request-uri).

- Să permită utilizatorului să configureze modul de operare al sistemului.

2.2.2 Specificații non-funcționale

Sistemul trebuie, de asemenea, să aibă următoarele caracteristici non-funcționale pentru a realiza obiectivele specificate:

- Să fie ușor de instalat și de folosit pentru orice utilizator, oricât de neexperimentat .
- Să poată intercepta traficul de pe orice/oricâte interfețe disponibile.
- Să poată rula pe orice sistem de operare Windows cu Python2 instalat.
- Să aibă o rată de blocare de 100% a IP-urilor de pe lista neagră, iar în cazul detecției de SQL injection să nu aibă detecții false pozitive mai mari 2-3% și o acuratețe generală de peste 90%.

Capitolul 3

Studiu bibliografic

În acest capitol sunt prezentate alte abordări similare ale problemelor tratate de proiectul propus, prin evidențierea asemănărilor și diferențelor dintre acestea și se explică tehnologiile și metodele folosite de proiect.

3.1 Abordări similare

Precum Richard Bassett, Cesar Urrutia și Nick Ierace susțin în articolul **Intrusion prevention systems** [5] ”sistemele de prevenire a intruziunilor sunt o componentă importantă a sistemelor de protecție IT, iar fără această tehnologie, datele noastre și rețelele ar fi mult mai predispuse activităților malițioase”.

În general, tentativele de exploatare a vulnerabilităților unei aplicații vin sub formă de input către o aplicație țintă. Acest input fiind generat de către un atacator ce intenționează o controleze sau să îi întrerupă activitatea. În cazul unui atac reușit, un astfel de atacator poate să dezactiveze temporar aplicația (atacuri de tipul denial of service) sau poate accesa, altera sau executa date/cod în interiorul aplicației. Un sistem de prevenire a intruziunilor are rolul de a examina traficul destinat unei aplicații și de intercepta și bloca astfel de tentative [6].

Un sistem de prevenire a intruziunilor este de regulă folosit alături de un sistem firewall respectiv alături de un sistem de detectare a intruziunilor. Deși au scopuri asemănătoare, aceste sisteme au funcționalități diferite și rezolvă diferite probleme de securitate. Un sistem de prevenție a instrucțiunilor este cel mai bine comparat cu sistemele de tip firewall. Un sistem firewall tipic este constituit dintr-o serie de reguli ce permit traficului să treacă. Aceste reguli sunt sub forma ”dacă traficul îndeplinește anumite condiții poate trece”, însă dacă nu există nici o regulă care să potrivească anumite pachete, acestea sunt blocate. Asemenea sistemelor firewall, sistemele de prevenire a intruziunilor prezintă un set de reguli de filtrare a pachetelor, request-urilor sau a clienților, însă aceste reguli sunt de regulă reguli de blocare. Astfel, dacă un anumit pachet nu potrivește nici o regulă sistemul de prevenire a intruziunilor îl lasă să treacă [7].

Spre deosebire de sistemele de tip firewall sau cele de prevenire a intruziunilor, care oferă control utilizatorului asupra traficului ce trece prin rețea, sistemele de detecție a intruziunilor permite acestuia să vizualizeze evenimentele din rețea. Precum și Joel Snyder susține în articolul **Do you need an IDS or IPS, or both?** [7] sistemele de detecție a intruziunilor oferă unui utilizator facilități asemănătoare unui analizator de pachete [8], însă din perspectiva de securitate a rețelei. Aceste informații furnizate de către sistem îi permit utilizatorului să decopere:

- Încălări ale politicilor de securitate, precum utilizatori sau sisteme ce desfășoară activități ce încalcă politicile prestabilite.
- Posibile sisteme infectate ce folosesc rețeaua pentru a se răspândi sau să atace alte sisteme.
- Scurgeri de informație cauzate de infectarea unor sisteme cu malware-i sau de utilizatori rău intenționați.
- Erori de configurare în sisteme sau aplicații cu setări de securitate incorecte sau configurări proaste ce consumă prea multă bandă de rețea.
- Detectarea unor clienți sau servere ce accesează sau sunt accesate în mod neautorizat, respectiv aplicații malițioase ce fac asta.

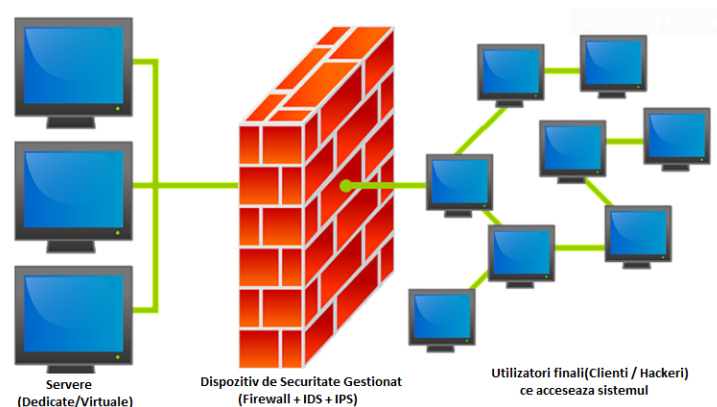


Figura 3.1: Administrarea securității unei aplicații

Figura 3.1 prezintă administrarea securității unei aplicații folosind combinația dintre cele trei sisteme.

În comparație cu sistemele de detecție a intruziunilor care sunt sisteme pasive și scanează rețeaua fără să interfereze cu traficul, sistemele de prevenire a intruziunilor sunt plasate între server și clienți, alterând în mod automat traficul în cazul în care acesta

declanșează una din regulile prezente în sistem. Precum sunt prezentate și în articolul **What is an intrusion prevention system?** [6], printre funcționalitățile unui sistem de prevenire a intruziunilor se număra:

- Notificarea unui administrator de rețea în cazul în care una sau mai multe reguli sunt declanșate.
- Oprirea pachetelor considerate malițioase pentru rețea.
- Blocarea unor utilizatori prin excluderea adreselor IP ale acestora.
- Resetarea unor conexiuni.

În cea ce privește funcționalitățile oferite de un sistem de prevenire a intruziunilor, acestea sunt specifice tipului sistemului. Conform autorului articolului **Intrusion Prevention System (IPS): Definition & Types** [9], Beth Hendricks, există patru tipuri primare de astfel de sisteme:

- Network-based: Protejează întreaga rețea.
- Wireless: Protejează doar rețeaua wireless.
- Network behavior: Examinează traficul din rețea.
- Host-based: Software cu scopul de a proteja un singur calculator.

Sistemele de tipul Network-based (reprezentând și categoria în care se încadrează *Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea*) presupun implementarea unor senzori în rețea care capturează și analizează traficul ce trece prin acesta. Acești senzori sunt plasați în puncte cheie ale rețelei pentru a putea captura în timp real traficul, iar în cazul interceptării unor activități malițioase să poată interveni imediat, fără să scadă performanța rețelei. Aceste sisteme oferă protecție rețelei indiferent de dimensiunile sau creșterea acesteia, adăugarea de noi device-uri fiind posibilă fără să necesite adăugarea de noi senzori. Adăugarea de noi senzori fiind nevoită doar în cazul în care traficul rețelei depășește capacitatea de procesare a senzorilor curenți, influențând astfel performanțele rețelei [10].

În funcție de nevoi, un sistem de prevenire a intruziunilor poate să ofere diferite opțiuni de protecție pentru diferite părți ale rețelei. Unele sunt capabile să oprească traficul malițios sau să limiteze lățimea de bandă către anumite părți ale rețelei. Conform [11] aceste sisteme pot oferi protecție împotriva următoarelor tipuri de atacuri:

- **ICMP Storms:** un volum mare de ecouri ICMP pot să indice activități malițioase precum cineva ce scanează rețeaua.
- **Ping to Death:** un utilizator poate să modifice comandă de ping, astfel încât să trimită un număr mare de pachete de dimensiune mare către o destinație țintă pentru a o scoate din uz.

- **SSL Evasion:** unele atacuri se pot folosi de criptarea SSL pentru a evita dispozitivele de securitate, întrucât în general acestea nu sunt decriptate.
- **IP Fragmentation:** constă în exploatarea faptului că pachetele sunt descompuse în fragmente pentru a satisface cerințele de dimensiune a rețelelor traversate, inundând o destinație țintă cu fragmente false pentru a îi consuma resursele.
- **SMTP mass mailing attacks:** un sistem infectat poate să se folosească de email-ul utilizatorului pentru a se răspândi, rezultând într-un trafic mare destinat serverului de mail.
- **DoS/DDoS attacks:** cu scopul de a face o resursa indisponibilă utilizatorilor, este realizată prin inundarea sistemului țintă cu un număr mare request-uri de la unul sau mai multe (în cazul Dos distribuit-DDoS) sisteme malițioase.
- **SYN Flood attacks:** atacatorul trimite un număr mare de pachete de inițiere a unei conexiuni fără să mai răspundă ulterior, epuizând astfel resursele de memorie.
- **Http obfuscation:** pentru a evita să fie detectate de anumite sisteme de protecție, unele atacuri folosesc tehnici de ofuscarea a request-urilor HTTP.
- **Port Scanning:** este folosit pentru descoperi ce porturi sunt deschise pe un sistem, ulterior permițându-i atacatorului să știe ce vulnerabilități ar putea prezenta sistemul.
- **ARP Spoofing:** un atacator trimite în rețea pachete false de ARP legându-și propria adresă MAC de adresa IP a unui alt sistem. Ca urmare, atacatorul va primi pachete destinate sistemului cu adresa IP folosită în pachetul de ARP.
- **CGI Attacks:** un atacator poate să trimită request-uri malițioase, determinând destinația să trateze request-ul primit ca și cod executabil, oferindu-i atacatorului acces pe sistem.
- **Buffer Overflow attacks:** presupune ca atacatorul să depășească limitele unui buffer de lungime fixă, excesul de date ajungând să suprascrie zone adiacente de memorie rezultând în căderea sistemului sau dându-i atacatorului oportunitatea să ruleze cod propriu.
- **OS Fingerprinting attacks:** presupune ca atacatorul să descopere ce sistem de operare rulează pe un sistem și folosindu-se de această informație să exploateze vulnerabilități specifice acelui sistem de operare.

Sistemul propus, *Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea* implementează un sistem de prevenire a intruziunilor folosindu-se de un reverse proxy pentru a intercepta tot traficul care intră și iese din rețea (reprezentând

senzorii ce au rol de a captura și analiza traficul) și oferind protecție împotriva a două categorii de atacuri: oprirea de URL-urilor malițioase destinate unei aplicații (protecție împotriva SQL injection) și blocarea adreselor IP cunoscute ca fiind folosite de utilizatori rău intenționați (protecție împotriva adreselor IP ale rețelei Tor).

Pentru prevenirea atacurilor de SQL injection, se folosește o metodă asemănătoare celei propuse de Eun Hong Cheon, Zhongyue Huang și Yon Sik Lee în lucrarea **Preventing SQL Injection Attack Based on Machine Learning** [12]. Pentru clasificarea request-urilor HTTP în SQL injection sau curate, se folosește un sistem bazat pe machine learning. Acest sistem este antrenat anterior cu date reale, ca și trăsături fundamentale în clasificare, folosindu-se cuvintele cheie și simbolurile specifice limbajului SQL (spre exemplu: SELECT, ADD, DELETE, ", ' etc.).

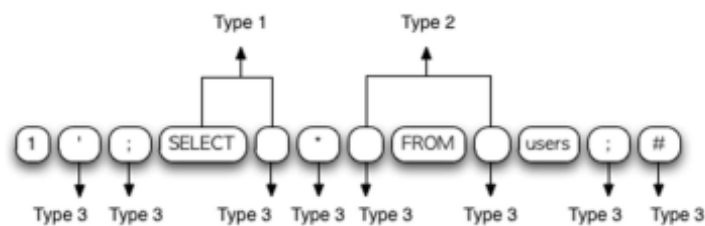


Figura 3.2: Tipuri de trăsături ale limbajului SQL

Figura 3.2 prezintă tipurile de trăsături luate în considerare în lucrarea **Preventing SQL Injection Attack Based on Machine Learning** în raport cu simbolurile sau cuvintele cheie folosite.

O altă abordare pentru prevenirea atacurilor SQLI este propusă de Fredrik Valeur, Darren Mutz și Giovanni Vigna în lucrarea **A Learning-Based Approach to the Detection of SQL Attacks** [13]. În această lucrare se prezintă folosirea unui sistem bazat pe detecția de anomalii pentru detectarea atacurilor ce exploatează o aplicație pentru a îi compromite baza de date. Asemeni abordării bazate pe machine learning, acest sistem presupune o fază anterioară de antrenare în care se învață comportamentul normal al utilizatorilor, alcătuiind astfel niște profile specifice. Astfel în faza de detecție, comportamentul ce nu coincide cu profilele alcătuite în faza de antrenate, este considerat malițios.

Pentru prevenirea utilizatorilor de Tor, în general lista de IP-uri este alcătuită din toate IP-urile care au utilizat rețeaua într-un anumit interval de timp, aceasta fiind actualizată periodic. O astfel de abordare este folosită și în cazul marelui firewall al Chinei [14] care identifică nodurile la prima accesare a rețelei de Tor. Însă precum precum se evidențiază și în articolul **Characterizing the Nature and Dynamics of Tor Exit Blocking** [15] o astfel de abordare nu este cinstită față de unii utilizatori de Tor, întrucât reputația acestora este împărțită între toți utilizatorii. Astfel un nod care este utilizat doar pentru câteva minute sau ore (probabil din motive de curiozitate) poate să ajungă să fie blocat, fiind tratat asemeni cu un nod ce funcționează de câteva zile. O astfel de discriminare a încercat să fie evitată prin implementarea aleasă a sistemului propus. Pentru

realizarea listei de IP-uri blocate se folosește un algoritm ce stabilește o limită de timp minima de funcționare pentru un anumit nod în intervalul a 30 de zile.

3.2 Tehnici/Tehnologii/Surse folosite

Pentru realizarea sistemului propus s-au folosit două limbaje de programare: Python2/3(pentru partea de back end) și C#(pentru partea de front end). În partea de back end a proiectului se realizează implementarea unui reverse proxy pentru a intercepta traficul uneia sau mai multor interfețe, un modul pentru clasificarea request-urilor împotriva atacurilor SQL injection și un modul pentru generarea listei negre de IP-uri ce utilizează frecvent rețeaua Tor. Toate aceste componente sunt realizate prin utilizarea de librării open-source pentru a ușura și eficientiza munca precum: twisted [16], beautiful soup [17], libsvm [18].

Motivul utilizării atât a limbajului Python3 cât și Python2 este datorat diferențelor de module și librării open-source disponibile pentru cele două limbaje dar și a faptului că suportul pentru Python2 se încheie în anul 2020. Conform documentațiilor oficiale [19] și [20], dar și articolului *Python 2 to python 3: why, and how hard can it be?* de Tim Grey [21], între cele două versiuni nu sunt modificări majore, însă în anumite cazuri pot exista librării care să ofere doar suport pentru una dintre acestea.

