VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Bakalauro darbas

Saugus pažeidžiamumų skaitytuvas sistemų auditavimui

Atliko:
Artur Ostrovski parašas

Vadovas:
Lektorius Virgilijus Krinickij

Vilnius
2023

Turinys

Sutartinis terminų žodynas **3**

Santrauka **4**

Summary **5**

Įvadas **6**

1. Docker technologija **8**
1.1. Įrankio aprašymas ... 8
1.2. Komponentai ... 8

2. Pažeidžiamumų ir programinių klaidų analizės metodai **9**
2.1. Išorinių pažeidžiamumų skenavimas ... 9
2.2. Vidinių pažeidžiamumų skenavimas ... 9
2.2.1. Vietinė failų sistemos apsauga ... 9
2.3. Kompiuterinių virusų skenavimas ... 10
2.4. Audito gerosios praktikos ... 10

3. Susijusių įrankių analizė **11**
3.1. Nmap ... 11
3.2. Rootkit Hunter ... 11
3.3. Lynis ... 11

4. Sistemų auditavimo įrankis **13**
4.1. Įrankio architektūra ... 13
4.2. Vartotojo sąsaja ... 14
4.3. Saugios aplinkos užtikrinimas ... 15
4.4. Įgyvendinami skenavimo metodai ... 16
4.4.1. Išorinis sistemos skenavimas ... 16
4.4.2. Failų skenavimas ... 16
4.4.3. Kenkėjiškų programų skenavimas ... 16
4.4.4. Vidaus sistemos stiprinimas ... 17
4.5. Skenavimo užklausų įgyvendinimas ... 17
4.6. Įrankio testavimas ... 20
4.7. Aptiktini pažeidžiamumai ... 20
4.8. Įgyvendinti uždaviniai ... 20
4.9. Darbo rezultatų apibendrinimas ... 21

Išvados ir rekomendacijos **22**

Ateities darbų planas **23**

Literatūros šaltiniai **24**

Priedai **26**

Sutartinis terminų žodynas

- Framework - karkasas, į kurį įeina kelios (ar daugiau) programinės įrangos;
- Daemon – programa, kuri veikia fone ir nėra valdoma vartotojo, bet laukia specifinio įvykio ar salygos suveikti;
- Slaptas įėjimas/atgalinių durų pažeidžiamumas – kodo dalis, kuri leidžia atakuotojui į sistemą patekti nepastebėtam;
- Kenkėjiška programinė įranga – bendras pavadinimas tokiai programinei įrangai, kuri yra kenkėjiška ir patenka į sistemą be vartotojo leidimo;
- Rootkit - virusas yra kenkėjiškų programų slaptas tipas, skirtas slėpti tam tikrų procesų ar programų buvimą jūsų kompiuteryje iš įprastų aptikimo metodų, kad leistų tai arba kitas kenksmingas procesas privilegijuotą prieigą prie jūsų kompiuterio.
- UI - vartotojo sąsaja.

Santrauka

Laikui bėgant technologinių inovacijų sparta natūraliai sukūria terpę plėstis saugumo spragoms. Todėl norint išvengti duomenų, jautrios informacijos nutekėjimo ar didelių finansinių nuostolių, didžiausios pasaulyje įmonės ir galingiausios valstybės investuoja į įrankių kūrimą kurie padėtų išvengti išvardintų problemų.

Darbo tikslas – sukurti saugų pažeidžiamumų auditavimo įrankį, kuris saugiai skenuotų Docker kontreinerį pagal pasirinktą vaizdą, jo įšorinę ir vidinę sistemą bei jame esančius failus ir pateiktų informaciją apie audito rezultatus PDF formatu.

Darbe pristatomas saugus pažeidžiamumų skaitytuvas skirtas Docker sistemos auditavimui. Taip pat analizuojama Docker sistemos veikla bei jos patogumai. Be to, pristatomi pažeidžiamumų ir programinių klaidų analizės metodai, jų analizė ir panaudojimas skaitytuve. Taip pat pristatoma susijusių skaitytuvų analizė, jų veiksmingumą ir panaudojimo galimybės. Skaitytuvas įgyvendintas naudojant Javascript programavimo kalbą, Node.js bei React karkasus, Bash skriptus, Docker konteinerių technologiją, MySQL duomenų bazę. Skaitytuvas geba nustatyti, ar Docker vaizdas yra saugus naudojimusi. Pažeidžiamumų skaitytuvas taip pat gali aptikti išorines sistemos spragas, kenkėjiškas programas Docker konteinerio failuose bei branduolio moduliuose. Be to, įrankis sugeba skenuoti atviras direktorijas, failus, prie kurių vartotojai neturėtų turėti teisės prieiti, sustiprinti sistemą ir pateikti gerąsias praktikas ir rekomendacijas. Skaitytuvo saugumas yra pagrįstas konteinerių technologija, kuri užtikrina visišką sistemos izoliaciją.

Summary

Secure Vulnerability Scanner for System Auditing

Over time, the pace of technological innovation naturally creates an environment for security vulnerabilities to expand. Therefore, in order to avoid the leakage of data, sensitive information or large financial losses, the world's largest companies and the most powerful countries invest in the development of tools that would help avoid the listed problems.

The purpose of the work is to create a secure vulnerability auditing tool that would securely scan a Docker container based on a selected image, its external and internal system and the files it contains, and provide information about the audit results in PDF format.

The work presents a secure vulnerability scanner for Docker system auditing. Moreover, purpose of the Docker system and its conveniences are also analyzed. In addition, the methods of analysis of vulnerabilities and software errors, their analysis and use in the scanner are presented. An analysis of related scanners, their effectiveness, and their potential uses are also presented. The scanner is implemented using Javascript programming language, Node.js and React frameworks Bash scripts, Docker container technology, MySQL database. The scanner is able to determine if a Docker image is safe to use. The vulnerability scanner can also detect external system vulnerabilities, malware in Docker container files and kernel modules. In addition, the tool is able to scan open directories, files that users should not have access to, harden the system and provide best practices and recommendations. Scanner security is based on container technology, which ensures complete system isolation.

Įvadas

Darbo motyvacija

Kibernetinis saugumas yra itin svarbus, nes jis apsaugo visų kategorijų duomenis nuo vagysčių ir sugadinimo. Tai apima neskelbtinus duomenis, asmenį identifikuojančią informaciją, saugomą informaciją apie sveikatą, asmeninę informaciją, intelektinę nuosavybę, jautrius duomenis bei vyriausybių užslaptintus dokumentus. Greita naujų technologijų sparta ir pasaulinis debesijos ir virtualizacijos paslaugų išpopuliarėjimas, susietas su vis sudėtingesniais technologiniais sprendimais, lėmia augančią riziką, kad organizacijos nukentės nuo kibernetinės atakos arba duomenų nutekėjimo. Institucijų vadovai nebegali pasikliauti vien tik paruoštais kibernetinio saugumo sprendimais, tokiais kaip antivirusinė programinė įranga ir ugniasienės, kibernetiniai nusikaltėliai tampa vis labiau patyrę, o jų naudojami atakos vektoriai vis labiau komplikuoti ir sunkiau pastebimi. Todėl laikui bėgant patapo svarbu aprėpti kuo daugiau kibernetinio saugumo sričių, kad naudojamos sistemos būtų gerai apsaugotos.

Kibernetinės grėsmės gali kilti iš bet kurio organizacijos lygio. Del to organizacijos pradėjo aktyviai kurti kibernetinio saugumo mokymus, kad darbuotojai būtų mokomi apie įprastas kibernetines grėsmes. Nepaisant to, tyrimas parodė, kad netgi sveikatos įstaigose pagrindinė problema išlieka žmogiškas faktorius ir netyčinės klaidos galimybė[2].

Siekiant išvengti žmogiškos klaidos ir pagerinti šių svarbių duomenų saugumą, žinomiausios organizacijos investuoja didelius pinigus į daugybę skirtingų apsaugos priemonių, įskaitant naujesnę aparatinę įrangą su įmontuotomis saugos funkcijomis, aukštesnio saugumo tinklo protokolus arba brangią kenkėjiškų programų aptikimo programinę įrangą[3].

Apibendrinant, situacija parodo, kad kibernetinio saugumas yra kaip niekad aktuali tema skirtingo tipo organizacijoms. Kibernetinių atakų įvairovė su laiku tik didėja, o išspręsti šias problemas tampa vis sudetingiau. Todėl ši situacija reikalauja naujų įnovacijų ir įrankių kurie padėtų aptikti pažeidžiamumus, žmogiškas klaidas ir laiku išvengti didesnių nuostolių institucijoms.

Darbo tikslas – sukurti saugų pažeidžiamumų skenavimo įrankį, kuris veiktų visiškai izoliuotoje terpėje, saugiai skenuotų Docker konteinerį pagal pasirinktą vaizdą bei jo failus ir pateiktų informaciją apie audito rezultatus PDF formatu.

Darbo uždaviniai:

- Išanalizuoti Docker technologiją, jos pranašumus ir palyginti kitus virtualizacijos įrankius;
- Ištirti dažniausiai pasitaikančius ir grėsmingiausius sistemų pažeidžiamumus;
- Apžvelgti dažniausiai pasitaikančius skenavimo metodus ir jų efektyvumą;
- Išanalizuoti skenavimo projektus, darbus, įrankius;
- Sukurti saugų skaitytuvą, kuris saugiai skenuotų Docker sistemą;

Pasiekti rezultatai:

- Išanalizuota Docker technologija, jos veikimo ypatybės ir pranašumai;
- Ištirti dažniausiai pasitaikantys ir grėsmingiausi pažeidžiamumai;
- Ištirti dažniasiai pasitaikantys skenavimo būdai ir apžvelgtas jų efektyvumas;
- Išanalizuoti skenavimo projektai, darbai, įrankiai ir panaudoti kuriant skaitytuvą;
- Sukurtas nuosavas įrankis kuris skenuoja Docker konteinerio failų sistemą;
- Sukurtas saugus skaitytuvas kuris skenuoja Docker sistemą ir peteikia audito rezultatą;

Pirmajame skyriuje apžvelgiama Docker technolgija ir jos komponentai. Vėliau analizuojami skenavimo metodai ir audito gerosios praktikos. Trečiame skyriuje pateikia projektų bei įrankių analizė. Ketvirtame skyriuje aprašomas saugaus pažeidžiamumų skaitytuvo veikimo logika ir galimybės

1. Docker technologija

Šioje skiltyje aprašomi Docker sistemos pagrindiniai veikimo principai ir naudojami komponentai.

