**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ŞI SECURITATE CIBERNETICĂ**

***Specializarea: Calculatoare şi sisteme informatice pentru apărare***

***şi securitate naţională***



**ATM-Scan**

CONDUCĂTOR ŞTIINŢIFIC:

Mai. Dr. Ing. Constantin GRUMĂZESCU

ABSOLVENT:

**Std. Sg. Maj. Denis-Alexandru-Andrei GRIGORE**

Conţine \_\_\_\_\_\_\_\_\_\_\_ file

Inventariat sub nr. \_\_\_\_\_\_

Poziţia din indicator: \_\_\_\_

Termen de păstrare: \_\_\_\_\_

**BUCUREŞTI**

**2022**

Raport plagiere fata

Raport plagiere verso

Scrisoare de apreciere fata

Scrisoare de apreciere verso

Detaliere fata

Detaliere verso

Abstract

Rezumat in engleza maxim 1 pagina

Rezumat

Rezumat in romana maxim 1 pagina

**CUPRINS**

[**Abstract 7**](#_heading=h.30j0zll)

[**Rezumat 9**](#_heading=h.1fob9te)

[**Listă de abrevieri 15**](#_heading=h.3znysh7)

[**Listă de figuri 17**](#_heading=h.2et92p0)

[**Listă de tabele 18**](#_heading=h.tyjcwt)

[**1. Introducere 19**](#_heading=h.3dy6vkm)

[*1.1. Prezentare generală 19*](#_heading=h.1t3h5sf)

[*1.2. Domeniul de interes 19*](#_heading=h.4d34og8)

[*1.3. Importanța temei 20*](#_heading=h.2s8eyo1)

[*1.4. Obiectivele lucrării 22*](#_heading=h.17dp8vu)

[*1.5. Metodologia de cercetare 22*](#_heading=h.3rdcrjn)

[*1.6. Rezumatul lucrării pe capitole 23*](#_heading=h.26in1rg)

[**2. Sisteme și standarde în automatizare 25**](#_heading=h.lnxbz9)

[*2.1. Puppet 25*](#_heading=h.1hmsyys)

[*2.2. Terraform 27*](#_heading=h.41mghml)

[*2.3. Progress Chef 29*](#_heading=h.2grqrue)

[*2.4. Ansible 29*](#_heading=h.vx1227)

[*2.5. Comparația framework-urilor de orchestrare 30*](#_heading=h.3fwokq0)

[2.5.1.](#_heading=h.1v1yuxt) Caracteristici fundamentale 30

[2.5.2.](#_heading=h.4f1mdlm) Ușurința de configurare și utilizare 31

[2.5.3.](#_heading=h.2u6wntf) Popularitatea în rândul utilizatorilor 31

[**3. Integrarea cu platforme cyber range 33**](#_heading=h.19c6y18)

[*3.1. Suita Apache 34*](#_heading=h.3tbugp1)

[3.1.1.](#_heading=h.28h4qwu) Apache HTTP Server 2.4 34

[3.1.2.](#_heading=h.nmf14n) Apache Any23 2.7 36

[3.1.3.](#_heading=h.37m2jsg) Apache Druid 0.22 38

[3.1.4.](#_heading=h.1mrcu09) Apache Struts 2.5.29 40

[3.1.5.](#_heading=h.46r0co2) Apache Unomi 1.5.4 41

[3.1.6.](#_heading=h.2lwamvv) Apache Tomcat 8.5.78 42

[*3.2. Baze de date 44*](#_heading=h.111kx3o)

[3.2.1.](#_heading=h.3l18frh) MySQL 8.0.28 44

[3.2.2.](#_heading=h.206ipza) MariaDB 10.6.8 46

[3.2.3.](#_heading=h.4k668n3) PostgreSQL 12.11 47

[3.2.4.](#_heading=h.2zbgiuw) Redis 5.0.7 48

[3.2.5.](#_heading=h.1egqt2p) MongoDB 3.6.8 50

[*3.3. Framework-uri de dezvoltare aplicații web 51*](#_heading=h.3ygebqi)

[3.3.1.](#_heading=h.2dlolyb) Laravel Framework 8.83.16 51

[3.3.2.](#_heading=h.sqyw64) Node.js 10.19.0 53

[*3.4. Servicii de rețea 54*](#_heading=h.3cqmetx)