În realizarea modulului pentru clasificarea request-urilor împotriva atacurilor SQL injection s-a folosit o colecție de date reale atât de atacuri cât și de trafic curat. Pentru uniformizarea acestor date și pentru a trata tentativele de păcălire a clasificatorului prin codarea unor caractere în valoarea lor în cod hexadecimal (exemplu 'https://www.google.ro/search?q=a' echivalent cu 'https://www.google.ro/search?q=%61') datele au fost preprocesate și transformate în întregime în coduri hexadecimale [22]. În procesarea datelor, pentru identificarea trăsăturilor relevante, s-au identificat caracterele specifice limbajului [23] și cuvintele cheie rezervare [24]. Ulterior, pentru antrenarea modelului de support vector machine și pentru clasificarea noilor request-uri s-a folosit software-ul open-source libsvm [25].

Figura 3.3 prezintă arhitectura unui sistem de clasificare a request-urilor HTTP de un sistem bazat pe machine learning. Structura este prezentată în lucrarea prezentată și anterior **Preventing SQL Injection Attack Based on Machine Learning** [12]. Acesta structură a reprezentat un model de pornire în realizarea modulului de prevenire a atacurilor SQL injection, implementarea modulului încercând să aducă îmbunătățiri de performanță prin modificarea algoritmului folosit pentru antrenarea modelului de support vector machine, dar și prin filtrarea trăsăturilor propus în lucrare în conformitate cu raportul dintre obișnuința de apariție a acestora atât în request-urile ce intenționează să execute un atac cât și în cele curate.

Pentru blocarea IP-urilor utilizate de rețeaua Tor s-a folosit un script scris în Python3. Programul interoghează periodic(din 6 în 6 ore) informațiile oferite de *Tor Network Status* [26] identificând astfel nodurile cu un "Uptime" mai mare de 7 zile în parcursul unei

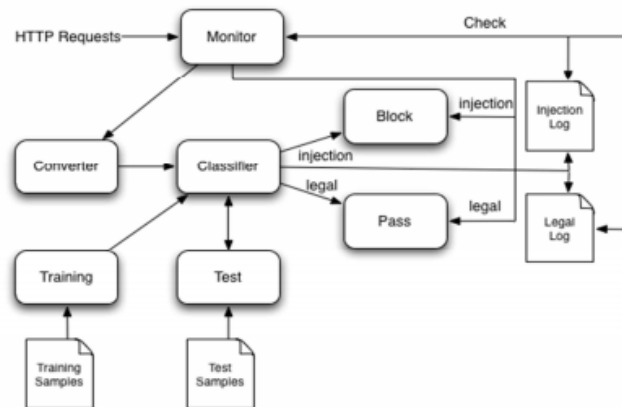


Figura 3.3: Arhitectura unui sistem de clasificare a request-urilor HTTP

luni. Blocarea IP-urilor se realizează prin compararea cu o astfel de listă generată lunar.

Componenta ce încorporează toate modulele de protecție, este cea de reverse proxy. Aici este monitorizat tot traficul ce vine de pe o anumită interfață (una sau mai multe, în funcție de configurația utilizatorului) și este trecut prin toate modulele disponibile pentru a verifica condițiile de securitate. Pentru testarea dacă o adresă IP este utilizată frecvent de rețeaua Tor, în momentul în care un client dorește să realizeze o conexiune la serverul protejat de sistem, adresa IP a acestuia este verificată să nu se afle pe lista IP-urilor blocate. Pentru actualitate, lista adreselor IP blocate este actualizată periodic cu adresele IP utilizate frecvent de rețeaua Tor în ultima luna. Modulul de prevenire a atacurilor SQL injection este integrat tot în componenta de reverse proxy, însă evaluarea request-urilor este făcută după realizarea conexiunii între client și server. Request-urile primite de către server sunt tratate asemănător celor folosite pentru antrenarea modulului de support vector machine, însă pentru clasificarea acestora este folosit modulul antrenat în fază inițială și software-ul de prezicere oferit tot de libsvm [18].

Capitolul 4

Fundamente teoretice

În acest capitol sunt evidențiate și explicate pe scurt aspectele teoretice pe care se bazează proiectul.

4.1 Reverse proxy

Un reverse proxy este un server intermediar care trimite mai departe request-urile pentru conținut, de la mai mulți clienți nedefiniți, către unul sau mai multe servere. Un reverse proxy este un tip de proxy care în mod normal este situat în spatele unui firewall într-o rețea internă și redirecționează traficul clienților către serverele asociate. Acesta introduce un nivel în plus de abstractizare și control, asigurând controlul fluxului de trafic [27].

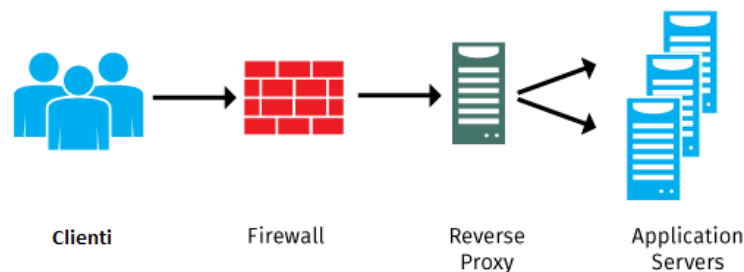


Figura 4.1: Folosirea unui reverse proxy în arhitectura unei aplicații.

Figura 4.2 prezintă modalitatea de integrare a unui reverse proxy în implementarea arhitecturii de back end a unei aplicații.

Cele mai obișnuite caracteristici ce pot fi oferite de utilizarea unui reverse proxy sunt:

- Load balancing-un reverse proxy poate să distribuie request-urile primite de la clienți, astfel încât nici un server să nu fie copleșit de requesturi. În cazul în care un server

este supraîncărcat cu request-uri sau este căzut, acesta poate să redirecționeze traficul carea alte servere funcționale.

- Web acceleration - un reverse proxy poate să realizeze compresia datelor sau să memoreze în memoria cache conținut ce este frecvent accesat sau poate să realizeze operațiile de criptare SSL executate în mod normal de server, îmbunătățind astfel în mod considerabil viteza de comunicare dintre client și serverul destinație.
- Securitate și anonimitate-prin interceptarea request-urilor primite de către server, acesta asigură anonimitatea serverului acționând că un nivel extra de securitate. De asemenea se asigură că mai multe servere pot fi accesate prin intermediul unui punct comun, indiferent de structura rețelei interne.

4.2 Support vector machine

Algoritmul de machine learning, support vector machine reprezintă un model obținut prin folosirea de diverși algoritmi pentru antrenarea acestuia, folosit pentru a clasifica date. Acest model intră în categoria de învățare supravegheată('supervised learning'), întrucât pentru obținerea lui se folosește un set de date că și exemplu, date pe care modelul le va folosi ca referință pentru clasificarea noilor evenimente.

Realizarea unui astfel de model se obține în urma executării unui proces elaborat ce implică mai mulți pași:

- Primul pas reprezintă identificarea datelor relevante în cea ce privește problema tratată(setul de antrenare). În conformitate cu scopul clasificării unor evenimente/date, în două(sau mai multe) categorii, inițial trebuie identificate o serie de astfel de evenimente și categorizate de către utilizator în evenimente ce sigur aparțin fiecărei dintre categoriile țintă.
- După obținerea datelor de antrenare, trebuie identificate toate trăsăturile relevante din aceste date, trăsături care să fie cât se poate de specifice fiecărei categorii în parte. Fiind recomandată evitarea trăsăturilor ce sunt prezente în mare parte din date sub aceeași formă (ex:caracterul '=' sau '?' într-un URI folosit pentru clasificarea atacurilor SQL injection), indiferent de categoria din care acestea fac parte.
- După obținerea trăsăturilor specifice datelor de antrenare, se realizează antrenarea modelului folosind un algoritm specific. În cazul proiectului propus s-a folosit algoritmul gata implementat, furnizat de biblioteca open source LIBSVM [18]. Pentru obținerea modelului, datele de antrenare au fost procesate folosind un kernel gaussian. Un kernel gaussian reprezintă modul în care modelul procesează datele de antrenare astfel încât clasificarea noilor date să fie realizată prin calcularea similarităților dintre acestea și cele de antrenare. În calcularea similarității dintre aceste două tipuri de date, un parametru foarte important este sigma. Acest parametru este ales pentru

întreg setul de date, iar valoarea lui este direct proporțională cu gradul de similaritate pe care algoritmul îl va asocia la două evenimente/date diferite.

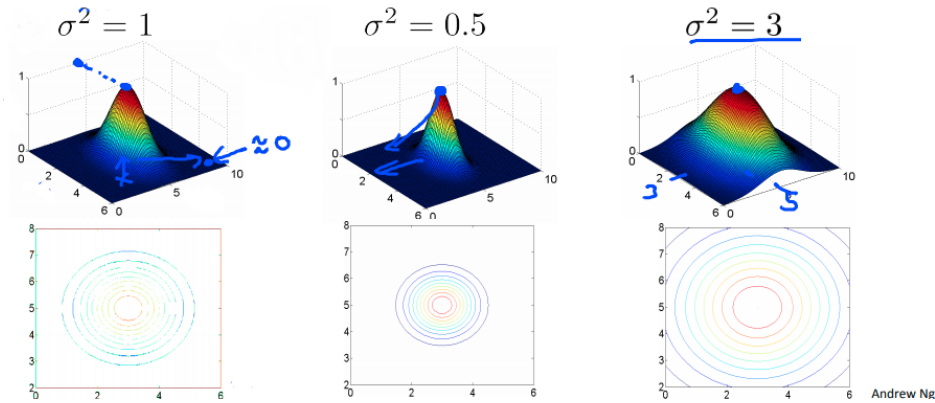


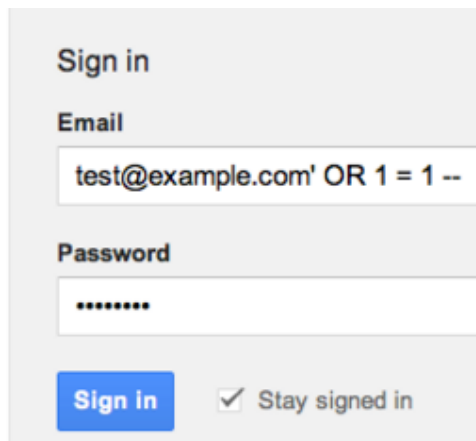
Figura 4.2: Influențele aduse algoritmului de modificarea parametrului sigma în algoritmul de antrenare.

Figura 4.2 prezintă cum influențează clasificarea unui nou eveniment valoarea parametrului sigma din formula algoritmului de support vector machine. Figura 4.2 a fost preluată din slide-urile cursului de machine learning susținut de Andrew Ng [28]

4.3 SQL injection

Atacurile de tipul SQL injection sunt realizate prin injectarea de cod executabil într-o bază de date.

Procesul de interacționare cu o bază de date presupune realizarea de interogări asupra acesteia. În formularea acestor interogări, utilizatorul trebuie să prezinte interpretorului, sub formă de șiruri de caractere, numele tabelelor interogate sau valorile unor câmpuri specifice din acestea. Aceste șiruri de caractere sunt delimitate folosind caracterul ”sau ’. Atacurile de tipul SQL injection exploatează folosirea acestor delimitatori de șiruri de caractere, trimițând șiruri de caractere eronate intenționat către baza de date. Un utilizator rău intenționat poate să furnizeze astfel de șiruri de caractere către o bază de date prin intermediul oricărui procesator de conținut disponibil unui client al unei aplicații ce comunică cu o bază de date. Aceste șiruri de caractere delimitează prematur valoarea care este folosită în interogare, introducând după aceasta o serie de caractere pe care interpretorul le va trata că și cod executabil, oferindu-i astfel utilizatorului să execute operațiuni asupra bazei de date la care nu ar avea accesul în mod normal. Aceste operațiuni pot să reprezinte alterarea bazei de date sau obținerea de date confidențiale.



The image shows a 'Sign in' form with two input fields: 'Email' and 'Password'. The 'Email' field contains the text 'test@example.com' OR 1 = 1 --'. Below the fields are a blue 'Sign in' button and a checkbox labeled 'Stay signed in' which is checked.

Figura 4.3: Exemplu de atac realizat prin SQL injection.

Figura 4.3 prezintă o tentativă de atac prin SQL injection în care în câmpul de validare a email-ului se încearcă injectarea de cod ce va fi executat în interogarea de validare a credentialelor. Prin prezența caracterului ' se escapează tot textul urmat după acesta ca fiind cod și nu un string ce face parte din câmpul de email. Operația logică "OR 1=1" va determina interpretorul să returneze adevărat(valid) pentru orice adresă de email introdusă înaintea caracterului '.

4.4 Adresele IP ale rețelei Tor

Rețeaua Tor reprezintă un softwrae gratis de anonimizare a traficului pe internet. Numele este de fapt un acronim pentru "The Onion Router"(router-ul ceapă) care sugerează modul de funcționarea al acestuia, fiecare nod din rețea adăugând un strat extra de securitate celor precedente. Modul de funcționare al rețelei se bazează pe rutarea traficului prin cât mai multe noduri pentru a anonimiza și a face cât mai greu de urmărit traficul unei anumite persoane. Aceste noduri prin care traficul este direcționat sunt susținute gratis de către voluntari/utilizatori de Tor din întreaga lume.

Pentru criptarea traficului rețeaua Tor folosește criptarea la nivelul aplicației în ceea ce privește structura rețelelor de calculatoare(modelul OSI sau TCP/IP). Datele transmise sunt criptate, incluzând destinația, cu excepția nodului următor, astfel creându-se structura de "ceapă" asupra unui pachet. Selecția nodurilor prin care se face rutarea pachetelor este aleasă random. Fiecare pachet decriptează un "strat", aflând nodul următor pentru pachetul respectiv, nodul final decriptând datele inițiale și realizând transmisia către destinație, fără să îi comunice sursa pachetului. Întrucât în comunicarea pachetelor, pe parcursul rutelor parcurse, se cunoaște în permanență doar nodul următor, acest lucru împiedică monitorizarea traficului între sursă și destinație.

Cu toate că rețeaua Tor oferă anonimitate de partea clientului, acest lucru nu se

realizează și față de ea. Rețeaua nu se ascunde față de serviciile accesate prin intermediul ei. Astfel un site anume poate să detecteze dacă un anumit client îl accesează folosind rețeaua Tor sau nu.

Întrucât rețeaua Tor nu își ascunde adresele IP folosite de către aceasta, identificarea lor și procurarea de date despre acestea este destul de ușoară. În proiectul propus s-a folosit un serviciu care furnizează gratuit astfel de date [26] (adresa IP, uptime etc.) și s-au folosit algoritmi proprii pentru procesarea acestor date, eliminând astfel necorectitudinea dintre utilizatorii rețelei.

