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

Bakalauro darbas

Saugus pažeidžiamumų skaitytuvas sistemų auditavimui

Atliko:
Jonas Gavėnavičius parašas

Vadovas:
Lektorius Virgilijus Krinickij

Vilnius
2019

Turinys

Sutartinis terminų žodynas

- FTP - *File Transfer protocol*, protokolas leidžiantis perkelti duomenis iš vienos sitemos į kitą.
- Buffer overflow - tai viena iš potencialių rizikų, kai dėl per didelio kiekio duomenų, informacija yra perrašoma gretimuose atminties blokuose.
- SSH - *Secure Shell*, tai tinklo protokolas kuris leidžia vartotojui saugiai pasiekti sistemą per nesaugu tinklą.
- Framework - Tai karkasas, į kurį įeina daug programinės įrangos ar kitų sistemų.
- NASL - *Nessus Attack Scripting Language*, tai paprastą kalbą, apibūdinanti atskiras grėsmes ir galimas atakas.
- IPS - *Intrusion prevention system*
- IDS - *Intrusion detection system*
- HTTP - *HyperText Transfer Protocol* tai informacijos priemimo perdavimo protokolas.
- Daemon - Tai programa kuri veikia fone ir nėra valdoma vartotojo, bet laukia specifinio įvykio ar salygos suveikti.
- Backdoor - Tai kodo dalis kuri leidžia atakuotojuj į sistemą patekti nepastebėtam.

Santrauka

Šiais laikais kai pasaulis tampa vis labiau ir labiau skaitmenizuotas, iškyla pati opiausia problema, tai yra kibernetine sauga. Vis daugiau pavyždžių matome kaip įsibrauna į sistemas kurias galima potencialiai išnaudoti dėl finansinių ar kitų priežasčiu. Taip pat dėl to nukenčia ir tų sistemų vartotojai - finansiškai ar morališkai, jų privati informacija būna pavogiama ir paviešinama. Niekas nėra saugus nuo tokių situacijų. Todėl valstybės, įmonės ar korporacijos vis daugiau investuoja į kibernetinę apsaugą, kibernetinė sauga tampa vis dažnesnė diskusija visuomenėja, vis daugiau dėmėsio ir resursų skiriama butent jai, stengemasi užkirsti kelią minėtoms situacijoms. Kuriami įrankiai kurie analizuoja sistemas ir randa jų spragas, pritaikomos įvairios technologijos ar metodai kurie perspėja apie galima ar vykstančią ataką prieš sistemą, stengiamasi rasti sistemoje spragas ir užlopyti jas kol jų nerado asmenys kurie išnaudotų jas.

Summary

Darbo pavadinimas kita kalba

This is a summary in English...

Įvadas

Šiame darbe kuriamas saugus įrankis, kuris neišduotų savo serverio vietos ir vartotojai pasinaudoję juo galėtų gauti išsamią informaciją apie jų internetinėje svetainėje egzistuojančias spragas bei pažeidžiamumus ar modifikuotus failus. Darbo tikslas – Sukurti saugų pažeidžiamumų skenavimo įrankį, kuris veiktu iš tam tikros slepiamos vietos, skenuotų internetinę svetainę bei jos failus, aptiktų pažeidžiamumus ar modifikuotus failus, ir pateiktų klientui informaciją apie skenavimo rezultatus. Siekiant, kad darbas būtų paklausus ir veiksmingas, keliami šie darbo uždaviniai:

1. Apžvelgti egzistuojančius panašius įrankius;
2. Išanalizuoti dažniausiai pasitaikančius internetinių svetainių pažeidžiamumus;
3. Išanalizuoti dažniausiai pasitaikančių pažeidžiamumų egzistuojančius atviro kodo įrankius;
4. Išanalizuoti dažniausiai pasitaikančius internetinių svetainių skenavimo metodus.
5. Sukurti įranki, kuris būtu saugus pačiam jo naudotojui.