1.1. Įrankio aprašymas

Docker yra programinės įrangos kūrimo įrankis ir virtualizacijos technologija, kuri leidžia lengvai kurti, diegti ir valdyti programas naudojant konteinerius. Konteineris reiškia lengvą, atskirą vykdomąjį programinės įrangos paketą, kuriame yra visos bibliotekos, konfigūracijos failai, priklausomybės ir kitos reikalingos dalys programai valdyti. Kitaip tariant, programos veikia vienodai, nepaisant to, kur jos yra ir kokiame įrenginyje jos veikia, nes konteineris suteikia aplinką per visą programos programinės įrangos kūrimo ciklą. Kadangi konteineriai yra izoliuoti, jie užtikrina saugumą, todėl viename serveryje vienu metu gali veikti keli konteineriai.[8]

1.2. Komponentai

Docker sistema susideda iš trijų komponentų.

Programinė įranga:

Docker demonas, vadinamas dockerd, yra nuolatinis procesas, valdantis Docker konteinerius ir tvarkantis konteinerių objektus. Demonas klauso užklausų, siunčiamų per Docker Engine API Docker kliento programa suteikia komandų eilutės sąsają (CLI), kuri leidžia vartotojams bendrauti su „Docker“ demonais.

Objektai:

- Docker konteineris yra standartizuota, izoliuota aplinka, kurioje veikia programos. Sudėtinis rodinys tvarkomas naudojant „Docker“ API arba CLI.
- Docker vaizdas yra skaitomas šablonas, naudojamas konteineriams kurti. Vaizdai naudojami programoms saugoti ir siųsti.

Registras:

- Docker registras yra vaizdų saugykla. Docker klientai prisijungia prie registrų, kad atsisiųstų ir įkeltų vaizdus, kuriuos jie sukūrė. Registrai gali būti vieši arba privatūs. Pagrindinis viešasis registras yra Docker Hub. Docker Hub yra numatytasis registras, kuriame Docker ieško vaizdų.

2. Pažeidžiamumų ir programinių klaidų analizės metodai

Sistemų pažeidžiamumų analizės metodų yra apstu. Todėl norint atrasti visas įmanomas spragas, verta aprėpti kuo platesnį spektrą metodų ir juos pritaikyti praktikoje audituojant sistemas.

2.1. Išorinių pažeidžiamumų skenavimas

Išorinis pažeidžiamumo nuskaitymas atliekamas ne organizacijos tinkle (t.y. neprieinant prie tinklo). Šie nuskaitymai taikomi tinklo išoriniams IP adresams ir aptinka atvirus prievadus bei protokolus, kad nustatytų saugos spragas.[17] Jie siūlo šiuos privalumus:

- Patikrinti išorinių paslaugų saugos padėtį;
- Parodyti žinomus tinklo struktūrų trūkumus, kurie gali sukelti pažeidimą;
- Atskleisti svarbiausias grėsmes ir rizikas organizacijos tinkle;
- Nustatyti įrenginius ar paslaugas, kurie gali kelti naujų grėsmių ar trūkumų įmonei;

2.2. Vidinių pažeidžiamumų skenavimas

Vidinis pažeidžiamumo nuskaitymas apžvelgia organizacijos IT infrastruktūrą ir saugos profilį iš viešai neatskleistos pusės – konkrečiai asmens, turinčio teisę pasiekti įmonės sistemas. Tai gali būti darbuotojas arba trečioji šalis.[15] Tokiu būdu tikrinant IT infrastruktūrą, vidiniai skaitytuvai ir įrankiai suteikia šiuos privalumus :

- Imituoti asmens, turinčio standartines privilegijas, elgesį ir veiksmus, siekiant nustatyti pažeidžiamumą, kuris gali pakenkti svarbioms sistemoms, funkcijoms ir operacijoms;
- Nustatyti rizikos sistemas ir teikti pirmenybę pažeidžiamumo pašalinimui;
- Pateikti naudingų įžvalgų, kaip sustiprinti sistemas;
- Išanalizuoti galimus sistemos atnaujinimus;

Vidinio ir išorinio pažeidžiamumo nuskaitymo procesas yra panašūs, tačiau skiriasi nuskaitymo įrenginio arba sistemos tinklo vieta. Išorinis pažeidžiamumo nuskaitymas turi galimybė nustatyti spragas, iš kibernetinio nusikaltėlio perspektyvos, nukreipiant į tinklą per viešuosius tinklo IP adresus, o vidinis pažeidžiamumo nuskaitymas veikia scenarijuje, kad auditorius yra asmuo turintis prieigą prie vidinio tinklo yra nukreipimas per privačius IP adresus. Nustatyti abu vidinės ir išorinės grėsmės yra labai svarbu bet kokiam kompiuterių tinklui, kad realiuoju laiku būtų galima susidaryti vaizdą apie tinklo saugumą, remiantis pažeidžiamumų skaičiumi.[15]

2.2.1. Vietinė failų sistemos apsauga

Viena svarbesnių dalių skenuojant vidinius pažeidžiamumus yra atkreipti demėsį failų ir aplankų prieigos nustatymus, ar nėra atidaryti tam tikri failai paprastiems vartotojams, ar teisingai yra sukurtos vartotojų grupės bei patikrinti ar nebuvo žmogiškos klaidos kuriant ar atnaujinant sistemas.[15]

2.3. Kompiuterinių virusų skenavimas

Yra keletas metodų, kurie yra naudojami siekiant galutinio tikslo, t.y. laiku įgyti tikslų supratimą apie virusines programas, prevencijai ir reaguoti, kad būtų galima kontroliuoti kompiuterių virusų protrūkius.[18] Skenuojant sistemas reikia suprasti tam tikrus pažeidžiamumus ir kenkėjišką kodą, kuris jas išnaudoja. Naudojant tinkamus įrankius ir panaudojant savo žinias laiku galima užkirsti:

- Makro virusams;
- Atminties rezidentiniams virusams;
- Oligomorfiniams ir polimorfiniams virusams;
- Kirmėlėms (Worm);
- Trojanams
- Rootkit‘ams

2.4. Audito gerosios praktikos

Verta paminėti, kad generuojant pažeidžiamumų ataskaitas pravartu laikytis sistemų auditavimo standartų. Standartiškai pasirenkami skenavimai be įsibrovimo technikų (angl. Non-Intruisive). Nors Nmap siūlo naudoti būtent tokias technikas ir naudoti įsibrovėliškus skanavimo metodus (angl. Intruisive), šiame darbe visi naudojami įrankiai yra naudojami pagal etiško auditavimo taisyklės. Pagrinde gerosios praktikos buvo panaudotos rėmintis straipsniu.[13]

3. Susijusių įrankių analizė

Susijusių įrankių analizėje yra analizuojami projektai, skaitytuvai, įrankiai, kurie skirtingai būdais ir taikydami skirtingus skanavimo metodus aptinka sistemos pažeidžiamumus. Šioje skiltyje aprašomi įrankiai kurie aprėpia skirtingas metodus ir aptinka platų spektrą pažeidžiamumų.

3.1. Nmap

Nmap („Network Mapper“) yra nemokama atvirojo kodo programa, skirta tinklo aptikimui ir saugos auditui. Nmap naudoja neapdorotus IP paketus naujais būdais, kad nustatytų, kokios mašinos yra prieinamos tinkle, kokias paslaugas (programos pavadinimą ir versiją) šie kompiuteriai siūlo, kokias operacines sistemas (ir OS versijas) jos naudoja, kokio tipo paketų filtrai / ugniasienės yra naudojami ir daugybė kitų savybių.[9].

Nmap gali asistuoti su tokiais auditavimo poreikiais:

- Atvirų prievadų paieška perimetro įrenginiuose (perimetras – nuo interneto krašto iki ekstraneto ar intraneto ribų);
- Atlikti tam tikrų sistemų versijų žvalgybą;
- Naudojant OS aptikimo funkciją, kad būtų galima tiksliai nustatyti pasenusias arba neleistinas sistemas tinkle;
- Neleistinų programų ir paslaugų aptikimas;

Nmap Scripting Engine (NSE) yra viena galingiausių ir lanksčiausių Nmap funkcijų. Tai leidžia vartotojams rašyti ir bendrinti paprastus scenarijus, kad būtų galima automatizuoti įvairias tinklo užduotis. Šie scenarijai vykdomi lygiagrečiai greitai ir efektyviai. Vartotojai gali pasikliauti augančiu ir įvairiu scenarijų rinkiniu, platinamu naudojant Nmap, arba parašyti savo, kad patenkintų pasirinktinius poreikius.[9]

3.2. Rootkit Hunter

Rootkit Hunter yra nedidelis projektas, skirtas rasti įtartinus rootkit komponentus. Taip pat gali būti slaptus įėjimus ar kenkėjišką programinę įranga, ypač jei jos tikslas yra pasislėpti. Įrankis naudoja įvairius skanavimo būdus, pvz., naudojant iš anksto nustatytas katalogų vietas ir lyginant sistemos paslaugų išvestį. Kitas naudojamas būdas yra prašant konkrečios išvesties ir tikrinti, ar ši išvestis nėra pakeista, taip apgaudinėjant rootkit komponentus, kad jie atskleistų save. Rkhunter nuskaito failus, tinklo nustatymus ir sistemos komandas, o rezultatus įrašo į žurnalo failą.[16]

3.3. Lynis

Lynis yra išbandytas saugos įrankis sistemoms, kuriose veikia Linux, MacOS arba Unix operacinės sistemos. Jis atlieka išsamų jūsų sistemų būklės nuskaitymą, kad palaikytų sistemos tvirtinimą ir atitikties testavimą. Projektas yra atvirojo kodo programinė įranga su licencija ir prieinama nuo 2007 m. Kadangi Lynis yra lankstus, jis naudojamas įvairiems tikslams.

Tipiški Lynis naudojimo atvejai:

- Saugumo auditas;
- Prasiskverbimo testavimas;
- Pažeidžiamumo aptikimas;
- Sistemos sustiprinimas;

Lynis nuskaitymas yra modulinis ir oportunistinis. Tai reiškia, kad jis naudos ir išbandys tik tuos komponentus, kuriuos gali rasti - turimus sistemos įrankius ir jų bibliotekas. Privalumas yra tas, kad nereikia diegti kitų įrankių, todėl galite išlaikyti savo sistemas švarias.[16]

Naudojant šį nuskaitymo metodą, įrankis gali veikti beveik be priklausomybių. Be to, kuo daugiau komponentų jis atras, tuo platesnis bus auditas. Kitaip tariant: Lynis visada atliks jūsų sistemai pritaikytus nuskaitymus.