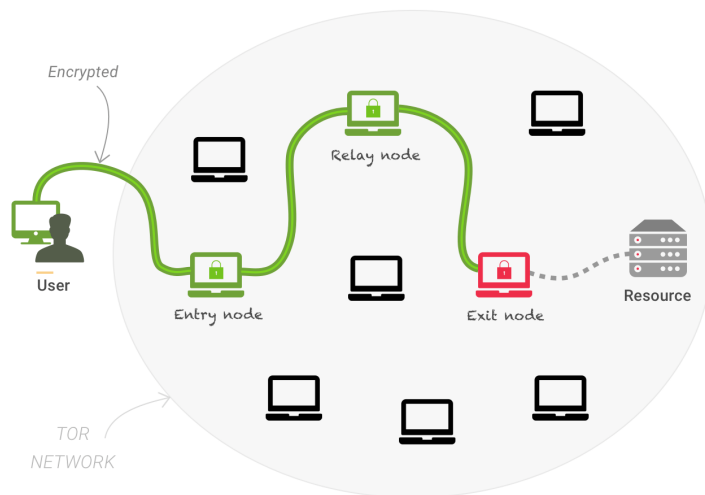


Figura 4.4: Exemplu de trafic realizat prin rețeaua Tor.

Figura 4.4 prezintă principalele elemente folosite la rutarea traficului de la client la destinație prin intermediul rețelei Tor.

4.5 Sistem de prevenire a intruziunilor

Conform scurtei descrieri prezentate în capitolul anterior, un sistem de prevenire a intruziunilor are rolul de a filtra traficul dintre clienții unui server și serverul propriu-zis.

Acest sistem funcționează liniar, adică este plasat direct între server și clienți acestuia. În cazul proiectului propus, componenta de bază pentru interceptarea traficului este realizată prin implementarea unui reverse proxy, oferind astfel caracteristica de interceptare și decriptare a traficului, ce permite analiza acestuia, dar și avantajele specifice utilizării unui reverse proxy.

Pentru filtrarea traficului, un sistem de prevenire a intruziunilor implementează anumiți senzori care au rolul să inspecteze tot traficul ce trece prin sistem, realizând această inspecție în timp real. Datorită acestei verificări, orice pachet considerat malițios

este oprit din a ajunge la serverul destinație. În proiectul propus, implementarea acestor senzori este realizată în două moduri. În cazul validării adreselor IP împotriva utilizatorilor de Tor se folosește o lista de IP-uri ce conține adrese frecvent utilizate de rețeaua Tor. În interiorul reverse proxy-ului, în momentul creării unei noi conexiuni, acesta verifică ca adresa IP ce solicită conectarea la server să nu fie conținută de lista menționată. În cazul detecției împotriva atacurilor de SQL injection, senzorul este implementat utilizând un model de support vector machine. În interiorul reverse proxy-ului în momentul interceptării unui request venit din exterior către rețeaua internă, acesta verifică dacă requestul poate fi clasificat ca și tentativă de atac. În caz afirmativ blocând trecerea acestuia mai departe către server.

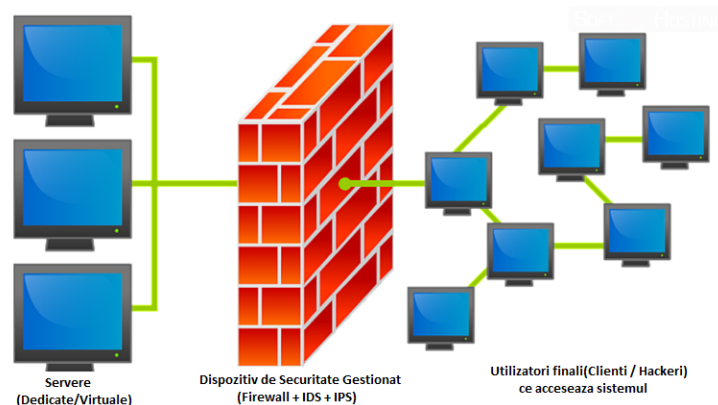


Figura 4.5: Integrarea unui sistem de prevenire a intruziunilor într-o rețea.

Figura 4.5 prezintă arhitectura unei rețele interne ce integrează un sistem de prevenire a intruziunilor pentru protejarea acesteia.

Un sistem de prevenire a intruziunilor poate să efectueze oricare din următoarele acțiuni în momentul detectării unui eveniment malițios [29]:

- Să întrerupă sesiunea dintre client și server, în cazul în care clientul desfășoară sau încearcă să desfășoare activități malițioase în rețeaua protejată de sistem. Acest lucru se poate realiza prin blocarea anumitor credentiale asociate cu utilizatorul respectiv sau prin blocarea adresei IP a acestuia.
- În condițiile în care un sistem de prevenire a intruziunilor detectează/clasifica o activitate ca fiind malițioasă, acesta poate să ia măsuri automat pentru a preveni un astfel de atac pe viitor(ex: în momentul detectării unei tentative de atac prin SQL injection, sistemul de prevenire a intruziunilor poate să blocheze în mod automat adresa IP a utilizatorului ce încearcă să facă atacul, nepermițându-i acestuia să se mai conecteze la serverul destinație pentru un anumit interval de timp sau până la intervenția unui administrator).

- O altă abordare posibilă în momentul declanșării unui eveniment malițios este alterarea traficului astfel încât să elimine conținutul malițios din acesta. Pentru realizarea acestui lucru, un sistem de prevenire a intruziunilor poate să șteargă atașamente infectate din interiorul unui mail, să altereze conținutul unui pachet sau să omită transmiterea mai departe a unor pachete.

Capitolul 5

Analiză și proiectare

În acest capitol este descris design-ul proiectului și cuprinde: cerințele sistemului, specificațiile cazurilor de utilizare, arhitectura sistemului, comportamentul sistemului, datele utilizate de sistem, dependențele sistemului și algoritmi esențiali și metodele folosite. Descrierea acestora se realizează prin asocierea cu diagramelor aferente.

5.1 Cerințele sistemului

Sistemul propus *Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea* reprezintă un software gratis care oferă clientului atât caracteristici specifice unui reverse proxy, cât și modalități de protecție asemeni unui sistem de prevenire a intruziunilor. Sistemul este ușor de instalat și de utilizat, chiar și de către utilizatorii neexperimentați, oferindu-le acestora o interfață clară și sugestivă, ce maschează logica complicată din spate. În spate, sistemul oferă un reverse proxy ce interceptează tot traficul destinat către un anumit server, de pe una sau mai multe interfețe. În procesul de interceptare, acesta implementează și câteva modalități de prevenire a unor tentative de intruziune. Sistemul blochează toți clienții ce folosesc adrese IP aflate pe o listă neagră (IP-urile utilizate frecvent de rețeaua Tor) și request-uri ce prezintă un URI cu conținut malițios (atacurile de SQL injection).

Sistemul trebuie să îndeplinească următoarele cerințe **funcționale**:

1. Să realizeze conexiunea la un server HTTP/HTTPS și să redirecționeze traficul primit către acesta.
2. Să intercepteze traficul venit pe o anumită interfață și port prestabilit.
3. Să prelucreze request-urile primite de la clienți într-un format specific pentru clasificatorul de URI-uri.
4. Să nu redirecționeze request-urile clasificate că și malițioase.

5. Să blocheze conectarea clienților ce folosesc IP-uri conținute de lista neagră de referință.
6. Să permită utilizatorului să editeze și să vizualizeze lista adreselor IP blocate.
7. Să prezinte în interfața grafică toate intervențiile realizate asupra traficului (blocări de conexiuni sau de request-uri).
8. Să permită utilizatorului să configureze modul de operare al sistemului.

Sistemul trebuie, de asemenea, să aibă următoarele caracteristici **non-funcționale**:

1. Să fie ușor de instalat și de folosit pentru orice utilizator, oricât de neexperimentat.
2. Să poată intercepta traficul de pe orice/oricâte interfețe disponibile.
3. Să poată rula pe orice sistem de operare Windows cu Python2 instalat.
4. Să aibă o rată de blocare de 100% a IP-urilor de pe lista neagră, iar în cazul detecției de SQL injection să nu aibă detecții false pozitive mai mari 2-3% și o acuratețe generală de peste 90%

5.2 Specificațiile cazurilor de utilizare

5.2.1 Actori, stakeholders si interese

Principalii actori și stakeholder-i sunt administratorii de servere, respectiv de baze de date. În implementarea sistemului propus se urmărește satisfacerea nevoilor acestor persoane, oferindu-le un plus de securitate asupra datelor ce sunt accesate de către clienți, respectiv împotriva clienților rău intenționați. Interesele acestor comunități de utilizatori sunt urmărite pentru a livra un produs care să ofere aceste protecții într-o manieră cât de prietenoasă pentru utilizator și cât mai eficientă.

5.2.2 Basic flow

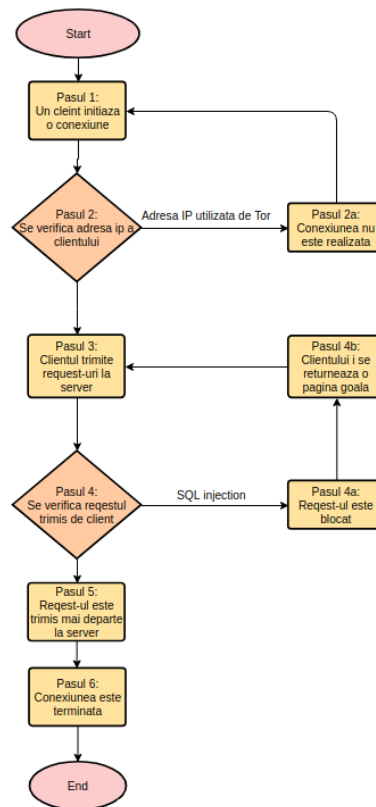


Figura 5.1: Basic flow pentru evenimente

Figura 5.1 prezintă cum arată basic flow-ul pentru evenimentele din sistemul propus.

Evenimentele sistemului:

1. **Start:** aceste cazuri de utilizare sunt inițiate în momentul în care sistemul este plasat între un server și utilizatorii acestuia.
2. **Pasul 1:** un client dorește și inițializeze o conexiune la server.
3. **Pasul 2:** adresa IP a clientului este verificată că acesta să nu fie un utilizator de Tor.
4. **Pasul 3:** clientul comunică cu serverul prin rețet-uri individuale.
5. **Pasul 4:** rețet-urile sunt verificate că acestea să nu fie atacuri de SQL injection.
6. **Pasul 5:** request-urile sunt trimise mai departe către server.
7. **Pasul 6:** conexiunea este terminată(de către client sau server).
8. **End:** la sfârșitul unui șir de evenimente, utilizatorul sistemului poate să vadă dacă clientul ce a inițializat evenimentele a realizat acțiuni considerate ca malițioase.

5.2.3 Alternative flow

Evenimentele alternative în cazurile de utilizare sunt următoarele:

1. **Pasul 2a:** în cazul în care un utilizator cu adresă IP de Tor încearcă să realizeze conexiunea la server aceasta este refuzată.
2. **Pasul 4a:** în cazul în care un utilizator încearcă să trimită la server un request ce reprezintă o tentativă de atac SQL injection, acesta este blocat.
3. **Pasul 4b:** pentru request-urile blocate ca și SQL injection clientului i se returnează o pagină goală.

5.3 Arhitectura sistemului si modulele principale

În conformitate cu cerințele sistemului și specificațiile cazurilor de utilizare, sistemul propus este alcătuit din module specifice care să trateze în mod individual problemele majore ridicate de sistem, printr-o implementare modulară, oferind astfel ușurința implementării de noi funcționalități.

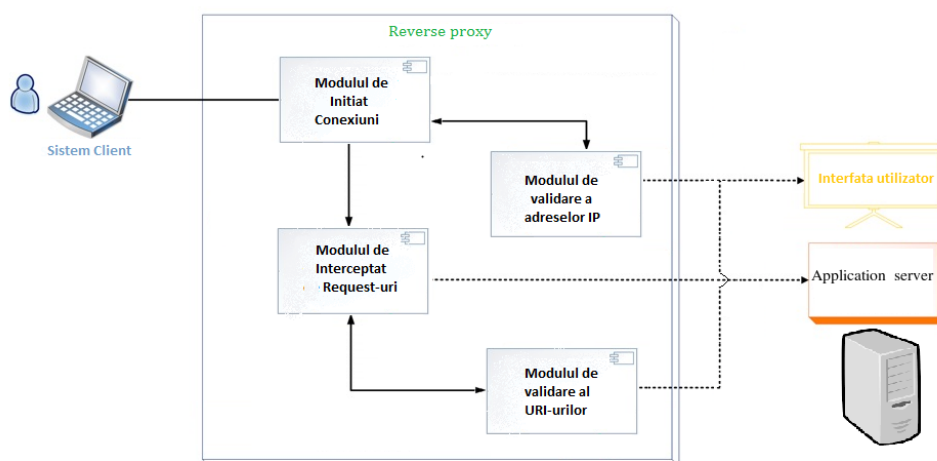


Figura 5.2: Principalele module ale sistemului propus

Figura 5.2 prezintă care sunt principalele module ale sistemului propus, precum și interacțiunea dintre acestea.

Principala și teoretic singura componentă a sistemului este reprezentată de cea de **reverse proxy**, însă pentru a obține rezultatele dorite și pentru a încorpora funcționalitățile de protecții în aceasta, logică ei a fost împărțită în cele 4 mari submodule: **modulul de inițiat conexiuni** și **modulul de interceptat request-uri**, ce reprezintă componentele ce suportă logica unui reverse proxy și **modulul de validare a adreselor IP** și **modulul de validare a URI-urilor**, ce reprezintă modulele care interacționează cu cele ce implementează structura de reverse proxy pentru a oferi funcționalitățile de securitate.

Sistemul Client și application server.

Aceste două componente reprezintă componentele clasice între care vin plasat sistemul propus. **Sistemul client** este reprezentat de orice client dorește să acceseze baza de date/partea de server a unei aplicații. **Application server** reprezintă serverul aplicație la care se pot conecta clienții pentru a avea acces la un anumit conținut. Sistemul propus are rolul de intermediere între cele două tipuri de componente, prevenind astfel eventuale tentative de exploatare a unor vulnerabilități din partea clientului către server.

Modulul de inițiat conexiuni.

Acest modul este constituit din componentele oferite de biblioteca open source **twisted**, fiind modificate ulterior pentru a permite integrarea modului **Modulul de validare a adreselor IP**. Modulul are rolul de a crea un socket pe sistemul pe care rulează acesta, care să asculte pentru posibile conexiuni. În cazul în care un client încearcă să inițieze o nouă conexiune acesta transmite adresa IP a clientului către modulul **Modulul de validare a adreselor IP**, iar în funcție de răspunsul acestuia conexiunea acestuia va fi realizată sau refuzată.

Modulul de interceptat request-uri.

Asemeni modului anterior, **Modulul de inițiat conexiuni**, acest modul este construit peste codul oferit de biblioteca open source **twisted**. Biblioteca oferă suport pentru interceptarea request-urilor dintre un client și server, implementându-se o logică suplimentară ce încorporează modulul de detecție a atacurilor SQLi (**Modulul de validare a URI-urilor**). În acest modul se interpretează natura URL-urilor trimise de către client către server. Interpretarea constând în analiza URI-ului unui request de către modulul de validare a request-urilor, iar în funcție de rezultatul acestuia returnând răspunsuri corespunzătoare atât clientului cât și serverului (în caz de tentativă de atac, la server nu va ajunge nimic).

