ROMÂNIA

MINISTERUL APĂRĂRII NAȚIONALE

ACADEMIA TEHNICĂ MILITARĂ "FERDINAND I"

FACULTATEA DE SISTEME INFORMATICE ŞI SECURITATE CIBERNETICĂ

Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională

Aplicație digital forensics pentru sistemul de operare Microsoft Windows

STUDENT:

Stud. Sg. Maj. Cristian Băluți

BUCUREȘTI

2024

Cuprins

1. Introducere

Într-o eră digitală unde volumele de date cresc exponențial și amenințările cibernetice devin din ce în ce mai sofisticate, nevoia de instrumente avansate de digital forensics este mai mare ca niciodată.

Lucrarea de diplomă intitulată "Aplicație digital forensics pentru sistemul de operare Microsoft Windows" are ca obiectiv dezvoltarea unei soluții software care să răspundă acestei cerințe esențiale.

Proiectul se concentrează pe crearea unei aplicații server-client care permite colectarea, analiza și raportarea datelor digitale într-un mod eficient și sistematic, oferind astfel o resursă valoroasă pentru investigații în domeniul IT.

1.1. Descrierea sumară a proiectului

Această aplicație este proiectată să execute o serie de pași critici în domeniul digital forensics.

În primul pas, se va implementa un sistem de monitorizare EDR (Endpoint Detection and Response), pentru a detecta activitățile suspecte pe un dispozitiv Windows.

Ulterior, un script personalizat va fi folosit pentru a colecta artefacte digitale relevante, care vor fi esențiale în construirea unei imagini comprehensive a activității sistemului.

Pasul trei implică utilizarea unui instrument precum Autopsy pentru achiziția și analiza memoriei RAM și a discului sistemului, esențiale pentru identificarea posibilelor metode de persistență și alte dovezi digitale, precum fișierele Prefetch și User Assist.

Finalizarea acestui proces va consta în generarea unui raport detaliat care va fi transmis înapoi clientului, oferind o imagine clară a situației de securitate și a oricăror compromiteri ale sistemului.

2. Cerințe Software

- **Monitorizarea EDR:** Aplicația trebuie să poată monitoriza activitatea sistemului în timp real pentru a detecta și a înregistra comportamente suspecte sau anomalii.
- **Colectarea Artefactelor:** Trebuie să existe un mecanism prin care aplicația să colecteze artefacte digitale din diverse locații ale sistemului de operare, cum ar fi fișiere de sistem, registri de Windows, fișiere temporare, și jurnale de evenimente.
- **Achiziția de Date:** Aplicația va avea capacitatea de a efectua achiziții de date volatile, cum ar fi conținutul memoriei RAM, și non-volatile, cum ar fi date de pe disc, într-un mod care nu alterează sau compromite integritatea datelor.
- **Analiza Datelor:** După colectarea datelor, aplicația trebuie să le analizeze pentru a identifica modele, și a descoperi dovezi de utilizare a sistemului sau de compromitere.
- **Generarea de Rapoarte:** Aplicația trebuie să genereze rapoarte detaliate, ușor de înțeles, care să includă toate descoperirile relevante ale analizei, împreună cu orice recomandări pentru acțiuni ulterioare.
- **Interfața Utilizator:** Trebuie să fie disponibilă o interfață grafică pentru utilizatori, care să permită accesul ușor la toate funcționalitățile aplicației, inclusiv configurarea, monitorizarea și revizuirea rapoartelor.
- **Securitatea Datelor:** Aplicația trebuie să asigure securitatea datelor pe întreg parcursul procesului de forensics, de la colectare la raportare, utilizând protocoale de criptare și autentificare.

3. Tehnologii Utilizate

3.1. Java FX

JavaFX este o platformă software folosită pentru crearea și livrarea aplicațiilor desktop. Este succesorul Swing în stack-ul UI Java și oferă o arhitectură modernă, orientată spre aplicații bogate în interfață grafică (rich Internet applications - RIAs). JavaFX permite dezvoltatorilor să integreze grafică, animații și design de interfață utilizator (UI) de înaltă fidelitate într-o manieră care este independentă de platformă. Dispune de un set bogat de API-uri care facilitează crearea de UI-uri complexe și dinamice, inclusiv suport pentru CSS și markup-ul specific JavaFX, FXML, care îmbunătățește separarea designului UI de logica aplicației.