Kibernetinis saugumas yra svarbi kiekvienos sistemos dalis. Kadangi kiekvieną sistemą kuria žmogus, ir į kiekvieną sistemą įeina žmogiškasis faktorius, dėl kurio sistemos beveik visada turi pažeidžiamumų, kuriuos gali išnaudoti puolėjas siekiantis tam tikros naudos. Dėl šios priežasties toks įrankis būtų ypač naudingas bet kokiai sistemai. Aptikus pažeidžiamumą, galima įtarti, kad tą pažeidžiamumą gali aptikti ir nuostolio siekiantys asmenys, arba jis jau buvo aptiktas, ir tam tikri failai buvo modifikuoti įdedant tam tikrą kodą, kuris leistų atakuojančiui asmeniui patekti į sistemą nepastebėtam. Dėl šių priežasčių saugus ir automatizuotas sistemų pažeidžiamumo skenavimo įrankis yra ypač aktualus. Tačiau šio įrankio kūrimą apsunkina šios priežastys:

- Įrankio kūrimas reikalauja didelio bagažo žinių;
- Daugumos sistemų, kurios turi pažeidžiamas vietas, išeities kodas nėra atviras, todėl kai kurios spragos negali būti aptiktos automatizuotu įrankiu. Taip pat negalima atlikti kai kurių sistemos kodo analizių, kas taip pat sumažina tokio įrankio efektyvumą;
- Sistemoje gali būti tiek daug skirtingų potencialių pažeidžiamų, jog jų aptikimo automatizavimas tampa problematiškas.

Šias problemas siūloma spręsti apskaičiuojant tam tikrą įrankio veiksmingumą procentaliai. Didžiausias tokio sprendimo privalumas yra tas, kad vartotojas supras, jog tokio įrankio rezultatų analizavimas ir pritaikymas neužtikrina visų pažeidžiamumų aptikimo ir kibernetinio saugumo užtikrinimo. Šio įrankio kūrimo metu, siekiant sukurti saugų pažeidžiamumų aptikimo įrankį, siekiama pagerinti sistemos saugumą, bet neužtikrinti jo. Įrankis bus skirtas tik internetinėms svetainėms ir nebus stengiamasi jo pritaikyti ir kitoms sistemoms.

1. Susijusių darbų analizė

1.1. Nessus skaitytuvas

1.1.1. Įrankio aprašymas

„Nessus“ įrankis yra tinklo pažeidžiamumų skaitytuvas, kuris naudoja bendrąją pažeidžiamumų architektūrą, kad lengvai susietų suderinamus kibernetinio saugumo įrankius. „Nessus“ naudoja *NASL*.

„Nessus“ turi modulinę architektūrą, susidedančią iš serverio *daemon* atliekančio nuskaitymą, ir nuotolinų kliento kuris yra valdomas administratoriaus. Administratoriai gali įtraukti *NASL* visų įtariamų pažeidžiamumų aprašus, kad sukurtų tinkintus nuskaitymus. Reikšmingos „Nessus“ galimybės:

- Suderinamumas su bet kokio dydžio kompiuteriais ir serveriais.
- Apsaugos spragų aptikimas vietiniuose ar nuotoliniuose kompiuteriuose.
- Trūkstamų sistemų ir programinės įrangos saugumo atnaujinimų aptikimas.
- Imituoti išpuoliai, skirti nustatyti pažeidžiamumą.
- Saugumo testų atlikimas uždaroje aplinkoje.
- Suplanuotas saugumo auditas.

„Nessus“ serverį šiuo metu galima naudoti „Unix“, „Linux“ ir „FreeBSD“. Klientas yra prieinamas „Unix“ arba „Windows“ operacinėms sistemoms.

1.1.2. Panaudojimas darbe

Darbe bus naudojama "Nessus" architektūra, vartotojo sąsaja nebus tiesiogiai bendraujanti su pačių serveriu kuris vykdys skanavimus, taip jie bus nepriklausomi vienas nuo kito. Taip pat serveris susidarys iš skirtingų modulių, kurie veiks padedami docker technologijos, bus kuriami atskiri konteineriai ir vykdomos atskiros izoliutos operacijos tam, kad sumažinti riziką infekuoti patį serverį.