Lynis pradeda nuo inicijavimo fazės, kai aptinka operacinę sistemą. Kai ši dalis baigta, atliekamas dvejetainių failų ir įrankių atradimas. Šis įrankių rinkimas yra pagrindas nuskaitymo varikliui, kuris atlieka šimtus individualių testų.

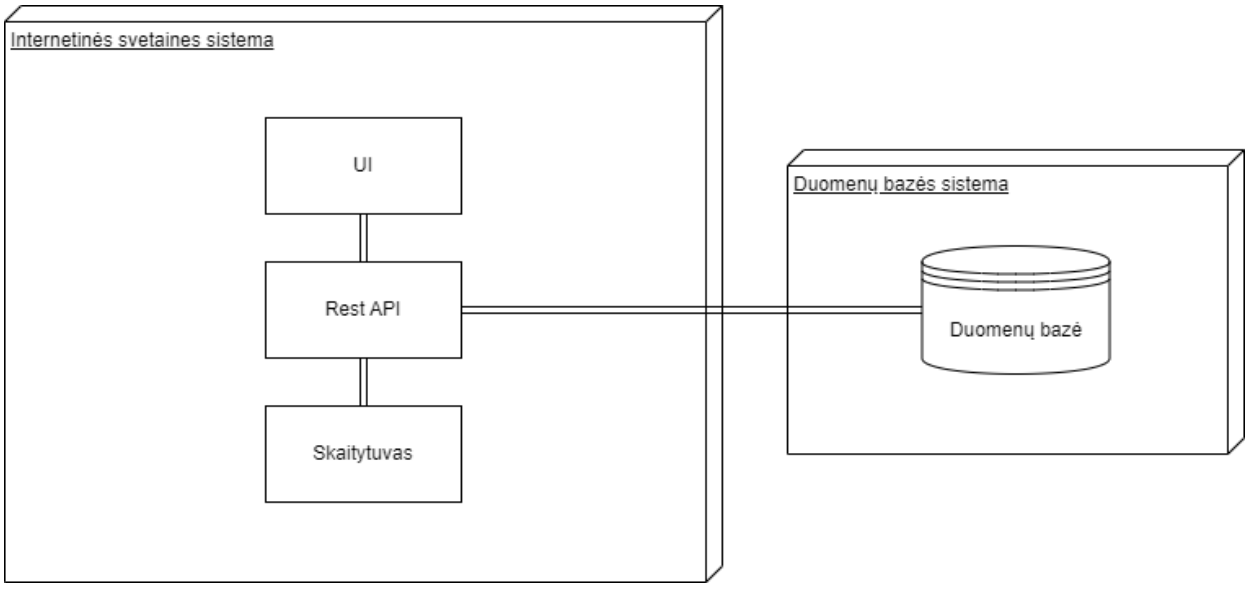
4. Sistemų auditavimo įrankis

Darbo tikslas buvo sukurti saugų pažeidžiamumų skaitytuvą kuris būtų izoliuotas nuo likusios sistemos ir galėtų efektyviai skenuoti vidinę bei išorinę Docker sistemos dalį, atrasti klaidas failų sistemoje bei aptikti įvairias kenkėjiškas programas. Darbo metu pavyko įgyvendinti šiuos rezultatus:

- Sukurta saugi, izoliuota aplinka kurioje galima naudoti užkrėstus Docker komponentus;
- Įgyvendintas išorinis sistemos skenavimas;
- Tikrinami įvairaus tipo pažeidžiamumai vidinio ir išorinio skanavimo metu;
- Skenuojama vidaus failų sistema;
- Vyksta sistemos stiprinimas ir pateikiamos rekomendacijos;

4.1. Įrankio architektūra

Skenavimo įrankio architektūra yra padalinta į dvi pagrindinės dalis – Internetinę svetainę ir duomenų bazę.

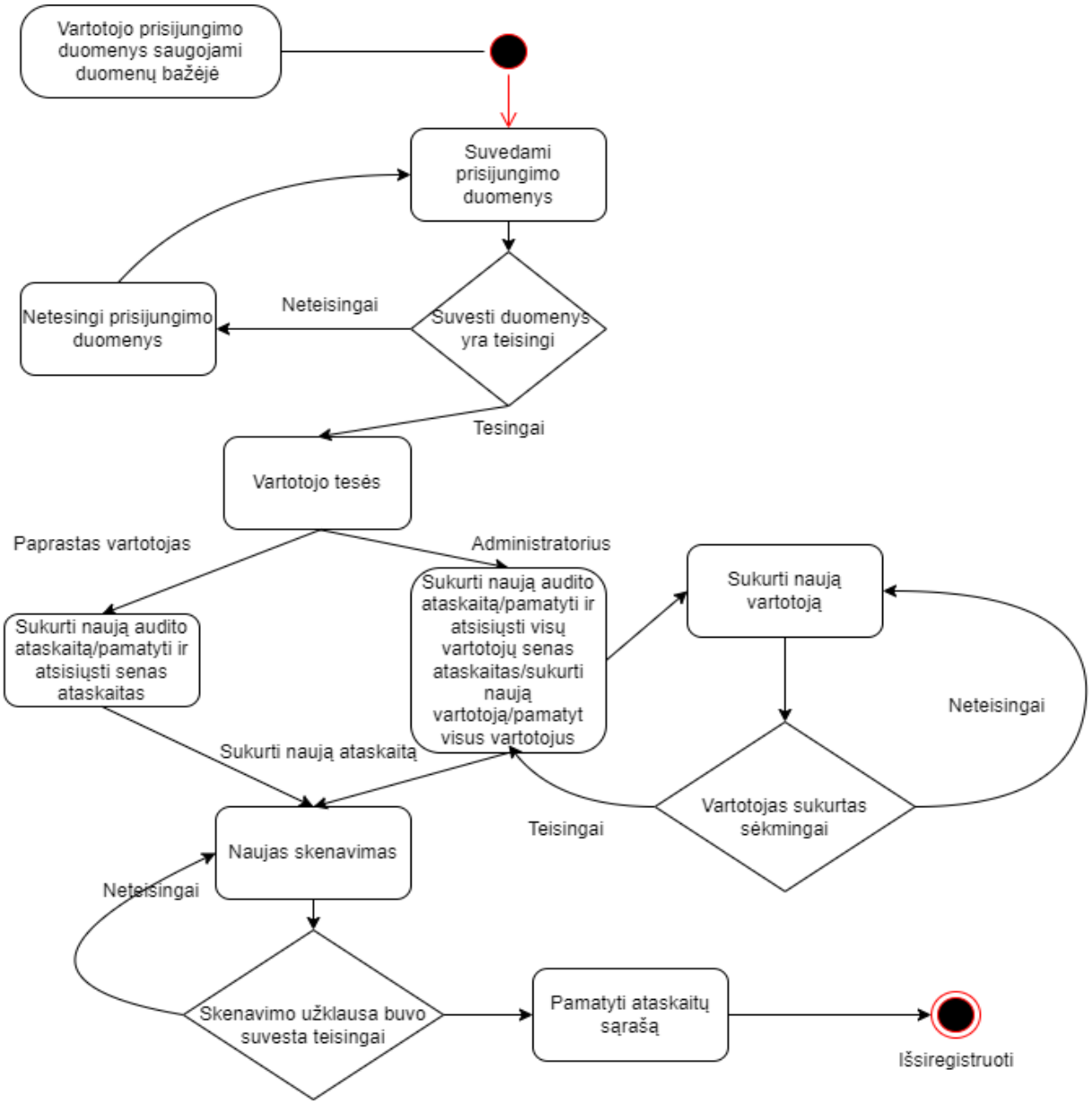


1 pav. Sistemos architektūros schema

Sistemos architektūros schemos 1 pavyzdyje galime matyti dvi sistemas – internetinės svetainės sistemą bei duomenų bazės sistemą. Internetinės svetainės sistemoje veikia pats internetinis puslapis, kuriame vyksta prisijungimas prie sistemos, skenavimo užklausų kūrimas ir audito ataskaitų parsisiuntimas bei vartotojų valdymas. Visa skenavimo logika yra įgyvendinta skaitytuvo posistemėje. Duomenų bazės sistemoje veikia pati duomenų bazė, kurios paskirtis yra saugoti skenavimo užklausas ir jų audito rezultatus, prisijungimo duomenis bei sesijas.

UI aplikacija internetinės svetainės sistemoje sukurta naudojant React. Rest API kuris yra atsakingas už visą bendravimą su duomenų baze bei skaitytuvo paleidimą sukurtas naudojant NodeJs. Duomenų bazė sukurta naudojant MySQL. Skaitytuvas yra sukurtas naudojant Bash skriptus bei išnaudojant Docker konteinerius kartu su jau paruoštais Docker vaizdais..