Caracteristici cheie

- Interfață grafică avansată: Suport pentru scenografie 2D și 3D, inclusiv forme, culori, efecte și transformări.
- Controluri UI bogate: O gamă largă de controluri UI, cum ar fi tabele, arborii, meniurile, tab-urile și multe altele, care pot fi stilizate cu CSS.
- Media: Capacități încorporate pentru redarea audio și video, precum și manipularea fișierelor media.
- Grafică Integrată: Suport pentru canvas și grafică vectorială, care permite desenarea de elemente grafice complexe și personalizabile.
- Animatie: Un framework puternic de animație care poate anima aproape orice proprietate a obiectelor UI.
- FXML: Un limbaj bazat pe XML pentru declararea structurii interfeței grafice, promovând o bună separare a logicii de prezentare.
- Suport pentru dispozitive tactile: Interacțiuni multi-touch gestionate nativ, ceea ce este esențial pentru dispozitivele moderne.

Motivul alegerii

Am ales JavaFX pentru aplicația mea deoarece ne permite să construim o interfață utilizator modernă și atractivă, care să fie și funcțională. JavaFX facilitează construirea unei interfețe grafice responsive și intuitivă, contribuind astfel la o experiență utilizator îmbunătățită. De asemenea, integrarea sa strânsă cu limbajul Java mi-a permis să mă folosesc de ecosistemul vast și bibliotecile existente pentru a dezvolta funcționalitățile dorite în mod eficient. În plus, suportul extensiv pentru CSS ne permite să personalizăm aspectul aplicației noastre pentru a se potrivi cu branding-ul și cerințele de design specifice.

3.2. Flask

Flask este un microframework pentru aplicații web scris în Python. Este clasificat ca "micro" pentru că nu necesită anumite unelte sau biblioteci suplimentare. Acest lucru nu înseamnă că aplicațiile dvs. sunt limitate la funcționalități de bază; Flask este extensibil și se poate adapta la nevoile complexe prin adăugarea de extensii care pot îmbunătăți funcționalitatea sa de bază. Flask este bazat pe Werkzeug WSGI (Web Server Gateway Interface) toolkit și Jinja2 template engine, ambele fiind Pallets Projects.

Caracteristici cheie

- Simplicitate și Flexibilitate: Flask este conceput să fie ușor de învățat și de implementat, permițând dezvoltatorilor să construiască o aplicație web rapid.
- Dezvoltare Agilă: Ușurința cu care se pot face modificări și adăugări la codul Flask o face ideală pentru dezvoltare agilă și iterativă.
- Testare Ușoară: Flask oferă suport pentru testarea unitară a aplicațiilor, ceea ce este esențial pentru ciclurile de dezvoltare rapide și sigure.
- Microframework: Flask este livrat cu doar componentele esențiale pentru a construi o aplicație web, dar poate fi extins cu "Flask extensions" pentru a adăuga funcționalități suplimentare cum ar fi validarea formularelor, încărcarea fișierelor, autentificarea utilizatorilor și multe altele.
- Route și View Function: Suportă un sistem de mapare a URL-urilor către funcții Python și permite o organizare clară a logicii de routing.
- Templating: Cu Jinja2, permite crearea de template-uri HTML pentru a genera dinamic pagini web pe partea de client.
- Suport pentru RESTful Request Dispatching: Flask are facilități pentru a crea API-uri web în stil RESTful, ceea ce îl face ideal pentru backend-uri de aplicații moderne web și mobile.