1.2. OpenVAS skaitytuvas

1.2.1. Įrankio aprašymas

„OpenVAS“ yra visa apimantis pažeidžiamumų skaitytuvas. Jo galimybės apima įvairių aukšto ir žemo lygio interneto ir pramoninių protokolų skanavimą, našumo derinimą didelės apimties nuskaitymams ir galingą vidinę programavimo kalbą, kad būtų galima įgyvendinti bet kokio tipo pažeidžiamumo testą.

1.2.2. Irankio ištakos

2006 m. Buvo sukurtos kelios „Nessus“ atviro kodo atšakos, kaip reakcija į "Nessus" įrankio komercilizavimą nebepalaikant atviro kodo. Iš šių šakų tik viena toliau rodė aktyvumą: „OpenVAS“, atviro kodo pažeidžiamumų skenavimo sistema. „OpenVAS“ buvo įregistruotas kaip „Software in the Public Interest, Inc.“ projektas, skirtas valdyti ir apsaugoti domeną „openvas.org“.

Dėl šios priežasties abu įrankiai yra panašus. Didžiausias tarp jų skirtumas yra tas, kad „Nessus“ įrankis yra komercializuotas, priešingai negu „OpenVAS“.

1.2.3. Panaudojimas darbe

1.3. Nmap

1.3.1. Įrankio aprašymas

„Nmap“ („Network Mapper“) yra nemokamas ir atvirojo kodo įrankis skirtas tinklo skanavimui. Daugelis sistemų ir tinklo administratorių mano, kad šis įrankis naudingas atliekant tokias užduotis kaip tinklo inventorizavimas ir pagrindinio kompiuterio ar paslaugos veikimo stebėjimas. „Nmap“ naudoja neapdorotus IP paketus, kas taip pat padeda įrankiui būti daug efektyviasniam. Įrankis buvo sukurtas greitai nuskaityti didelius tinklus, tačiau puikiai veikia su atskirais kompiuteriais ar serveriais. „Nmap“ veikia visose pagrindinėse kompiuterių operacinėse sistemose „Linux“, „Windows“ ir „Mac OS X“. [4]

Irankio skanavimo galimybes:

- Tinklo skanavimas: "Nmap" gali identifikuoti tinkle visus esančius įrenginius, tokius kaip serverius, maršrutizatorius, kelvedžius, taip pat kaip jie yra sujungti;
- Operacinės sistemos aptikimas: "Nmap" gali identifikuoti, kokia operacinė sistema veikia pasirinktame įrenginyje, kiek laiko įrenginys jau yra aktyvus, programinės įrangos versijas;
- "Nmap" įrankis ne tik aptinka įrenginius tinkle, bet taip pat kokia jų paskirtis, ar tai yra internetinės sistemos serveris ar pašto serveris, ar kažkas kito, taip pat jis aptinka su tuo susijusios programinės įrangos versijas;
- Saugumo auditavimas: Taip pat "Nmap" aptinka kokias ugniasienes ar paketų filtrus pasirinktasis irenginys naudoja.

1.3.2. Panaudojimas darbe

Įrankis yra puikiai pritaikytas naudoti bet kokioje sistemoje, taip pat jį ypač patogu pritaikyti bet kokia scenarijuje. Šis įrankis yra plačiai naudojamas kibernetinio saugumo bendruomeneje ir taip pat yra ypač efektyvus atliekant tinklo skanavimus ar analizę. Atsisžvelgus į šiuos faktus, šis įrankis tampa butinybė bet kokioje spragų skanavimo sistemoje dėl savo didelio potencialo ir bendruomenės pasitikėjimo.

