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

Informacinių technologijų 4 kurso bakalauro baigiamasis darbas

Įterptinių instrukcijų atakų (XSS) aptikimas naudojant dirbtinius neuroninius tinklus

Cross-Site Scripting (XSS) Detection Using Artificial Neural Networks

Atliko:
Domantas Baliukynas parašas

Vadovas:
lekt. Eduardas Kutka

Vilnius
2024

Turinys

Sutartinis terminų žodynas **4**

Santrauka **5**

Summary **6**

Įvadas **7**

1. Įterptinių instrukcijų atakos (XSS) **9**
1.1. Grėsmės . . . 9
1.2. Klasifikacija . . . 10
1.2.1. Atsispindinčioji ataka . . . 10
1.2.2. Patalpintoji ataka . . . 10
1.2.3. Dokumento objekto modelio (DOM) ataka . . . 11
1.2.4. Mutuotoji ataka . . . 11
1.3. Pažeidžiamumų aptikimas be papildomos programinės įrangos . . . 11
1.3.1. XSS poligloto naudojimas . . . 11
1.3.2. Vienetų testavimas . . . 12
1.4. Pažeidžiamumų aptikimas naudojant pažeidžiamumų skenerius . . . 12
1.4.1. Klasifikacija . . . 12
1.4.2. Veikimo principas . . . 13
1.4.3. Kylančios problemos . . . 13
1.5. Prevencija . . . 14
1.5.1. Kodavimas . . . 14
1.5.2. HTML kodo apvalymas . . . 14
1.5.3. Karkasų naudojimas . . . 14
1.5.4. Svetainių aplikacijų užkarda . . . 14

2. Dirbtiniai neuroniniai tinklai **15**
2.1. Neurono principai . . . 15
2.2. Architektūra . . . 15
2.3. Treniravimas . . . 16

3. Konvoliuciniai neuroniniai tinklai **16**
3.1. Pagrindiniai skirtumai nuo tradicinių neuroninių tinklų . . . 16
3.2. Architektūra . . . 16
3.2.1. Konvoliucinis sluoksnis . . . 17
3.2.2. Sutelkimo sluoksnis . . . 17
3.2.3. Pilnai sujungtasis sluoksnis . . . 17
3.3. Panaudojimas . . . 18

4. Neuroninių tinklų naudojimas XSS atakų aptikimui **18**
4.1. Naudojami modeliai . . . 19
4.2. Kylantys iššūkiai . . . 20
4.3. Aptikimas . . . 20
4.3.1. Duomenų paruošimas neuroninio tinklo treniravimui . . . 20

4.3.2. Klasifikavimas 21

5. Tyrimas **22**
5.1. Pagrindinės technologijos 22
5.2. Techninė įranga 22
5.3. Modelis 23
5.4. Duomenys 23
5.4.1. Treniravimui skirti duomenys 23
5.4.2. Validacijai skirti duomenys 24
5.5. Eksperimentai 24
5.5.1. Optimalaus epochų kiekio nustatymas 24
5.5.2. Optimalaus rinkinio dydžio nustatymas 25
5.5.3. Optimalaus mokymosi koeficiento nustatymas 26
5.5.4. Optimalaus svorio nykimo koeficiento nustatymas 26
5.5.5. Optimalaus išmetimo koeficiento nustatymas 27
5.5.6. Skirtingi vektorizacijos metodai 27
5.5.7. Skirtingi tokenizacijos metodai 28
5.5.8. Modelio rezultatai 29

6. Pritaikymas **29**

Išvados ir rekomendacijos **32**

Ateities tyrimų planas **33**

Literatūros šaltiniai **34**

Sutartinis terminų žodynas

- ASCII – simbolių kodavimo formatas, naudojamas tekstui kompiuteriuose ir kituose įrenginiuose.
- DOM – aplikacijų programavimo sąsaja, skirta HTML ar XML dokumentams.
- HTML – hiperteksto ženklinimo kalba, su kuria yra aprašoma svetainių struktūra.
- HTTP – protokolas, skirtas hipermedijos duomenų perdavimui.
- OSI – modelis, aprašantis septynis abstrakčius kompiuterio tinklų sluoksnius.
- OWASP – pelno nesiekianti organizacija, kurios tikslas yra tobulinti svetainių saugumą nuo kibernetinių atakų.
- URL – svetainės puslapio adresas.
- VM – fizinio kompiuterio virtualizacija.
- WWW – pasaulinis tinklalapių tinklas.
- WYSIWYG – redagavimo programinė įranga, kurioje redaguojamas tekstas yra atvaizduojamas taip pat, kaip ir galutinis teksto variantas.
- XSS – tokios kibernetinės atakos, kuriose kenkėjo kodas yra įterpiamas svetainėse, su tikslu, jog tas kodas bus įvykdytas pas kitą vartotoją.

Santrauka

Įterptinių instrukcijų atakos (XSS) yra vienos iš dažniausiai pasitaikančių kibernetinių atakų svetainėse. Jos dažniausiai yra klasifikuojamos į tris grupes – atsispindinčias, patalpintąsias ir DOM. Pažeidžiamumus surasti galima naudojantis be papildomos programinės įrangos – naudojant XSS poliglotą, vienetų testavimą. Vis dėlto, šiais būdais aptikti įterptinių instrukcijų pažeidžiamumus didesnėse svetainėse gali būti sudėtinga ir neefektyvu, todėl buvo sukurtos programos, kurios gali palengvinti šį darbą.

Tam, kad atakų būtų išvengta, taip pat yra naudojami tam tikri metodai ir programinė įranga. Prevencijai yra naudojamas kodavimas, HTML kodo apvalymas, karkasų naudojimas, svetainių aplikacijų užkarda. Tačiau šie metodai dažnai nerodo patenkinamų rezultatų, todėl buvo pradėta ieškoti alternatyvų. Viena iš jų – dirbtinių neuroninių tinklų naudojimas. Įterptinių instrukcijų atakų dinamiškumas yra viena iš didžiausių problemų, kurią dirbtiniai neuroniniai tinklai gali spręsti geriau nei kiti metodai.

Bakalauro baigiamajame darbe buvo sukurtas konvoliucinių neuroninių tinklų modelis, skirtas aptikti įterptinių instrukcijų atakas. Tyrimo metu, modelis buvo treniruojamas su daugiau nei 13 tūkst. kodo eilučių turinčia duomenų aibe. Buvo atliekami eksperimentai, siekiant išsiaiškinti, su kokiu epochų kiekiu, rinkinio dydžiu, mokymosi koeficientu, svorio nykimo koeficientu, išmetimo koeficientu ir vektorizacijos, tokenizacijos metodais, modelis gali pasiekti didžiausią vidutinį tikslumą.

Galiausiai, sukurtas modelis buvo implementuotas žurnalizavime. Buvo sukurta svetainė, kurioje buvo implementuotas modelis, galintis aptikti ataką HTTP užklausoje. Svetainė siunčia užklausas į „Seq“ serverį, „Seq“ serveris renka ir išsaugo visas užklausas, taip pat serveris sugrupuoja užklausas ir siunčia el. laiškus apie galimas atakas kas 10min.

Summary

Cross-Site Scripting (XSS) Detection Using Artificial Neural Networks

Cross-site-scripting (XSS) attacks are one of the most common cyberattacks against web applications. According to often in literature cited cybersecurity organization "OWASP", injection attacks (XSS included) take third place in vulnerability occurrence. XSS is usually classified into the three groups – reflected, stored and DOM. It is possible to find vulnerabilities without using additional software – using XSS polyglot, unit testing. However, detecting cross-site-scripting vulnerabilities using these methods could be difficult and ineffective in larger websites. Therefore, software that could ease this job was created.

In order to prevent a cyberattack, preventive methods are being used such as encoding, HTML sanitization, frameworks, web application firewall. However, these methods are not always giving satisfactory results. Therefore, alterantive methods such as usage of artificial neural networks, were considered.

In this bachelor's work, a convolutional neural network model, that is able to detect XSS attacks, was created. The conducted experiments determined the optimal epoch number, batch size, learning rate, weight decay rate, dropout rate and best vectorization, tokenization methods. Lastly, a created model was implemented in logging. A website was created that sends requests to "Seq" server, "Seq" server logs the requests, groups possible attacks and sends e-mails about them every 10 minutes.

Įvadas

Kibernetinės atakos svetainėse tampa vis dažnesnės. Kibernetinio saugumo įmonė „Akamai“ aptiko, jog nuo 2017-ųjų metų lapkričio mėn. iki 2019-ųjų metų balandžio mėn. buvo įvykdyta 4.5 milijardai atakų prieš svetaines. Taip pat, dar viena kibernetinio saugumo kompanija „Symantec“ pastebėjo, jog 2019-aisiais metais atakos prieš svetaines išaugo 56%. [24]

Kibernetinių atakų pagausėjimą lemia kelios priežastys – toliau augantis interneto svetainių populiarumas, modernių svetainių sudėtingumas, apsunkinantis atakų prevenciją, konfidencialios informacijos gavimo nauda. Anksčiau informacijos gavimas nebuvo toks aktualus, kadangi tokios paslaugos, kurios reikalauja išskirtinio duomenų saugumo, kaip elektroninė bankininkystė ir elektroninė komercija, nebuvo populiarios. [24]

Svetainės dažniausiai nukenčia nuo duomenų viliojimo (angl. phishing), įterptinių instrukcijų (angl. cross-site scripting), struktūrizuotų užklausų kalbos įskverbties (angl. SQL injection), paskirstytų paslaugos trikdymo (angl. distributed denial of service), įsiterpusio žmogaus per vidurį (angl. man-in-the-middle), įsiterpusio žmogaus naršyklėje (angl. man-in-the-browser) atakų. [29].

Pasak dažnai literatūroje cituojamos kibernetinio saugumo įmonės „OWASP“, įskverbties atakos yra trečioje vietoje pagal tai, kiek daug svetainių yra pažeidžiamos. Į įskvierbties atakas patenka ir įterptinių instrukcijų atakos (XSS), apie kurias ir bus kalbama šiame darbe. [29].

Įterptinių instrukcijų atakos (XSS) atsirado XX a. 10-ojo dešimtmečio pabaigoje. Vis dėlto, jos išlieka vienos iš didžiausių kibernetinių grėsmių interneto svetainėse. To priežastis yra pačios saityno technologijos pobūdis. Tam, kad vartotojai galėtų pasiekti svetaines ir naudotis jų funkcionalumu, yra siunčiamos užklausos į serverį. Todėl yra svarbu, jog svetainių kūrėjas tinkamai įgyvendintų vartotojo užklausos patikras, užklausos turinio kodavimą ir HTML apvalymą, t.y. potencialiai pavojingų HTML žymų, atributų ar reikšmių panaikinimą. Tačiau, tai reikalauja žinių, kurios yra sudėtingai įvaldomos. [29].

Tam, kad apsisaugojimas nuo kenkėjų būtų palengvintas, galima pasinaudoti programine įranga, kuri aptinka svetainėje egzistuojančius pažeidžiamumus. Tačiau vien pažeidžiamumų aptikimas nėra pakankamas sprendimas, norint išvengti atakų. Dažnai programinė įranga, analizuojanti pažeidžiamumus, klysta arba nepatikrina visų vietų, kurios yra potencialiai pažeidžiamos. Todėl yra reikalingas papildomas saugumo sluoksnis – atakų aptikimas. [28]

Vienas iš atakų aptikimo būdų yra svetainių aplikacijų užkarda. Tai programinė įranga, filtruojanti HTTP užklausas – nusprendžianti, ar užklausos turinyje yra kenksmingo kodo ir atmetanti užklausą. Tačiau svetainių aplikacijų užkarda naudoja žmogaus aprašytas taisykles, pagal kurias yra priimami filtravimo sprendimai. Žmogaus aprašytos taisyklės gali būti klaidingos, jas atnaujinti reikia rankiniu būdu, o atakų įvairiapusiškumas gali taisyklių rašymo procesą ženkliai pasunkinti. [11]

Pastaruoju metu pradėjo populiarėti dirbtinių neuroninių tinklų panaudojimas įterptinių instrukcijų atakų aptikime. Vienas tai lemančių veiksnių – viršuje aptartos svetainių aplikacijų užkardų problemos. Vienas iš dirbtinių neuroninių tinklų privalumų – sugebėjimas aptikti atakas, kurios dar nėra žinomos. XSS atakų aptikime dažnai naudojami konvoliuciniai neuroniniai tinklai (CNN), rekurentiniai neuroniniai tinklai (RNN), dvikrypčiai rekurentiniai neuroniniai tinklai (bi-RNN), ilgos-trumpos atminties (LSTM) neuroniniai tinklai. [27]

Tačiau, naudojant dirbtinius neuroninius tinklus taip pat yra susiduriama su iššūkiais, tokiais kaip: poreikis nuolatos atnaujinti duomenų aibę, atakos, kurios klaidina dirbtinių neuroninių tinklų modelį, didelis sistemos resursų naudojimas, tinkamo modelio ir duomenų aibės pasirinkimas. [27]

Bakalauro baigiamojo darbo tyrimo metu, buvo treniruojamas konvoliucinių neuroninių tinklų

modelis, naudojant duomenų aibę, kurioje virš 13 tūkst. kodo dalių, yra suskirstytos į kenksmingo ir nekenksmingo kodo kategorijas.