Motivul alegerii

Flask a fost ales pentru acest proiect datorită naturii sale ușoare și a capacității de a se adapta rapid la cerințe. Flask este suficient de flexibil pentru a fi extins în funcție de cerințele proiectului. În plus, ecosistemul său larg de extensii ne permite să adăugăm funcționalități în plus fără a încărca aplicația cu componente inutile. De asemenea, simplitatea sa face ca Flask să fie ușor de înțeles pentru dezvoltatorii noi, favorizând colaborarea și accelerând procesul de dezvoltare.

3.3. Docker

Docker este o platformă de containerizare open-source care permite dezvoltatorilor să ambaleze aplicații și dependențele lor în containere. Un container Docker encapsulează o bucată de software într-un pachet complet, care conține tot ce este necesar pentru a rula software-ul: codul, un sistem de operare runtime, biblioteci și setări de sistem. Această containerizare asigură că software-ul va funcționa la fel în orice mediu, fie că este vorba de dezvoltare, testare sau producție, rezolvând astfel problema "la mine funcționează".

Caracteristici cheie

- Izolare: Containerele rulează procese în mod izolat de restul sistemului, ceea ce înseamnă că fiecare container are propriul său set de resurse, inclusiv sistemul de fișiere, CPU, memorie și procesul de execuție.
- Portabilitate: Odată creat, un container poate fi rulat pe orice sistem care are Docker instalat, indiferent de mediu, ceea ce reduce incompatibilitățile dintre medii.
- Eficiență: Containerele împărtășesc kernelul sistemului gazdă și nu necesită un sistem de operare suplimentar, ceea ce le face mai eficiente în utilizarea resurselor decât mașinile virtuale tradiționale.
- Microservicii: Docker este ideal pentru arhitectura microserviciilor, deoarece fiecare serviciu poate fi ambalat în propriul container, facilitând scalarea și menținerea izolată.
- Docker Hub: O bibliotecă publică de imagini Docker gata de utilizat, care permite utilizatorilor să găsească și să distribuie containere pentru diferite aplicații și servicii.
- Orchestrare: Docker se integrează cu unelte de orchestrare, cum ar fi Docker Swarm și Kubernetes, care ajută la gestionarea, scalarea și desfășurarea automată a containerelor în medii de producție.

Motivul alegerii

Am ales Docker ca parte a infrastructurii noastre deoarece aduce o consistență extraordinară în procesul de dezvoltare, testare și producție, eliminând problemele legate de mediu. De asemenea, Docker simplifică procesul de CI/CD (Continuous Integration/Continuous Delivery), permițând actualizări rapide și eficiente ale aplicației. Utilizarea containerelor ne-a permis să scalăm serviciile noastre eficient și să profităm de un model de desfășurare care îmbunătățește disponibilitatea și utilizarea resurselor.

3.3.1. Docker Hub

Docker Hub este un serviciu de registry pentru containere Docker care permite dezvoltatorilor să stocheze și să distribuie imagini de container. Este un hub centralizat care conține o colecție vastă de imagini de container, de la aplicații open-source până la software propriu. Utilizatorii pot împinge (push) și pot trage (pull) imagini spre și dinspre Docker Hub, facilitând partajarea și colaborarea. Este similar cu GitHub, dar pentru imagini Docker în loc de cod sursă. Docker Hub integrează funcționalități de automatizare a build-ului și de scanare a securității, oferind utilizatorilor posibilitatea de a crea lanțuri de integrare și livrare continuă (CI/CD) pentru dezvoltarea containerelor.

3.4. Kubernetes

Kubernetes, cunoscut și sub acronimul k8s, este un sistem open-source de automatizare a desfășurării (deployment), scalării și gestionării aplicațiilor containerizate. Este proiectat pentru a facilita și a oferi o abordare declarativă pentru gestionarea serviciilor și a suportului pentru diferite tipuri de modele de desfășurare a aplicațiilor. Kubernetes a fost inițial dezvoltat de Google și acum este menținut de Cloud Native Computing Foundation.