2. Sistemų auditavimas

2.1. Statinė kodo analizė

2.1.1. Panaudojimas

Statinė analizė suteikia galimybę gauti informacijos apie galimą programos elgesį programos vykdymo metu, nevykdant programos. Statinė analizė tiria išeities kodą ir ieško įtartinų kodo segmentų kurie galėtu turėti spragą. Atlikus teisingai statinę analizę, galima aptikti akivaizdžias klaidas kurių programuotojas galėjo nepastebėti, tai sutaupo laiko bei sumažina spragų kiekį, taip pat galimai aptinkami nenumatyti scenarijai. Kai kurios programavimo aplinkos (Visual Studio, IntelliJ...) atlieka pastovią statine analizę tam, kad programuotojai pamatytų potencialias klaidas prieš sistemos startą. [1]

Statinė analizė padeda aptikti:

- Neįcituotus kintamuosius;
- Potencialias klaidas sistemos išeities kode;
- *Buffer overflow* spragas.

2.1.2. Panaudojimas darbe

Vartotojui davus *FTP* serverio adresą iš jo bus bandoma atsisiūsti išeities kodą. Atsisiuntus sistemos išeities kodą bus atliekama statinė analizė ir taip bus bandoma aptikti spragas ar scenarijus, kurie kelią potencialią grėsmę pačiai sistemai. Tokios grėsmės būtų - įsilaužimas, arba sistemos veikimo paveikimas, sugadinimas.

2.2. Dinaminė kodo analizė

2.2.1. Panaudojimas

Dinaminė analizė vykdoma kai programa jau yra vykdomo stadijoje. Dynaminės analizės metu bandoma igyvendinti visus įmanomus scenarijus ir išbandyti visas imanomas įvesčių variacijas suvedant jas į programos ivestį.

Veikimo metu programa gali neatlaisvinti atminties atgal į operacinę sistemą, to pasekoje serveris kuriame programa veikia, pritruks atminties ir pradės veikti lečiau kol galiausiai sustos. Nuo to padėtų apsaugoti dinaminė analizė, atlikus ją teisingai, galima aptikti didžiają dalį spragų kurios potencialiai labiausiai įtakos sistemą. Jas ištaisius, sistema veikimo stabilumas padidėja, nenumatytų scenarijų skaičius taip pat pamažėja.

Dinaminė analizė padeda aptikti:

- Atminties nutekėjimus;
- Netikėtus scenarijus;
- Opiausias spragas;

2.2.2. Panaudojimas darbe

Dinaminė analizė bus taikoma spragų skenavimo sistemoje su kliento sutikimu. Analizės metu, bus bandoma į visas sistemos įvestis pateikti nenumatytu reikšmių, taip išbandant kuo daugiau nenumatytų scenarijų.

2.3. Išorinių spragų skenavimas

2.3.1. Panaudojimas

Išorinis pažeidžiamumų skenavimas atliekamas iš sistemos tinklo išorės, o pagrindinis jo tikslas yra aptikti perimetro gynybos spragas, pavyzdžiui: atvirus tinklo užkardos prievadus ar specializuotą žiniatinklio programų užkardą. Išorinis pažeidžiamumų skenavimas gali padėti organizacijoms išspręsti saugumo problemas, kurios įsilaužėliams galėtų suteikti prieigą prie organizacijos tinklo.
Išorinis pažeidžiamumų skenavimas aptiks:

- Didžiausios tiesioginės grėsmės sistemoje;
- Programinę įrangą kuriai reikia atnaujinimų bei priežiuros;
- Atidaryti prievadus ir protokolus - įėjimo taškus į sistemos tinklą;

2.3.2. Panaudojimas darbe

Testavimo įrankyje bus implementuota prievadų skanavimo funkcija naudojant "Nmap" atviro kodo įrankį , taip pat bus bandoma išgauti informacija apie pačią sistemą, kokia operacinę sistemą pati sistema naudoja, kokia tos operacinės sistemos versija, atidaryti prievadai, taip pat bus tikrinama ar prie tam tikrų prievadų galima bandyti jungtis. Pagal šia informaciją, galima suprasti, koki atakos vektorių galimai pasirinktų pats puolėjas.