Šis bakalauro baigiamasis darbas yra kursinio darbo tęsinys. Dalis įvado, dalis santraukos, visas pirmasis skyrius yra nekeisti. [6]

Bakalauro baigiamojo darbo uždaviniai:

- Aptarti literatūros šaltiniuose keliamas įterptinių instrukcijų atakų problemas, sprendimus ir iššūkius.
- Aptarti dirbtinių neuroninių tinklų privalumus ir trūkumus įterptinių instrukcijų atakų kontekste.
- Sukurti konvoliucinių neuroninių tinklų modelį tyrimui ir žurnalizavimo implementacijai.
- Surasti duomenų aibes skirtas konvoliucinių neuroninių tinklų modelio treniravimui ir validacijai.
- Paruošti duomenų aibes treniravimui (dekoduoti, pervadinti stulpelių pavadinimus).
- Atlikti tyrimą, naudojant sukurtą konvoliucinių neuroninių tinklų modelį.
- Implementuioti ištreniruotą konvoliucinių neuroninių tinklų modelį žurnalizavime.

Bakalauro baigiamojo darbo tikslas: Sukurti konvoliucinių neuroninių tinklų modelį, skirtą XSS atakų aptikimui ir ištirti su kokiu optimaliausiu epochų kiekiu, rinkinio dydžiu, mokymosi koeficientu, svorio nykimo koeficientu, išmetimo koeficientu, tinkamiausiu vektorizacijos ir tokenizacijos metodu, sukurtas modelis gali pasiekti aukščiausią vidutinį tikslumą.

1. Įterptinių instrukcijų atakos (XSS)

1.1. Grėsmės

Svetainės, kurios yra pažeidžiamos XSS atakų, kelia grėsmę ne tik savo reputacijai, bet ir vartotojams. Įmonės, valdančios tokias svetaines, gali patirti didelę finansinę žalą. Atakos pasekmės vyrauja nuo nereikšmingų iki pavojingų [21]:

- Paveikto vartotojo susierzinimas.
- Nukreipimas į kitą svetainę.
- Apsimetimas kitu asmeniu.
- Duomenų modifikavimas.
- Kenkėjiškos programinės įrangos įdiegimas į vartotojo sistemą.
- Prieiga prie įrenginių, kurie yra prijungti prie vartotojo kompiuterio. Pavyzdžiui, kamera, mikrofonas.
- Vartotojo duomenų pavogimas.
- Prieiga prie vartotojo lokalios failų sistemos (tuo atveju, jei vartotojas parsisiunčia kenkėjišką programinę įrangą).

XSS atakų poveikis vartotojų duomenų saugumui priklauso nuo pačios svetainės pobūdžio. Svetainėje, kurioje beveik visa informacija yra viešai prieinama, atakos poveikis būtų nereikšmingas. Tai galėtų būti diskusijų forumas, naujienų portalas, sporto statistikos svetainė ir t. t. Tačiau jeigu tinklalapyje yra duomenų, kurie neturėtų būti prieinami tam tikriems asmenims, atakos pasekmės gali būti pačios didžiausios. Tokių tinklalapių pavyzdžiai – bankas, paštas, gydymo įstaiga. [14]

Kenkėjiškos programinės įrangos įdiegimas yra vienodai pavojingas, nepriklausomai nuo svetainės tipo. Taip yra todėl, kad taikiniu tampa visi duomenys, esantys naudotojo kompiuteryje. Naudotojas gali nepastebėti kenkėjo modifikuotos atsiuntimo nuorodos ir paspausti ant jos. [14]

Atakų poveikis priklauso ir nuo pačios atakos tipo. Didžiąja dalimi atvejų, teoriškai ataką išvengti gali ir pats vartotojas. Pavyzdžiui, vartotojas, gavęs el. laišką su nuoroda į svetainę, pastebi nuorodos viduje esantį kenksmingą kodą ir nespaudžia ant nuorodos, jog išvengtų nukreipimo į pažeidžiamą svetainę. [14]

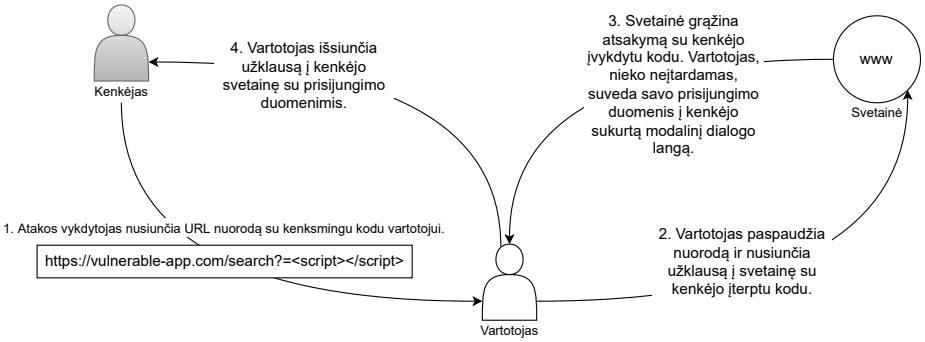
Kodo įterpimo atakos yra grėsmingos ne tik dėl galimų padarinių, bet ir dėl to, kad jas atlikti yra ganėtinai paprasta, o apsisaugoti nuo jų – atvirkščiai, sudėtinga. Yra keletas priežasčių, kurios XSS puolimą padaro nesudėtingu: svetainę užpulti galima rankiniu būdu, neautomatizuotai. Pradėti įgyvendinti paprastas atakas tereikia HTML ir JavaScript pagrindų. Puolimas dažnai vyksta patikimose svetainėse, todėl naudotojai lengviau apsigauna. Sudėtingą apsisaugojimą nuo kodo įterpimo lemia šie faktoriai: kodo įterpimas turi daug atakos vektorių – skirtingų būdų, kuriais galima pakenkti. Svetainėse atsiranda vis daugiau funkcijų ir sudėtingėja jų implementacija, todėl padaugėja atakos vektorių. [14]

1.2. Klasifikacija

Dažniausiai XSS atakos yra skirstomos į tris pagrindines grupes: **atsispindinčioji / trumpalaikė**, **patalpintoji / ilgalaikė** ir **dokumento objekto modelio** ataka. Tačiau neseniai literatūroje pradėta naudoti dar vieną atakos klasę, vadinamą **mutuotaja** XSS ataka. [15]

1.2.1. Atsispindinčioji ataka

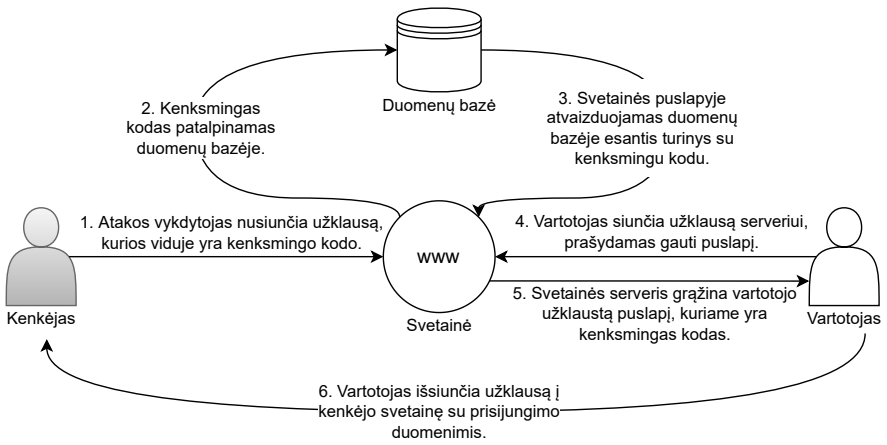
Tai dažniausiai pasitaikantis atakos tipas svetainėse. Kenkėjo nusiųsta užklausa į serverį su kenksmingu kodu yra apdorojama ir serveris grąžina atsakymą, kuris atsispindi svetainėje. Pavyzdžiui, kenkėjas sukuria URL, kurio viduje yra kodas, kuris aktyvuosis, vartotojui paspaudus nuorodą. Tarkime, įvykdžius kodą, atsidarys modalinis dialogo langas, prašantis suvesti prisijungimo vardą ir slaptažodį. Patvirtinimo mygtukas yra su nuoroda į kenkėjo svetainę. Paspaudus šį mygtuką, išsiunčiama užklausa į kenkėjo svetainę su prisijungimo duomenimis. Kenkėjas pamato užklausoje esančius duomenis ir taip pavagia asmens paskyrą. [20]



1 pav. Atsispindinčiosios atakos procesas. [20]

1.2.2. Patalpintoji ataka

Šiai grupei priklauso atakos, kurių kenksmingas kodas yra patalpinamas svetainių duomenų bazėje. Tokios atakos yra pačios pavojingiausios, kadangi vartotojui tereikia užeiti į svetainės puslapį, net nespaudžiant kenksmingos nuorodos, kaip viršuje minėtoje atsispindinčioje atakoje, kad būtų įvykdytas kenkėjo kodas. Kitaip sakant, pats vartotojas šios atakos išvengti negali (neskaitant nesilankymo pačioje svetainėje). [20]



2 pav. Patalpintosios atakos procesas. [20]

1.2.3. Dokumento objekto modelio (DOM) ataka

Dokumento objekto modelio ataka yra išskirtinė, kadangi HTTP atsakymas, siunčiamas į vartotojo naršyklę, nepasikeičia, kaip pavyzdžiui, su jau viršuje minėtais atakos tipais, kai kenksmingas kodas yra patalpinamas į HTTP atsakymą. Tai reiškia, jog ataka vyksta tik kliento pusėje – naršyklėje. Puolimo įgyvendinimas yra beveik identiškas atsispindinčiajam tipui. Užpuolikas nusiunčia nuorodą vartotojui su įterptine instrukcija ir vartotojas paspaudžia ant nuorodos. Tačiau šį kartą vartotojas gaus atsakymą iš svetainės be įvykdyto kenkėjiško kodo, bet JavaScript kode gali būti naudojamas `document.location` objektas, iš kurio yra pasiimamas URL su įterptu kodu. Tai reiškia, jog įterptinė instrukcija bus įvykdyta po to, kai puslapis yra užkrautas, o ne tada, kai puslapis yra dar kraunamas, kaip atsispindinčiose ir patalpintose atakose. [29]

1.2.4. Mutuotoji ataka

Šiam tipui priklauso tokios atakos, kuriose kodas yra modifikuojamas pačios naršyklės. Naršyklė paverčia netaisyklingą HTML kodą į nesaugų. Taip gali atsitikti, jeigu svetainės kode yra naudojama `innerHTML` savybė arba jeigu užklausos turinys nėra koduojamas. Pavyzdžiui, kenkėjas gali įvesti netinkamą HTML kodą, tačiau naršyklė pakeis tam tikras žymas taip, kad kodas veiktų. [12]

1 išeities kodas. Netaisyklingas HTML kodas [23]

```
1 <noscript><p title="</noscript><img src=x onclick=alert(1)>">
```

2 išeities kodas. Mutuotasis HTML kodas [23]

```
1 <noscript><p title="</noscript>
2 <img src="x" onclick="alert(1)">
3 "">"
```

1.3. Pažeidžiamumų aptikimas be papildomos programinės įrangos

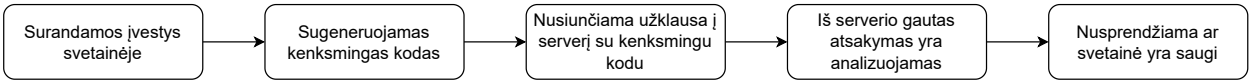
Dėl XSS atakų įvairumo ir sudėtingėjančios svetainių logikos, aptikti esamus XSS pažeidžiamumus svetainėje gali būti sudėtinga. Todėl ypatingai svarbu naudoti žinomus metodus, kurie padėtų išspręsti šią problemą.