[3.4.1.](#_heading=h.1rvwp1q) FTP 3.0.3 54

[3.4.2.](#_heading=h.4bvk7pj) LDAP (389 - ds) 56

[3.4.3.](#_heading=h.2r0uhxc) SMTP 56

[3.4.4.](#_heading=h.1664s55) SNMP 5.8 57

[3.4.5.](#_heading=h.3q5sasy) Kerberos 59

[3.4.6.](#_heading=h.25b2l0r) Openssh 61

[3.4.7.](#_heading=h.kgcv8k) Samba 62

[*3.5. Alte tipuri de servicii 63*](#_heading=h.34g0dwd)

[3.5.1.](#_heading=h.1jlao46) Nginx 63

[3.5.2.](#_heading=h.43ky6rz) Citrix Server 64

[3.5.3.](#_heading=h.2iq8gzs) Lotus Core CMS 1.0.1 66

[3.5.4.](#_heading=h.xvir7l) VNC 67

[*3.6. Tipuri de exerciții și competiții de securitate cibernetică 68*](#_heading=h.3hv69ve)

[3.6.1.](#_heading=h.1x0gk37) Concursuri de tip Capture The Fag (CTF) 69

[3.6.2.](#_heading=h.4h042r0) Concursuri de tip Red Team – Blue Team 69

[**4. Descrierea implementării 71**](#_heading=h.35nkun2)

[*4.1. Prezentarea framework-urilor folosite 71*](#_heading=h.2w5ecyt)

[4.1.1.](#_heading=h.1baon6m) Python Flask 71

[4.1.1.](#_heading=h.3vac5uf) React Redux 71

[4.1.2.](#_heading=h.2afmg28) MongoDB 72

[4.1.1.](#_heading=h.pkwqa1) Ansible 73

[*4.1.1.1. Ce este Ansible? 73*](#_heading=h.39kk8xu)

[*4.1.1.2. Cum funcționează? 74*](#_heading=h.1opuj5n)

[*4.1.1.3. Modulele Ansible 75*](#_heading=h.48pi1tg)

[*4.1.1.4. Plugin-urile Ansible 76*](#_heading=h.2nusc19)

[*4.1.1.5. Inventarul 76*](#_heading=h.1302m92)

[*4.1.1.6. Playbook-urile 76*](#_heading=h.3mzq4wv)

[*4.1.1.7. Conexiunea SSH 77*](#_heading=h.2250f4o)

[*4.2. Structura proiectului 77*](#_heading=h.haapch)

[4.2.1.](#_heading=h.319y80a) Cerințele aplicației web 77

[4.2.2.](#_heading=h.1gf8i83) Cerințele modulului software de tip REST API 78

[4.2.3.](#_heading=h.40ew0vw) Cerințele de scriere a playbook-urilor 78

[4.2.4.](#_heading=h.2fk6b3p) Definirea arhitecturii proiectului 78

[4.2.5.](#_heading=h.upglbi) Prezentarea cazurilor de utilizare 80

[*4.3. Implementarea software 81*](#_heading=h.3ep43zb)

[4.3.1.](#_heading=h.1tuee74) Modulul software de tip REST API 81

[4.3.1.](#_heading=h.4du1wux) Implementarea aplicației web 85

[4.3.2.](#_heading=h.2szc72q) Integrarea modulelor în cyber range 88

[**5. Concluzii 91**](#_heading=h.1ksv4uv)

[*5.1. Probleme întâmpinate 91*](#_heading=h.44sinio)

[*5.2. Direcții viitoare 91*](#_heading=h.2jxsxqh)

[**6. Bibliografie 93**](#_heading=h.z337ya)

[**7. Anexe 94**](#_heading=h.3j2qqm3)

[*7.1. Anexa A 94*](#_heading=h.1y810tw)

[*7.2. Anexa B 95*](#_heading=h.2xcytpi)

[*7.3. Anexa C 96*](#_heading=h.3whwml4)

[*7.4. Anexa D 98*](#_heading=h.3as4poj)

[*7.5. Anexa E 99*](#_heading=h.49x2ik5)

[*7.6. Anexa F 100*](#_heading=h.147n2zr)