2.4. Vidinių pažeidžiamumų skenavimas

2.4.1. Panaudojimas

Vidinis pažeidžiamumo patikrinimas atliekamas iš organizacijos perimetro gynybos. Jos tikslas yra aptikti pažeidžiamumus, kuriuos galėtų išnaudoti įsilaužėliai arba nepatenkinti darbuotojai, sėkmingai įsiskverbiantys į perimetro gynybą, arba turintys teisėtą prieigą prie organizacijos tinklo.
Vidinių pažeidžiamumų skenavimas aptiks:

- Sistemos komponentus kurie galimai gali sukelti gresmę;
- Pasenusi programinė įranga, kuriai reikia atnaujinimų;

2.4.2. Panaudojimas darbe

Įrankio vartotojas savo noru gali pateikti prisijungimus prie sistemos naudojant *SSH* protokolą. Pasinaudojus šia informaciją įrankis prisijungs prie sistemos, ir naudodamas vidinės analizės komponentą, bandys surinkti kuo daugiau informacijos. Surinkus visą galimą informaciją, bus ieškomos potencialios spragos.

2.5. Oligomorfinių virusų skenavimas

Virusų kurėjai greitai suprato, kad užšifruotą virusą antivirusinei programinei įrangai aptikti yra paprasta, kol paties iššifruotojo kodas yra pakankamai ilgas ir pakankamai unikalus. Norėdami apgauti antivirusinius produktus, jie nusprendė įgyvendinti techninį ieškojimą, jie nusprendė įdiegti mutavusių iššifruoklių kūrimo būdus. [5]

2.6. Polimorfinių virusų skenavimas

Polimorfiniai virusai gali iššifruoti jų iššifratorius iki daugybės skirtingų atvejų, kurie gali pasireikšti milijonais skirtingų formų. [5]

2.7. Metamorfinių virusų skenavimas

Metamorfiniai virusai neturi iššifruotojo ar nuolatinio viruso kūno, tačiau sugeba sukurti naujas kartas, kurios atrodo kitaip. jie nenaudoja duomenų srities užpildo su styginių konstantomis, tačiau turi vieną vieno kodo pagrindą, kuris duomenis kaupia kaip kodą. [5]

3. Gerosios praktikos

3.1. Atviro ir uždaro kodo programinė įranga

Visame pasaulyje vis daugiau dėmesio skiriama atvirojo kodo programinei įrangai, ypač operacinei sistemai "Linux" ir įvairioms programoms kurios būtent veikia su šia operacinę sistema. Įvairios didžiosios įmonės ir vyriausybės vis labiau priema atviro kodo modelį. Dėl to yra daugybė publikacijų apie atviro kodo pranašumus ir trūkumus. Vykstančios diskusijos apima platų temų spektrą, pavyzdžiui, „Windows“ lyginimas su „Linux“, išlaidų klausimus, intelektinės nuosavybės teises, kūrimo metodus ir panašias temas. Atkreipiant dėmesį būtent į saugumo problemas susijusias su atviro kodo metodika, kompiuterių saugumo bendruomenėje tapo gana nusistovėjęs įsitikinimas, kad dizaino ir protokolų publikavimas prisideda prie jų pagrindu sukurtų sistemų saugumo. [2] Bet ar iš tiesų išeities kodo publikavimas prisideda sistemos saugumo daugiau negu uždaras išeities kodas? Šis klausimas sukelia daug diskusijų ir vieno aiškaus atsakymo niekada nebūna, dauguma specialistų sutinka su tokia nuomone, kad tiek uždaras kodas, tiek atviras kodas turi savų pliusų ir minusų. Todėl peršasi išvada, kad paprasto atsakymo nėra į šį klausimą, ir vienintelis sprendimas tokiai dilemai yra įsigilinti į abi šias metodikas, ir išsiaiškinti, kuo viena metodika pranašesnė už kitą, ir kur atsiranda trūkūmų

3.1.1. Trūkumai atviro kodo programinės įrangos

Argumentai prieš atvirą kodą:

- Atviras kodas suteikia didelį pranašumą atakuojančiui asmeniui dėl spragų radimo, atakuojančiam asmeniui reikia surasti vieną spragą su kuria jis galėtu sekmingai užpulti sistemą, o programuotojams reikia ištaisyti visas spragas, kurios leistu atakuojančiam asmeniui sekmingai užpulti sistemą.[2]

- Yra didelis skirtumas tarp atviro dizaino ir atviro kodo. Atviras dizainas gali atskleisti logines klaidas kurios gali pakenkti sistemos saugumui. Bet skyrus pakankamai dėmesio tam, šios klaidos gali būti rastos ir ištaisytos, skirtingai negu atvirame kode kur klaidos yra aptinkamos daug sunkiau. [2]

- Atakuotojai gali apsimesti programuotojais kurie nori prisidėti prie atviro kodo sistemos kurimo ir palaikymo siulydami savo pataisymus kuriuose slepiasi *backdoor* ar kitas klaidinantis kodas kuris iš pirmo žvilgsnio atlieka savo funkciją, bet įsigilinus matosi, kad tas kodas skirtas suteikti pranašumą puolėjuj. [6]

- Kodo uždarymas užkerta kelią atakuojančiui asmeniui lengvai gauti informacijos apie sistemą ir jos spragas, priešingai negu laikant kodą atvira. Laikant kodą atvira atakuojantis asmuo gali labai lengvai rasti spragas, kurios jam padėtų įsilaužti į sistemą, arba jai pakenkti. [2]

- Viena iš didžiausių priežasčių kodėl atviras kodas nėra idealus pasirinkimas yra tai, kad kodo atvirumas negarantuoja, kad kodą peržiurės kvalifikuoti specialistai, kurie suteiks savo įžvalgas. Atviro kodo projektai daug dažniau nustoja būti aktyvus negu uždaro kodo projektai, kadangi jie būna nebepalaikomi, rastos klaidos nebėra taisomos, o sistemą kuri naudoja tokį kodą turi didelią saugumo spagą. [6]

3.1.2. Trūkumai uždaro kodo programinės įrangos

Argumentai prieš uždarą kodą:

- Uždaro kodo programinės įrangos kodo kokybė dažnai nėra tokia aukšta kaip galima būtų tikėtis, dažnai manoma kad atviro kodo projektuose kokybės kontrolė būna beveik neegzistuojanti, ir ją užtikrinti yra gan sunku dėl didelio skaičiaus žmonių kurie nori prisidėti prie projekto, jų kodas buna tikrinamas pavirsutiniskai dėl laiko taupymo to rezultatas yra prasta kodo kokybė ir didejantis potencialas spragoms. Bet kaip galima pastebėti, paviešistas kodas kuris visada buvo uždaras dažnai būna ypatingai prastos kokybės ir iš to paviešinto kodo spragos būna išgaunamos labai greitai. Saugumo specialistai analizuojantys greitai atranda spragas ir paviešina įrankius, skirtus išnaudoti tokias spragas, vieni geriausių pavyzdžių yra kai kaikurios Microsoft Windows NT 4.0 produkto kodo dalys buvo paviešintos, keliu dienų eigoje pirmieji įrankiai buvo sukurti išnaudoti spragas esančias šiame produkte. [2]

- Uždaras kodas neužtikrina, kad spragos nebus rastos, nors kodas ir yra neprieinamas, vis tobulėjantys irankiai skirti dekompiliuoti produktą tam, kad išgauti jame esanti kodą ir rasti spragas, palengvina darbui saugumo specialistams ir asmenims kurie nori pasinaudoti išgautomis spragomis. Dažni atvėjai kai uždaro kodo produkto spragos būna paviešinamos, šios spragos būna užtaisomos atnaujinimais, bet kaikurios spragos būna rastos ir nepaviešintos tam, kad niekad jų nesutvarkytų, ir atakuojantys asmenys turetu ilgesnį laiką išnaudoti šias spragas. Iš to galima teigti, kad nors ir spragos buna lečiau ir sunkiau surandamos uždarame produkte, tų spragų paviešinimas yra daug greitesnis potencialiai atviro kodo produktuose. [2]