Modulul de validare a adreselor IP.

Rolul acestui modul este de a valida o adresă IP în conformitate cu adresele conținute pe lista neagră de referință a sistemului. La pornirea sistemului modulul încarcă în memorie o lista cu toate adresele IP care trebuie să fie blocate. Această lista este construită la pornirea sistemului pe baza unor date salvate local și reactualizate periodic. Pentru validarea unei adrese, în cazul în care această se află în lista cu IP-uri blocate modulul returnează valoarea "false" codului apelant, iar în caz contrar valoarea "true".

Modulul de validare a URI-urilor.

Modulul de validare a URI-urilor este constituit din modelul de detecție a atacurilor SQL injection. Logica modelului de detecție a atacurilor SQL injection este împărțită în două componente. În prima parte acesta identifică trăsăturile specifice unui atac SQL injection într-un URI. Aceste trăsături sunt reprezentate de caractere și cuvinte cheie ce pot fi găsite în acesta. În cazul identificării unor astfel de trăsături, URI-ul este pasat mai departe celei de a doua componentă. Aceasta din urmă este reprezentată de un model de support vector machine antrenat anterior, care va decide dacă URI-ul primit se încadrează în cele specifice atacurilor de SQL injection sau nu. Asemeni modului anterior **Modulul de validare a adreselor IP**, în funcție de rezultatul obținut acesta va returna "true" sau

”false” codului apelant.

5.4 Comportamentul sistemului. Diagrama de stare si de secventa

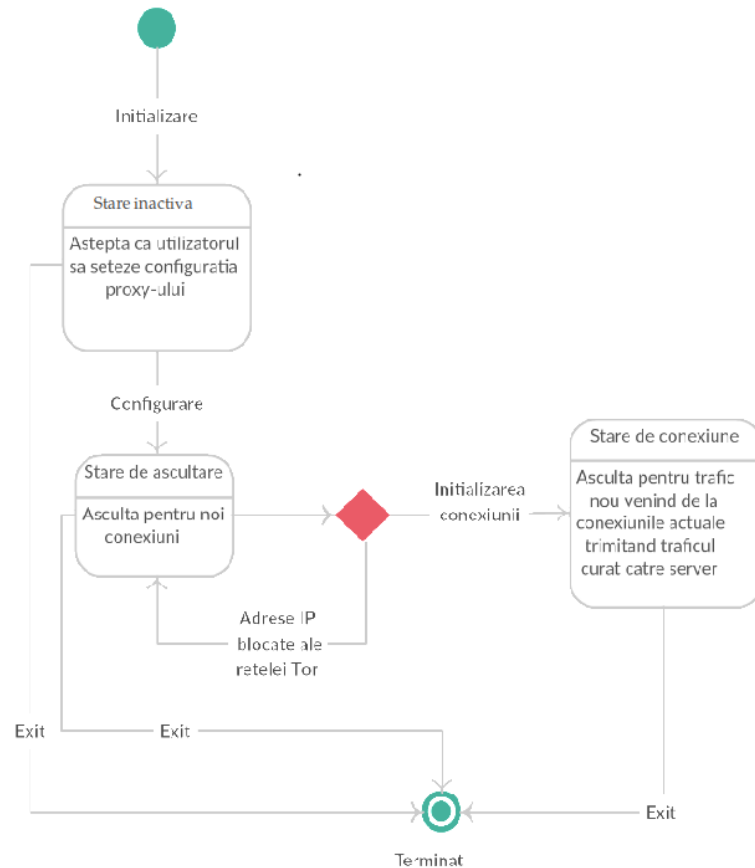


Figura 5.3: Diagrama de stare a sistemului propus.

Figura 5.3 prezintă diagrama de stare a sistemului propus.

După pornirea aplicației reprezentată prin inițializare, utilizatorului i se pune la dispoziție interfața de utilizator în care sistemul așteaptă să fie configurat și pornit("Stare inactivă"). După configurarea aplicației în meniul "Configure", prin setarea specificațiilor serverului de reverse proxy, prin apăsarea butonului "start", sistemul trece în următoarea stare "Stare de ascultare". În această stare sistemul acceptă noi conexiuni de la diferiți clienți ce doresc să acceseze server-ele protejate de acesta. Tot aici are loc și verificarea adreselor IP ale clienților pentru protejarea împotriva utilizatorilor Tor. După stabilirea

unei noi conexiuni, sistemul intră în starea "Stare de conexiune", în care acesta ascultă pentru noi request-uri între client și server și le transmite mai departe către destinația dorită. Tot aici are loc și validarea request-urilor trimise de clienți către server pentru prevenirea atacurilor SQL injection. În această stare utilizatorul poate să urmărească activitatea sistemului în meniul "Monitor" din interfața de utilizator, unde sunt logate toate evenimentele generate de sistem. Pentru oprirea aplicației, utilizatorul poate pur și simplu să o închidă de la butonul din drapta sus, aceasta terminându-și toată activitatea indiferent de starea în care se află.

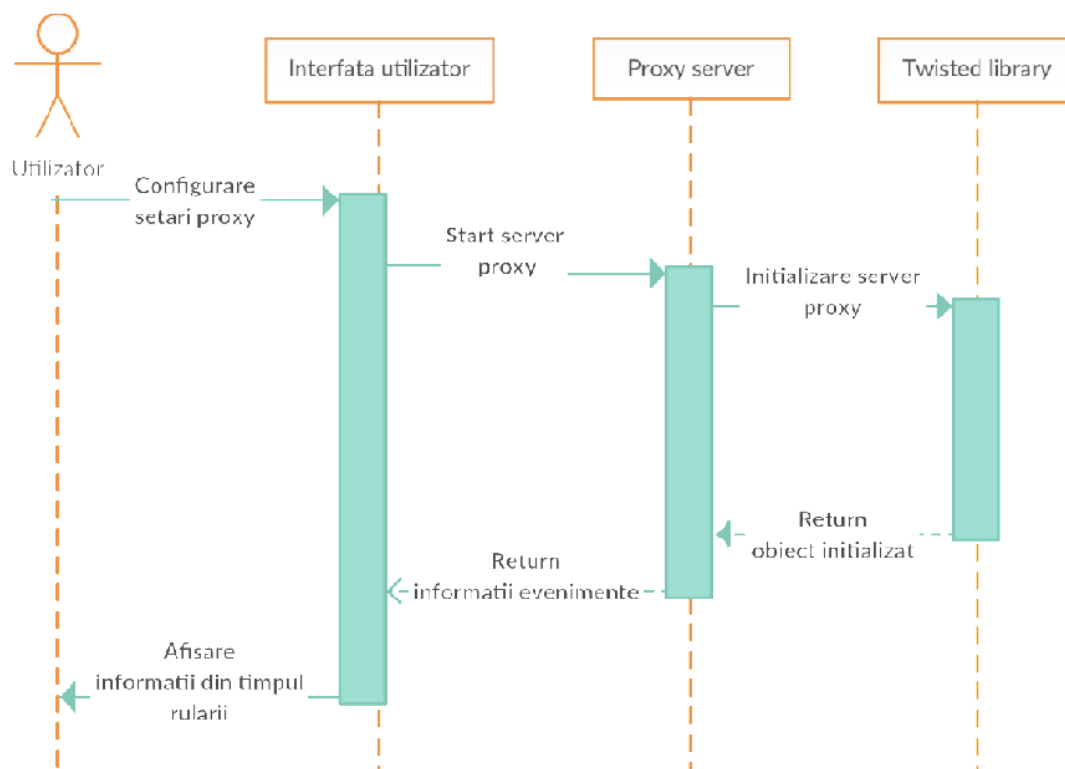


Figura 5.4: Diagrama de secvență a sistemului propus.

Figura 5.4 prezintă diagrama de secvență a sistemului propus.

După pornirea sistemului, utilizatorul configurează setările pentru server-ul de reverse proxy din interfața de utilizator (în meniul "Configure"). După configurarea sistemului, prin apăsarea butonului de start, este lansată în execuție server-ul de reverse proxy cu caracteristicile specificate de utilizator. Scriptul de pornire a server-ului de reverse proxy face apel la fișierele bibliotecii Twisted, care vor returna către acesta o instanță a server-ului inițializat. În interiorul server-ului de reverse proxy, apariția de noi evenimente este semnalată către interfața grafică. Aceste evenimente sunt puse la dispoziția utilizatorului pentru vizualizare.

5.5 Dependințele sistemului

Pentru funcționarea corectă a sistemului propus, acesta are nevoie de următoarele dependențe:

1. Sistem de operare Windows cu Python2 instalat pe sistem.
2. Conexiune stabilă la internet pentru rularea script-urilor ce obțin adresele IP utilizate de Tor.
3. Sistemul să aibă acces direct la interfețele prin care clienții se pot conecta la server-ul protejat
4. Sistemul să aibă o conexiune directă la server-ul ce trebuie protejat.

5.6 Algoritmi si metode

Pentru prevenirea atacurilor de SQL injection s-a folosit un algoritm de machine learning, support vector machine. Acest algoritm presupune antrenarea anterioară a unui model cu un set de date de referință etichetate în prealabil corect de către utilizator. Aceste date sunt folosite de algoritm pentru a raporta noi seturi de date, natura acestora fiind necunoscută, încercând să găsească similitudini între cele noi și cele de referință, determinându-le astfel natură celor noi. Algoritmul de support vector machine rezultă într-un model obținut prin folosire a diverși algoritmi pentru antrenarea acestuia, folosit pentru a clasifica date.

Realizarea unui astfel de model sa realizat în urmă executării unui proces elaborat ce implică mai mulți pași:

- Primul pas reprezintă identificarea datelor relevante în cea ce privește problema tratată(prevenirea atacurilor SQL injection). În conformitate cu scopul clasificării evenimentelor/datelor în două categorii (tentativă de atac și request-uri clean) inițial au fost identificate o serie de astfel de evenimente și categorizate în evenimente ce sigur aparțin fiecărei dintre categoriile țintă.
- După obținerea datelor de antrenare, au fost identificate toate trăsăturile relevante din aceste date, trăsături care să fie cât se poate de specifice fiecărei categorii în parte. Aceste trăsături au reprezentat cuvintele cheie a limbajului SQL dar și caractere specifice folosite frecvent de acesta.
- După obținerea trăsăturilor specifice datelor de antrenare, sa realizat antrenarea modelului folosind un algoritm specific. În cazul proiectului propus s-a folosit algoritmul

gata implementat, furnizat de biblioteca open source LIBSVM [18]. Pentru obținerea modelului datele de antrenare au fost procesate folosind un kernel gaussian. Un kernel gaussian reprezintă modul în care modelul procesează datele de antrenare astfel încât clasificarea noilor date să fie realizată prin calcularea similarităților dintre acestea și cele de antrenare. În calcularea similarității dintre aceste două tipuri de date, un parametru foarte important este sigma. Acest parametru este ales pentru întreg setul de date, iar valoarea lui este direct proporțională cu gradul de similaritate pe care algoritmul îl va asocia la două evenimente/date diferite.

Pentru blocarea IP-urilor utilizate de rețeaua Tor s-a folosit un script scris în Python3. Programul interoghează periodic(din 6 în 6 ore) informațiile oferite de *Tor Network Status* [26], salvând uptime-ul fiecărei adrese din ultimele 6 ore. Identificarea adreselor malițioase se face prin însumarea timpului total din ultima lună de uptime, iar cele cu o valoare mai mare de 7 zile sunt considerate malițioase. Blocarea IP-urilor se realizează prin compararea cu o astfel de lista generată lunar.

5.7 Diagrama de desfășurare

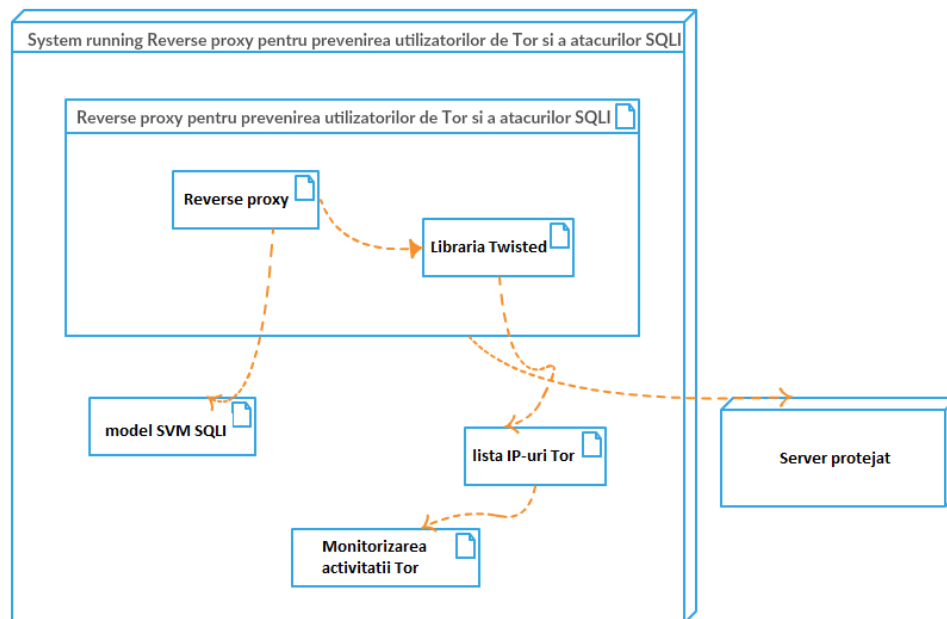


Figura 5.5: Diagrama de desfășurare a sistemului propus.

Figura 5.5 prezintă diagrama de desfășurare a sistemului propus. Pentru a putea rula sistemul *Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea* pe un sistem, acesta trebuie să aibă acces direct la server-ul ce se dorește a

fi protejat. Sistemului trebuie să îi fie furnizate modelul de SVM folosit pentru prevenirea atacurilor de SQL injection. De asemenea scriptul pentru monitorizarea activității rețelei Tor și cel pentru generarea listei de adrese IP cu un uptime mai mare de 7 zile, trebuie să se afle pe sistem, pentru ca acesta să utilizeze date "up-to-date".

5.8 Justificarea design-ului

Toate deciziile de design au fost luate în vederea obținerii unor funcționalități cât mai importante pentru utilizatorii țintă, dar și pentru a obține o eficiență cât mai mare a sistemului, atât din punct de vedere al corectitudinii cât și a timpului de execuție.

Pentru o acuratețe cât mai mare în prevenirea atacurilor de SQL injection s-a ales folosirea unui abordări bazate pe machine learning, întrucât detecțiile statice prezintă o logică foarte complicată, pot omite multe cazuri esențiale și sunt limitate de capacitatea de observare a programatorului.

Pentru identificarea adreselor utilizate de rețeaua Tor se putea aborda o implementare cu lista fixă, întrucât aceste liste prezintă un set mic de adrese ce rămân neschimbate în timp, însă pentru o acuratețe cât mai mare, s-a ales folosirea celor două scripturi suplimentare ce vor obține periodic toate adresele utilizate de rețea în perioada respectivă de timp.

De asemenea dezvoltarea modulară a sistemului asigură ușurința de adăugare de noi module sau funcționalități fără a necesita modificări majore sistemului actual.