Caracteristici cheie

- Orchestrare de containere: Permite utilizatorilor să ruleze aplicații în containere distribuite peste un cluster de mașini.
- Automatizare: Realizează automat desfășurarea containerelor și le reînvie dacă acestea eșuează, sunt eliminate sau sunt terminate.
- Scalare: Kubernetes facilitează scalarea serviciilor în sus sau în jos, manual sau automat, bazându-se pe utilizarea resurselor sau alte metrici definite de utilizator.
- Servicii de descoperire și echilibrare a încărcării: Asigură adrese IP și un nume DNS unic pentru un set de containere și poate echilibra încărcătura traficului de rețea pentru a menține stabilitatea sistemului.
- Gestionare a configurațiilor și a secretelor: Permite stocarea și gestionarea informațiilor sensibile, cum ar fi parolele, OAuth tokens și cheile ssh, fără a le expune în configurația stack-ului.
- Auto-reparare: Capacitatea de a înlocui și a resuscita containerele care eșuează, a omite mașini care nu răspund și a preveni desfășurarea containerelor care nu sunt gata pentru serviciu până când nu sunt pregătite în mod corespunzător.

- Gestionare declarativă a configurațiilor: Permite utilizatorilor să descrie starea dorită pentru aplicațiile lor prin fișiere YAML sau JSON, iar Kubernetes se ocupă de realizarea acelei stări și de menținerea ei.

Motivul alegerii

Am ales să folosim Kubernetes datorită capacității sale de a automatiza și a optimiza gestionarea aplicațiilor containerizate. Kubernetes ne oferă flexibilitatea de a desfășura rapid noi versiuni ale aplicațiilor, de a răspunde la schimbările de încărcare și de a asigura disponibilitatea și reziliența serviciilor noastre. De asemenea, Kubernetes ne permite să utilizăm infrastructura noastră de hardware în mod mai eficient și să menținem un control mai bun asupra resurselor și costurilor operaționale.

3.4.1. Kind

Kind (Kubernetes IN Docker) este un instrument puternic destinat rulării clusterelor Kubernetes în interiorul containerelor Docker. Creat pentru a oferi un mediu de testare consistent și izolat, Kind permite dezvoltatorilor să configureze rapid și eficient mai multe noduri Kubernetes pe o singură mașină, facilitând astfel testarea, dezvoltarea și integrarea continuă. Acesta este special conceput pentru a fi ușor de utilizat și se integrează perfect în fluxurile de lucru CI/CD, devenind o opțiune preferată pentru simulari și experimente într-un mediu controlat. Kind reproduce fidel un cluster Kubernetes, oferind dezvoltatorilor posibilitatea de a testa caracteristici avansate și scenarii de clustering, fără a implica complexitatea și costurile asociate cu desfășurarea unui cluster real la scară largă. Este o soluție ideală pentru cei care doresc să aprofundeze capabilitățile Kubernetes într-un mediu ușor de gestionat și accesibil.

3.5. Microservicii

Microservicii se referă la o abordare arhitecturală în dezvoltarea de software unde o aplicație este structurată ca o colecție de servicii mici, independente și desfășurate modular. Fiecare microserviciu rulează un proces unic și comunicații cu alte servicii prin intermediul unei interfețe bine definite, de obicei HTTP API-uri. Această metodă este opusă arhitecturii monolitice, unde toate funcționalitățile aplicației sunt strâns cuplate și desfășurate ca o singură unitate.

Microserviciile sunt proiectate pentru a fi:

- Loosely Coupled: Fiecare serviciu este independent de celelalte, ceea ce înseamnă că pot fi dezvoltate, desfășurate și scalate independent.
- Mentenabile și Testabile: Serviciile mai mici sunt mai ușor de înțeles, de modificat și de testat.
- Deținute de echipă: Fiecare microserviciu poate fi condus de o echipă mică, care ia propriile decizii tehnice și organizatorice.
- Specializate: Fiecare serviciu este construit pentru un set specific de funcții și poate fi scris în cel mai potrivit limbaj de programare pentru acele funcții.
- Scalabile: Serviciile pot fi scalate independent, permițând o alocare mai eficientă a resurselor în funcție de cerințele fiecărui serviciu.
- Reziliente: Dacă un serviciu eșuează, celelalte servicii pot continua să funcționeze fără perturbări majore.