1.3.1. XSS poligloto naudojimas

XSS poliglotas – tai vienos eilutės atakos kodas, kuris gali būti panaudojamas skirtingose situacijose. [19]

Įprastai, kenkėjiškas kodas turi būti keičiamas, atsižvelgiant į tai, ką norima testuoti. Pavyzdžiui, `input` žymoje, `value` atributo reikšmė gali būti tarp viengubų arba dvigubų kabučių. Tam, kad patikrinti ar kenkėjas negali uždaryti kabučių, būtų galima bandyti du atskirus kodo gabalus – vieną su viengubom kabutėm, o kitą su dvigubom. Tačiau, naudojant XSS poliglotą abiem atvejais kodo keisti nereikia. Kodas yra sukonstruotas taip, kad būtų tinkamas naudoti įvairiose vietose – skirtingose kabutėse, HTML komentaruose, žymose, paieškos reiškiniuose (angl. regular expressions), tam tikrose JavaScript funkcijose. [19]

Taigi šio metodo privalumas yra tas, kad viena kodo eilutė apima skirtingus atakos vektorius. Natūralu, kad toks metodas nepatikrins daugybės kitų atakos vektorių. Tai pagrindinis XSS poligloto trūkumas.



3 pav. XSS poligloto pavyzdys. [8]

1.3.2. Vienetų testavimas

Vienetų testavimas (angl. unit testing) – tai smulkiausios kodo dalies tinkamo funkcionalumo patikrinimas. Vienetų testavimas gali būti naudojamas aptikti XSS pažeidžiamumus, kilusius dėl netinkamo kodavimo (angl. encoding). [22]

Šiuo atveju, turėtų būti tikrinama funkcija, kuri koduoja iš vartotojo įvestus duomenis. Visų pirma, parašomas kenksmingas testo kodas, kuris bus paduodamas tikrinamai funkcijai. Tam, kad būtų žinoma ar mūsų funkcija veikia tinkamai ar ne, pavyzdžiui, testo kode galima pakeisti svetainės pavadinimą. Antra, funkcijai yra paduodamas testinis kodas. Trečia, patikrinamas svetainės pavadinimas. Jeigu funkcija koduoja tinkamai – svetainės pavadinimas nepakis, jeigu netinkamai – pasikeis į „ATAKA SĖKMINGA“. Funkcijos kvietimo, svetainės pavadinimo tikrinimo procesas turėtų būti automatizuotas. [22]

Taigi šio metodo privalumas – automatinis pažeidžiamumo patikrinimas, naudojant neribotą kiekį skirtingų atakos būdų, atvirkščiai, nei su XSS poligloto naudojimo metodu. Vis dėlto, testinio kodo rašymas reikalauja XSS, HTML ir JavaScript išmanymo. [22]

1.4. Pažeidžiamumų aptikimas naudojant pažeidžiamumų skenerius

Pažeidžiamumų skeneriai – tai programinė įranga, skirta automatiškai aptikti XSS pažeidžiamumus. Dalis tokių skenerių yra mokami. Natūralu, jog ne visi svetainių kūrėjai turi finansinių išteklių naudotis mokama programine įranga. Tada tektų naudoti atvirojo kodo skenerius. Būtent atvirojo kodo skeneriai ir bus analizuojami tyrimo dalyje.

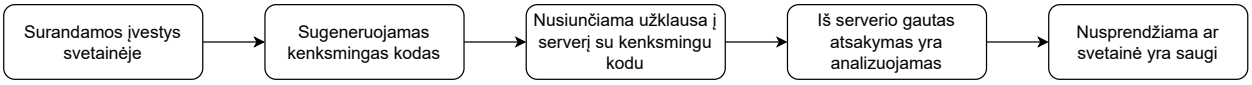
1.4.1. Klasifikacija

Dažniausiai skeneriai yra skirstomi į du tipus: **statinius** ir **dinaminius**. Statiniai skeneriai turi prieigą prie svetainės išeities kodo ir tikrina patį kodą. Toks metodas yra vadinamas baltosios dėžės testavimu. Dinaminiai skeneriai naudoja juodosios dėžės metodą – tai reiškia, jog skeneriai neturi prieigos prie išeities kodo ir naudoja pačią svetainės aplikaciją, kaip ir patys vartotojai, kad aptiktų pažeidžiamumus. Atvirkščiai nei dinaminiai, statiniai skeneriai yra retai naudojami, kadangi jų veikimas priklauso nuo to, ant kokios programavimo kalbos ir karkaso yra parašytas svetainės kodas. [18] Todėl šiame darbe toliau bus kalbama tik apie dinaminius skenerius.

Pagrindinis šių programų privalumas yra tas, kad pagrindiniai veiksmai yra automatizuoti – surandami galimi atakos vektoriai, kenksmingas kodas įterpiamas į galimus atakos vektorius, parašoma ataskaita su informacija apie tai, kiek ir kokios vietos yra pažeidžiamos. Dėl šių privalumų tai yra pagrindinis metodas aptinkant XSS pažeidžiamumus. [28]

1.4.2. Veikimo principas

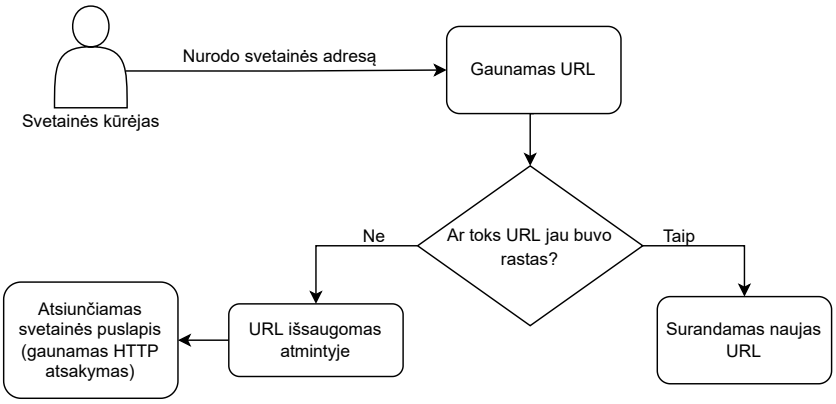
Skeneriai įprastai susideda iš dviejų dalių: pasyvaus ir aktyvaus skenavimo. Pasyvaus skenavimo dalyje programa skenuoja ir analizuoja užklausas ir jų atsakymus realiu laiku. Tuo tarpu, aktyvaus skenavimo dalyje, programa simuliuoja tikrą ataką, sukurdama kenksmingą kodą. Kodas yra išsiunčiamas per HTTP užklausą. Gavus atsakymą iš serverio, programa analizuoja informaciją ir nusprendžia ar svetainė yra saugi, ar ne. Toks būdas dar yra žinomas, kaip tikslintinas testavimas (angl. fuzzy testing). [28]



4 pav. Aktyvaus skenavimo procesas. [28]

URL ieškojimui yra naudojamas saityno peržiūros robotas (angl. web crawler). Tai programa, automatiškai surenkanti informaciją, esančią svetainėse. Tai yra svarbiausia skenerio dalis, kadangi nesurinkus tam tikrų nuorodų, programa automatiškai patikrinti ar jos yra pažeidžiamos, nebegali. [17]

Visų pirma, skeneriui reikia nurodyti svetainės adresą, kad saityno peržiūros robotas žinotų nuo ko pradėti rinkti informaciją. Gavęs nuorodą, robotas atsisiunčia svetainės puslapį. Jį išanalizuoja ir suranda esančias nuorodas. Šis procesas yra kartojamas tol, kol nebelieka naujų nuorodų. [17]



5 pav. Saityno peržiūros roboto darbo eiga. [17]

1.4.3. Kylančios problemos

Visų pirma, skenerio aptikimo galimybės priklauso nuo to, kaip gerai yra surandamos nuorodos svetainėje. Gali atsitikti taip, jog skeneris, o dar tiksliau, saityno peržiūros robotas, neras tam tikros vietos, kur vartotojas gali išsiųsti užklausą. Tokiu atveju, programa nepatikrins ar ta nuoroda yra saugi. Antra problema – tam, kad patikrinti vietas, kurios svetainėse yra pasiekiamos tik registruotų arba tam tikras teises turinčių vartotojų, patys skeneriai turi prisijungti, kaip vartotojai. Kai kurie skeneriai tą atlikti gali, tačiau net ir tada nėra garantijos, jog skeneriai patikrins visas reikiamas vietas. Pavyzdžiui, skeneris gali atsijungti nuo paskyros, tikrindamas atsijungimo nuorodą nuo atakų. Trečia problema – skeneriai dažnai suteikia galimybę juos konfigūruoti pagal reikiamus poreikius. Pavyzdžiui, rankiniu būdu pridėti URL adresus, kurių programinė įranga pati neaptinka. Tačiau tai svetainės kūrėjui prideda papildomo darbo ir ilgainiui, konfigūracijos failas gali pasidaryti toks didelis, kad jį bus sunku modifikuoti. [10]

1.5. Prevencija

1.5.1. Kodavimas

Kodavimas – tai tam tikrų simbolių pavertimas kitais, siekiant, jog kenksmingi duomenys siunčiami į serverį taptų nekenksmingais. Nors simboliai ir yra paverčiami kitokiais, jų išvaizda HTML dokumente nepasikeičia. Pavyzdžiui, ženklas „>“ (daugiau nei) yra pakeičiamas į „>“. Todėl jeigu užkoduotume ženklus „daugiau nei“ ir „mažiau nei“, kodas, turintis šiuos ženklus: `<script>...</script>` būtų pakeistas į `<script>...</script>`. Vadinasi, kodas, esantis `script` žymos viduje neįsivykdys. [21]

1.5.2. HTML kodo apvalymas

HTML kodo apvalymas yra panašus į kodavimo metodą, tačiau šiuo atveju, tam tikri simboliai ar jų kombinacijos yra panaikinamos. Šis metodas galėtų būti naudingas, kai HTML kode nėra tikimasi tam tikrų simbolių. Pavyzdžiui, jeigu yra prašoma įvesti vartotojo vardą ir pavardę, tokių ženklų, kaip „<“ ar „>" neturėtų būti. Vis dėlto, jeigu, pavyzdžiui, vartotojas naudojasi WYSIWYG teksto redaktoriumi svetainėje, šis metodas nebetiktų, kadangi viskas, ką įveda vartotojas, turėtų būti atvaizduojama. Tai reiškia, kad reikėtų naudoti kodavimo metodą. [21]

1.5.3. Karkasų naudojimas

Šiuolaikinėse svetainėse karkasai (angl. frameworks) yra dažnai naudojami dėl jau paruoštos struktūros, išplečiamumo, dokumentacijos ir saugumo. Iš XSS perspektyvos, karkasai yra naudingi tuo, kad automatizuoja HTML kodavimą ir suteikia iš anksto sukurtus šablonus, kurie padeda išvengti saugumo spragų. Tačiau naudojant karkasus vis tiek galima susidurti su atvejais, kai svetainė tampa pažeidžiama. Pirma, karkasas turi funkcijas, kurios modifikuoja DOM elementus, kurie gali būti pažeidžiami. Antra, pats svetainės kūrėjas gali nurodyti, jog HTML kodas nebūtų apvalomas. Trečia, tam tikri karkaso komponentai gali būti pasenę. Todėl naudojant karkasą, kitų prevencijos metodų pritaikymas išlieka aktualus. [25]

1.5.4. Svetainių aplikacijų užkarda

Svetainių aplikacijų užkarda – tai programinė įranga, stebinti HTTP srautą ir filtruojanti nepageidautinas užklausas į svetainės serverį. Jos veikimo principas yra panašus į tradicines užkardas. Pagrindinis skirtumas yra tas, kad stebimas tinklo srautas ne OSI ketvirtajame ar žemesniuose sluoksniuose, bet OSI aplikacijos sluoksnyje. [11]

Tradicinėse užkardose programai aprašyti taisykles, pagal kurias bus filtruojamas koks nors srauto vienetas, pavyzdžiui segmentas ar paketas, yra paprasčiau. To priežastis – baigtinis skaičius IP adresų, prievadų ar protokolų. Tačiau svetainių aplikacijų užkardos filtruoja HTTP užklausas, kurios turi žymiai daugiau unikalių atvejų. Pavyzdžiui, kodo gabalas `<script>...</script>` iš pradžių galėtų atrodyti, jog yra nepageidaujamas. Tačiau, jeigu svetainė yra diskusijų forumas, kuriame yra diskutuojama apie atakas, vartotojas turi galėti įkelti kodo pavyzdį su `script` žyma. Tokiu atveju, reikėtų papildomos konfigūracijos. [11]

Taigi nors svetainių aplikacijų užkarda ir padeda išvengti XSS atakų, ji to negarantuoja. Todėl ji turėtų būti naudojama kartu su kitais minėtais metodais.