Capitolul 6

Detalii de implementare

6.1 Structura codului sursa

Codul folosit pentru obținerea sistemului *thesistitle* este împărțit în 3 proiecte separate:

- **Tor activity monitor** ce constituie logica necesară obținerii listei de adrese IP blocate de sistem.
- **SQLi SVM** scopul acestuia fiind obținerea modelului de SVM folosit pentru prevenirea atacurilor de SQL injection.
- **Implementarea unui reverse proxy pentru prevenirea atacurilor la nivel de rețea** reprezentând sistemul propus și încorporează rezultatele obținute de celelalte două proiecte.

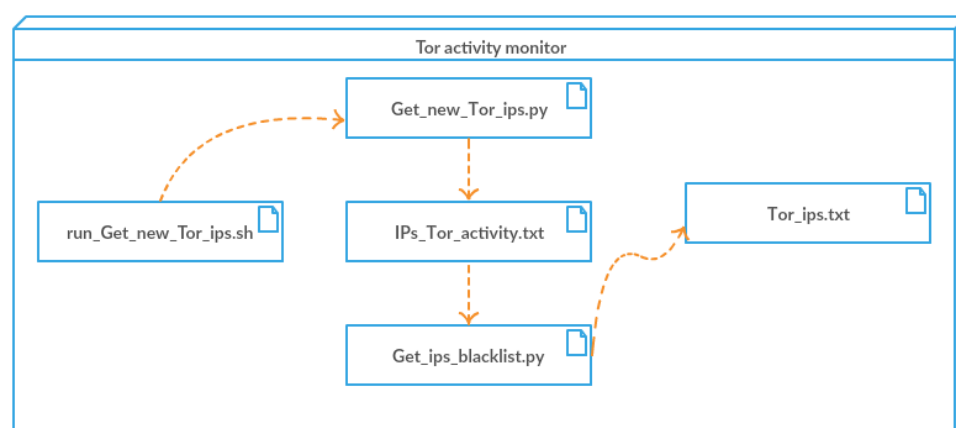


Figura 6.1: Modulul pentru monitorizarea activității rețelei Tor

Figura 6.1 prezintă care sunt fișierele folosite pentru monitorizarea activității rețelei Tor și pentru obținerea listei cu adresele IP ce trebuie blocate de către sistem și relațiile dintre acestea.

run_Get_new_Tor_ips.sh este un fișier de bash ce are rolul de a rula **Get_new_Tor_ips.py**. Fișierul este programat să lanseze în execuție **Get_new_Tor_ips.py** la ore fixe, acesta rulând încontinuu pe un sistem cu acces la internet neîntrerupt. Lansările în execuție au loc o dată la 6 ore, respectiv la oră 12 am și pm și 6 am și pm.

Get_new_Tor_ips.py are rolul de a documenta modificările de uptime din ultimele 6 ore ale nodurilor rețelei Tor. Datele sunt furnizate de pe pagina "Tor Network Status" [26] și pentru fiecare adresă IP prezentă în date, se verifică care este uptime-ul din ultimele 6 ore, informațiile acestea fiind stocate în **IPs_Tor_activity.txt**.

IPs_Tor_activity.txt are rolul de a stoca informațiile despre toate adresele IP utilizate de rețeaua tor din ultima luna. Acestea sunt salvate în liste, pentru fiecare adresă IP în parte o listă.

Get_ips_blacklist.py are rolul de a genera fișierul **Tor_ips.txt** folosit de către sistemul propus pentru blocarea adreselor IP. Acesta folosește datele din interiorul fișierului **IPs_Tor_activity.txt**, pentru a genera o listă cu toate adresele IP ce au un uptime total mai mare de 7 zile în ultimele 30 de zile.

Tor_ips.txt reprezintă rezultatul proiectului. Acesta este alcătuit dintr-o listă formată din toate adresele IP ce vor fi blocate de sistemul propus în timpul rulării.

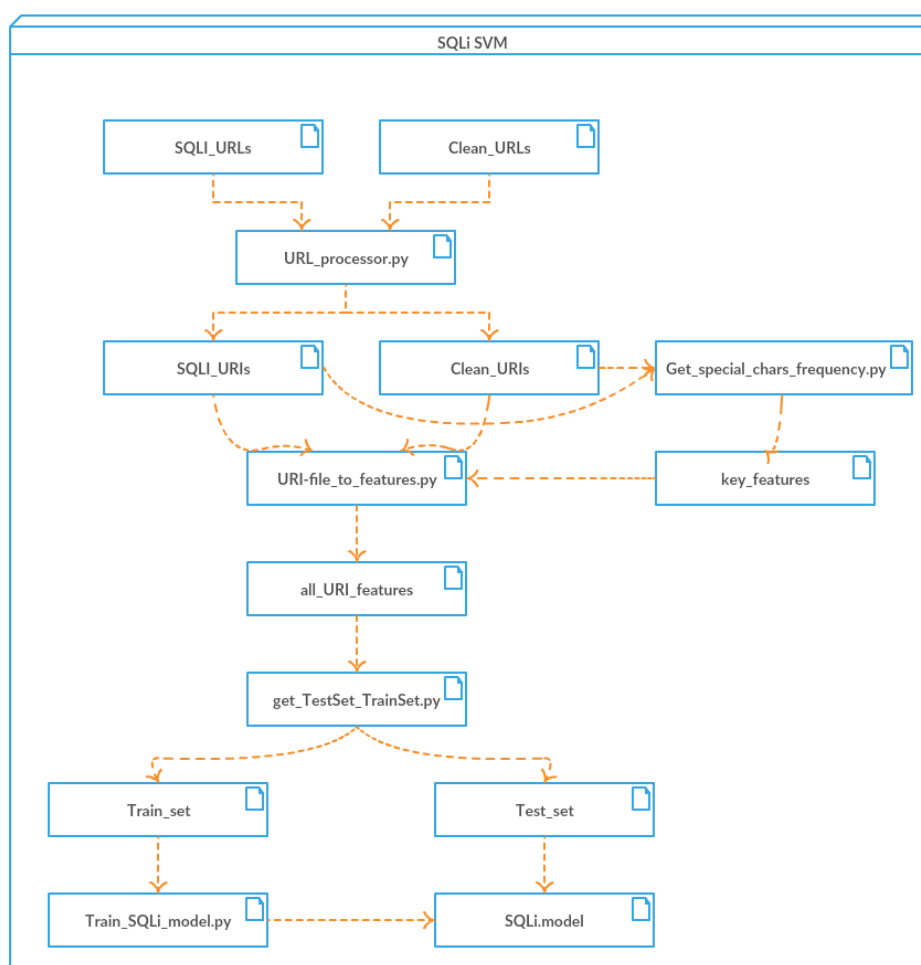


Figura 6.2: Modulul pentru prevenirea atacurilor SQL injection

Figura 6.2 prezintă care sunt fișierele utilizate pentru procesarea datelor necesare în procesul de antrenare a modelului de SVM, până la obținerea modelului propriu-zis și care sunt relațiile dintre acestea.

SQLURLs reprezintă setul inițial de date "infected" folosite pentru identificarea atacurilor SQL injection. Acest set este constituit din URL-uri ce au fost identificate de un produs autorizat ca fiind tentative de SQL injection.

Clean_URLs reprezintă setul inițial de date "clean" folosite pentru identificarea atacurilor SQL injection. Acest set este constituit din URL-uri ce au fost identificate de un produs autorizat ca fiind URL-uri curate.

URL_processor.py primește ca input un fișier sau string și are rolul de a procesa URL-uri într-un format uniform (se elimină encodările) și specific pentru pașii următori.

SQLURLs constituie noua listă rezultată din procesarea fișierului SQLURLs de către URL_processor.py.

Clean_URLs constituie noua lista rezultată din procesarea fișierului `Clean_URLs` de către `URL_processor.py`.

Get_social_chars_frequency.py are rolul de a calcula frecvența de apariție a unor caractere speciale în cele două seturi de date și de a decide în funcție de frecvența lor de apariție, care din acestea sunt relevante în vederea alegerii trăsăturilor de clasificare.

key_features este o listă alcătuită din toate cuvintele cheie a limbajului SQL, dar și din caracterele speciale utilizate în acesta și considerate ca fiind relevante în urma execuției scriptului `Get_social_chars_frequency.py`.

URI-file-to-features.py are rolul de a procesa cele două fișiere de date și pe baza trăsăturilor din `key_features` să constituie un nou fișier ce conține pentru fiecare URL din cele două fișiere tipul acestuia și trăsăturile găsite în el, precum și frecvența lor.

all_URI_features este rezultatul rulării scriptului `URI-file-to-features.py` și conține pentru fiecare URL din cele două fișiere de date, tipul acestuia și trăsăturile găsite în el, precum și frecvența lor, acestea fiind folosite pentru antrenarea și testarea modelului de SVM.

get_TestSet_TrainSet.py are rolul de a împărți datele prezente în `all_URI_features` în două seturi de proporție 70-30. Aceste două seturi fiind folosite pentru antrenarea și testarea modelului.

Train_set constituie 70% din totalul de exemple acumulate pentru antrenarea modelului, doar acestea fiind de fapt folosite pentru antrenarea sa.

Test_set constituie 30% din exemplele acumulate, aceste date fiind folosite pentru testarea acurateții modelului după antrenare.

Train_SQLi_model.py realizează obținerea modelului de SVM folosit de sistem pentru prevenirea atacurilor SQL injection. Pentru antrenare modelului este folosit setul de date din fișierul `Test_set` și algoritmi puși la dispoziție de biblioteca open source `libsvm` [18].

SQLi.model reprezintă rezultatul proiectului. Acesta este testat cu ajutorul setului de date din fișierul `Test_set` și ulterior integrat în sistemul propus pentru a fi folosit pentru prevenirea atacurilor SQL injection.

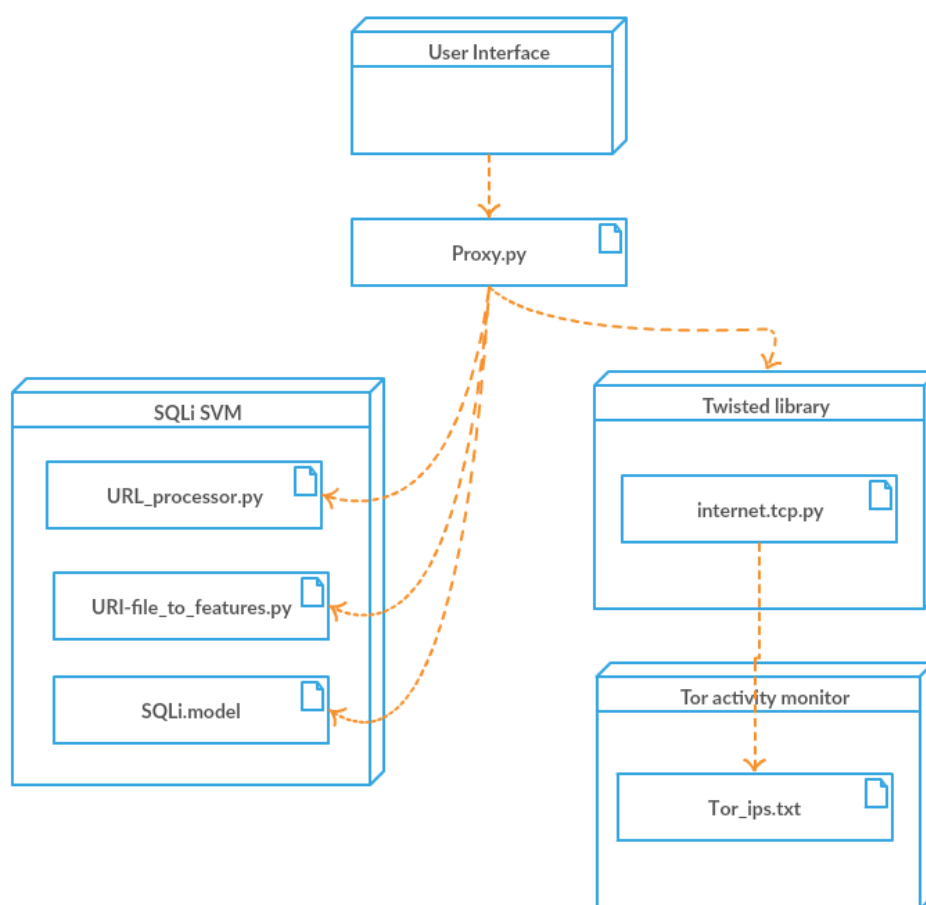


Figura 6.3: Interacțiunea dintre interfața grafică și celelalte module

Figura 6.3 prezintă care sunt fișierele, specific fiecărui modul, care sunt accesate în mod direct de către interfață sau indirect de alte fișiere, în timpul rulării și relațiile dintre acestea.

User Interface reprezintă întreaga componentă ce realizează interfața de utilizator cu toate fișierele necesare realizării ei incorporate în compoziția sa. Aceasta a fost realizată ca un proiect dezvoltat în .Net implicând multe fișiere cu scopul realizării elementelor grafice. Aceste elemente nu vor fi tratate în această secțiune.

Proxy.py reprezintă fișierul ce încorporează, respectiv leagă, tot codul ce constituie partea de "back end" a proiectului. În acest script de python se realizează instanțierea elementului de reverse proxy cu parametri de adrese IP și porturi aferente, precum și furnizarea metodelor de detecție componentei de reverse proxy.

Twisted library este o bibliotecă open source scrisă în Python, ce oferă suport pentru diferite protocoale (TCP, UDP, SSL/TLS). Biblioteca a fost folosită pentru partea de cod ce oferă implementarea unui reverse proxy. Mare parte din fișierele oferite de acesta

biblioteca au fost folosite fără a fi suprascrise sau a li se aduce modificări ulterioare, însă în vederea atingerii scopului propus, asupra unor fișiere sursă au fost aduse mici modificări (internet.tcp.py).

internet.tcp.py este scriptul din biblioteca twisted ce oferă suportul pentru protocolul TCP. Acest fișier a fost modificat pentru introducerea detecției împotriva utilizatorilor de Tor. Script-ul integrează lista realizată de proiectul "Tor activity monitor" pentru verificarea adresei utilizatorilor ce doresc să stabilească o conexiune TCP.

6.2 Algoritmi, metode si API-uri

În acesta secțiune se urmărește descrierea codului sursă folosit la realizarea sistemului propus și explicarea amănunțită a codului/metodelor considerate mai relevante, precum și a principalelor api-uri utilizate în implementarea acestuia. Abordarea codului sursă se realizează conform sub-proiectelor prezentate în secțiunea anterioară (Tor activity monitor, AQLi SVM, Interfața utilizator).

6.2.1 Tor activity monitor

Conform figurii 6.1, acest proiect este realizat din 5 fișiere, acestea având o relație liniară între ele.

Fișierul ce începe ciclul de execuție, `run_Get_new_Tor_ips.sh`, este un fișier de bash ce rulează în buclă infinită. Fișierul trebuie să ruleze pe un sistem ce este funcțional non-stop și cu acces nelimitat la internet. La ore fixe (12 am și pm și 6 am și pm), acesta lansează în execuție scriptul de python `Get_new_Tor_ips.py`.