În contextul aplicatiei, serverele ce funcționează ca microservicii sunt:

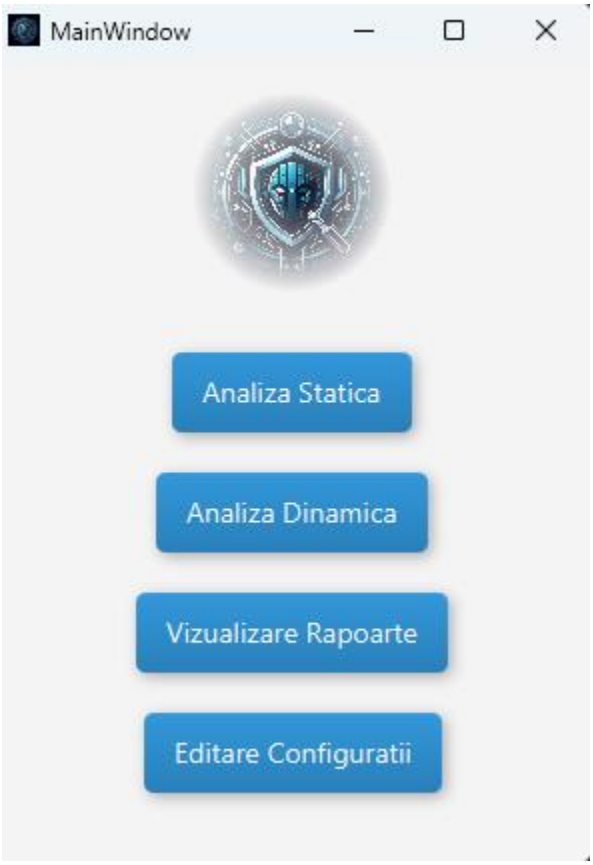
- Serviciul de Gestionare a Rapoartelor: Acest microserviciu este responsabil pentru generarea, stocarea și furnizarea de rapoarte. Este focalizat exclusiv pe procesele legate de raportare și poate fi optimizat pentru aceste sarcini.
- Serviciul de Analiză a Datelor: Acest microserviciu prelucrează și analizează datele primite. Este specializat în executarea de algoritmi și operații de analiză de date, furnizând insights și rezultate necesare altor componente ale sistemului.

Utilizarea microserviciilor în acest mod permite serverului principal să se concentreze pe coordonarea și gestionarea cererilor între servicii, în timp ce fiecare microserviciu îndeplinește sarcini specifice în mod eficient și izolat. Această separare a responsabilităților contribuie la un sistem mai robust, scalabil și ușor de întreținut.