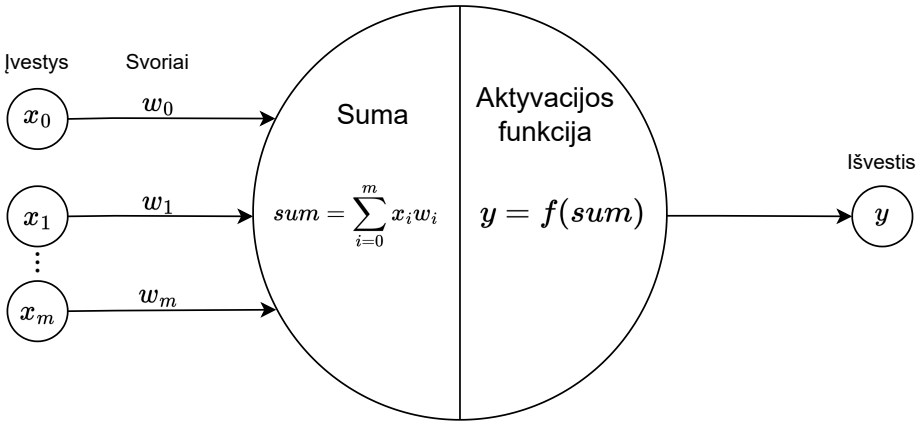
2. Dirbtiniai neuroniniai tinklai

Dirbtiniai neuroniniai tinklai (**toliau – neuroniniai tinklai**) – tai sujungtų matematinių funkcijų visuma, bandanti atkartoti žmogaus smegenų funkciją ir struktūrą. Kiekvienas neuroninis tinklas yra sudarytas iš dirbtinių neuronų (**toliau – neuronų**) – matematinių funkcijų, įgyvendinančių tokį algoritmą: įvestys sudauginamos su svoriais, sudaugintos reikšmės yra sumuojamos, sumuota reikšmė yra paduodama aktyvacijos funkcijai. [2]

2.1. Neurono principai

1943m. du tyrėjai – Warren McCullock ir Walter Pitts pasiūlė pirmąjį matematinį neuroninio tinklo modelį. 1958m. toks modelis buvo pirmą kartą implementuotas. Kiekvienas neuronas yra sujungtas su kitais neuronais ir turi įvestis, per kurias gauna duomenis iš išorės arba iš kitų neuronų. Kiekviena įvestis turi individualius svorius, kurie nulemia galutinį rezultatą.

Neuronas turi dvi pagrindines dalis: sumos apskaičiavimą ir aktyvacijos funkciją. Sumos dalis yra atsakinga už įvesčių, sudaugintų su svoriais, reikšmių sumavimą. Aktyvacijos funkcija – atsakinga už išvesties (neurono galutinio rezultato) apskaičiavimą. [2]

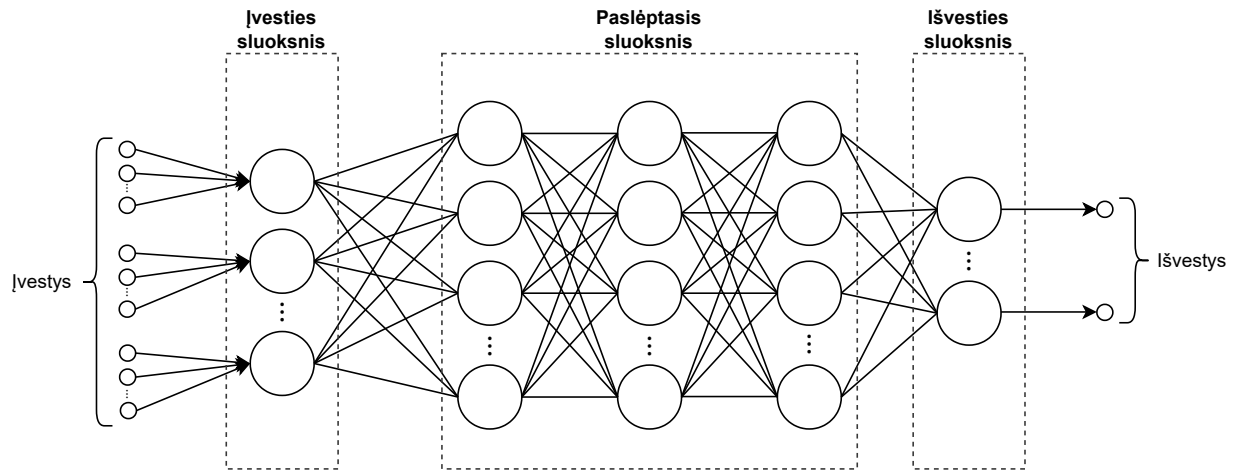


6 pav. Neurono struktūra. [2]

Aktyvacijos funkcijos pasirinkimas priklauso nuo sprendžiamos problemos pobūdžio. Pavyzdžiui, dvejetainė laiptuotoji funkcija galėtų būti naudojama dvejetainėms klasifikacijos užduotims, sigmoidinė funkcija – kelių kategorijų klasifikacijoms. [26]

2.2. Architektūra

Neuronų principai yra paprasti, tačiau jungiant neuronus su kitais neuronais ir taip sudarant tinklą, atsiranda sudėtingumas ir papildomi iššūkiai. Neuroniniai tinklai, tam, kad galėtų išmokti sudėtingus ryšius tarp duomenų, yra dažnai grupuojami į tris sluoksnius – įvesties, paslėptojo ir išvesties. Įvesties sluoksnis susideda iš vieno ar daugiau neuronų, kurie gauna duomenis iš išorės. Paslėptasis sluoksnis gauna duomenis ne iš išorės, bet iš įvesties sluoksnio neurono. Išvesties sluoksnis, gaunantis reikšmę iš paslėptojo sluoksnio, grąžina galutinį rezultatą. Toks sluoksniavimas leidžia atlikti netiesiškas transformacijas, išreikšti duomenis hierarchijoje, juos reguliarizuoti. Visa tai leidžia spręsti sudėtingas praktines problemas, tokias kaip, vaizdo, balso atpažinimas, natūralios kalbos apdorojimas, nestruktūrizuotų duomenų analizė. [26]



7 pav. Neuroninio tinklo architektūra. [26]

2.3. Treniravimas

Neuroninio tinklo treniravimas – tai svorių koregavimas, siekiant gauti kuo tikslesnį rezultatą norimam rezultatui. Tam, kad galėtume pradėti koreguoti svorius, juos visų pirma reikia turėti, todėl svoriai iš pradžių yra atsitiktiniai. [2]

Tarkime, jog mūsų norimas rezultatas yra o, dabartinis rezultatas yra y. Jeigu $y \neq o$, tada visi svoriai turėtų būti modifikuojami pagal 2.1 formulę:

$$\Delta w_i = \eta(y - o)x_i, \text{ kur } w_i\text{– svoris, } \eta\text{– treniravimo greitis, } x_i\text{– įvestis. [2]} \tag{2.1}$$

Pakeitus svorius, yra kartojamas svorių modifikavimas, iki tol, kol gaunami norimi rezultatai.

3. Konvoliuciniai neuroniniai tinklai

Konvoliuciniai neuroniniai tinklai – tai tokie neuroniniai tinklai, kurie turi konvoliucinį sluoksnį, kuris yra skirtas išgauti tam tikras duomenų charakteristikas, kaip pavyzdžiui nuotraukoje esančias spalvas, kampus, kraštus. [4]

3.1. Pagrindiniai skirtumai nuo tradicinių neuroninių tinklų

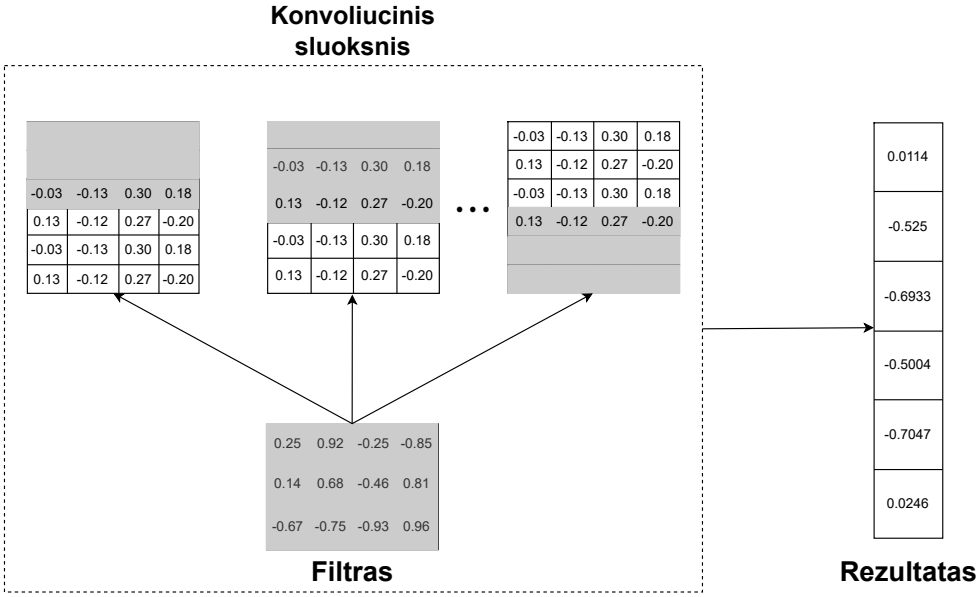
Atvirkščiai nei tradiciniuose neuroniniuose tinkluose, įvestys yra trijų dimensijų, neuronai nėra sujungti su visais kitais neuronais, o tik su tam tikra dalimi neuronų. Neuronai dalinasi tuo pačiu svoriu (yra naudojamas filtras). Treniravimo eigoje, duomenų dimensijos yra sumažinamos. Dėl šių savybių, yra naudojama mažiau atminties ir atliekama mažiau skaičiavimų. Todėl neuroninis tinklas gali būti treniruojamas su didesne duomenų aibe, greičiau ir efektyviau, nei su tradiciniais neuroniniais tinklais. [4]

3.2. Architektūra

Konvoliuciniai neuroniniai tinklai turi tris pagrindinius sluoksnius – konvoliucinį, sutelkimo (angl. pooling) ir pilnai sujungtąjį (angl. fully-connected). Konvoliucinis sluoksnis išgauna duomenų charakteristikas, sutelkimo sluoksnis sumažina duomenų dimensijas, o pilnai sujungtasis sluoksnis apjungia duomenis galutiniam rezultatui. [4]