Fișierul principal din acest proiect îl reprezintă `Get_new_Tor_ips.py`. Acesta descarcă pagina "Tor Network Status" [26] și procesează datele de pe acestea, introducând în fișierul `IPs_Tor_activity.txt` informații referitoare la adresele IP găsite pe pagină și uptime-ul lor din ultimele 6 ore. Procesarea adreselor IP și extragerea valorii lor de uptime se poate observa în următoarele două secvențe de cod.

```
time_up = row.findAll('td')[4].contents[0]
ip = row.findAll('td', attrs={'class': 'iT'})[0].findAll('a',
    attrs={'class': 'who'})[0].contents[0]
```

Pentru prelucrarea conținutului paginii, acesta a fost download-at în memoria programului, iar codul HTML rezultat a fost prelucrat cu ajutorul bibliotecii de python open source, BeautifulSoup. Codul de mai sus reprezintă extragerea valori de timp (uptime) și adresa IP căreia aceasta corespunde. Variabila "row" fiind un element din obiectul iterabil rezultat din inițializarea bibliotecii BeautifulSoup cu codul HTML al paginii.

```
days = time_up.split()[1]
```

```
if days == 'd':
    hours = 6
else:
    hours = int(time_up.split()[0])
    if hours > 6:
        hours = 6
```

În secvența de mai sus de cod, este identificata valoarea corectă de uptime din ultimele 6 ore. Pentru cazul în care valoare de uptime este sub formă de zile sau aceasta este mai mare de 6 ore, ea se setează pe 6 ore, întrucât nu ne interesează decât activitatea din ultimele 6 ore.

```
from bs4 import BeautifulSoup
from urllib.request import urlopen

pagesource = urlopen(page)
soup = BeautifulSoup(pagesource.read())
table = soup.findAll('table', attrs={'class': 'displayTable'})
```

Pentru procesarea conținutului unei pagini web("Tor Network Status" [26]) s-au folosit bibliotecile de python, open source, urllib și BeautifulSoup [17]. Biblioteca urllib oferă funcții și clase ce pot fi folosite pentru deschiderea de URL-uri (urlopen). Funcția folosită în codul de mai sus primește ca parametru "page" adresa sub formă de string a paginii ce se dorește a fi citită. Biblioteca BeautifulSoup este o bibliotecă de python folosită pentru extragerea de date din fișiere de HTML sau XML. Acesta este inițializată cu un document HTML/XML și că rezultat oferă un obiect ce permite navigarea prin codul sursă asemeni unei structuri de date imbricate. Astfel, spre exemplu, pentru identificarea structurii corespunzătoare tabelului de adrese IP din codul sursă al paginii s-a folosit o singură linie de cod, în care s-au specificat numele și attributele specifice acestuia.

Rezultatele rulării fișierului Get_new_Tor_ips.py sunt actualizate în IPs_Tor_activity.txt. Acest fișier este de fapt un json în care se realizează dump la noul dicționar obținut de Get_new_Tor_ips.py. Dicționarul este constituit din adresa IP ca și cheie și o listă. Structura acestor liste este realizată dintr-o serie de numere între 0 și 6 ce reprezintă timpul total de uptime corespunzător sfertului respectiv de zi. Dimensiunea acestei liste este fixată la 30 (zile) *4 (sferturi de zi), pe măsură ce un element nou este adăugat, primul element din lista fiind scos.

Următorul pas este filtrarea tuturor adreselor IP ce au un uptime mai mare de 7 zile. Acest lucru este realizat de scriptul Get_ips_blacklist.py, iar rezultatele sunt stocate în Tor_ips.txt, o adresă IP pe linie.

6.2.2 SQLi SVM

Desfășurarea proiectului începe de la cele două fișiere SQLi_URLs si Clean_URLs. În aceste două fișiere se află URL-uri complete din categoria conformă cu numele fiecărui fișier. Aceste fișiere sunt procesate de scriptul URL_processor.py care are rolul de a uniformiza datele în aceeași encodare. Următoarea bucată de cod prezintă secvența ce transformă valorile hexa dintr-un URI în caractere și modul în care erorile de encodare sunt tratate (sunt afișate pentru a fi tratate manual de către programator):

```
for index, sub_uri in enumerate(uri.split('%')):
    if sub_uri:
        if index == 0:
            new_uri = sub_uri
            continue
        try:
            hex_val = bytearray.fromhex(sub_uri[:2]).decode()
        except UnicodeDecodeError:
            print(uri + ' --- ' + sub_uri[:2])
            return ''
        new_uri += hex_val + sub_uri[2:]
```

Întrucât într-un URL caracterul '%' nu poate să apăra decât dacă acesta este encodat ('%25'- valoarea pentru encodarea caracterului '%'), identificarea tuturor caracterelor encodate a fost făcută prin identificarea tuturor caracterelor de tipul '%' în URI. În cazul în care un astfel de caracter este găsit, se încearcă conversia următoarelor două caractere (așteptându-se, conform convenției, să fie cifre sau litere de la A la F), din valoarea în hexa corespunzătoare unui anumit caracter în caracterul în sine.

În urma procesării datelor, rezultă cele două fișiere SQLi_URLs si Clean_URLs, acestea având același conținut ca cele anterioare însă în aceeași encodare. În urma obținerii acestor două fișiere, a fost realizată completarea listei de trăsături (lista inițială este constituită din toate cuvintele cheie a limbajului SQL) cu caracterele speciale întâlnite în acest limbaj. Scriptul Get_social_chars_frequency.py are rolul de a determina frecvența de apariție a fiecărui caracter special în cele două seturi de date, iar pe baza unei observații umane, a fost realizată determinarea caracterelor ce constituie trăsături rentabile. Următoarea bucată de cod este din scriptul Get_social_chars_frequency.py și realizează numărarea URI-urilor(pentru comparare) din setul de date și frecvența caracterelor speciale("dict" este un dicționar cu cheile fiind caracterele speciale utilizate în limbaj):

```
with open(args.f, 'r') as fd:
    for lines in fd.readlines():
        lines_nr += 1
        line = lines.strip()
        for keys in dict:
```

```

if keys in line:
    dict[keys] += 1

```

Pentru identificarea caracterelor relevante din limbajul SQL în comparație cu cele folosite într-un URL obișnuit, s-a ales numărarea frecvenței de apariție a acestora în URL-uri normale, dar și în URL-uri ce conțin atacuri de SQL injection. Numărarea se face alternativ, rezultatele fiind furnizate ca două seturi de date separate, bucata anterioară de cod reprezentând procesul doar pentru una dintre cele două categorii. Dicționarul referit în cod este alcătuit dintr-un dicționar ce are ca și chei valoarea caracterelor speciale căutate, iar ca valoare, acestea sunt inițializate pe zero pentru a fi incrementate o dată cu numărul de apariții ale caracterelor căutate. Pentru uniformizarea setului de date, întrucât distribuția acestor caractere nu este tot timpul uniformă, se ține cont și de numărul de linii (pe fiecare linie se află un URI distinct) în care au fost identificate frecvențele lor de apariție.

După determinarea caracterelor relevante, acestea au fost completate manual în fișierul `key_features`, fișier ce însumează toate trăsăturile ce sunt folosite în antrenarea modelului (fiecare linie conține o trăsătură, numărul liniei fiind și indicele de referință a trăsăturii).

Pentru obținerea datelor într-un format ce poate fi procesat de biblioteca `libsvm` [18] a fost necesară transformarea acestora într-un anumit format:

```

+1 23:1 37:4 103:2
-1 54:1 77:1

```

Conversia URI-urilor în formatul de mai sus este realizată de scriptul `URI-file_to_features.py`. Formatul de mai sus este reprezentarea fiecărui URL în funcție de tipul acestuia și indicele trăsăturii găsite în interiorul sau precum și frecvența de apariție a acestora (tip trăsătură : trăsătură : frecvență). Următoarele bucăți de cod sunt extrasă din fișierul `URI-file_to_features.py` și realizează conversia unui URI în echivalentul sau în trăsături:

```

if keys in uri:
    if keywords_list.index(keys) > 184:
        keywords[keys] = uri.count(keys)
    if not uri.count(keys) == 0:
        ok = True

```

În lista cu trăsături, primele 184 de poziții corespund cuvintelor cheie ale limbajului SQL, următoarele 8 poziții aparținând trăsăturilor de tipul caracter special (determinate la pasul anterior de script-ul `"Get_special_chars_frequency.py"`). În cazul în care o astfel de trăsătură este găsită într-un URI, pur și simplu se găsește numărul total de apariții a caracterului respectiv în URI.

```
sub_uri = uri.split(keys)
for index, ele in enumerate(sub_uri):
    if index + 1 < len(sub_uri):
        if ele and sub_uri[index + 1] and not ele[-1].isalpha()
            and not sub_uri[index+1][0].isalpha():
            keywords_aux[keys] += 1
    elif not ele and sub_uri[index - 1]:
        keywords_aux[keys] += 1
```

Pentru trăsăturile de pe primele 184 de poziții, adică cuvintele cheie ale limbajului SQL, s-a abordat o logică mai complexă de procesare. O dată ce un cuvânt este găsit într-un URI, pentru fiecare apariție a acestui cuvânt se verifică ca primul caracter anterior cuvântului și primul următor cuvântului să nu fie literă, astfel eliminându-se cazurile în care unele cuvinte sunt incluse în altele (ex: **all** și **deallocate**).

În urma procesării fișierelor de date, rezultă un fișier (**all_URI_features**) ce însumează toate datele utilizate în antrenarea modelului (exemple atât pozitive cât și negative), însă sub formă prezentată anterior (tip trăsătură : trăsătură : frecvență). Acest fișier este împărțit în două fișiere cu proporția de 70% pentru **Train_set** și 30% **Tes_set** de scriptul **get_Trainset_Testset.py**. cu scopul de a obține un grup de date atât pentru antrenarea modelului cât și pentru testarea ulterioară a acestuia.

Pentru antrenarea modelului, s-a folosit biblioteca open source **libsvm**. Această bibliotecă furnizează fișierele necesare antrenării modelului de svm și de prezicere a unui rezultat pe un model existent. Apelul către executabilul "svm-train.exe" se face prin intermediul scriptului de python **Train_SQLi_model.py**. Executabilul de windows furnizat de biblioteca **libsvm** poate fi apelat conform exemplului următor:

```
svm-train.exe [options] training_set_file [model_file]
```

6.2.3 Interfață utilizator

Componenta de interfață reprezintă atât proiectul în .net în care a fost realizată interfața sistemului, dar și fișierele/script-urile ce au rolul de a interconecta rezultatele celor două proiecte anterioare cu acesta.

Scriptul **Proxy.py** încapsulează toate elementele, din partea de back end, a proiectului propus. Acesta de asemenea furnizează ca output într-un format specific (format ce este cunoscut și interpretat de interfața grafică) toate evenimentele ce apar în timpul rulării sistemului. În acest script se realizează instanțierea elementului de reverse proxy cu parametri de adrese IP și porturi aferente, precum și furnizarea metodelor de detecție componentei de reverse proxy. Inițializarea elementului de reverse proxy se realizează cu ajutorul codului oferit de biblioteca **twisted**. Acest cod este prezentat în "Anexa A" a acestei lucrări.

În următoarele imagini sunt prezentate posibilele pagini ale interfeței utilizator precum și o scurtă descriere ce ilustrează funcționalitățile oferite de paginile respective pentru utilizator.

]

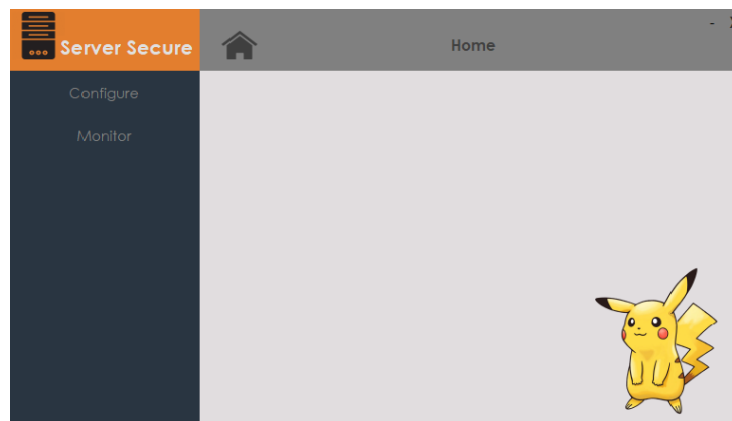


Figura 6.4: Meniul "Home" in interfata grafica

Figura 8.1 prezintă design-ul paginii de "Home" din interfața grafică pusă la dispoziție utilizatorilor sistemului. Acesta pagină este afișată la lansarea în execuție a aplicației, dar poate fi accesată de către utilizator și prin apăsarea butonului în formă de "căsuță" din dreapta logoului aplicației.

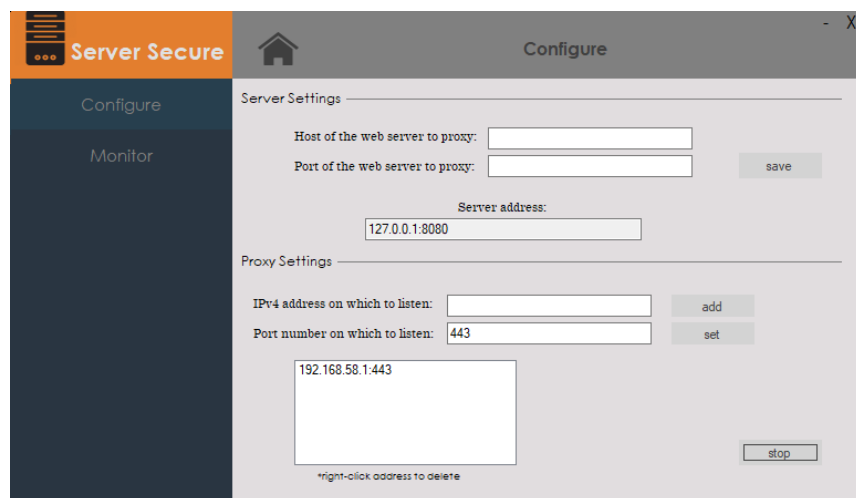


Figura 6.5: Meniul "Configure" in interfața grafica

Figura 8.2 prezintă design-ul paginii de "Configure" din interfața grafică, în care utilizatorul sistemului poate să seteze interfețele și porturile care vor fi tratate în timpul rulării. Partea superioară a meniului reprezintă setările aferente părții de server, precum sugerează și eticheta din stânga sus "Server Settings". În partea inferioară se pot seta detaliile interfețelor ce se pot conecta la server-ul setat mai sus (aceste interfețe pot fi multiple).