4. Module

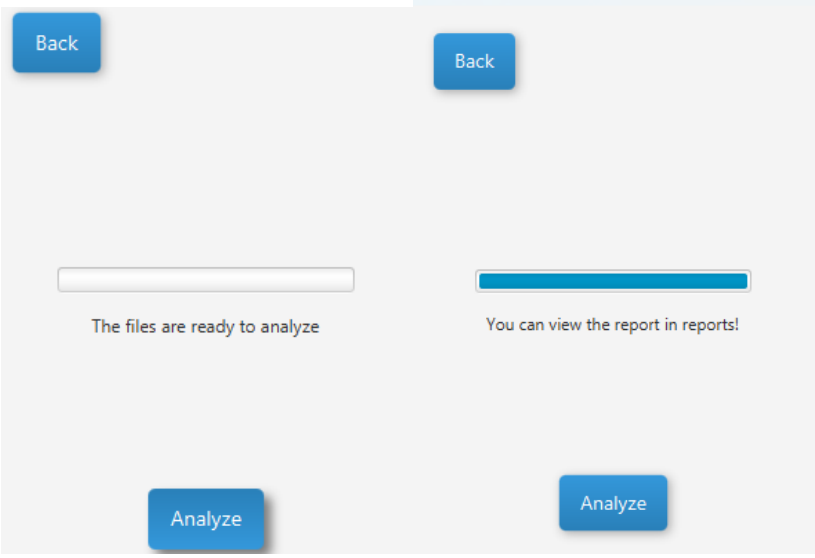
4.1. Aplicatie desktop

Aplicația desktop Java FX este un instrument puternic, dezvoltat pentru a sprijini activitățile de digital forensics printr-un set robust de funcționalități de colectare, trimitere și vizualizare a datelor. Interfața utilizator grafică (GUI) este construită folosind framework-ul JavaFX, oferind o experiență de utilizare interactivă și intuitivă.



Caracteristici Principale:

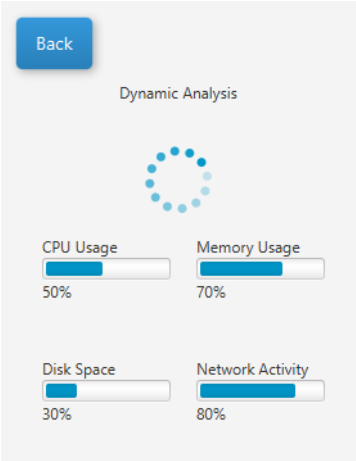
- Colectarea Datelor: Aplicația este echipată cu mecanisme avansate de colectare a datelor, care permit captarea informațiilor necesare în investigațiile de digital forensics. Colectarea este realizată în mod eficient, asigurând integritatea și confidențialitatea datelor.



- Trimiterea Datelor către Server: După colectarea datelor, aplicația le împachetează și le trimite în siguranță către serverele back-end. Procesul de transfer este securizat, asigurând că datele sunt protejate în timpul transmisiei.

- Vizualizarea Rapoartelor: Utilizatorii pot solicita și vizualiza rapoarte generate de server. Această funcție permite o analiză aprofundată a datelor colectate și o înțelegere clară a rezultatelor investigației.

- Analiză Dinamică în Timp Real: Aplicația oferă capabilități de analiză în timp real, permițând utilizatorilor să observe datele pe măsură ce sunt colectate și procesate. Această caracteristică asigură că deciziile critice pot fi luate rapid, pe baza celor mai actuale informații.



- Interfață Intuitivă: Cu ajutorul JavaFX, aplicația are o interfață utilizator prietenoasă și ușor de navigat, care permite utilizatorilor să acceseze și să gestioneze funcționalitățile aplicației cu efort minim.

- Integrarea cu Microservicii: Aplicația funcționează în armonie cu arhitectura bazată pe microservicii, interacționând eficient cu serviciile de gestionare a rapoartelor și de analiză a datelor, și oferind o soluție completă pentru nevoile de digital forensics.

4.2. Server principal

Serverul principal în cadrul unei arhitecturi bazate pe microservicii joacă rolul de orchestrator și punct central de comunicare între diferite servicii și clienți, inclusiv aplicația desktop Java FX.

Caracteristici Principale:

- API Gateway: Serverul poate funcționa ca un API Gateway, care expune interfețele API ale microserviciilor către clienți externi. Este responsabil pentru acceptarea cererilor de la aplicația desktop, aplicarea politicilor de securitate, și rutarea acestora către microserviciile corespunzătoare.

- Load Balancing: Implementează algoritmi de echilibrare a încărcării pentru a distribui sarcinile în mod egal între microservicii, asigurând astfel performanță optimă și timp de răspuns rapid.

- Service Discovery: Gestionează descoperirea serviciilor în cadrul rețelei, permițând microserviciilor să se găsească și să comunice între ele fără configurare statică.

- Securitate: Oferă straturi de securitate, cum ar fi autentificarea, autorizația și criptarea, pentru a proteja datele sensibile și a asigura conformitatea cu standardele industriei de digital forensics.

- Comunicare Inter-Servicii: Orchestrează comunicarea și tranzacțiile între diferite microservicii, asigurând o coerență și consistență a datelor pe tot parcursul proceselor.