3.2.1. Konvoliucinis sluoksnis

Konvoliuciniuose neuroniniuose tinkluose įvestis yra matricos struktūros. Kiekvienas matricos elementas yra skaičius. Pavyzdžiui, nuotraukų atpažinime, įvesties reikšmė būtų trimatė matrica, kur pirmoji dimensija būtų nuotraukos aukštis, antroji dimensija – ilgis, o trečioji dimensija – spalvų kanalų skaičius (RGB atveju, tai būtų 3). Teksto apdorojimo užduotyse, tarp kurių patenka ir XSS atakų aptikimas, tekstas yra paverčiamas į skaičius, naudojant vektorizaciją. [32]



8 pav. Konvoliucijos pavyzdys. [32]

Konvoliuciniame sluoksnyje yra naudojami filtrai – tam tikros dimensijos matricos, kurios slenkasi per įvestį. Kiekvienas filtro elementas yra sudauginamas su kiekvienu atitinkamu įvesties elementu. Sudauginti elementai yra susumuojami ir taip gaunamas vienas rezultato elementas. [32]

3.2.2. Sutelkimo sluoksnis

Sutelkimo sluoksnis yra atsakingas už duomenų dimensijų sumažinimą. Tokiu būdu neuroninis tinklas atlieka mažiau skaičiavimų ir yra sumažinamas persimokymas – atvejis, kai neuroninis tinklas duoda tikslius rezultatus, jeigu neuroniniam tinklui yra paduodami duomenys, ant kurių jis buvo mokomas, bet prastus rezultatus su naujais duomenimis. [32]

Sutelkimui dažniausiai yra naudojamos dvi funkcijos – vidutinio ir maksimalaus sutelkimo. Per įvestį slenkasi tam tikros dimensijos matrica. Jeigu yra naudojamas vidutinis sutelkimas, visos įvesties elementų reikšmės, patenkančios į tą matricą yra susumuojamos ir padalinamos iš matricos elementų kiekio, gaunamas vidurkis, kuris tampa sutelktos matricos elementu. Maksimalaus sutelkimo atveju, sutelktos matricos elementu tampa į besislenkančią matricą patenkantis įvesties elementas, turintis didžiausią reikšmę. [32]

3.2.3. Pilnai sujungtasis sluoksnis

Pilnai sujungtasis sluoksnis yra paskutinis konvoliucinių neuroninių tinklų sluoksnis. Jis atlieka klasifikavimo funkciją. Kiekvienas neuronas yra sujungtas su kiekvienu kitu neuronu. Prieš

tai, kol duomenys patenka į pilnai sujungtąjį sluoksnį, jie turi būti suplokštinami – paversti vienos dimensijos duomenimis. Taip yra todėl, kad norint klasifikuoti duomenis, turime gauti visas duomenų charakteristikas. [32]

3.3. Panaudojimas

Konvoliuciniai neuroniniai tinklai gali būti pritaikyti daugelyje sričių, tačiau vieni iš veiksmingiausių pritaikymų yra nuotraukų atpažinimas ir natūralios kalbos apdorojimas.

Nuotraukų atpažinimas: vienas iš labiausiai paplitusių konvoliucinių neuroninių tinklų panaudojimų. Pagrindinės sprendžiamos problemos: objektų aptikimas nuotraukose, jų klasifikavimas, nuotraukų rūšiavimas pagal turinį, veido atpažinimas, nuotraukų atgaminimas. Nuo 2010m. iki 2017m. kasmet vykęs ImageNet konkursas, skirtas efektyviausių nuotraukų atpažinimo algoritmams kurti, pastūmėjo naujų algoritmų, tokių kaip AlexNet ir ResNet atsiradimą. AlexNet Top5 tikslumas buvo 86%, o ResNet – 96.3%. Top5 tikslumas reiškia teisingos klasės patekimą į penkių klasių, su didžiausiomis tikimybėmis, aibę. [31]

Natūralios kalbos apdorojimas: į šią kategoriją patenka šnekos atpažinimas ir teksto klasifikavimas. Tyrimai parodė, jog šnekos atpažinimo užduotyje, konvoliuciniai neuroniniai tinklai, lyginant su tradiciniais, gali būti efektyvesni – santykinė paklaida be išankstinio neuroninio tinklo treniravimo sumažėja apie 8%, su išankstiniu treniravimu – apie 6%. Teksto klasifikavimo užduotyje, konvoliuciniai neuroniniai tinklai, su tam tikromis duomenų aibėmis, taip pat pranoksta kitus modelius. Įterptinių instrukcijų atakų aptikimas taip pat yra teksto klasifikavimo užduotis. [1]

4. Neuroninių tinklų naudojimas XSS atakų aptikimui

Šiame darbe aptarti tradiciniai XSS atakų aptikimo metodai, dažnai turi būti naudojami kartu su kitais metodais, nėra pilnai automatizuoti ir negarantuoja aptikimo. Dėl šių priežasčių, buvo pradėta ieškoti kitų būdų ir buvo pastebėta, jog neuroninių tinklų mokymas gali būti veiksminga alternatyva egzistuojantiems metodams. Neuroninių tinklų mokymas, skirtas XSS atakų aptikimui, gali būti atliekamas keliais būdais: prižiūrimuoju, neprižiūrimuoju ir sustiprintuoju mokymu.

Prižiūrimasis mokymas: tai toks mokymo būdas, kai neuroniniam tinklui yra duodama suklasifikuota kodo dalis – yra pažymima ar ji kenksminga, ar ne. 2023m. atliktame tyrime modelių treniravimui buvo naudojamas prižiūrimasis mokymas. Modeliai pasiekė 99.8% XSS atakų aptikimo tikslumą. Tačiau šis mokymo metodas turi ir trūkumų: didesnė persimokymo tikimybė, klasifikavimą turi atlikti žmogus, o jeigu klasifikavime yra klaidų, modelis bus apmokytas neteisingai klasifikuoti ir taip generuoti klaidingus rezultatus. [3]

Neprižiūrimasis mokymas: tai toks mokymo būdas, kai neuroniniam tinklui yra duodama nesuklasifikuota kodo dalis. Toks būdas gali būti naudingas, nes lyginant su prižiūrimuoju mokymu, persimokymo tikimybė yra mažesnė ir kiekvienos kodo dalies klasifikavimas duomenų aibėje rankiniu būdu gali būti neefektyvus. Tačiau neprižiūrimasis mokymas duoda geriausius rezultatus klasterizavimo, o ne klasifikavimo užduotyse. [16]

Sustiprintasis mokymas: tai toks mokymo būdas, kai neuroninis tinklas mokosi per klaidas ir bandymus. 2022m. atliktame tyrime, sustiprintasis mokymas buvo naudojamas kenksmingo kodo duomenų aibės modifikavimui. Duomenys buvo pakeičiami taip, kad XSS aptikimo sistemoms, juos būtų sudėtinga atpažinti (angl. obfuscation). Tokia pakeista duomenų aibė gali būti naudinga

treniruojant neuroninius tinklus, skirtus atpažinti sudėtingesnes XSS atakas. [7]

3 išeities kodas. Originalus HTML kodas [7]

```
1 <video onerror=alert(0);>
```

4 išeities kodas. Modifikuotas HTML kodas [7]

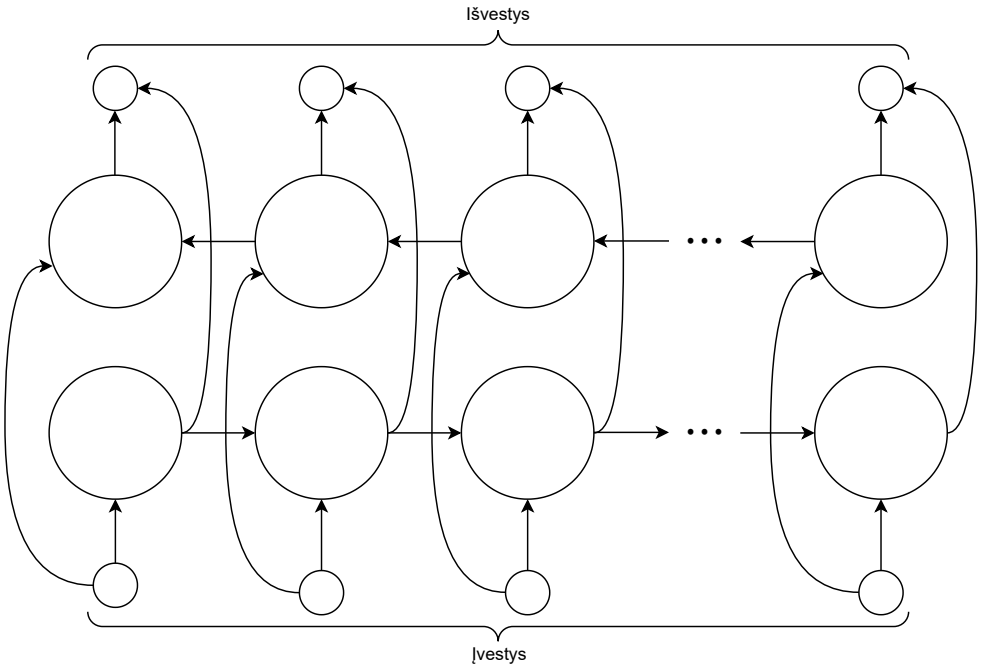
```
1 <video onERRor=top[/al/.source+/ert/.source](0);>/dfssdfsdfsfd/>
```

4.1. Naudojami modeliai

Konvoliuciniai neuroniniai tinklai (CNN): daugiausiai naudojami nuotraukų atpažinimo užduotyse, tačiau taip pat naudojami dvejetainės klasifikacijos užduotyse, tokiose kaip, XSS atakų aptikimas. Konvoliuciniai neuroniniai tinklai geba išgauti skirtingas charakteristikas kodo dalyje, taikant filtrus ir sumažinti persimokymo tikimybę taikant sutelkimo metodą. [27]

Rekurentiniai neuroniniai tinklai (RNN): daugiausiai naudojami natūralios kalbos apdorojimui ir šnekos atpažinimui. Pačiame tinkle, gali būti naudojami ciklai tarp neuronų, todėl jie ir yra vadinami rekurentiniais. RNN naudoja daugiau resursų negu konvoliucinai neuroniniai tinklai. [27]

Dvikrypčiai rekurentiniai neuroniniai tinklai (bi-RNN): rekurentinio neuroninio tinklo atšaka. Pagrindinis skirtumas nuo RNN – duomenys per įvestis gali patekti iš dviejų krypčių. Iš esmės, yra sujungiami du atskiri RNN su skirtingomis kryptimis. [27]



9 pav. Dvikrypčiai rekurentiniai neuroniniai tinklai (bi-RNN). [13]

Ilgos-trumpos atminties (LSTM) neuroniniai tinklai: rekurentinio neuroninio tinklo atšaka. Pagrindinis skirtumas nuo RNN – nykstančio gradiento problemos (atvejis, kai svorių pakeitimai tampa beveik neegzistuojantys) sprendimas. [27]

4.2. Kylantys iššūkiai

Poreikis nuolatos atnaujinti duomenų aibę: bėgant laikui, atsiranda naujų XSS atakų, kurių neuroninis tinklas gali neatpažinti. Tokiu atveju, reikėtų atnaujinti esamą duomenų aibę ir iš naujo treniruoti tinklą. Taip pat, šiuo metu nėra sukurtas viešai prieinamas ir nuolatos atnaujinamas suklasifikuotas XSS atakų duomenų rinkinys. [27]

Atakos, skirtos suklaidinti neuroninių tinklų modelį (angl. adversarial attacks): nors neuroniniai tinklai yra efektyvūs XSS atakų aptikime, kodo modifikavimas, paverčiant jį mažiau suprantamu neuroniniam tinklui, išlieka viena iš didžiausių problemų. [27]

Resursų reikalaujantis sprendimas: neuroninio tinklo mokymas ir naudojimas reikalauja daug sistemos resursų, todėl toks sprendimas XSS atakų aptikimui gali būti ne visiems tinkamas. [27]

Tinkamo modelio pasirinkimas: nėra vieno būdo nustatyti, koks modelis būtų tinkamiausias užduoties įgyvendinimui. Atlikus literatūros analizę, galima išsiaiškinti, kurie modeliai yra labiausiai tinkami XSS atakų aptikimui, tačiau kiekvienas iš modelių, duoda skirtingus rezultatus. [27]

Tinkamos duomenų aibės pasirinkimas: nuo to, kokie duomenys buvo parinkti neuroninio tinklo treniravimui, priklausys pačio neuroninio tinklo efektyvumas. Duomenų aibės dydis, duomenų kokybė, duomenų įvairumas yra svarbūs faktoriai, į kuriuos turėtų būti atsižvelgiama, prieš treniruojant neuroninį tinklą. [27]

4.3. Aptikimas

Tam, kad būtų galima aptikti XSS atakas, visų pirma reikia ištreniruoti neuroninį tinklą. Tam reikia atitinkamai paruošti duomenis. Atlikus treniravimo procesą, neuroninį tinklą galima naudoti klasifikavimo užduočiai.