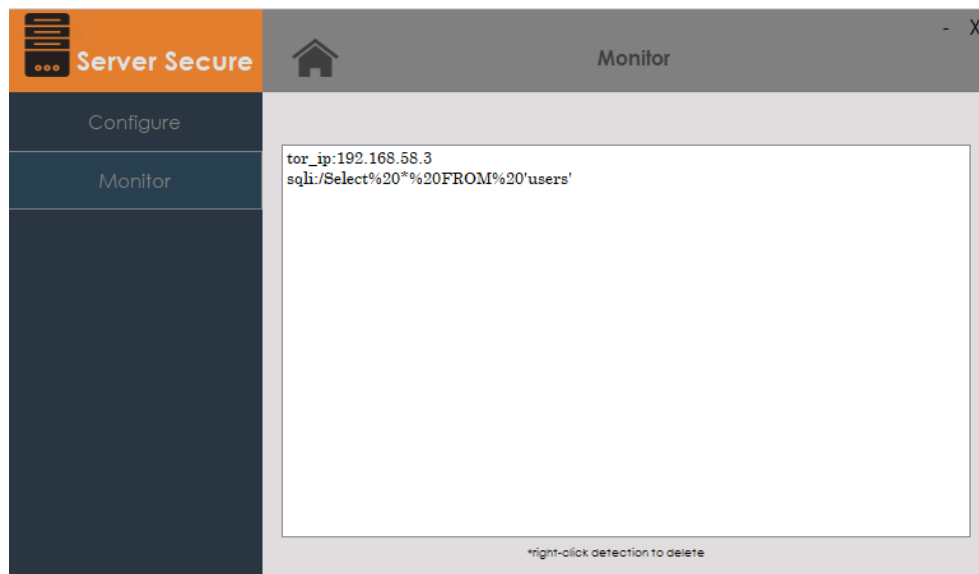


Figura 6.6: Meniul "Monitor" in interfața grafica

Figura 8.3 prezintă design-ul paginii de "Monitor" din interfața grafică, în care utilizatorul poate să urmărească activitatea sistemului în timpul rulării. În această fereastră apar evenimentele apărute în timpul rulării aplicației, evenimente ce sunt sub formă de detecții.

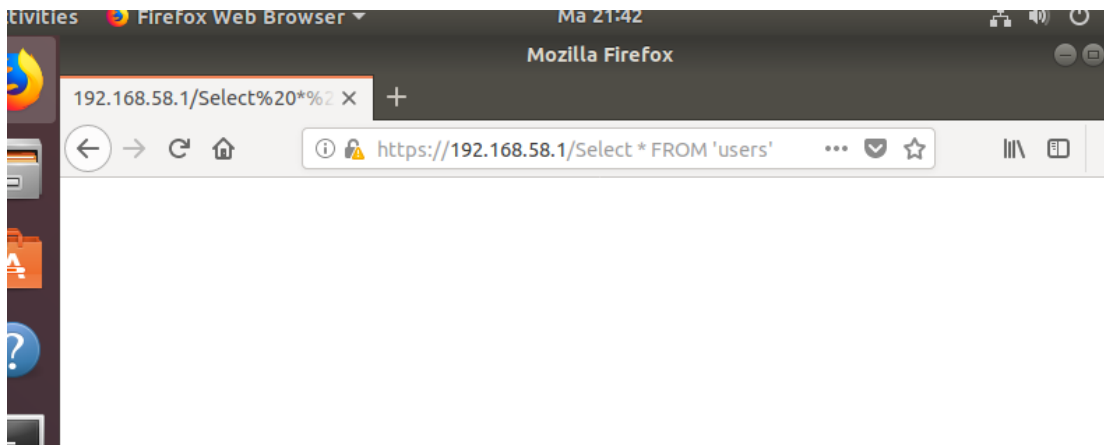


Figura 6.7: Raspunsul pentru SQL injection URL

Figura 6.7 prezintă cum arată răspunsul primit de la server de către un utilizator după trimiterea unui request către server, cu intenția de a realiza un atac de tipul SQL injection. Utilizatorului i se returnează o pagină goală.

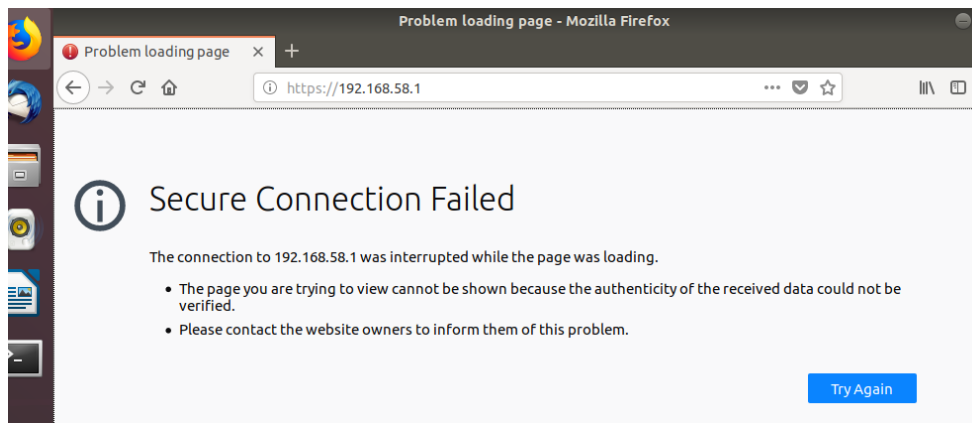


Figura 6.8: Principalele module ale sistemului propus

Figura 6.8 prezintă cum arată răspunsul primit de la server de către un utilizator al rețelei Tor ce intenționează să se conecteze la acesta. Acestor utilizatori li se refuză conexiunea.

Capitolul 7

Teste și rezultate experimentale

7.1 Teste de funcționalitate

În testarea sistemului s-a pus accentul pe testare celor două funcționalități de protecție împotriva atacurilor de SQL injection și a utilizatorilor de Tor.

Pentru testarea modelului folosit în prevenirea atacurilor de SQL injection s-a folosit setul de date de test specificat și în capitolul 6. Acest set a fost obținut din setul inițial de date de antrenare, acesta fiind împărțit în 2 seturi separate cu proporția de 70%(set antrenare) și respectiv 30%(set testare).

Tip set de testare	Preziceri corecte	Dimensiune set	Acuratete(%)
Clean & infected	187596	189278	99.11
Clean	5466	6258	87.34
Infected	182130	193020	99.51

Pe baza tabelului de mai sus se pot observa diferențe majore de performanță între detecția pe setul "Clean" și pe "Infected". Aceste diferențe, respectiv scăderi de performanță în cazul setului Clean, se datorează dimensiunii mult mai mici a setului de Clean folosit și în antrenarea modelului.

Pentru testarea eficienței sistemului de blocare a adreselor IP ale rețelei Tor, s-a realizat o referință între datele colectate de modulul de monitorizare a activității rețelei Tor.

În prima diagramă este prezentată diferența procentuală dintre adresele IP cu un uptime mai mare de 7 zile în ultima lună și cele cu un uptime mai mic. În cea de a doua diagramă este evidențiată diferența dintre uptime-ul total ale acestor adrese IP din ultima lună. Printr-o analiză simplă în paralel a datelor din cele două diagrame, se poate observa că deși doar 20.31% din IP-urile folosite de rețeaua Tor sunt blocate, din timpul total de uptime din ultima lună, 75.6% aparține acestor adrese IP .

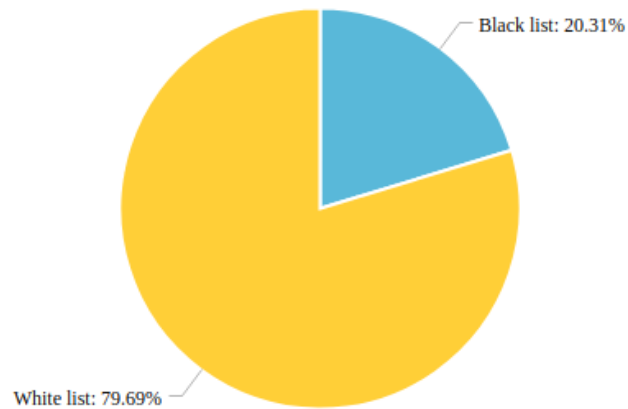


Figura 7.1: Raportul dintre numărul IP-urilor de pe Blacklist și Whitelist dintr-o lună

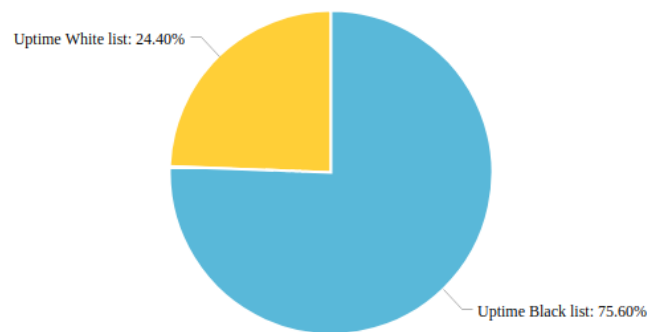


Figura 7.2: Raportul dintre uptime-ul IP-urile de pe Blacklist și Whitelist din aceeași lună

7.2 Teste de performanță

Sistemul propus a fost testat pe două configurații diferite pentru PC-ul gazdă: Intel Core i7-6600U CPU cu 4 nuclee și o frecvență de 2.6 GHz, cu 16 GB RAM DDR4 și i7-4790k CPU cu 4 nuclee și o frecvență maximă de 4.0 GHz, cu 16 GB RAM DDR4. Ambele sisteme furnizând mult mai multe resurse decât cele necesare unei funcționari optime. În cea ce privește resursele minime necesare, acestea trebuie să fie cele necesare rulării unui sistem de operare Windows 10: un procesor cu o frecvență mai mare de 1.0 GHz și o memorie mai mare sau egală cu 2 GB de RAM. În cea ce privește capacitățile sistemului de a suporta conexiuni exterioare, acesta nu a fost testat decât manual, prin intermediul unor mașini virtuale.

Capitolul 8

Manual utilizator

8.1 Instalarea proiectului

Pentru instalarea sistemului, atât timp cât toate dependențele acestuia sunt împlinite, utilizatorul nu mai trebuie să facă nimic. Toate fișierele necesare sistemului au fost împachetate în modulul principal, singurul pas ce poate fi executat de utilizator este crearea unui shortcut către executabilul ce lansează în execuție sistemul.

8.2 Utilizare

După deschiderea aplicației, utilizatorul este întâmpinat de meniul de Home al acesteia. În următoarea fereastră se poate observa designul acestuia:

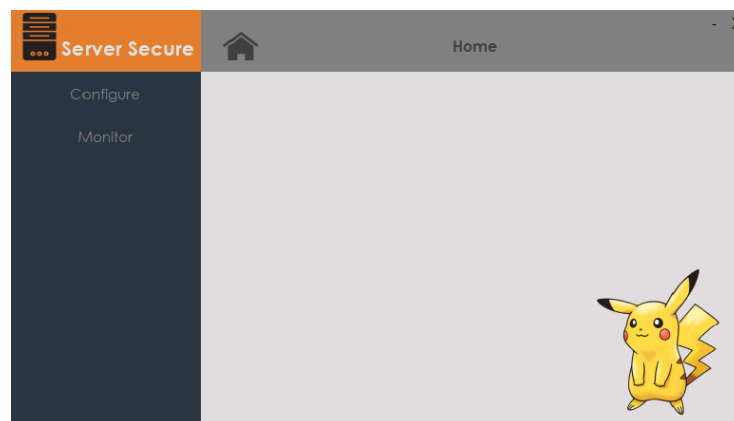


Figura 8.1: Meniul "Home" în interfața grafică

În partea superioară a ferestrei se află în permanență numele ferestrei curente (în cazul de față fereastra Home). În meniul prezent în stânga ferestrei sunt situate celelalte

două ferestre disponibile utilizatorului: Configure și Monitor. Utilizatorul poate să navigheze între aceste ferestre dând un click pe numele ferestrelor, respectiv pe pictograma în formă de casuță pentru meniul Home.

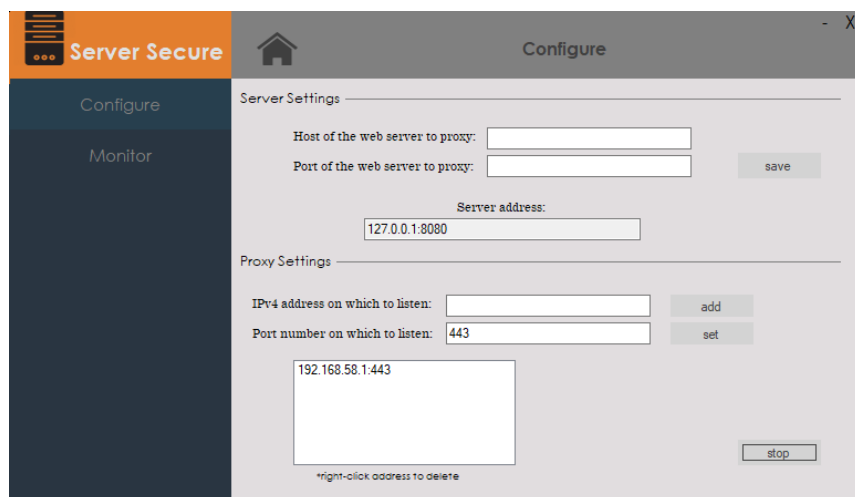


Figura 8.2: Meniul "Configure" în interfața grafică

În fereastra Configure, utilizatorul poate să seteze parametrii de rulare a sistemului. Aceștia i se prezintă două rubrici: Server Settings și Proxy Settings. În partea de Server Settings, utilizatorul setează datele server-ului: adresa IP a acestuia și portul aferent pe care acesta acceptă conexiuni. Salvarea sau suprascrierea acestor date se realizează prin apăsarea butonului "save" din rubrica respectivă. În partea de Proxy Settings, utilizatorul poate să introducă mai multe adrese IP și un port, pe care sistemul să accepte conexiuni și să le redirecționeze către server. După setarea parametrilor de mai sus, sistemul se poate porni/opri prin apăsarea butonului din dreapta jos(cu textul "start"/"stop" după caz).

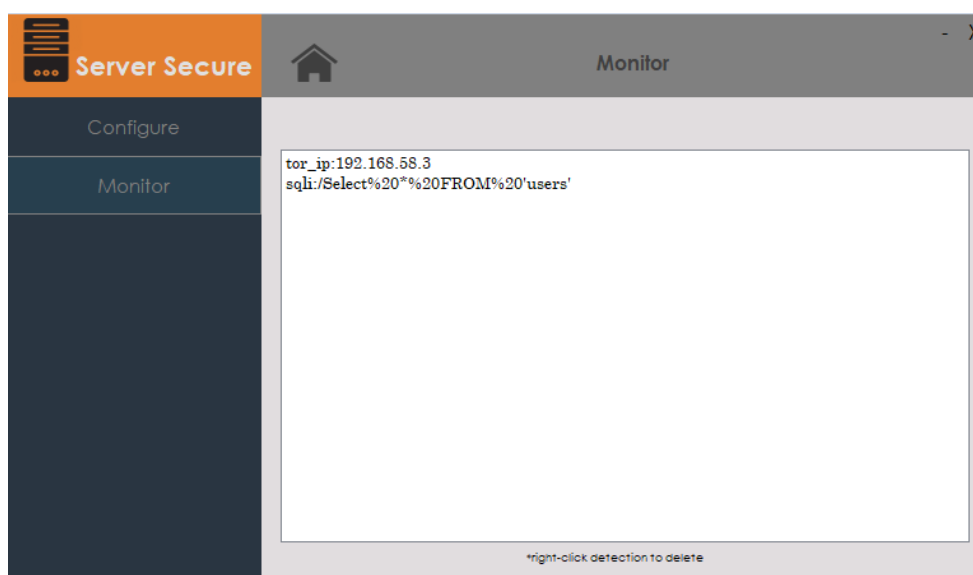


Figura 8.3: Meniul "Monitor" în interfața grafică

În fereastra de Monitor, utilizatorul poate să urmărească activitatea sistemului. În cazul în care sistemul detectează un eveniment, acesta este afișat în interfața grafică în această fereastră (conform exemplului din imagine). Dacă utilizatorul dorește ștergerea evenimentelor anterioare, aceasta se poate realiza prin click dreapta pe eveniment.