4.2. Vartotojo sąsaja



2 pav. Aktyvumo diagrama

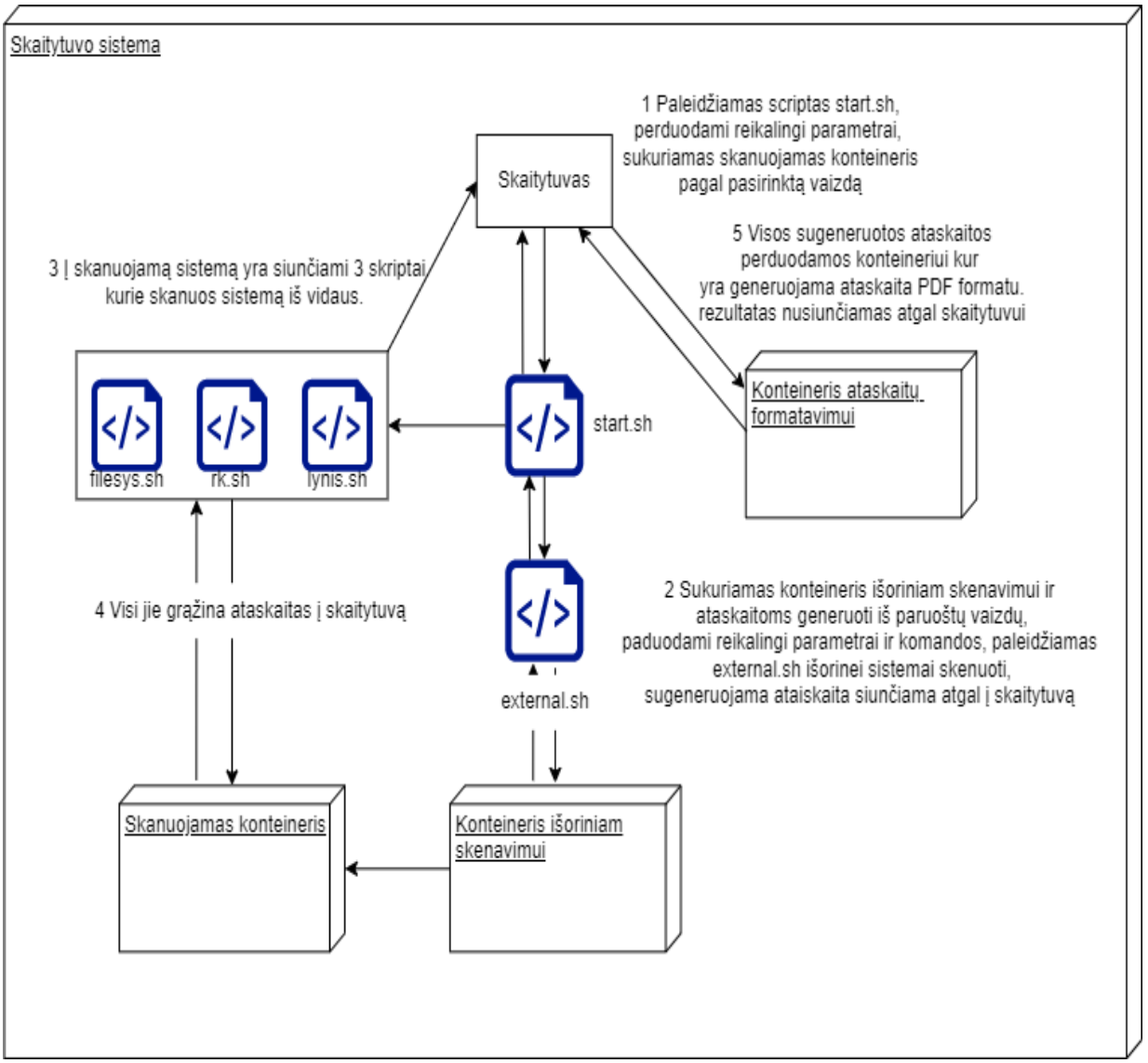
Aktyvumo diagramoje, kuri yra pavaizduota 2 pavyzdyje, galima matyti pasirinkimus, kuriuos turi vartotojas ir administratorius. Vartotojo sąsaja sukurta naudojant React, visi atliekami veiksmai keliauja į Rest API, kuris yra sukurtas su NodeJS. Rest API įvyksta visa svetainės logika – autentifikacijos, sesijų bei duomenų valdymas. Jungimosi metu į UI suvedami duomenys keliauja į Rest API, kuriame prisijungimo duomenys yra tikrinami kartu su duomenų baze. Priklausomai nuo vartotojo tipo, galimi skirtingi svetainės funkcionalumai. Paprastas vartotojas sėkmingai prisijungęs gali skurti naują audito ataskaitą pagal pasirinktą Docker vaizdą, peržiūrėti prieš tai skanuotus vaizdus bei parsisiųsti savo generuotas ataskaitas. Administratorius papildomai gali tvarkyti vartotojus (Sukurti naują arba ištrinti esamą) bei peržiūrėti visų vartotojų audito ataskaitas.

4.3. Saugios aplinkos užtikrinimas

Docker sistemos skenavimo įrankyje saugi aplinka užtikrinama naudojant Docker technologiją, kuri leidžia izoliuoti skenavimo ir auditavimo procesus nuo likusios internetinės svetainės sistemos. Plačiau apie Docker sistemos izoliaciją ir jos komponentus yra aprašyta 1. skyriuje. Docker konteineriai yra kuriami operacinės sistemos lygyje, skirtingai negu kiti virtualizacijos sprendimai, kurie kuriami geležies lygyje, nereikalauja pernelyg daug resursų, todėl yra žymiai greitesni už kitus virtualizacijos įrankius. Tai ypač svarbu kuriant pažeidžiamumų skaitytuvą, kuris reikalauja labai skirtingų programų, bibliotekų bei paketų ir laukiant audito ataskaitos užtikrinti vartotojui kuo greitesnį rezultatų pateikimą.

Konteinerių valdymas yra vykdomas naudojant Bash skriptus. Jie yra atsakingi už tinkamų vaizdų atsiuntimą ir konteinerių paleidimą, reikalingų parametrų pateikimą, failų parsisiuntimą bei pateikimą.

Visos sistemos naudoja Linux 20.4 versiją. Docker sistemos skaitytuvas pritaikytas Debian (nuo 9 versijos) bei palaiko Docker vaizdus nuo Ubuntu 16.04 versijos.



3 pav. Saugios aplinkos užtikrinimo veikimas

3 pavyzdyje matome visą procesą, kuris užtikrina sistemos bei skaitytuvo saugumą. Žemiau yra paaiškinamas kiekvienas proceso žingsnis:

1. Iš pradžių yra gaunama užklausa iš UI kokį Docker vaizdą norima praskanuoti, Rest API sutvarko užklausą ir perduoda parametrus start.sh skriptui. Jis sukuria Docker konteinerį pagal gautus parametrus bei konteinerį ataskaitų formatavimui.

2. Sekančiame žingsnyje paleidžia external.sh skriptą ir paduoda reikiamus parametrus. External.sh paleidžia konteinerį iš paruošto vaizdo ir skanuoja Docker sistemą išoriškai. Skenavimo rezultatus grąžina skaitytuvui.

3. Tuo pačiu skaitytuvas vykdant išorinį skenavimą pradeda skenuoti ir iš vidaus. Jis nusiunčia tris skirtingus paruoštus skriptus į skanuojamą Docker konteinerį ir laukia rezultatų.

4. Po atliktų vidinių skenavimų rezultatai yra taip pat grąžinami į skaitytuvą.

5. Skaitytuvas turėdamas visus skenavimo rezultatus nusiunčia juos į ataskaitų formatavimo konteinerį. Pagal nurodytas komandas yra formatuojama PDF auditavimo ataskaita ir yra nusiunčiama atgal skaitytuvui. Visi naudojami konteineriai yra sunaikinami.

4.4. Įgyvendinami skenavimo metodai

Bendrai buvo įgyvendinti keturi skenavimo metodai. Naudojant šiuos metodus galima aptikti platų spektrą pažeidžiamumų. Įgyvendintas auditas apima išorinės ir vidinės sistemos spragas bei atrastus pažeidžiamumus. Taip pat tikrinama sistema ieškant kenkėjiškų programų bei paslėptų kernelio modulių, tikrinama sistemos failų sistema bei stiprinama sistema.

4.4.1. Išorinis sistemos skenavimas

Išorinis sistemos skenavimas pagrinde buvo naudojamas tam, kad aptiktų atvirius prievadus, patikrintų, kokią informaciją pateikia pati sistema iš išorės – kokią operacinę sistemą sistema naudoja, kokia tos operacinės sistemos versija, kokios naudojamos aplikacijos ar sistemos veikia toje sistemoje. Taip pat naudojami įrankio skriptai kurie nuodugniai ieško pažeidžiamumus priklausomai nuo naudojamos sistemos. Skenavimo įgyvendinimo metu yra sukuriamas konteineris, iš kurio yra atliekamas skenavimas. Pasibaigus skenavimui ataskaitos rezultatai yra grąžinami skaitytuvui formatavimo procesui.[9]

4.4.2. Failų skenavimas

Failų skenavime pagrinde ieškoma atvirų direktorijų, rizikingų failų arba keistų sutapimų kurie galėtų potencialiai kelti grėsmę sistemos saugumui. Šiam žingsniui buvo sukurtas asmeninis įrankis kuris naudoja Nmap apipavidalinimo konfigūraciją bet visa logika yra paremta Bash skriptų įgyvendinimu. Failų skenavimas vyksta Docker konteinerio viduje įdedant skriptą auditui įgyvendinti. Teisingai naudojant šį skanavimo metodą galima išvengti neteisingų prieigos konfigūracijų bei žmogiškos klaidos faktoriaus. [15]

4.4.3. Kenkėjiškų programų skenavimas

Kenkėjiškų programų skenavimas yra skirtam tam, kad rasti įvairaus pobūdžio virusus. Tai gali būti rootkit‘ai paslėpti Linux kernelio modulyje, kirmėlės (Worms) arba kitokio pobūdžio virusai. Šiam skenavimui irgi naudojamas Bash skriptas patalpintas skanuojamo konteinerio