- Managementul Tranzacțiilor și a Stării: Menține starea globală a aplicației și gestionează tranzacțiile care necesită coordonare între mai multe servicii, asigurând integritatea datelor și gestionarea corectă a erorilor.

4.2. Modul gestionare rapoarte

Modulul de Gestionare Rapoarte este o componentă esențială într-o aplicație dedicată digital forensics. Acest modul este responsabil pentrucrearea, stocarea și distribuirea rapoartelor care rezumă rezultatele analizelor și investigațiilor forensice. El servește drept mecanism pentru organizarea informațiilor colectate și furnizarea de insight-uri valoroase și concluzii bazate pe datele analizate.

Funcționalități Cheie:

- Generarea de Rapoarte: Modulul permite generarea automată sau la cerere a rapoartelor detaliate, ce pot include grafice, statistici, rezumate ale datelor și concluzii ale analizelor. Rapoartele pot fi configurate să includă diferite niveluri de detaliu, în funcție de audiența țintă sau de cerințele specifice ale cazului.

- Stocarea Sigură a Rapoartelor: Asigură stocarea rapoartelor într-un sistem sigur, care respectă protocoalele de securitate și confidențialitate. Aceasta este crucială pentru păstrarea integrității datelor în context legal.

- Acces și Distribuție Controlată: Permite controlul accesului la rapoarte, asigurând că doar utilizatorii autorizați pot vizualiza sau descărca rapoartele. Funcționalitatea de distribuție poate include trimiterea automată a rapoartelor prin e-mail sau prin alte canale securizate de comunicare.

- Exportul Datelor: Permite exportul datelor în diferite formate, cum ar fi PDF, Excel sau formatelor specificate pentru integrare cu alte sisteme, facilitând schimbul de informații și utilizarea datelor în alte contexte.

Modulul de Gestionare Rapoarte este un instrument crucial în procesul de digital forensics, deoarece rapoartele generate sunt adesea utilizate ca probe în proceduri legale. Prin urmare, acest modul trebuie să funcționeze la standarde înalte de precizie și fiabilitate, asigurând că informațiile sunt prezentate clar, sunt ușor de înțeles și susțin concluziile anchetei.

4.3. Modul analiza date

Modulul de Analiză a Datelor este o componentă cheie în cadrul unui sistem de digital forensics și joacă un rol crucial în interpretarea, procesarea și extragerea de informații valoroase din seturi de date mari și complexe. Acest modul folosește tehnici avansate de analiză de date și algoritmi pentru a identifica modele, a face corelații și a extrage insight-uri utile pentru investigații.

Funcționalități Principale:

- Prelucrarea și Curățarea Datelor: Modulul primește date din diverse surse și formate, care trebuie curățate și normalizate pentru a asigura consistența și acuratețea analizelor ulterioare.

- Analiză Statistică: Realizează analize statistice pentru a identifica tendințe, anomalii și modele în date. Aceasta poate include calculul mediei, deviației standard, și realizarea testelor statistice pentru a determina relevanța unor modele observate.

În domeniul digital forensics, analiza datelor este esențială pentru a distila informațiile relevante din cantități mari de date adesea neorganizate și fără structură clară. Modulul de Analiză a Datelor trebuie să fie capabil să gestioneze:

- Analiza Registrilor Windows: Extrage și analizează date din registrii sistemului de operare.

- Analiza Rețelelor: Examinează traficul de rețea și logurile pentru a identifica activități suspecte sau neautorizate.

- Analiza Fișierelor și Sistemelor de Fișiere: Analizează structura sistemelor de fișiere și recuperează fișierele șterse sau ascunse.

- Cripto-Analiză: Decriptează datele criptate unde este legal și tehnic posibil, pentru a accesa informații ascunse sau protejate.

5. Deployment de Microservicii cu Docker și Kubernetes

Pentru desfășurarea efectivă a microserviciilor, am proiectat și redactat Dockerfile-uri personalizate pentru fiecare componentă a sistemului. Această abordare mi-a permis să construiesc imagini Docker corespunzătoare, reflectând specificațiile precise necesare pentru fiecare serviciu.