Capitolul 9

Conluzii

9.1 Privire de ansamblu asupra sistemului

În urma eforturilor depuse, s-a realizat un sistem de prevenire a intruziunilor care îndeplinește aproape în întregime obiectivele inițial propuse, mici inconveniente fiind la partea de performanță în detecție a sistemului. În cea ce privește performanța sistemului, în cazul detecției atacurilor de SQL injection(precum se poate vedea și în capitolul 7) procentul de fals pozitiv este mult mai mare decât se intenționa inițial(12.66% în loc de 3-4%), însă acest lucru datorându-se în principal setului mic de date de antrenare avute la dispoziție.

Prin implementarea detecției atacurilor SQL injection folosind tehnici de machine learning, s-au dobândit cunoștințe valoroase și experiență ce pot fi folosite în viitoare proiecte. Întrucât în practică, abordarea acestui subiect a reprezentat un lucru nou, o mare parte din timpul investit în dezvoltarea sistemului a reprezentat documentarea și acumularea de noi cunoștințe și experiență pentru rezolvarea unor probleme cu tehnici de machine learning.

9.2 Dezvoltări ulterioare

Datorită unei abordări modulare a dezvoltării sistemului, acestuia îi pot fi adăugate cu ușurință noi funcționalități.

Sistemul prezintă două posibilități majore de dezvoltări ulterioare, acestea fiind afe-rente celor două tipuri de protecție implementată. În cazul protecției împotriva adreselor IP malițioase, lista folosită momentan se poate extinde cu ușurință sau prin mici modificări în cod se pot adăuga noi liste.

Protecția bazată pe detecția unui anumit conținut în URL poate fi cu ușurință extinsă, prin adăugarea de noi model de machine learning sau noi logici de detecție, ambele cazuri necesitând mici modificări în codul sursă.

Bibliografie

- [1] J. C. Villanueva, “Top 8 Benefits of a Reverse Proxy,” <https://www.jscape.com/blog/bid/87841/Top-8-Benefits-of-a-Reverse-Proxy>, Aug 2012.
- [2] “OWASP Top 10 Application Security Risks - 2017,” https://www.owasp.org/index.php/Top_10-2017_Top_10, Last accessed: September 30st, 2018.
- [3] J. V. William G.J. Halfond and A. Orso, “A Classification of SQL Injection Attacks and Countermeasures,” 2006, College of Computing Georgia Institute of Technology, Proceedings of the IEEE International Symposium on Secure Software Engineering.
- [4] M. Prince, “The Trouble with Tor,” <https://blog.cloudflare.com/the-trouble-with-tor/>, Mar 2016.
- [5] R. Bassett, C. Urrutia, and N. Ierace, “Intrusion prevention systems,” <https://dl.acm.org/citation.cfm?id=1071927>, June 2005, ACM New York, NY, USA .
- [6] “What is an intrusion prevention system?” <https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips>.
- [7] J. Snyder, “Information Security,” <https://searchsecurity.techtarget.com/Do-you-need-an-IDS-or-IPS-or-both>.
- [8] M. Rouse, “Network management and monitoring: The evolution of network control,” <https://searchnetworking.techtarget.com/definition/network-analyzer>.
- [9] B. Hendricks, “Intrusion Prevention System (IPS): Definition & Types,” <https://study.com/academy/lesson/intrusion-prevention-system-ips-definition-types.html>.
- [10] C. Paquet, “Implementing cisco ios network security (iins): (ccna security exam 640-553) (authorized self-study guide),” Apr 2009, by Cisco Press.
- [11] K. Rajesh, “An overview of ips – intrusion prevention system and types of network threats,” <https://www.excitingip.com/626/an-overview-of-ips-intrusion-prevention-system-and-types-of-network-threats/>, October 2009.

- [12] E. H. Cheon, Z. Huang, and Y. S. Lee, "Preventing SQL Injection Attack Based on Machine Learning," 2013, Department of Computer Engineering, Woosuk University and Department of Computer Information Engineering, Kunsan National University .
- [13] F. Valeur, D. Mutz, and G. Vigna, "A Learning-Based Approach to the Detection of SQL Attacks," 2005, Reliable Software Group, Department of Computer ScienceUniversity of CaliforniaSanta Barbara .
- [14] P. Winter and S. Lindskog, "How the Great Firewall of China is Blocking Tor," 2012, Karlstad University .
- [15] R. Singh, R. Nithyanand², S. Afroz, P. Pearce, M. C. Tschantz, P. Gill¹, and V. Paxson, "Characterizing the Nature and Dynamics of Tor Exit Blocking," 2017, University of Massachusetts – Amherst, Stony Brook University, University of California – Berkeley, International Computer Science Institute .
- [16] "Framework pentru programarea retelelor scris in python." [Online]. Available: <https://twistedmatrix.com/documents/8.1.0/api/twisted.html>
- [17] "Librarie pentru procesarea de cod html/xml scris in python." [Online]. Available: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
- [18] "Software integrat cu suport pentru svm." [Online]. Available: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [19] "Documentation for python 3.7.1rc1." [Online]. Available: <https://docs.python.org/3/>
- [20] "Documentation for python 2.7.15." [Online]. Available: <https://docs.python.org/2.7/>
- [21] Tim Gray, "Python 2 to python 3: why, and how hard can it be?" [Online]. Available: <https://optimalbi.com/blog/2018/01/19/python-2-to-python-3-why-and-how-hard-can-it-be/>
- [22] "Tabela cu toate caracterele ascii si codurile lor specifice in diferite baze." [Online]. Available: <https://www.asciitable.com/>
- [23] "Caracterele speciale ce pot fi folosite intr-o interogare sql." [Online]. Available: https://docs.oracle.com/cd/A97630_01/text.920/a96518/cqspcl.htm
- [24] "Cuvinte rezervate specifice limbajului sql." [Online]. Available: <https://docs.microsoft.com/en-us/sql/t-sql/language-elements/reserved-keywords-transact-sql?view=sql-server-2017>
- [25] "A practical guide to support vector classification." [Online]. Available: <https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>

- [26] “Statusul nodurilor utilizate de tor.” [Online]. Available: <https://torstatus.blutmagie.de/index.php>
- [27] by NGINX Page, “What is a reverse proxy server?” [Online]. Available: <https://www.nginx.com/resources/glossary/reverse-proxy-server/>
- [28] Andrew Yan-Tak Ng, “Machine learning by stanford university.” [Online]. Available: <https://www.coursera.org/learn/machine-learning/home/welcome>
- [29] by Firewalls.com Page, “How intrusion prevention systems (ips) work in firewall.” [Online]. Available: <https://community.spiceworks.com/topic/362007-how-intrusion-prevention-systems-ips-work-in-firewall>

Anexa A

Codul sursa(principalele fisiere)

A.1 Reverse proxy

```
class BadURL(Resource):
    def render(self, request):
        return ""

class HTTPSReverseProxyResource(proxy.ReverseProxyResource, object):
    def getChild(self, path, request):
        global m

        features = uri_convertor.uri_to_features(request.uri)
        x0, max_idx = gen_svm_nodearray(features)
        label = libsvm.svm_predict(m, x0)

        if int(label) == 1:
            print('blocked-->sqli:' + request.uri)
            return BadURL()

        child = super(HTTPSReverseProxyResource, self).getChild(path, request)
        return HTTPSReverseProxyResource(child.host, child.port, child.path, child.reactor)

def main(args):

    ap = argparse.ArgumentParser()
    ap.add_argument('-c', type=str, default='./server.crt')
    ap.add_argument('-k', type=str, default='./server.key')
    ap.add_argument('-m', type=str, default='./sqli.model')
    ap.add_argument('--server-ip', type=str, default='localhost')
    ap.add_argument('--server-port', type=int, default=8080)
    ap.add_argument('--listen-ip', type=str, default = ['192.168.58.1', 'localhost'])
    ap.add_argument('--listen-port', type=int, default=443)
    ns = ap.parse_args(args[1:])

    global m
    m = svm_load_model(ns.m)

    if type(ns.listen_ip) == str:
        ns.listen_ip = ast.literal_eval(ns.listen_ip)
    myProxy = HTTPSReverseProxyResource(ns.server_ip, ns.server_port, '')
    site = server.Site(myProxy)
    if ns.c:
```

```
with open(ns.c, 'rb') as fp:
    ssl_cert = fp.read()
if ns.k:
    with open(ns.k, 'rb') as fp:
        ssl_key = fp.read()
        certificate = ssl.PrivateCertificate.load(ssl_cert, ssl.KeyPair.load(ssl_key, crypto.FILETYPE_PEM),
                                                crypto.FILETYPE_PEM)
else:
    certificate = ssl.PrivateCertificate.loadPEM(ssl_cert)
for ele in ns.listen_ip:
    try:
        reactor.listenSSL(ns.listen_port, site, certificate.options(), interface=ele)
    except error.CannotListenError:
        print('Error: ' + ele + ' not a valid interface in this context')
        exit(0)
    except Exception as e:
        print('Error: ' + e.message)
        exit(0)
else:
    for ele in ns.listen_ip:
        try:
            reactor.listenTCP(ns.listen_port, site, interface=ele)
        except error.CannotListenError:
            print('Error: ' + ele + ' not a valid interface in this context')
            exit(0)
        except Exception as e:
            print('Error: ' + e.message)
            exit(0)
reactor.run()

if __name__ == '__main__':
    main(sys.argv)
```

A.2 Convertor de URI in trasaturi pentru modelul SVM

```
def convert_uri(uri):
    new_uri = ''

    for index, sub_uri in enumerate(uri.split('%')):
        if sub_uri:
            if index == 0:
                new_uri = sub_uri
                continue
            try:
                hex_val = bytearray.fromhex(sub_uri[:2]).decode()
            except UnicodeDecodeError:
                hex_val = ''
            except ValueError:
                print(uri + ' --- ' + sub_uri[:2])
                return ''
            except Exception as e:
                print(str(e))
                print(uri + '----' + sub_uri[:2])
                exit(0)
```

```
    new_uri += hex_val + sub_uri[2:]
    new_uri = new_uri.upper()

return new_uri

def uri_to_features(uri):

    global keywords
    global keywords_list

    uri = convert_uri(uri)

    keywords_aux = keywords.copy()
    ok = False

    try:
        for keys in keywords_aux:
            if keys in uri:
                if keywords_list.index(keys) > 184:
                    keywords_aux[keys] = uri.count(keys)
                    if not uri.count(keys) == 0:
                        ok = True
            else:
                sub_uri = uri.split(keys)
                for index, ele in enumerate(sub_uri):
                    if index + 1 < len(sub_uri):
                        if ele and sub_uri[index + 1]:
                            if not ele[-1].isalpha() and not sub_uri[index+1][0].isalpha():
                                keywords_aux[keys] += 1
                                ok = True
                    else:
                        if not ele and sub_uri[index - 1]:
                            keywords_aux[keys] += 1
                            ok = True
    except:
        ok = False

    if ok:
        features = {}
        for keys in keywords_list:
            if not keywords_aux[keys] == 0:
                features[keywords_list.index(keys)+1] = keywords_aux[keys]
        keywords_aux.clear()
        return features
    else:
        keywords_aux.clear()
        return {}

def main(argv):

    parser = argparse.ArgumentParser()
    parser.add_argument('-u', type=str, help='uri to convert')

    args = parser.parse_args(argv[1:])

    if not args.u:
        print('No given string(uri)')
        exit(0)

    uri = convert_uri(args.u)
```



```
print(uri)

if __name__ == '__main__':
    main(sys.argv)
```

A.3 Monitorizarea adreselor IP ale rețelei Tor

```
import re
import os
import json
from urllib.request import urlopen
from bs4 import BeautifulSoup

regex = re.compile(
    r'^(?:http|ftp)s?://' # http:// or https://
    r'(?:(?:[A-Z0-9](?:[A-Z0-9-]{0,61}[A-Z0-9])?\.)+(?:[A-Z]{2,6}\.?|[A-Z0-9-]{2,}\.?)|' #domain...
    r'localhost|' #localhost...
    r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})' # ...or ip
    r'(?::\d+)?' # optional port
    r'(?:/?|[/?]\S+)$', re.IGNORECASE)

all_ip = {}

def isValidUrl(url):
    if regex.match(url) is not None:
        return True
    return False

def sort_ip_list(ip_list):

    from IPy import IP
    ipl = [(IP(ip).int(), ip) for ip in ip_list]
    ipl.sort()
    return [ip[1] for ip in ipl]

def init_dictionary():

    all_ip['0.0.0.0'] = []

def pars():

    global all_ip
    if not os.path.isfile('/home/avid/work/scenarii/bitbucket/craw_tor/ips_tor_activity'):
        init_dictionary()
    else:
        with open('/home/avid/work/scenarii/bitbucket/craw_tor/ips_tor_activity', 'r', encoding='utf8') as fd:
            all_ip = json.load(fd)

    page = 'https://torstatus.blutmagie.de/index.php?SR=Uptime&SO=Desc'

    pagesource = urlopen(page)
    s = pagesource.read()
    soup = BeautifulSoup(s)
    table = soup.findAll('table', attrs={'class': 'displayTable'})
    rows = table[0].findAll('tr', attrs={'class': 'r'})
```

```
current_tor_ips = {}

for row in rows:

    try:
        time_up = row.findAll('td')[4].contents[0]
        ip = row.findAll('td', attrs={'class': 'iT'})[0].findAll('a', attrs={'class': 'who'})[0].contents[0]

        days = time_up.split()[1]
        if days == 'd':
            hours = 6
        else:
            hours = int(time_up.split()[0])
            if hours > 6:
                hours = 6
    except:
        continue
    current_tor_ips[ip] = hours

for ips in all_ip:
    if ips == '0.0.0.0':
        continue
    if ips not in current_tor_ips:
        all_ip[ips].append(0)
        continue
    all_ip[ips].append(current_tor_ips[ips])
    del current_tor_ips[ips]

for ips in current_tor_ips:
    all_ip[ips] = all_ip['0.0.0.0'].copy()
    all_ip[ips].append(current_tor_ips[ips])

all_ip['0.0.0.0'].append(0)

with open('/home/avid/work/scenarii/bitbucket/craw_tor/ips_tor_activity', 'w', encoding='utf8') as fd:
    json.dump(all_ip, fd)

pars()
```