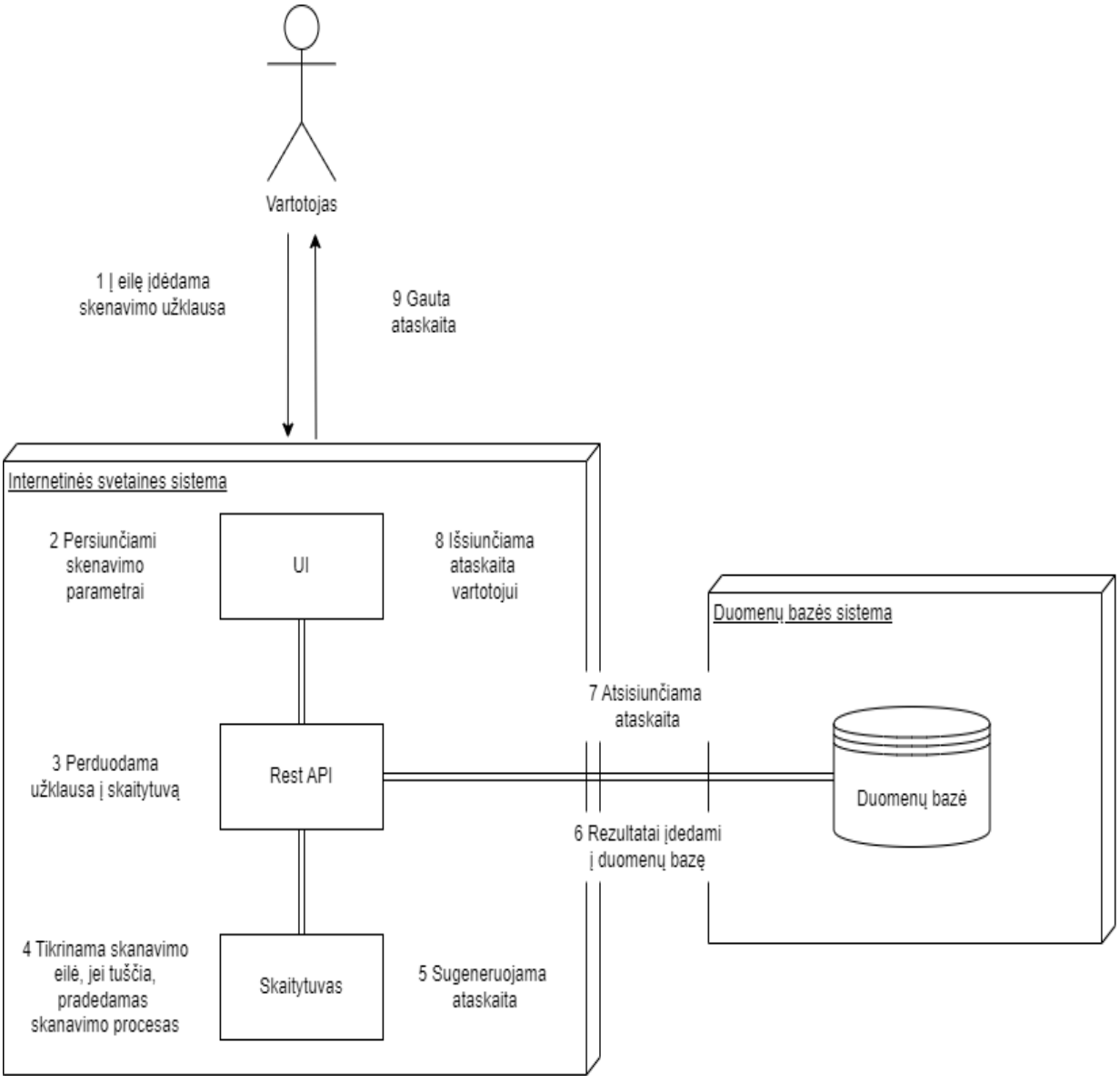
viduje taip padidinant tikimybę atrasti daugiau kenkėjiškų programų. Rezultatas yra grąžinamas skaitytuvui tolimesniam formatavimui.[15]

4.4.4. Vidaus sistemos stiprinimas

Šis metodas vėlgi yra vidaus sistemos skenavimas pasižymintis tuom, kad apima labai platų spektrą skanavimų. Esant sistemos viduje užduotis atrasti sistemos spragas žymiai palengvėja, nes skenavimo nestabdo jokios ugniasienės ar kiti saugumo sprendimai. Verta paminėti, kad šis skanavimo būdas nebando pakenkti sistemai ir nenaudoja jokių įsibrovėliškų technikų. Kaip ir praeitose pavyzdžiuose, šis skanavimo metodas irgi naudoja Bash skriptą kuris yra paleidžiamas sistemos viduje ir grąžina skenavimo ataskaitą skaitytuvui tolimesniems formatavimo žingsniams.

4.5. Skenavimo užklausų įgyvendinimas

Šiame skyriuje apžvelgiamas skenavimo užklausų įgyvendinimas ir procesų eiliškumas. Nuodugniai bandoma paaiškinti sistemos veiklą ir skenavimo procesų logiką.

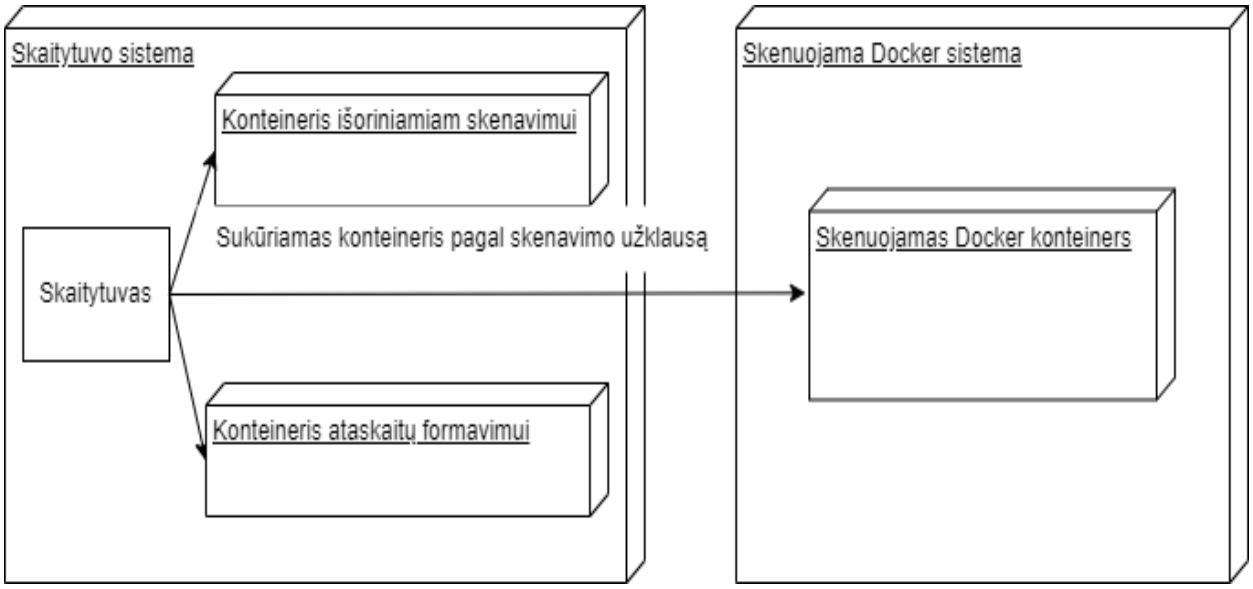


4 pav. Įrankio veikimo schema

Įrankio veikimo schemoje pavaizduotoje 4 pavyzdyje, galima matyti procesus vykstančius sistemos auditavimo metu. Kiekvieno proceso paaiškinamas yra apžvelgtas žemiau:

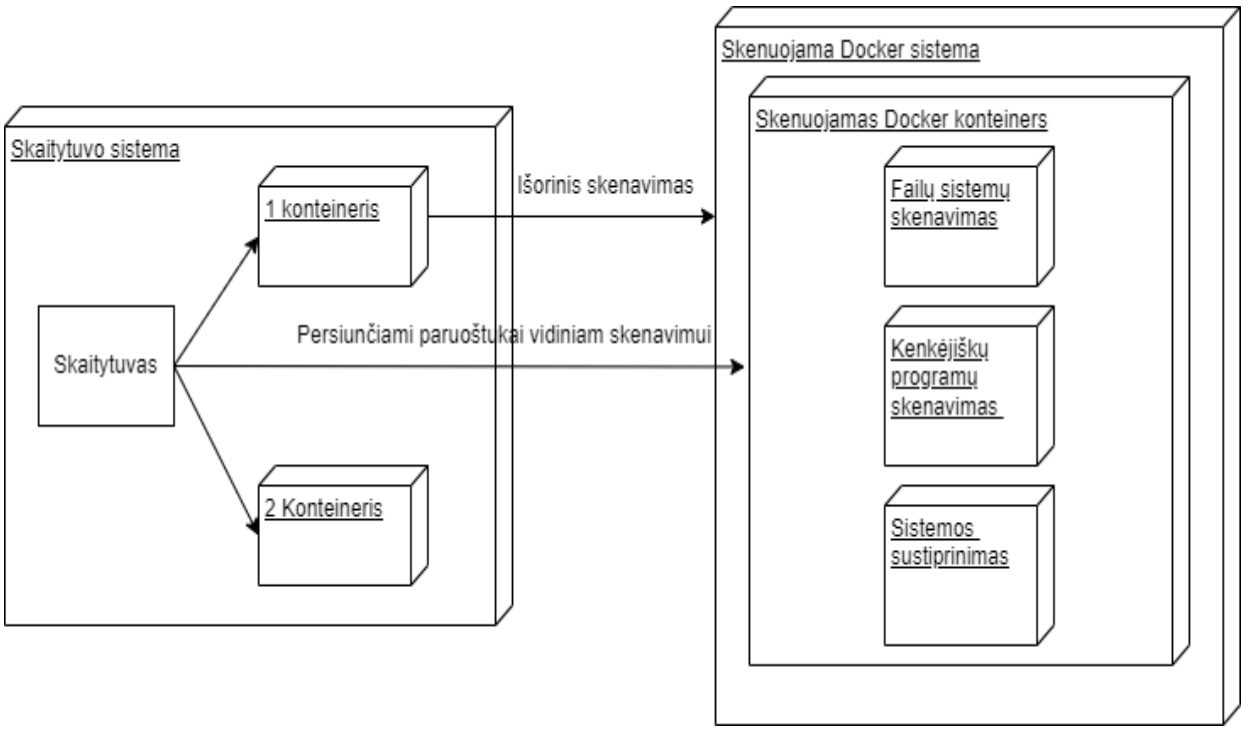
1. Po sėkmingo prisijungimo vartotojas suveda pavadinimą norimo Docker vaizdo kurį norėtų patikrinti ir pradeda generuoti ataskaitą;
2. Duomenys yra siunčiami į Rest API kuriame vyksta visi autorizacijos procesai su duomenų baze ir yra perduodami skenavimo parametrai;
3. Skenavimo parametrai yra perduodami skaitytuvui;
4. Visų pirma užklausa yra dedama į eilę. Jeigu tuo metu nevyksta jokių kitų skanavimų, pradedamas auditavimo procesas. Skaitytuvas pirmiausiai bando pagal parametrus paleisti Docker konteinerį ir patikrinti ar egzistuoja Docker vaizdas pagal turimą parametrą. Po sėkmingo bandymo pereinama prie skenavimo procesų.;
5. Rezultate yra sugeneruojama ataskaita PDF formatu;
6. Audito ataskaita kartu su kitais duomenimis įkeliami į Duomenų bazę;
7. Vartotojas grąžinamas į puslapį kur nurodyti visų skanavimų duomenys. Paspaudus atsisiuntimo mygtuką, pasirinktos ataskaitos duomenys yra parsisiunčiami iš duomenų bazės;
8. Ataskaita yra parsisiunčiama naršyklėje;
9. Vartotojas gauna audito ataskaitą;

.