- Atviro kodo produktai pasižymi savo bendruomenė, dažnai prie atviro kodo produkto gali prisidėti visi kas tik sugeba. Todėl dažnai kodas būna tvarkomas daug greičiau, vartotojai patys randa problemas kode, informuoja apie tai kurėjus, dažnai net pasiulo savo sprendimus, tokiais atvėjais kurėjams nebereikia patiems investuoti laiko sprendžiant problemą, užtenka tiesiog peržiurėti vartotojo sprendimą, ir jei viskas tinka jį pritaikyti. Tuo nepasižymi uždaro kodo produktai, vartotojai beveik negali, arba išvis negali prisidėti prie produkto kurimo, jie negali kurėjams patarti su produkto kurimu, ar padėti ištaisyti klaidas. Kurėjai turi patys aiškintis tas klaidas, ir dažnai tokių problemų sprendimas užtrunka ilgai nes tam reikia investuoti laiko ir resursų. Labai opių spragų sprendimas kartais užtrunka savaites ar net mėnesius. [2]

3.1.3. Privalumai atviro kodo programinės įrangos

Argumentai už atviro kodo programinę įranga

- Atidarant kodą visiems, yra lengviau ir greičiau identifikuoti problemas, negu uždarant kodą. Atidarius koda visiems, jį vertina ne tik kurėjai, bet ir vartotojai, bei kitos grupės žmonių, kurių deka spragos yra pamatos daug ankščiau lyginant su uždaro kodo produktais, taip pat jas pastebėti yra žymiai lengviau dėl būtent bendruomenės, išvengiama rizika, kad spraga bus nepastebėta ilgą laiką kol ją išnaudos nuostolio siekentys žmonės.

- Žmonės naudojantys atviro kodo programinė įrangą galės patys surasti ir sutvarkyti problemas kilusias su produktu, tuo atvėju jie gali savo pataisymus siulyti į pagrindinę produkto

repozitoriją, tokie pataisymai bus patvirtinti ir atsiras pačiame produkte, tuomet kiti žmonės galės parsisiusti šiuos pakeitimus pas save, taip padidindami savo sistemos saugumą. "*Linus's law: Given enough eyeballs, all bugs are shallow*" [3]

3.1.4. Privalumai uždaro kodo programinės įrangos

3.1.5. Apibendrinimas

Atvirojo kodo programinė įranga yra iš esmės saugesnė nei uždarojo šaltinio ir kiti teigė, kad taip nėra. Nei vienas iš atvejų nėra tiesa: jos iš esmės yra tos pačios monetos skirtingos pusės. Atviras šaltinis suteikia tiek užpuolikams, tiek gynėjams didesnę analitinę galią ką nors padaryti dėl programinės įrangos spragų. Jei gynėjas nieko nedaro dėl saugumo, atvirasis šaltinis tiesiog suteikia tą pranašumą užpuolikui. Tačiau atvirasis šaltinis taip pat siūlo didelius pranašumus gynėjui, suteikdamas prieigą prie saugumo metodų, kurių paprastai neįmanoma pasiekti uždarojo kodo programinėje įrangoje. Uždaras šaltinis verčia vartotojus sutikti su tokio kruopštumo lygiu, kokį pasirenka pardavėjas, tuo tarpu atvirojo kodo vartotojai (ar kiti žmonių kolektyvai) gali pakelti saugos juostą taip aukštai, kaip jie nori.(Cowan2003)(netaisytas)