Listă de abrevieri

|  |  |  |
| --- | --- | --- |
| **Nr. crt.** | **Abreviere/Acronim** | **Forma completă** |
|  | API | Application Programming Interface |
|  | IT | Information Technology |
|  | DevOps |  |
|  | IaC | Infrastructure as Code |
|  | YAML | Yet Another Markup Language |
|  | IRC | Internet Relay Chat |
|  | DSL | Domain Specific Language |
|  | VM | Virtual Machine |
|  | DNS | Domain Name System |
|  | SaaS | Software as a Service |
|  | SSH | Secure Shell Protocol |
|  | VPC | Virtual Private Cloud |
|  | CM | Configuration Management |
|  | HCL | Hashicorp Configuration Language |
|  | CR | Cyber Range |
|  | LMS | Learning Management System |
|  | HTTP | Hypertext Transfer Protocol |
|  | URL | Uniform Resource Locator |
|  | CGI | Common Gateway Interface |
|  | RDF | Resource Description Framework |
|  | URI | Uniform Resource Identifier |
|  | MIME | Multipurpose Internet Mail Extensions |
|  | XML | Extensible Markup Language |
|  | XXE | XML External entity injection |
|  | RCE | Remote Code Execution |
|  | CVE | Common Vulnerabilities and Exposures |
|  | OLAP | Online Analytical Processing |
|  | JDBC | Java Database Connectivity |
|  | OGNL | Object-Graph Navigation Language |
|  | JDK | Java Development Kit |
|  | DGPR | General Data Protection Regulation |
|  | JSON | JavaScript Object Notation |
|  | Java EE | Java Platform, Enterprise Edition |
|  | JPS | Jakarta Server Pages |
|  | SQL | Structured Query Language |
|  | BDMS | Database management system |
|  | CMS | Content Management System |
|  | ACID | Atomicity, Consistency, Isolation, and Durability |
|  | DoS | Denial of Service |
|  | GIS | Geographic Information System |
|  | I/O | Input/Output |
|  | SAN | Subject Alternative Names |
|  | XSS | Cross Site Scripting |
|  | FTP | File Transfer Protocol |
|  | SSFR | Server-side request forgery |
|  | LDAP | Lightweight Directory Access Protocol |
|  | SNMP | Simple Network Management Protocol |
|  | MIB | Management Information Base |
|  | SP | Service Processor |
|  | KDC | Key Distribution Center |
|  | TGS | Ticket Granting Server |
|  | SS | Service Server |
|  | AS | Authentication Server |
|  | SMB | Server Message Block |
|  | CIFS | Common Internet File System |
|  | RDS | Remote Desktop Services |
|  | VNC | Virtual Network Computing |
|  | WSGI | Web Server Gateway Interface |
|  | UI | User Interface |
|  | CTF | Capture The Fag |

Listă de figuri

[Figura 1 – Topologia Puppet Master/Agent](about:blank) 27

[Figura 2 – Etapele fluxului de lucru cu Terraform](about:blank) 28

[Figura 3 – Fluxul de lucru al serverului web Apache](about:blank) 34

[Figura 4 – Exemplu minimalist de graf RDF](about:blank) 36

[Figura 5 –Arhitectura Apache Any23](about:blank) 37

[Figura 6 – Diagrama simplificată a arhitecturii MVC în cazul Apache Struts](about:blank) 40

[Figura 7 – Logica arhitecturii Apache Unomi](about:blank) 42

[Figura 8 – Exemplu de arhitectură cu 3 niveluri folosind Apache Tomcat](about:blank) 43

[Figura 9 – Exemplu minimalist de arhitectură ce folosește MySQL Server 45](#_heading=h.zu0gcz)

[Figura 10 – Arhitectura sistemului PostgreSQL](about:blank) 48

[Figura 11 – Comparație între o bază de date clasică și Redis](about:blank) 49

[Figura 12 – Arhitectura unei aplicații Laravel](about:blank) 52

[Figura 13 – Arhitectura framework-ului NodeJS](about:blank) 53

[Figura 14 – Procesul de transfer de fișiere FTP](about:blank) 55

[Figura 15 – Arhitectura de gestionare a rețelei bazată pe SNMP](about:blank) 58

[Figura 16 – Arhitectura sistemului Kerberos](about:blank) 60

[Figura 17 – Controlul domeniului Samba cu LDAP [2]](about:blank) 62

[Figura 18 – Structura unui script Ansible](about:blank) 74

[Figura 19 – Schema bloc a aplicației web](about:blank) 79

[Figura 20 – Diagrama cazurilor de utilizare](about:blank) 80

[Figura 21 – Diagrama cazurilor de utilizare după ce s-a generat un playbook](about:blank) 81

[Figura 22 – Structura directorului ce se va încărca în baza de date 84](#_heading=h.14ykbeg)

[Figura 23 – Răspuns la apelarea rutei /get-services](about:blank) 84

[Figura 24 – Structura aplicației web](about:blank) 86

[Figura 25 – Structura cyber range-ului în care a fost integrat proiectul](about:blank) 89

Listă de tabele

[Tabel 1 – Comparația caracteristicilor de bază ale framework-urilor 30](#_heading=h.338fx5o)

[Tabel 2 – Comparația accesibilității utilizării framework-urilor 31](#_heading=h.1idq7dh)

[Tabel 3 – Evoluția comunității IaC între septembrie 2016 și mai 2019 31](#_heading=h.42ddq1a)

# Introducere

În ultima perioadă, amenințările pe plan cibernetic devin din ce în ce mai frecvente. Oricine își dorește să știe ce vulnerabilități prezintă sistemul sau rețeaua peste care este responsabil înainte ca acestea să fie exploatate la rândul lor de agenți malițioși. Este necesară testarea integritații și securității sistemului într-o manieră periodică.