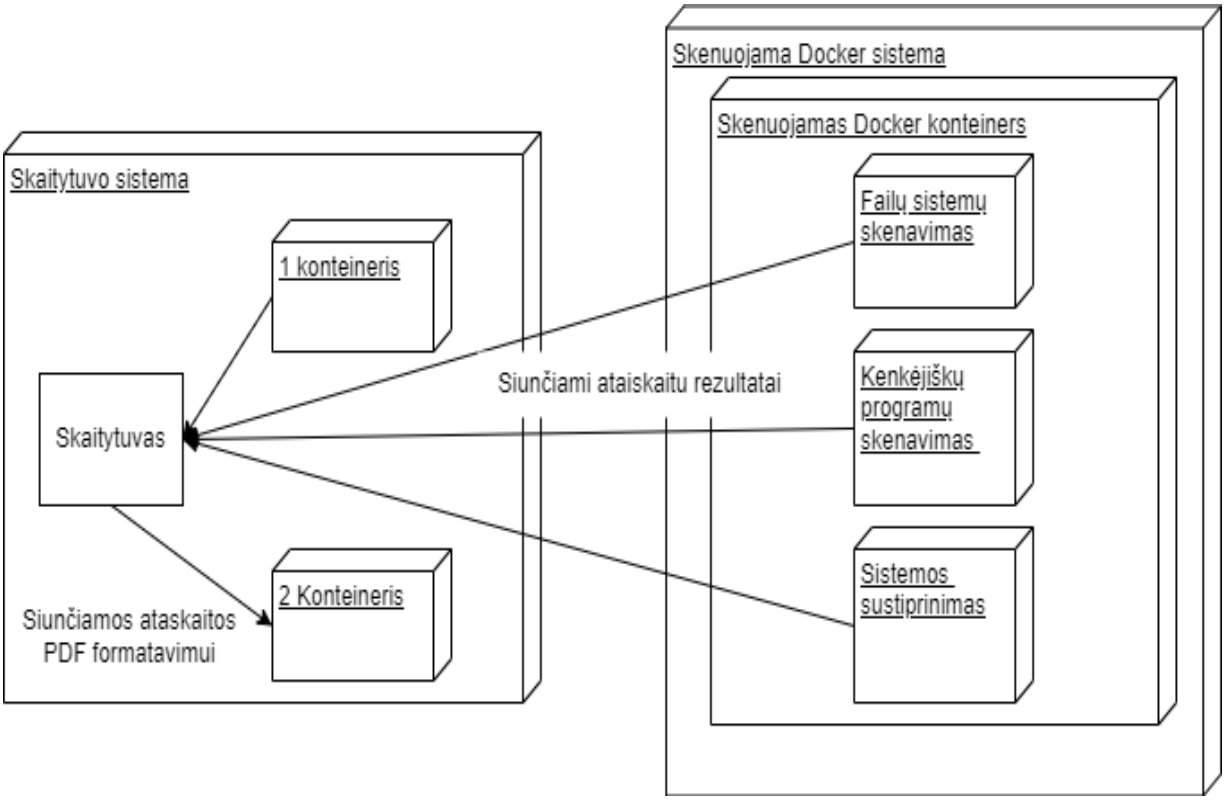
5 pav. Konteinerių sukurimas įgyvendinant skenavimo užklausas

Skaitytuvo sistemoje skenavimo procese yra kuriami trys Docker konteineriai pagal pasirinktus vaizdus kaip pavaizduota 5 pavyzdyje. Apie patį konteinerių kūrimo procesą plačiau buvo rašyta 4.3 skyriuje. Kiekvienam Docker konteinerio sėkmingam veikimui reikalingi skirtingi parametrai. Jie gaunami 4 pavyzdžio pirmajame punkte.



6 pav. Skaitytuvo skenavimo veikimas

Aiškesnis skaitytuvo veikimas pavaizduotas 6 pavyzdyje ir jame plačiau matoma kiekvieno proceso eiga. Pirmasis konteineris atlieka išorinę sistemos analizę kuri yra plačiau aprašyta 4.4.1 poskyryje. Viduje skenuojamo Docker konteinerio yra atliekami vidiniai sistemos skanavimai. Bandoma patikrinti failų sistemą, atrasti kenkėjiškas programas bei atlikti sistemos stiprinimą. Plačiau apie tai aprašyta 4.4 skyriuje atitinkamose poskyriuose.



7 pav. Rezultatų gražinimas

Įgyvendinus visus skenavimus rezultatai yra grąžinami skaitytuvui kuris nusiunčia juos į antrą konteinerį kuriame visi failai yra formatuojami iš HTML į PDF formatą. Visas aprašytas procesas pavaizduotas 7 pavyzdyje. Įvykdžius formatavimą audito ataskaita grąžinama skaitytuvui kur Rest API atlieką tolimesnius žingsnius ir nusiunčia visus duomenis į duomenų bazę.

4.6. Įrankio testavimas

Saugaus pažeidžiamumų skaitytuvo testavimams pirminėje kūrimo fazėje buvo naudojami atsitiktiniai parinkti Docker vaizdai kuriuose pavyko iš karto aptikti tam tikrus pažeidžiamumus. Gilesnei analizei ir pavyzdžiams buvo specialiai pasirinkti ir modifikuoti trys Docker vaizdai kuriuose apstu įvairaus pobūdžio sistemos spragų ir klaidų. Tokiu būdu norima buvo parodyti skaitytuvo efektyvumą ir įrodyti, kad kiekvienas iš pasirinktų įrankių veikia tinkamai ir pateikia kokybišką audito ataskaitą.

4.7. Aptiktini pažeidžiamumai

Šiuo metu skenavimo įrankis gali aptikti šiuo pažeidžiamumus:

- Aptinkami sistemos pažeidžiamumai iš išorės ir vidaus;
- Aptiktinos atgalinių durų pažeidžiamumus;
- Aptinkami infekuoti ir paslėpti Linux kernelio moduliai;
- Aptiktini failai ir direktorijos, prie kurių vartotojas neturėtų galimybės prieiti. Kaip pavyzdį galima būtų pateikti prieigą prie administratoriaus direktorijos failų sistemoje;
- Aptiktinkamos kenkėjiškos programos, virusai bei infekuoti failai;
- Aptinkami atidaryti prievadai.
- Ir dar papildomai begalė kitų smulkesnių spragų bei pažeidžiamumų;

4.8. Įgyvendinti uždaviniai

Įgyvendinti šie apsibrėžti uždaviniai:

- Sukurta saugi sistema kuri leidžia vartotojui generuoti audito ataskaitas pagal pasirinktus vaizdus ir atsisiųsti juos naršyklėje.;
- Sukurtas universalus skaitytuvas kuri galima lengvai modifikuoti ir atnaujinti poreikiui esant;
- Sukurta saugi aplinka kuri leidžia vartotojui skenuoti pavojingus Docker vaizdus nesibaiminant užkrėsti nuosavos sistemos;
- Aptinkami plataus spektro pažeidžiamumai;
- Sukuriama audito ataskaita PDF formatu kuri aprėpia daug naudingos informacijos ir yra lengvai įskaitoma vartotojui..

4.9. Darbo rezultatų apibendrinimas

Šio darbo metu buvo sėkmingai sukurtas saugus pažeidžiamumų skaitytuvas sistemų auditavimui pasinaudojant įvairiomis technologijomis. Pažeidžiamumų skaitytuvas turi dvi pagrindinės sistemas – internetinę svetainę ir duomenų bazę. Internetinėje svetainėje kaip posistemė buvo sukurtas skaitytuvas.

Skaitytuvas auditavimo metu įgyvendina keturis skirtingus skenavimus skirtingiems pažeidžiamumams aptikti. Sukurtas įrankis geba aptikti išorinius pažeidžiamumus skenuodamas sistemos išorę ir jos pažeidžiamumus, tikintis pastebėti atidarytus prievadus, sistemos operacinę versiją, veikiančias kitas programas bei sistemas. Taip pat besinaudojant specifiniais skriptais bandoma atrasti pažeidžiamumus iš išorės įimituojant kibernetinių nusikaltėlių veiksmus. Be to, įrankis sugeba patikrinti ar viduje tinkamai sukonfigūruota failų sistema ir uždėti tinkami ribojimai. Taip pat skaitytuvas aptinka įvairaus pobūdžio kenkėjiškas programas, net jei jos yra paslėptos sistemoje bei pavyksta pateikti rekomendacijas kaip sustiprinti sistemą.

Internetinė sistema yra sukurta bendrauti su vartotoju, pateikti auditavimo ataskaitas naršyklėje bei generuoti naujus skenavimus. Administratorius gali papildomai prisijungęs prie sistemos valdyti vartotojus ir matyti visas ataskaitas.

Išvados ir rekomendacijos

Išvados

Kuriant pažeidžiamumų skaitytuvą ir analizuojant susijusią medžiagą kyla natūrali išvada, kad pažeidžiamumų ir spragų sistemose tik daugės. Tai įrodo atlikti paprasti testai ir skanavimai. Jų metu pavyko surasti begalę klaidų ir grėsmių sistemose kurie gali būti kertinė organizacijų grėsmė. Todėl tikėtina, kad ir toliau institucijos ir didžiausios korporacijos nesustos investuoti į kibernetinę saugumo rinką. Šituos veiksnius pagrinde lemia didelis kiekis technologijų, įvairių virtualizacijos ir debesijos sprendimų bei naujų produktų kiekis.

Skaitytuvo kūrimas taip pat dėl technologijų įvairovės yra pakankamai sudėtinga užduotis reikalaujanti daug įvairių žinių ir patirties. Situaciją komplikuoja ir tai, kad virusai ir pažeidžiamumui tampa su kiekviena diena vis labiau komplikuoti ir patampa sudėtinga juos aptikti. Norint turėti tinkamą auditavimo įrankį tokioms sistemoms, procesas reikalauja daug darbo ir nuolatinės priežiūros kadangi nuolatos keičiasi įrankių versijos, atnaujinamos žinomų pažeidžiamumų duomenų bazės arba atsiranda nauja spraga sistemoje.

Rekomendacijos

- Vietoje MySQL naudoti PostgreSQL duomenų bazę taip palengvinant Rest API darbą bendraujant su duomenų baze ir autorizacijos procesus kuriant skenavimo užklausas;
- Skaitytuvą atskirti nuo internetinės svetainės taip įnešant papildomo stabilumo ir saugumo sistemai. Dabartiniu momentu skaitytuvą negalima naudoti kaip atskiro įrankio, kas apriboja sprendimo universalumą;
- Permąstyti Rest API architektūrą siekiant pagreitinti skanavimo greiti. Šiame darbe naudojamas visai nedidelis kiekis įrankių. Skaičiui padidėjus galimai atsiras problema, kad skenavimas užtrunka pernelyg ilgą laiką ir nėra patogus paprastam vartotojui;

Ateities darbų planas

Ateities darbų plano gairės:

- Papildyti skaitytuvą, kad būtų įmanoma skanuoti daugiau įvairių operacinių sistemų;
- Pritaikyti sistemą skanuoti Kubernečio klusterius;
- Papildyti skenavimu, kuris aptiktų spragas leidžiančias įgyvendinti SQL injekcijos atakas.