Padidėjęs žiniasklaidos ir plačiosios visuomenės dėmesys atviram šaltiniui pavertė atvirojo kodo frazę beveik visa apimančia. Čia mes naudojame ją originalia, gana specifine prasme. Atvirojo kodo programinė įranga yra programinė įranga, kurią vartotojas gali patikrinti, naudoti, modifikuoti ir perskirstyti atitinkamą šaltinį (ir visą susijusią dokumentaciją)1. Mes neišskiriame jokios kūrimo metodikos (pvz., Katedra ar Turgus [10]). Mums taip pat nerūpi kainodaros modelis (nemokama programinė įranga, bendro naudojimo programinė įranga ir kt.). Tačiau mes darome prielaidą, kad vartotojams (iš esmės) leidžiama ir jie gali atstatyti sistemą iš (modifikuotų) šaltinių ir kad jie turi prieigą prie tinkamų įrankių, kad tai padarytų. Kai kuriais atvejais taip pat būtina leisti vartotojui perskirstyti modifikuotus šaltinius (visiškai arba per pataisas) (pvz., Nemokama programinė įranga ir GNU viešoji licencija2). Daugelis mūsų argumentų taip pat galioja turimai šaltinio programinei įrangai, kur licencija neleidžia perskirstyti (modifikuoto) šaltinio. (0801,3924)(netaisytas)

4. Sistemų auditavimo įrankis

4.1. Įrankio aprašas

Kurimo darbo metu buvo vadovaujamasi šiuo darbo tikslu: Sukurti saugų pažeidžiamumų skenavimo įrankį, kuris veiktu iš tam tikros slepiamos vietos, skenuotų internetinę svetainę bei jos failus, aptiktų pažeidžiamumus ar modifikuotus failus, ir pateiktų klientui informaciją apie skenavimo rezultatus. Šis darbas buvo pasiektas tokiais veiksmais:

- Užtikrinti saugų ryšį tarp įrankio ir skanuojamos sistemos naudojant VPN, šiuo atveju yra naudojama OpenVPN technolgija.
- Sukurti saugią aplinką į kurią galima būtų siusti potencialiai užkrėtus failus, šiam veiksmui įgyvendinti buvo pasitelkta Docker konteinerių technologija.
- Išorinis internetinės svetainės skanavimas pažeidžiamumu skanavimas naudojant Nmap įrankį, bandand silpnus prisijungimus...
- Parsiustus failus tikrinti viešai prieinamuosia API, kurie patikrina ar failas arba jo turinys nėra potencialiai infekuotas ir kelią grėsmę, tam buvo pasitelkta VirusTotal API.

Apie šių tikslų igyvendima ir sistemos struktūra bus rašoma detaliau

4.1.1. Saugaus ryšio užtikrinimas

4.1.2. Saugios aplinkos užtikrinimas

4.1.3. Išorinis sistemos skanavimas

4.1.4. Sistemos failu patikrinimas

4.2. Aptiktini pažeidžiamumai

Pateikiamas 4.5 poskyrio tekstas. Vienas iš šaltinių [?]. Visas turinys priklauso 4 skyriui.

4.3. Įgyvendinti lukeščiai

Pateikiamas 4.5 poskyrio tekstas. Vienas iš šaltinių [?]. Visas turinys priklauso 4 skyriui.

4.4. Trūkumai

Pateikiamas 4.5 poskyrio tekstas. Vienas iš šaltinių [?]. Visas turinys priklauso 4 skyriui.

Išvados ir rekomendacijos

Išvados bei rekomendacijos.

Ateities tyrimų planas

Pristatomi ateities darbai ir/ar jų planas, gairės tolimesniems darbams....

Literatūros šaltiniai

[1] Crispin Cowan. Software Security for Open-Source Systems. *IEEE Security and Privacy*, 1(1):38–45, jan 2003.

[2] Jaap-Henk Hoepman and Bart Jacobs. Software Security Through Open Source. 2004.

[3] Andrew Meneely and Laurie Williams. Secure Open Source Collaboration: An Empirical Study of Linus' Law. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 453–462, New York, NY, USA, 2009. ACM.

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

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

[6] B Witten, C Landwehr, and M Caloyannides. Does open source improve system security? *IEEE Software*, 18(5):57–61, 2001.

Priedai

Dokumentą sudaro du priedai: A priede ....

A. Pirmojo priedo pavadinimas

Pirmojo priedo tekstas ...

B. Antrojo priedo pavadinimas

Antrojo priedo tekstas ...