4.3.1. Duomenų paruošimas neuroninio tinklo treniravimui

Tam, kad neuroninis tinklas galėtų teisingai išmokti aptikti XSS atakas, yra būtina atitinkamai paruošti duomenis. Visų pirma, kiekvienai kodo daliai yra pritaikoma tokenizacija – kodo išskaidymas į atskirus vienetus – žetonus (angl. tokens). Tokiu būdu, neuroninis tinklas geba išskirti skirtingas charakteristikas, tokias kaip HTML žymas, atributus, simbolius ar kitus raktažodžius. [27]

Originalus kodas	
<script>alert('ATTACK')</script>	
Tokenizacijos metodas	Tokenizuotas kodas
Žodžio tokenizacija	['<', 'script', '>', 'alert', '(', "'ATTACK", "'", ')', '<', '/script', '>']
Simbolio tokenizacija	['<', 's', 'c', 'r', 'i', 'p', 't', '>', 'a', 'l', 'e', 'r', 't', '(', "'", 'A', 'T', 'T', 'A', 'C', 'K', "'", ')', '<', '/', 's', 'c', 'r', 'i', 'p', 't', '>']
Žodžio tokenizacija (HTML žyma laikoma vienu žodžiu)	['<script>', 'alert', '(', "'ATTACK'", ')', '</script>']

1 lentelė. Skirtingi tokenizacijos metodai. [27]

Toliau, kodui yra pritaikoma teksto vektorizacija – teksto pavertimas į skaičius. Teksto vektorizacija yra reikalinga, kadangi neuroniniai tinklai apdoroja duomenis skaičiais. Kiekvienas žetonas yra konvertuojamas į skaitmenų masyvą. [27]

Teksto vektorizacijai gali būti naudojami skirtingi modeliai, tokie kaip „Word2Vec“, „GloVe“, „FastText“. Vienas iš „Word2Vec“ ir „GloVe“ trūkumų – negebėjimas vektorizuoti teksto, kurio nebuvo duomenų aibėje, kuri buvo naudojama modelio treniravimui. „FastText“ modelis šią problemą išsprendžia išskaidydamas žetoną į dar smulkesnius vienetus, juos paversdamas skaitmenimis ir apskaičiuodamas jų vidurkį. Tarkime, jeigu žetonas yra `alert` ir norime skaidyti žetoną po 2 simbolius, išskaidytas žetonas atrodytų taip: `al`, `le`, `er`, `rt`. [5]

2017m. atliktame tyrime, „FastText“ modelis parodė geresnius tikslumo rezultatus, negu „Word2Vec“ ir „GloVe“ modeliai. Tyrimas taip pat parodė, jog „FastText“ skaičiavimo sudėtingumas yra panašus į „Word2Vec“ ir „GloVe“ algoritmus. [5]

Žetonas	Vektorizuotas žetonas
<script>	[-0.05151805, -1.6048255, -1.699632, 0.40161118]
alert	[-0.37749016, -1.876152, -1.1220449, 0.29296082]
(	[-0.13210526, -1.0963918, -0.4318175, 0.39499637]
'ATTACK'	[0.4627794, -0.42491367, -0.23195651, -0.02324018]
)	[0.02772295, -0.9239012, -0.4520524, 0.48856288]
</script>	[0.00752187, -0.9768646, -2.7441828, 0.9101639]

2 lentelė. Žetonų vektorizacijos pavyzdys naudojant „Genim“ bibliotekos „FastText“ modelį. [27]

2 lentelėje yra pavaizduotas kodo `<script>alert('ATTACK')</script>` vektorizavimas, naudojant 4 dimensijų vektorių ir žodžio tokenizaciją, kai HTML žyma yra laikoma, kaip vienas žodis. Įprastai, dimensijos dydis dažniausiai būna nuo 100 iki 300. Kiekvienas žetonas yra konvertuojamas į vektorių.

Atlikus tokenizaciją ir vektorizaciją, modifikuotus duomenis galima paduoti neuroniniam tinklui, kad būtų pradėtas treniravimas.

4.3.2. Klasifikavimas

Visų pirma, kenkėjo kodas, kurį analizuos neuroninis tinklas, turėtų būti dekoduojamas. Kenkėjas gali naudoti skirtingus kodavimo metodus, kurių neuroninis tinklas neatpažins, jeigu buvo treniruotas su duomenimis, kuriuose nebuvo naudojamas kodavimas. [27]

Originalus kodas	
alert('ATTACK')	
Kodavimo metodas	Kodas
JSFuck	[][(![]+[])[+[]]+(![]+[])[!+[]+!+[]]+(![]+[])[+!+[]]+(!![]+[]) <...tęsiasi kodas>
Obfuskacija	function _0x22db(){var _0x196114=['4YQNIzL','7ygBLuL' <...tęsiasi kodas>
Unicode	\u0061\u006c\u0065\u0072\u0074('ATTACK')

3 lentelė. Skirtingi kodavimo metodai. [27]

Toliau, analogiškai, kaip ir prieš neuroninio tinklo treniravimo procesą, kodo dalį, kurią norima klasifikuoti, turėtume tokenizuoti ir vektorizuoti. Gautus duomenis, neuroninis tinklas apdoros ir galiausiai grąžins rezultatą – kenksminga ar nekenksminga kodo dalis. [27]

5. Tyrimas

5.1. Pagrindinės technologijos

Python: dinamiškai tipizuota programavimo kalba, dažniausiai naudojama duomenų analizei, dirbtinių neuroninių tinklų mokymui, svetainių kūrimui.

Spyder: atvirojo kodo integruota kūrimo aplinka, skirta Python progamavimo kalbai.

PyTorch: paketas, skirtas dirbtinių neuroninių tinklų mokymui. Pradedant tyrimą, buvo pastebėta, jog pasirinkta TensorFlow biblioteka naujausiose versijose nepalaiko GPU naudojimo Windows operacinėje sistemoje. Treniravimas ant CPU yra lėtesnis negu treniruojant su GPU. Dėl šios priežasties, buvo pasirinkta PyTorch biblioteka.

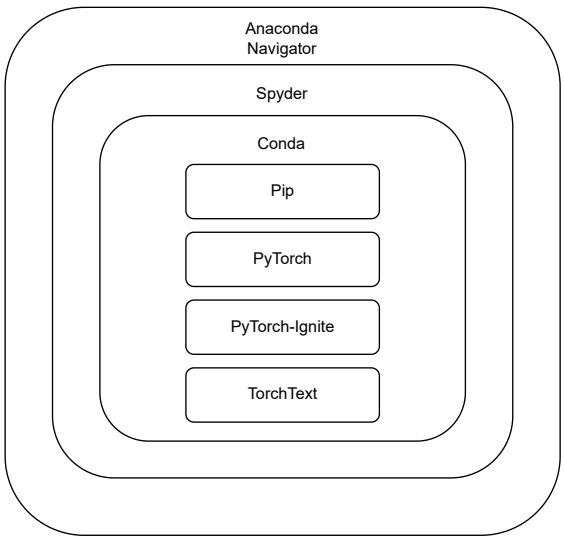
PyTorch-Ignite: tai paketas, skirtas supaprastinti ir abstraktuoti PyTorch kodą. Tai palengvina kodo skaitomumą ir valdymą.

TorchText: paketas, skirtas natūralios kalbos apdorojimo užduotims įgyvendinti.

Pip: atvirojo kodo paketų valdymo įrankis, skirtas Python programavimo kalbai.

Conda: atvirojo kodo paketų valdymo įrankis, skirtas tokioms programavimo kalboms, kaip Python, R, Java, JavaScript, C/C++ ir daugiau.

Anaconda Navigator: programa, skirta aplikacijų paleidimui, Conda paketų valdymui, naudojantis grafine vartotojo sąsaja.



10 pav. Aplinkos diagrama

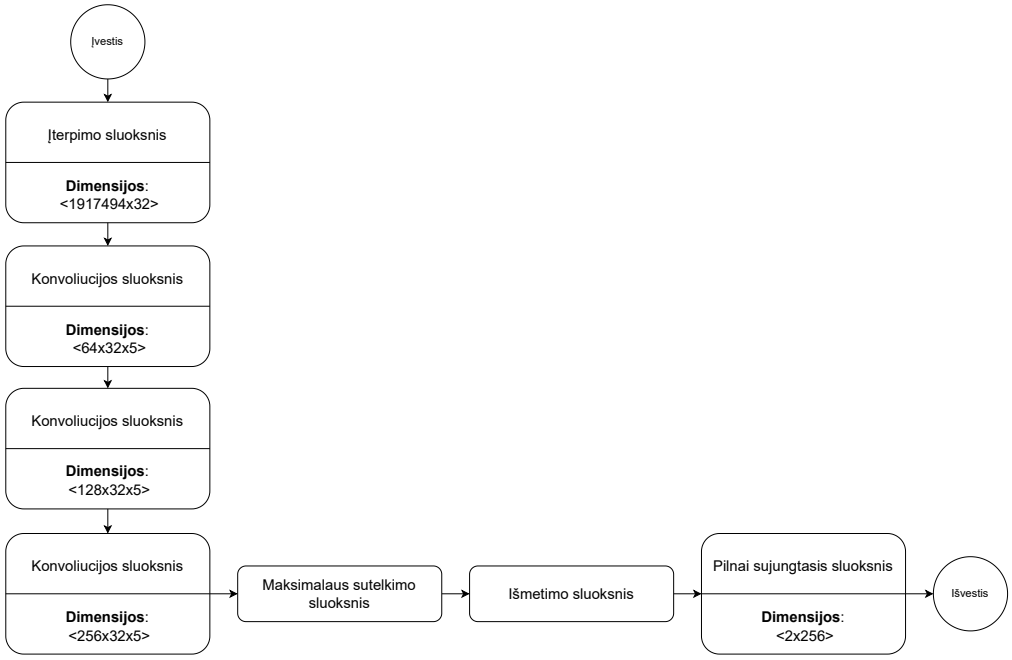
5.2. Techninė įranga

- Operacinė sistema: Microsoft Windows 11 Home
- Procesorius: Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz, 6 branduoliai

- Grafikos apdorojimo įrenginys: NVIDIA GeForce RTX 2060
- Atmintis: G.Skill Aegis 16GB (2666MHz)
- Talpykla: Kingston A2000 (1TB)

5.3. Modelis

Konvoliucinių neuroninių tinklų modelis yra sudarytas iš įterpimo sluoksnio – šiame sluoksnyje yra konvertuojamos įvesties reikšmės į skaitinį formatą, konvoliucijos sluoksnių, kuriuose atliekama konvoliucijos operacija, maksimalaus sutelkimo sluoksnio, kuriame yra panaikinama dalis duomenų, išmetimo sluoksnio, kuriame dalis duomenų paverčiama nuliais ir galiausiai pilnai sujungtojo sluoksnio, kuriame yra apjungiami duomenis galutiniam rezultatui.



11 pav. Konvoliucinių neuroninių tinklų modelis, naudojamas tyrimui ir implementacijai.

5.4. Duomenys

Neuroninio tinklo modelio treniravimui ir validacijai bus naudojamos atskiros duomenų aibės. Treniravimui yra naudojama mažesnė duomenų aibė, negu validacijai skirtos duomenų aibės, todėl, jog validacijos duomenų aibės nebuvo specialiai kuriamos neuroninio tinklo treniravimui.

5.4.1. Treniravimui skirti duomenys

Neuroninio tinklo modelio treniravimui bus naudojama viena duomenų aibė iš viso turinti 13686 kodo eilučių, 7373 iš jų – kenksmingos, 6313 – nekenksmingos.