Literatūros šaltiniai

[1] Ergen, Ahu & Ünal, Ahmet & Saygili, Mehmet. (2021). Is It Possible to Change the Cyber Security Behaviours of Employees? Barriers and Promoters. *Academic Journal of Interdisciplinary Studies*. 10. 210. 10.36941/ajis-2021-0111.

[2] Nifakos, Sokratis & Chandramouli, Krishna & Nikolaou, Charoula & Papachristou, Panagiotis & Koch, Sabine & Panaousis, Emmanouil & Bonacina, Stefano. (2021). Influence of Human Factors on Cyber Security within Healthcare Organisations: A Systematic Review. Sensors. 21. 10.3390/s21155119.

[3] Harkin, Diarmaid & Molnar, Adam. (2022). Exploring the social implications of buying and selling cyber security. Crime, Law and Social Change. 1-18. 10.1007/s10611-022-10037-y.

[4] Bjorn Egil Asbjornslett. Assess the vulnerability of your production system. *Production Planning & Control*, 10(3):219–229, 1999.

[5] Roy Fielding, Jim Gettys, Je rey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1, 1999.

[6] Ron Gula. Passive vulnerability detection. *Network Security Wizards*, 9:7, 1999.

[7] Paul R McWhirter, Kashif Kifayat, Qi Shi, and Bob Askwith. Sql injection attack classification through the feature extraction of sql query strings using a gap-weighted string subsequence kernel. *Journal of information security and applications*, 40:199–216, 2018.

[8] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[9] Angela Orebaugh and Becky Pinkard. *Nmap in the Enterprise: Your Guide to Network Scanning*. Syngress Publishing, 2008.

[10] Eric S Raymond and Tim O'Reilly. *The Cathedral and the Bazaar*. O'Reilly & Associates, Inc., USA, 1st edition, 1999.

[11] Sapiah Sakri. Intrusion detection and prevention. 2004.

[12] Carla Sayan, Salim Hariri, and George L Ball. Semantic knowledge architecture for cyber security. In *Proceedings of the International Conference on Security and Management (SAM)*, pages 69–76. The Steering Committee of The World Congress in Computer Science, Computer …, 2019.

[13] Bela, Genge & Graur, Flavius & Enachescu, Calin. (2015). Non-intrusive Techniques for Vulnerability Assessment of Services in Distributed Systems. Procedia Technology. 19. 10.1016/j.protcy.2015.02.003.

[14] Simon Parkinson, Andrew Crampton, Richard Hill. (2018). Guide to Vulnerability Analysis for Computer Networks and Systems: An Artificial Intelligence Approach

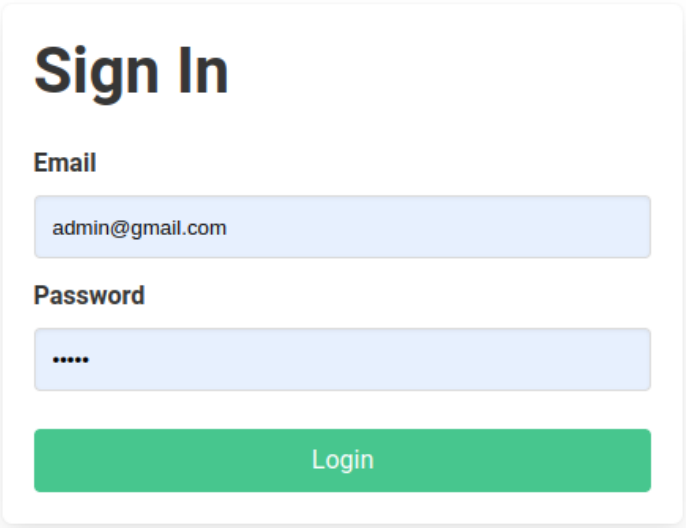
[15] Kalsi T. Practical Linux Security Cookbook : Secure Your Linux Environment from Modern-Day Attacks with Practical Recipes. Second ed. Birmingham UK: Packt Publishing; 2018.

[16] Hixon M Hutchens J. Kali Linux Network Scanning Cookbook : Take Your Penetration-Testing Skills to the Next Level. Second ed. Birmingham UK: Packt Publishing; 2017.

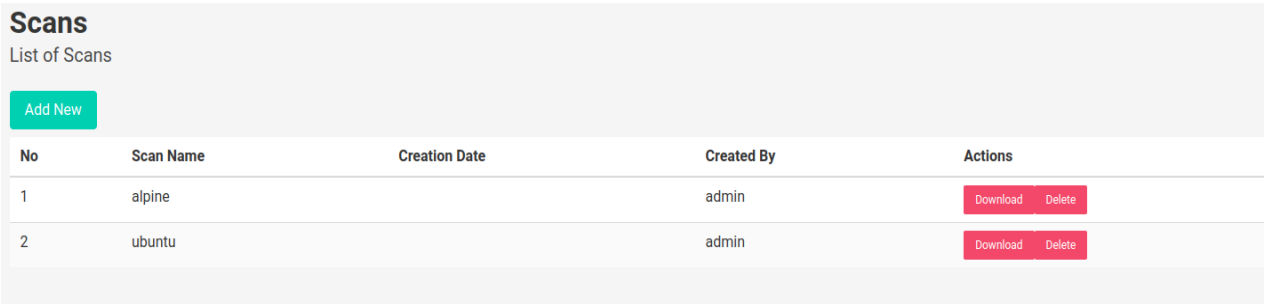
[17] Sairam J. Network Scanning Cookbook. Packt Publishing; 2018.

[18] Peter Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley Professional, 2005.