Lucrarea își propune să ușureze dezvăluirea și testarea vulerabilităților comune ce pot apărea în cadrul sistemelor informatice și al rețelelor de sisteme prin implementarea unui framework automat de pentest ce aduce pe o interfață grafică informații ce pot fi critice pentru securitatea unui sistem.

Acest framework, ATM-Scan, s-a dezvoltat pentru a omogeniza procesul de pentest, produsul final fiind un raport detaliat despre starea și securitatea sistemului sau rețelei. Scris în Python, permite utilizatorilor să introducă cu ajutorul interfeței grafice IP-ul sau plaja de IP-uri ce urmează a fi scanate și diferite opțiuni de customizare a scanării.

## Prezentare generală

Proiectul este bazat pe trei componente principale:

Prima componentă prezintă modului software al unui port scanner scris în limbajul Python, ce va expune în funcție de intrările utilizatorului IP-urile, porturile, serviciile și sistemele de operare ce rulează pe acestea găsite în rețeaua dată.

A două compomentă reprezintă un enumerator implementat folosind mai multe programe utilizate des în pentest. În functie de datele de ieșire din prima componentă, enumeratorul va dispune utilizatorului date găsite în anumite servicii ce rulează pe sistemele țintă.

A treia componentă expune toate datele utile pentru utilizator într-o interfață grafică. Aceasta sintetizează informația in liste, tabele, grafice. De asemenea, utilizatorul va putea salva raportul, configurația și țintele în sistemul de fișiere local, urmând să le încarce înapoi în program folosind interfața grafică.

## Domeniul de interes

Proiectul face parte din domeniul securității infomației în cadrul pentest, obținerea accesului în sisteme informatice, deoarece realizează automatizarea și omogenizarea pașiilor ce ar fi urmați de către un inginer de securitate cibernetică.

Pentest, un test de penetrare, este un atack simulat autorizat, ce țintește sisteme informatice pentru a evalua securitatea acestora. Un penetration tester folosește aceleași resurse, tehnici și procese precum atacatorii pentru a găsi și demonstra impactul slăbiciunilor unui sistem.

Testele de penetrare, de obicei, simulează o variatate de atacuri ce ar putea amenința un sistem. Se poate determina dacă un sistem este destul de robust pentru a susține atacuri din surse autentice sau neautentice. Testele se fac folosind unelte pentru recunoașterea instanțelor dintr-o rețea și porturi deschise, pentru scanarea de vulnerabilitați în cadrul serviciilor din rețea, aplicații web și API-uri, precum și unelte pentru proxy, exploatare și post-exploatare ce să mențină accesul, extindă și să primească obiective de atac. Aceste unelte pot fi lente iar informațiile obținute greu de interpretat eficient.

Chiar dacă pen testing este un efort în principal manual, pen testerii folosesc și tehnicii automate pentru scanare și testare. Totuși ei trebuie să își folosească cunoștințele despre ultimele tehnici de atac pentru a putea testa mai în detaliu decât se poate automat.

Testarea manuală decoperă vulnerabilități și slăbiciuni ce nu sunt incluse în liste populare (exemplu: OWASP Top 10) și testează logica companiei ce testarea automată poate trece cu vederea (exemplu: validarea de date, integritatea verificărilor). O testare manuală poate să identifice și fals pozitive raporate de testarea automată. Deoarece un pen tester este un expert ce gândește ca adversarul, el poate analiza datele astfel încât să țintească atacurile și testele în metode ce testarea automată ce urmează o rutină scriptată nu ar putea.

Testarea automată generează rezultate mult mai rapide și are nevoie de mai puțin profesioniști specializați comparativ cu un test complet manual. Framework-urile și uneltele automate pentru testare urmăresc rezultatele automat și uneori le pot exporta către o platformă centralizată. De asemenea, rezultatele testelor manuale pot varia de la test la test, spre deosebire de testarea automata ce va produce rezultate consistente la fiecare rulare.

Soluția propusă de această temă este un framework de pentest ce va folosi mai multe script-uri integrate și unelte deja existente de testare automată pentru a creea un raport cât mai sintetizat în funcție de rezulatele testărilor, precum și exploatarea automată a vulnerabilitațiilor comune. Toate datele vor fi expuse în fața utilizatorului într-o interfață grafică.