Kodo eilutės tipas	Kodo eilučių kiekis
Kenksmingas	7373
Nekenksmingas	6313

4 lentelė. Treniravimui skirtos duomenų aibės informacija.

5.4.2. Validacijai skirti duomenys

Neuroninio tinklo modelio validacijai bus naudojamos dvi duomenų aibės, viena jų turi vien kenksmingas kodo eilutes, kita – nekenksmingas.

Duomenų aibė	Kodo eilutės tipas	Kodo eiliučių kiekis
1	Kenksmingas	33426
2	Nekenksmingas	31407

5 lentelė. Validacijai skirtų duomenų aibių informacija.

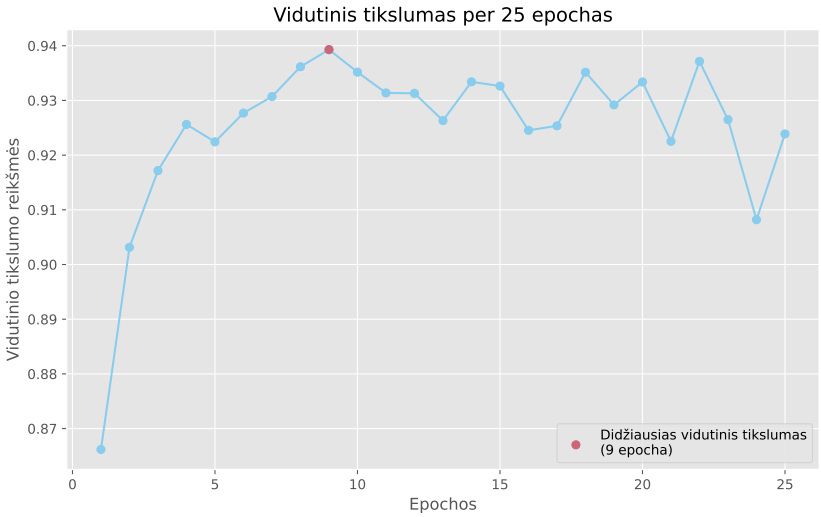
5.5. Eksperimentai

Optimalaus epochų kiekio, mokymosi koeficiento ir išmetimo koeficiento nustatyme bus naudojamas „GloVe“ vektorizacijos metodas. Tokenizacija bus naudojama simbolių lygmenyje.

Iš pradžių, bus nustatomas optimalus epochų kiekis, tam, jog kitiems eksperimentams nebūtų be reikalo naudojami resursai. Toliau, bus surandamas optimalus mokymosi koeficientas ir optimalus išmetimo koeficientas. Nustatyti tinkamiausius koeficientus bus naudojamas vidutinio tikslumo rodiklis, imant geriausios epochos duomenis.

5.5.1. Optimalaus epochų kiekio nustatymas

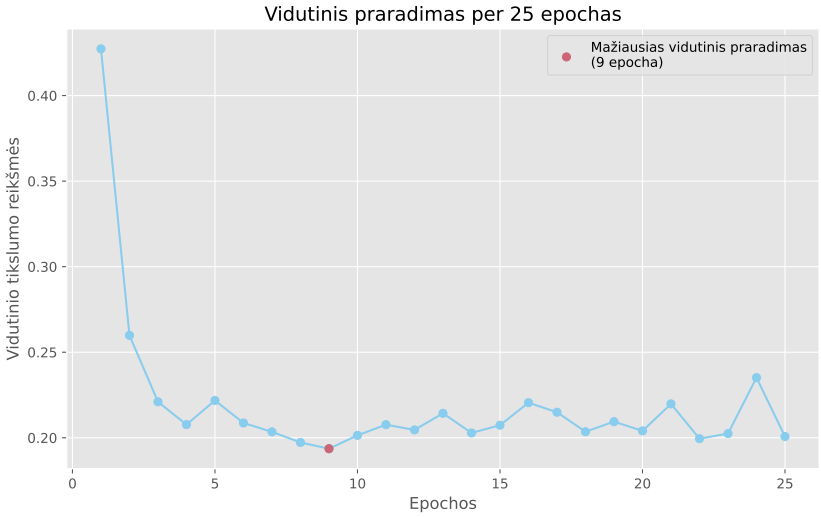
Prieš pradedant atlikinėti kitus eksperimentus, yra svarbu nustatyti optimalų epochų kiekį. Nustačius šį rodiklį, žinosime per kiek epochų neuroninis tinklas pasieks geriausią vidutinį tikslumą. Šiam eksperimentui bus naudojamos 25 epochos. Epocha – perėjimas per visą treniravimui skirtą duomenų aibę. Nustačius optimalų epochų kiekį, nereikės treniruoti neuroninio tinklo per visas 25 epochas, kadangi dažniausiai neuroninis tinklas pasiekia geriausią vidutinį tikslumą anksčiau.



12 pav. Vidutinis tikslumas per 25 epochas.

Eksperimentas buvo vykdomas su tokiais modelio parametrais: rinkinio dydis – 64, epochų skaičius – 25, mokymosi koeficientas – 0.00001, svornio nykimo koeficientas – 0.00001, išmetimo koeficientas – 0.5.

Atlikus eksperimentą, 12 pav. galima pastebėti, jog modelis pasiekia didžiausią vidutinį tikslumą devintoje epochoje.



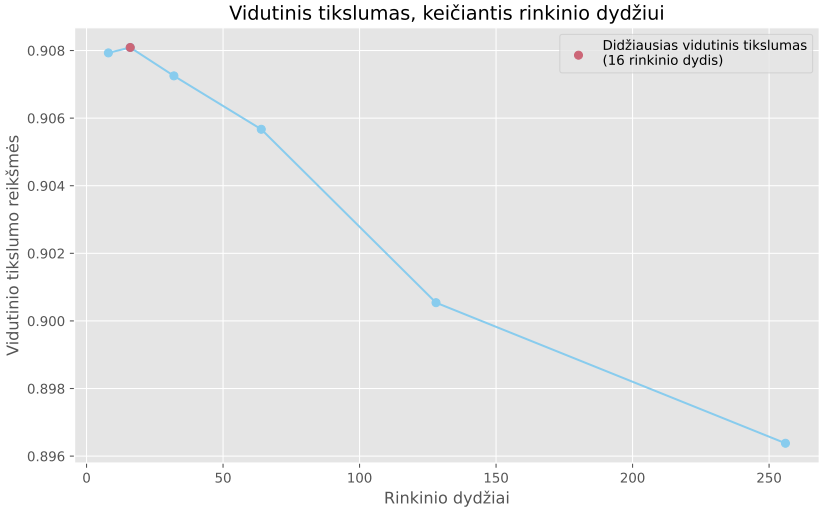
13 pav. Vidutinis praradimas per 25 epochas.

Taip pat buvo tikrinama, kurioje epochoje modelis pasiektų mažiausią vidutinį praradimą. Vidutinis praradimas rodo, kiek stipriai modelio rezultatai yra nutolę nuo tikrųjų. Pagrindinis skirtumas nuo vidutinio tikslumo yra tas, jog skaičiuojant vidutinį tikslumą nėra atsižvelgiama su kokia tikimybe modelis nuspėjo rezultatą.

13 pav. galima matyti, jog mažiausią vidutinį praradimą modelis pasiekia taip pat devintoje epochoje. Todėl, tolesniuose eksperimentuose bus naudojamos devynios epochos.

5.5.2. Optimalaus rinkinio dydžio nustatymas

Rinkinio dydis – kiek duomenų (šio tyrimo atveju, tai būtų kodo eilutės) bus paduodama modeliui vienu metu. Su didesniu rinkinio dydžiu neuroninio tinklo modelį treniruoti galima greičiau, negu su mažesniu. Eksperimento metu buvo išsiaiškinta, kokį rinkinio dydį naudojant buvo gaunamas aukščiausias vidutinis tikslumas.



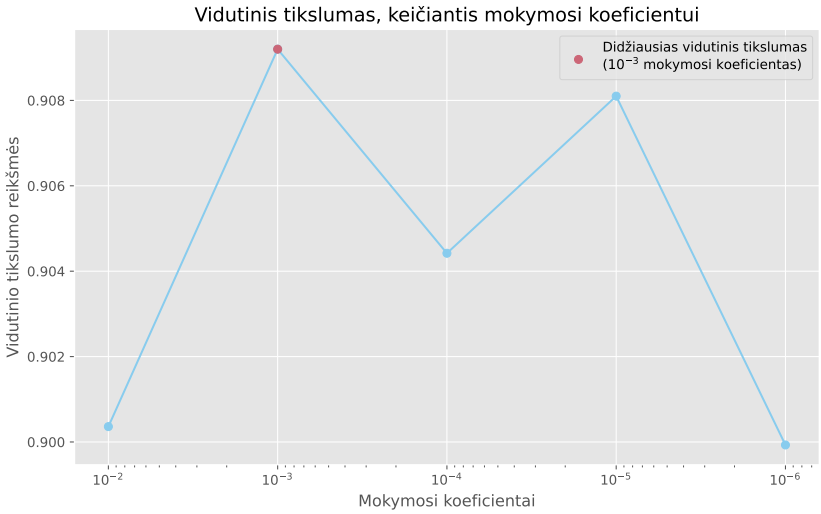
14 pav. Vidutinis tikslumas, keičiantis rinkinio dydžiui.

Eksperimento rezultatai, 14 pav. rodo, jog didžiausias vidutinis tikslumas yra pasiekiamas su rinkinio dydžiu – 16. Taip pat galima pastebėti, jog mažesnis rinkinio dydis dažniausiai rodo geresnius rezultatus. Tolimesniuose eksperimentuose bus naudojamas būtent šis rinkinio dydis.

5.5.3. Optimalaus mokymosi koeficiento nustatymas

Norint išvengti persimokymo, reikia nustatyti tinkamą mokymosi koeficientą. Mokymosi koeficientas nusako kiek smarkiai keisis svoriai kiekvienos iteracijos metu. Kuo mažesnis mokymosi koeficientas, tuo mažesnė įtaka svorio pokyčiams. Bandyme buvo tikrinama, su kokia mokymosi koeficiento reikšme modelis gali pasiekti didžiausią vidutinį tikslumą.

Eksperimento metu buvo naudojamos reikšmės nuo 10^{-2} iki 10^{-6}. 15 pav. atvaizduojamas vidutinio tikslumo pokytis, keičiantis mokymosi koeficientui. Su 10^{-3} mokymosi koeficiento reikšme yra pasiekiamas geriausias vidutinio tikslumo rezultatas. Ši reikšmė bus naudojama tolimesniuose eksperimentuose.

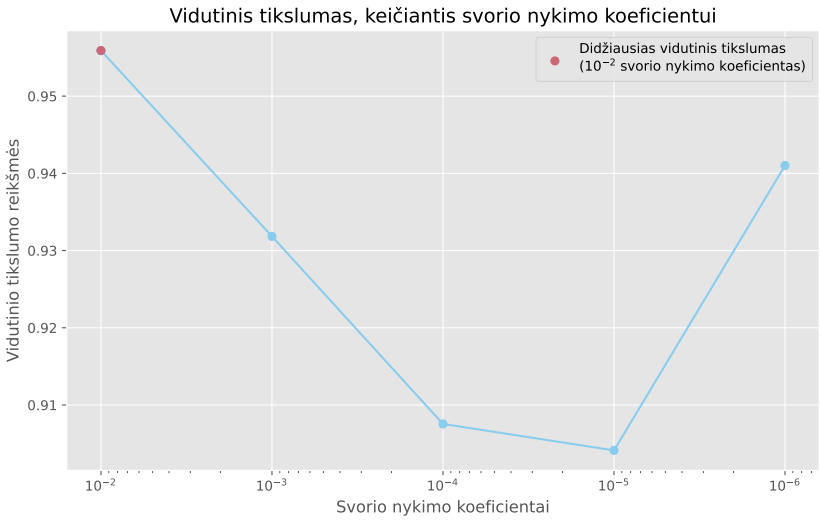


15 pav. Vidutinis tikslumas, keičiantis mokymosi koeficientui.