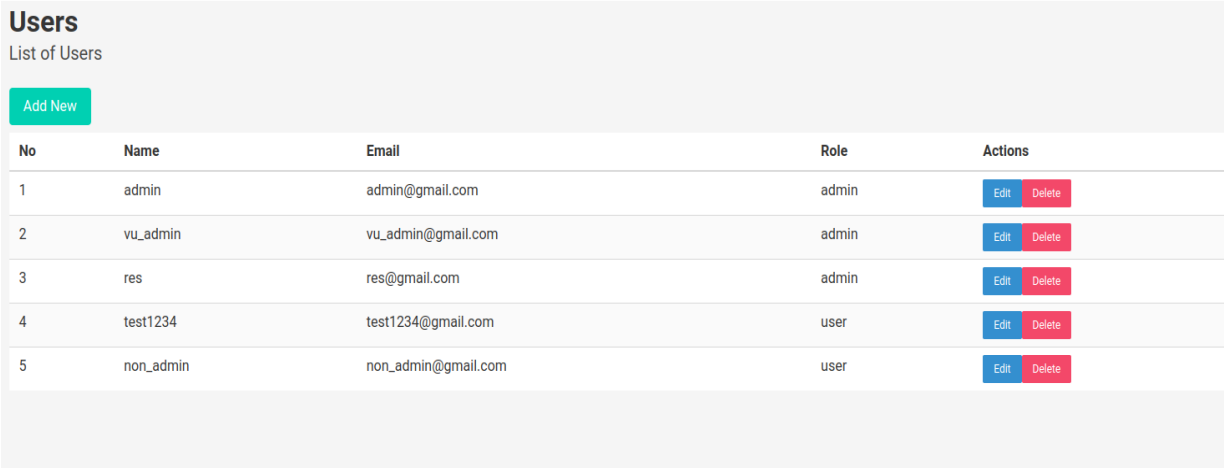
Priedai



8 pav. Prisijungimo puslapis



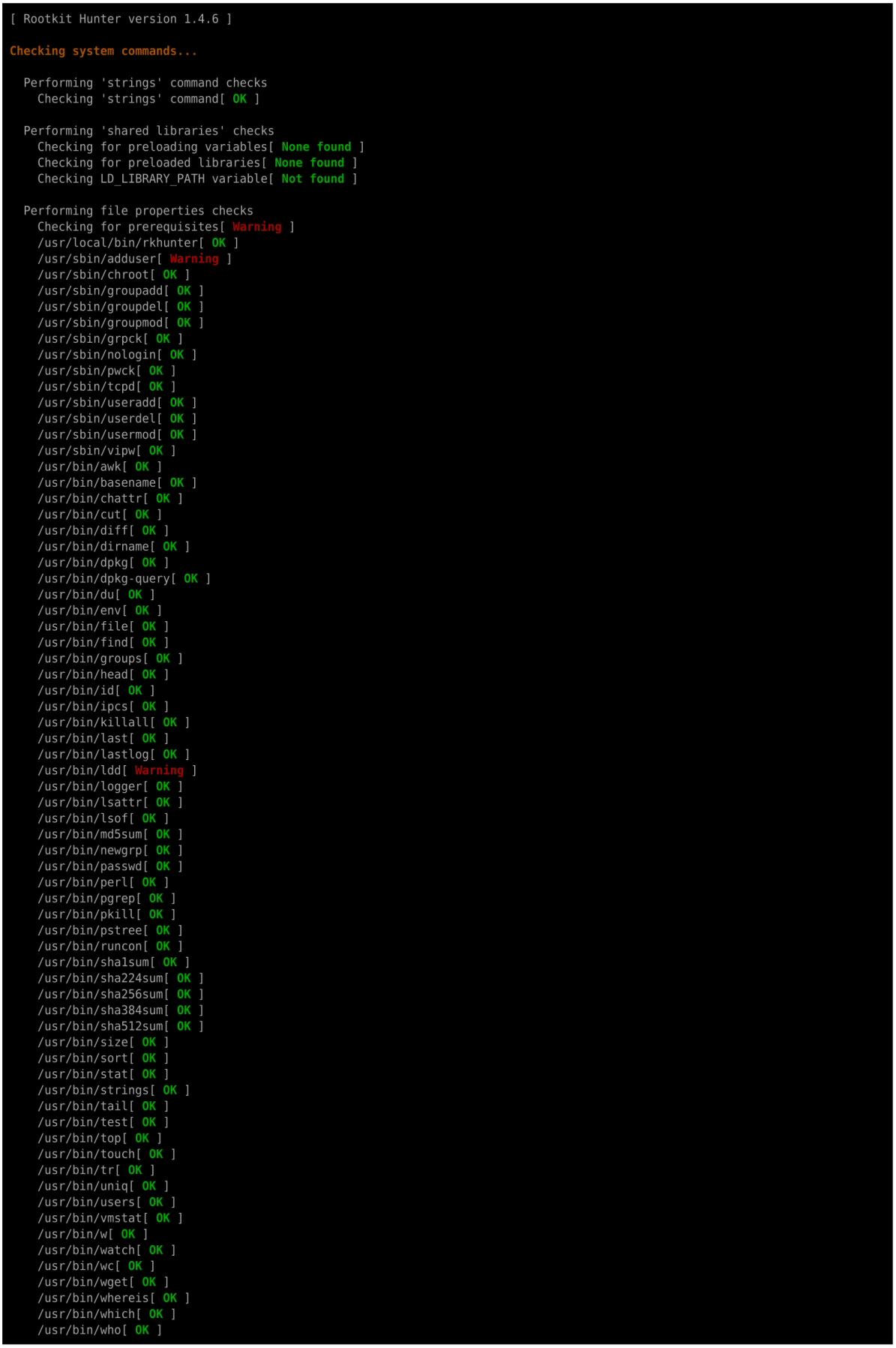
9 pav. Skenavimo rezultatų puslapis



10 pav. Vartotojų valdymo puslapis



11 pav. Skenavimo pavyzdys



12 pav. Skenavimo pavyzdys

Nmap Scan Report - Scanned at Mon Dec 19 15:21:39 2022

Scan Summary | **172.17.0.3**

Scan Summary

Nmap 7.80 was initiated at Mon Dec 19 15:21:39 2022 with these arguments:
nmap -sV --script vulners -oX /nmap2.xml 172.17.0.3

Verbosity: 0; Debug level 0

Nmap done at Mon Dec 19 15:21:47 2022; 1 IP address (1 host up) scanned in 7.15 seconds

172.17.0.3

Address

- 172.17.0.3 (ipv4)
- 02:42:AC:11:00:03 (mac)

Ports

The 999 ports scanned but not shown below are in state: **closed**

- 999 ports replied with: **resets**

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
80	tcp	open	http	syn-ack	Apache httpd	2.4.25	(Debian)

	http-server-header	Apache/2.4.25 (Debian)
	vulners	cpe:/a:apache:http_server:2.4.25: CVE-2022-31813 7.5 https://vulners.com/cve/CVE-2022-31813 CVE-2022-23943 7.5 https://vulners.com/cve/CVE-2022-23943 CVE-2022-22720 7.5 https://vulners.com/cve/CVE-2022-22720 CVE-2021-44790 7.5 https://vulners.com/cve/CVE-2021-44790 CVE-2021-39275 7.5 https://vulners.com/cve/CVE-2021-39275 CVE-2021-26691 7.5 https://vulners.com/cve/CVE-2021-26691 CVE-2017-7679 7.5 https://vulners.com/cve/CVE-2017-7679 CVE-2017-3169 7.5 https://vulners.com/cve/CVE-2017-3169 CVE-2017-3167 7.5 https://vulners.com/cve/CVE-2017-3167 CNVD-2022-73123 7.5 https://vulners.com/cnvd/CNVD-2022-73123 CNVD-2022-03225 7.5 https://vulners.com/cnvd/CNVD-2022-03225 CNVD-2021-102386 7.5 https://vulners.com/cnvd/CNVD-2021-102386 EXPLOITPACK:44C5118F831D55FAF4259C41D8BDA0AB 7.2 https://vulners.com/exploitpack/EXPLOITPACK:44C5118F831D55FAF4259C41D8BDA0AB *EXPLOIT* EDB-ID:46676 7.2 https://vulners.com/exploitdb/EDB-ID:46676 *EXPLOIT* CVE-2019-0211 7.2 https://vulners.com/cve/CVE-2019-0211 1337DAY-ID-32502 7.2 https://vulners.com/zdt/1337DAY-ID-32502 *EXPLOIT* FDF3DFA1-ED74-5EE2-BF5C-BA752CA34AE8 6.8 https://vulners.com/githubexploit/FDF3DFA1-ED74-5EE2-BF5C-BA752CA34AE8 *EXPLOIT* CVE-2021-40438 6.8 https://vulners.com/cve/CVE-2021-40438 CVE-2020-35452 6.8 https://vulners.com/cve/CVE-2020-35452 CVE-2018-1312 6.8 https://vulners.com/cve/CVE-2018-1312 CVE-2017-15715 6.8 https://vulners.com/cve/CVE-2017-15715 CNVD-2022-03224 6.8 https://vulners.com/cnvd/CNVD-2022-03224 8AFB43C5-ABD4-52AD-BB19-24D7884FF2A2 6.8 https://vulners.com/githubexploit/8AFB43C5-ABD4-52AD-BB19-24D7884FF2A2 *EXPLOIT* 4810E2D9-AC5F-5B08-BFB3-DDAFA2F63332 6.8 https://vulners.com/githubexploit/4810E2D9-AC5F-5B08-BFB3-DDAFA2F63332 *EXPLOIT* 4373C92A-2755-5538-9C91-0469C995AA9B 6.8 https://vulners.com/githubexploit/4373C92A-2755-5538-9C91-0469C995AA9B *EXPLOIT* 0095E929-7573-5E4A-A7FA-F6598A35E8DE 6.8 https://vulners.com/githubexploit/0095E929-7573-5E4A-A7FA-F6598A35E8DE *EXPLOIT* CVE-2022-28615 6.4 https://vulners.com/cve/CVE-2022-28615 CVE-2021-44224 6.4 https://vulners.com/cve/CVE-2021-44224 CVE-2019-10082 6.4 https://vulners.com/cve/CVE-2019-10082 CVE-2017-9788 6.4 https://vulners.com/cve/CVE-2017-9788 CVE-2019-0217 6.0 https://vulners.com/cve/CVE-2019-0217 CVE-2022-22721 5.8 https://vulners.com/cve/CVE-2022-22721 CVE-2020-1927 5.8 https://vulners.com/cve/CVE-2020-1927 CVE-2019-10098 5.8 https://vulners.com/cve/CVE-2019-10098 1337DAY-ID-33577 5.8 https://vulners.com/zdt/1337DAY-ID-33577 *EXPLOIT* SSV:96537 5.0 https://vulners.com/seebug/SSV:96537 *EXPLOIT* EXPLOITPACK:C8C256BE0BFF5FE1C0405CB0AA9C075D 5.0 https://vulners.com/exploitpack/EXPLOITPACK:C8C256BE0BFF5FE1C0405CB0AA9C075D *EXPLOIT* EDB-ID:42745 5.0 https://vulners.com/exploitdb/EDB-ID:42745 *EXPLOIT* CVE-2022-30556 5.0 https://vulners.com/cve/CVE-2022-30556 CVE-2022-29404 5.0 https://vulners.com/cve/CVE-2022-29404 CVE-2022-28614 5.0 https://vulners.com/cve/CVE-2022-28614 CVE-2022-26377 5.0 https://vulners.com/cve/CVE-2022-26377 CVE-2022-22719 5.0 https://vulners.com/cve/CVE-2022-22719 CVE-2021-34798 5.0 https://vulners.com/cve/CVE-2021-34798 CVE-2021-33193 5.0 https://vulners.com/cve/CVE-2021-33193 CVE-2021-26690 5.0 https://vulners.com/cve/CVE-2021-26690 CVE-2020-9490 5.0 https://vulners.com/cve/CVE-2020-9490 CVE-2020-1934 5.0 https://vulners.com/cve/CVE-2020-1934 CVE-2019-17567 5.0 https://vulners.com/cve/CVE-2019-17567 CVE-2019-10081 5.0 https://vulners.com/cve/CVE-2019-10081 CVE-2019-0220 5.0 https://vulners.com/cve/CVE-2019-0220 CVE-2019-0196 5.0 https://vulners.com/cve/CVE-2019-0196 CVE-2018-17199 5.0 https://vulners.com/cve/CVE-2018-17199 CVE-2018-17189 5.0 https://vulners.com/cve/CVE-2018-17189 CVE-2018-1333 5.0 https://vulners.com/cve/CVE-2018-1333 CVE-2018-1303 5.0 https://vulners.com/cve/CVE-2018-1303 CVE-2017-9798 5.0 https://vulners.com/cve/CVE-2017-9798 CVE-2017-7668 5.0 https://vulners.com/cve/CVE-2017-7668 CVE-2017-7659 5.0 https://vulners.com/cve/CVE-2017-7659 CVE-2017-15710 5.0 https://vulners.com/cve/CVE-2017-15710 CNVD-2022-73122 5.0 https://vulners.com/cnvd/CNVD-2022-73122 CNVD-2022-53584 5.0 https://vulners.com/cnvd/CNVD-2022-53584 CNVD-2022-53582 5.0 https://vulners.com/cnvd/CNVD-2022-53582 CNVD-2022-03223 5.0 https://vulners.com/cnvd/CNVD-2022-03223 1337DAY-ID-28573 5.0 https://vulners.com/zdt/1337DAY-ID-28573 *EXPLOIT* CVE-2020-11993 4.3 https://vulners.com/cve/CVE-2020-11993 CVE-2019-10092 4.3 https://vulners.com/cve/CVE-2019-10092 CVE-2018-1302 4.3 https://vulners.com/cve/CVE-2018-1302 CVE-2018-1301 4.3 https://vulners.com/cve/CVE-2018-1301 CVE-2018-11763 4.3 https://vulners.com/cve/CVE-2018-11763 4013EC74-B3C1-5D95-938A-54197A58586D 4.3 https://vulners.com/githubexploit/4013EC74-B3C1-5D95-938A-54197A58586D *EXPLOIT* 1337DAY-ID-35422 4.3 https://vulners.com/zdt/1337DAY-ID-35422 *EXPLOIT* 1337DAY-ID-33575 4.3 https://vulners.com/zdt/1337DAY-ID-33575 *EXPLOIT* CVE-2018-1283 3.5 https://vulners.com/cve/CVE-2018-1283 PACKETSTORM:152441 0.0 https://vulners.com/packetstorm/PACKETSTORM:152441 *EXPLOIT*

Go to top

Toggle Closed Ports

Toggle Filtered Ports

Misc Metrics (click to expand)

13 pav. Skenavimo pavyzdys