În continuare, cu ajutorul unui script PowerShell și prin intermediul Docker Engine, am automatizat procesul de încărcare a acestor imagini pe Docker Hub. Această metodologie nu doar că asigură o distribuție eficientă și organizată a imaginilor containerizate, dar și facilitează versionarea și actualizarea continuă a infrastructurii.

Continuând demersul spre o desfășurare eficientă a infrastructurii, am ales să utilizez un cluster Kubernetes gestionat prin intermediul utilitarului kind, pe care am desfășurat toate imaginile Docker elaborate anterior. Pentru fiecare microserviciu, am creat manifeste în format YAML, care nu doar că detaliază procedura de deployment dar includ și servicii de tip LoadBalancer asociate, asigurând astfel disponibilitatea serviciilor în exteriorul clusterului. Acest pas este esențial pentru integrarea reușită cu aplicația client și pentru garantarea accesibilității resurselor.

Pentru a optimiza procesul de actualizare și menținere a mediului de producție, am îmbogățit scriptul PowerShell existent cu comenzi suplimentare. Acestea sunt proiectate pentru a elimina orice instanță anterioară în mod automat și pentru a aplica noile manifeste, asigurând astfel că clusterul reflectă cea mai recentă configurație fără întreruperi. Această automatizare contribuie la un ciclu de dezvoltare mai agil și reduce semnificativ posibilitatea erorilor umane în procesul de deployment.

6. Concluzii

La finalul etapei curente a proiectului meu de licență, am reușit să dezvolt și să pun în funcțiune infrastructura completă esențială pentru realizarea obiectivelor propuse. Această infrastructură cuprinde un set de microservicii bine definit și scalabil, pregătit pentru extinderi viitoare, care constituie o bază solidă pentru progresul continuu al proiectului.

Am conceput și implementat o interfață front-end de bază, optimizată pentru a asigura o integrare eficientă și o comunicare fluidă cu infrastructura back-end. Acest aspect al proiectului este crucial, deoarece facilitează interacțiunea utilizatorilor cu funcționalitățile sistemului și subliniază importanța unei arhitecturi coerente.

Utilizarea tehnologiei Kubernetes în desfășurarea infrastructurii a fost o decizie strategică, permițându-mi să beneficiez de flexibilitatea și scalabilitatea pe care o oferă acest sistem de orhestrare a containerelor. Aceasta a deschis calea către o gestionare eficientă a resurselor și a distribuției sarcinilor între microservicii, asigurând o bază robustă pentru creșterea capacității sistemului în pas cu necesitățile proiectului.

În continuare, agenda proiectului presupune dezvoltarea și integrarea algoritmilor de analiză statică în cadrul microserviciului dedicat analizei. Urmează să elaborez și să implementez mecanisme de colectare a datelor necesare, care să permită trimiterea și procesarea eficientă a acestora.

De asemenea, voi aborda implementarea componentei de analiză dinamică, care va rula în mediul local, pentru a completa capacitatea de analiză și pentru a oferi o imagine cât mai completă asupra datelor examinate.

În privința generării de rapoarte, urmăresc să elaborez documente cuprinzătoare și pertinente, care să reflecte cu acuratețe rezultatele obținute prin algoritmii de analiză. Aceste rapoarte vor fi strâns legate de datele extrase și vor juca un rol esențial în interpretarea și prezentarea concluziilor analizei.

În sinteză, proiectul meu stă acum pe o fundație tehnologică solidă, pregătit pentru fazele avansate de dezvoltare, și anticipez că următoarele etape vor consolida și mai mult calitatea și eficiența soluției pe care o construiesc.

7. Bibliografie

- Digital forensics - André Årnes
- Handbook of Digital Forensics and Investigation - Eoghan Casey
- https://resources.infosecinstitute.com/topics/digital-forensics/free-open-source-computer-forensics-tools/
- https://recfaces.com/articles/digital-forensics