## Importanța temei

Înlăturarea sacrinilor repetitive ce pot apărea în folosirea unei suite de instrumente de pentest este un scop important pentru orice companie. Folosind ATM-Scan, personalul specializat în securitate pot să preia rapid informații legate de arhitectura unei rețele sau sistem informatic, acest lucru îi lasă să se concentreze pe testarea manuală având deja un front asupra căruia să lucreze, reducând mult din timpul petrecut în activități de scanare și enumerare ce sunt, de obicei, repetitive și folosesc același principiu. Automatizarea accelerează întregul proces salvând cea mai importantă resursă, timpul.

ATM-Scan este un framework automat pentru scanare, enumerare și exploatare de vulenrabilități ce poate scana o infrastructură întreagă în paralel, expune utilizatorului diferite tipuri de scanare și enumerare. Intefața prietenoasă poate ajuta un utilizator mai puțin bine pregătit, toate opțiunile ce acesta le poate alege sunt bine documentate în aplicație cu exemple concrete. Automatizarea nu înlocuiește total sarciniile unui tester, ci îi pune la dispoziție un mod de a accelera anumite sarcini triviale.

Un tester profesionist va fi sceptic cu privire la orice software nou apărut, astfel, pentru a demonstra eficiența soluției propuse de această temă, lista următoare prezintă multiple beneficii în utilizarea unui framework automat de pentest în rutina unui specialist de securitate.

* 1. Economisirea timpului și creșterea productivității

Timpul este una din cele mai imporante resurse pentru oricine, nu poate fi cumpărat și nici recuperat. Ne propunem să eficientizăm testarea automată, crescând astfel productivitatea prin scăderea timpului petrecut pentru scanare și enumerare.

* 1. Eliminarea sarcinilor repetitive

În cadrul activităților de pentest este comun ca scanarea și enumerarea să se efectueze folosind unelte software asemănătoare sau identice cu cele folosite în activitatea anterioară. Prin utilizarea unui framework automat de pentest, folosim instrumente software comun folosite de specialiști de securitate. Specialistul va da datele pe care le-ar da suitei lui de instrumente doar o dată, frameworkul va paraleliza cât mai mult în functie de sistemul gazdă.

* 1. Scăderea numărului de greșeli și erori

Un specialist pentest are nevoie de o suită de instrumente pe care să o cunoască extrem de bine și să fie familiar cu ea. Pentru personalul începător acest lucru poate fi o provocare, lipsa de experiență duce la erori neforțate ce pot duce la posibile breșe de securitate și trecerea cu veredea a unor vulnerabilități critice..

* 1. Depășirea complexității

Domeniul securității cibernetice este unui extrem de vast și de dinamic. Specialiștii trebuie să se adapteze mereu în funcție de situație. Soluția prezentată este menită de a prezenta un front de unde un specialist poate porni testarea securității sistemelor țintă.

O parte complicată este cea de a găsi o cale de atac. Încercăm să depășim acest lucru prin prezentarea sintetizată a datelor de ieșire folosite de scripturile integrate în cadrul framework-ului.

## Obiectivele lucrării

Este evident ca evaloarea securității unei infrastructuri este un proces meticulos dus la capăt fie manual, fie cu ajutorul instrumentelor special create pentru pentest. Am propus dezvoltarea unui framework automat de pentest pentru automatizarea sarcinilor ce trebuie îndeplinite de către un specialist de securitate în cadrul procesului de testare a infrastructurii țintă prin scanarea porturilor, enumerarea serviciilor, detectarea și exploatarea vulnerabilitățiilor folosind limbajul Python și integrarea API a unei suite de software folosite des în pentest.

Lucrarea își propune crearea unei interfețe grafice prietenoase în cadrul căreaia utilizatorul va introduce IP-ul sau rețeaua ce va fi testată, va selecta ce instrumente vor fi active în testare, numarul de threaduri pe care dorește aplicația să îl folosească și va putea configura detalii pentru fiecare script ce va rula în spate. Framework-ul va apela pentru începul un port scanner pentru detectarea sistemelor dintr-o rețea, porturile deschise pe mașinile respective, sistemele de operare și serviciile ce rulează pe aceste sisteme.

# State of the art

# ARHITECTURA SOLUTIEI

# Descrierea implementării

# Rezultate

# Concluzii

## Probleme întâmpinate

## Direcții viitoare

# Bibliografie

# Anexe

## Anexa A

## Anexa B

## Anexa C

## Anexa D

## Anexa E

## Anexa F