5.5.4. Optimalaus svorio nykimo koeficiento nustatymas

Svorio nykimas – technika, skirta sumažinti neuroninio tinklo modelio persimokymą. Svorio nykimas neleidžia modelio svoriams būti dideliems. Kuo didesnis svorio nykimo koeficientas, tuo mažesni svoriai yra naudojami. Bandymo tikslas – išsiaiškinti su kokiu svorio nykimo koeficientu neuroninio tinklo modelis gali pasiekti didžiausią vidutinį tisklumą.

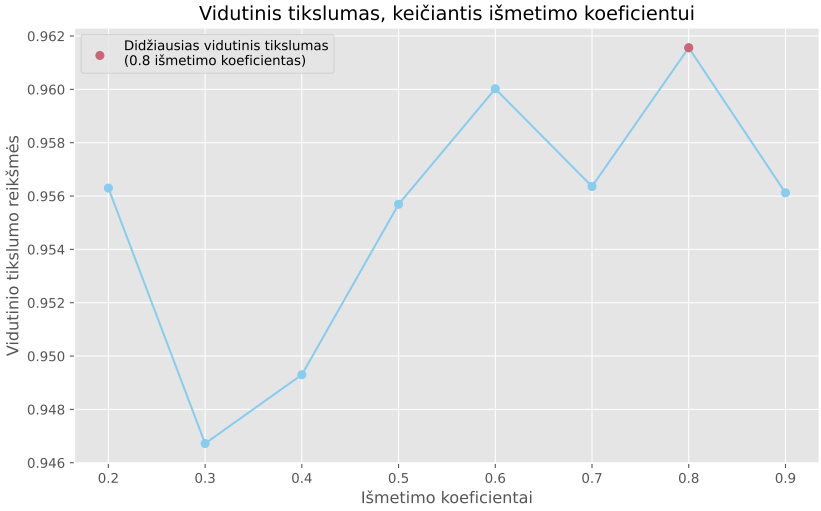
Eksperimente buvo naudojami šie svorio nykimo koeficientai: nuo 10^{-2} iki 10^{-6}. 16 pav. pavaizduotos vidutinio tikslumo reikšmės, kintant svorio nykimo koeficientui rodo, jog geriausi rezultatai yra pasiekiam su 10^{-2} koeficientu.



16 pav. Vidutinis tikslumas, keičiantis svorio nykimo koeficientui.

5.5.5. Optimalaus išmetimo koeficiento nustatymas

Išmetimas – technika, skirta sumažinti neuroninio tinklo modelio persimokymą, pašalinant tam tikrą dalį duomenų. Eksperimento tikslas – rasti maksimalaus sutelkimo koeficientą, su kuriuo būtų gaunamas maksimalus vidutinis tikslumas.



17 pav. Vidutinis tikslumas, keičiantis išmetimo koeficientui.

Atlikus eksperimentą, buvo išsiaiškinta, jog geriausias vidutinio tikslumo rezultatas yra gaunamas su 0.8 išmetimo koeficientu. Ši reikšmė bus naudojama tolimesniems bandymams. 17 pav. išmetimo koeficientui didėjant, pastebimas ir vidutinio tikslumo kylimas.

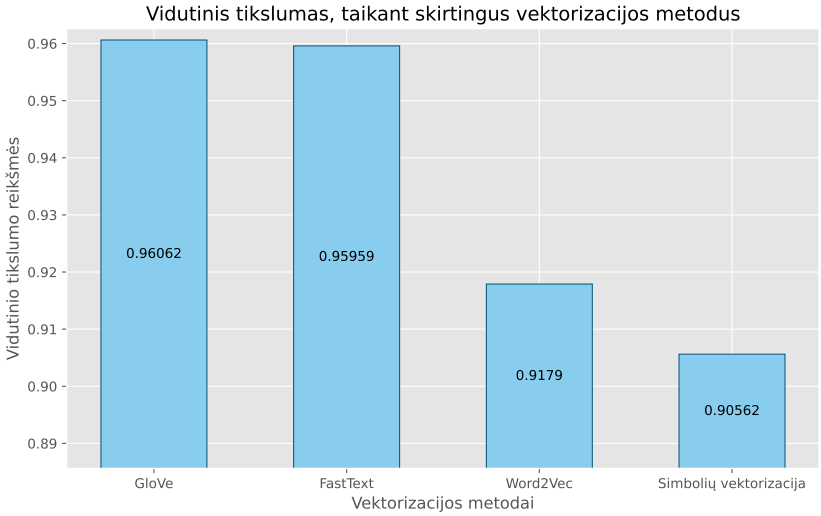
5.5.6. Skirtingi vektorizacijos metodai

Vektorizacija – teksto konvertavimas į skaitinį formatą. Vektorizacijai dažniausiai yra naudojami neuroninio tinklo modeliai, tokie kaip „GloVe“, „FastText“, „Word2Vec“. Yra ir kitų būdų, kaip paversti tekstą į skaičius, nenaudojant neuroninių tinklų modelių, pavyzdžiui – naudojant ASCII kodą, simbolių skaičių pagal abėcėlės tvarką (jeigu simboliai nėra raidės, tam tikra tvarka išdėstyti po abėcėlės raidžių).

Šiame bandyme neuroninis tinklas buvo treniruojamas su keturiais skirtingais vektorizacijos metodais – „GloVe“, „FastText“, „Word2Vec“ ir simbolių vektorizacija.

5 išeities kodas. Simbolių vektorizacijoje naudojamas simbolių eiliškumas [9]

```
1 abcdefghijklmnopqrstuvwxyz0123456789 — ,;.!?: ’\"/\\|_@#$%^&*~‘+—=<>()[]{}
```

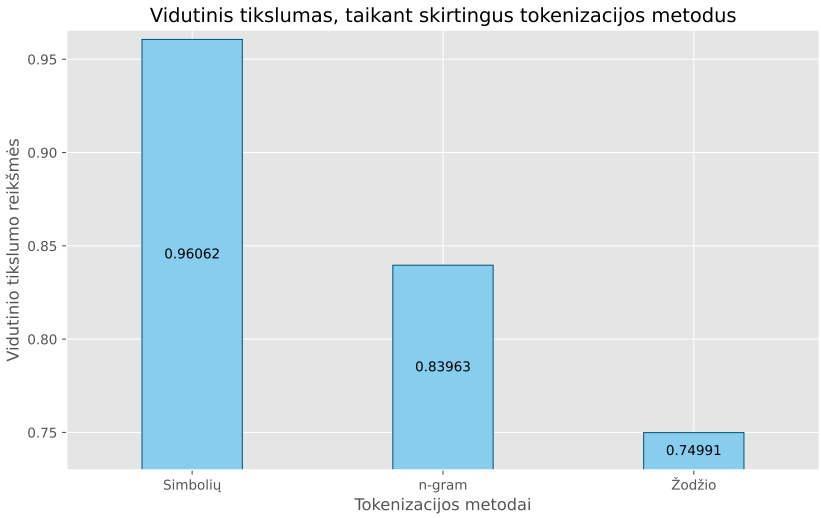


18 pav. Vidutinis tikslumas, taikant skirtingus vektorizacijos metodus.

18 pav., eksperimento rezultatai rodo, jog didžiausią vidutinį tikslumą neuroninio tinklo modelis pasiekia su „GloVe“ vektorizacijos metodu. Prasčiausias rezultatas yra gaunamas, taikant simbolių vektorizaciją. Toliau, būsimuose eksperimentuose, neuroninis tinklas bus treniruojamas taikant „GloVe“ vektorizacijos metodą.

5.5.7. Skirtingi tokenizacijos metodai

Tokenizacija – teksto išskaidymas į atskirus simbolius/simbolių aibes – žetonus. Šio eksperimento metu buvo naudojami trys tokenizacijos metodai – simbolių, n-gram ir žodžio.

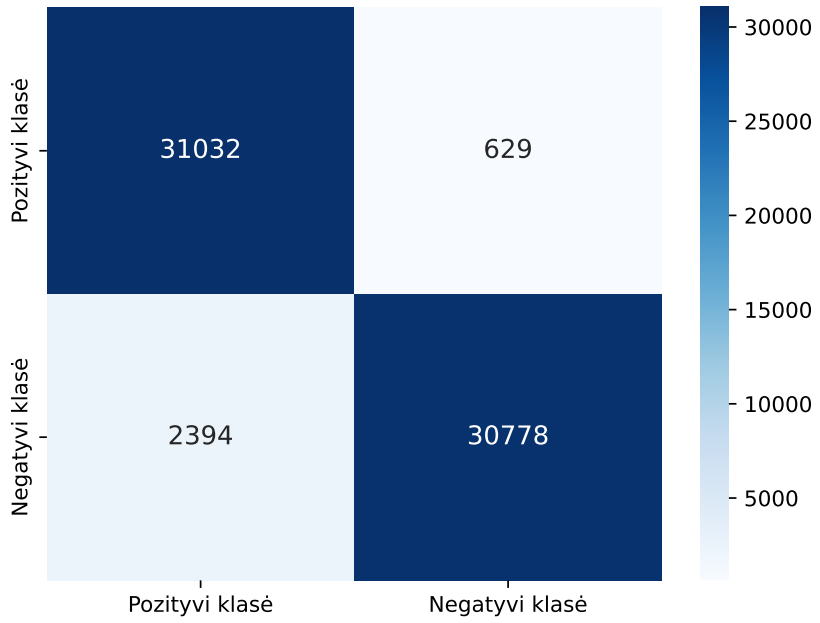


19 pav. Vidutinis tikslumas, taikant skirtingus tokenizacijos metodus.

Atlikus bandymą, 19 pav. galima pamatyti, jog simbolių tokenizacija yra ženkliai pranašesnė už n-gram ir žodžio tokenizacijos metodus. Tolimesniuose bandymuose bus taikomas simbolių tokenizacijos metodas.

5.5.8. Modelio rezultatai

Atlikus eksperimentus ir išsiaiškinus tinkamiausius parametrus neuroninio tinklo modelio treniravimui, galima aptarti galutinius modelio rezultatus.



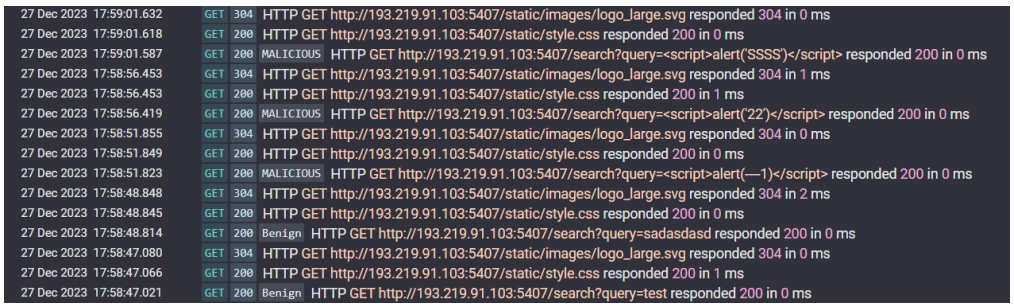
20 pav. Modelio rezultatai klasifikavimo matricoje.

Modelio rezultatai yra atvaizduojami 20 pav., klasifikavimo matricoje. Pozityvi klasė reiškia kenksmingą klasę, negatyvi – nekenksmingą. Modelis sugebėjo teisingai aptikti 31032 kenksmingas kodo eilutes iš 33426, t. y, daugiau nei 92.8%. 629 kodo eilutės buvo klasifikuotos, kaip kenksmingos, kai iš tiesų jos buvo – nekenksmingos, t. y, daugiau nei 2%. 2394 kodo eilučių buvo atpažintos kaip nekenksmingos, kai iš tiesų jos buvo kenksmingos, t. y, daugiau nei 7.1%. Galiausiai, modelis teisingai atpažino 30778 nekenksmingas kodo eilutes iš 31407, t. y, daugiau nei 97.9%.

6. Pritaikymas

Sukurtas ir apmokytas modelis buvo pritaikytas žurnalizavime (angl. logging). OpenNebula MIF aplinkoje buvo sukurtos dvi virtualios mašinos **defending_vm** – atsakinga už svetainės paleidimą ir neuroninio tinklo modelio integraciją ir **seq_vm** – atsakinga už „Seq“ serverio paleidimą.

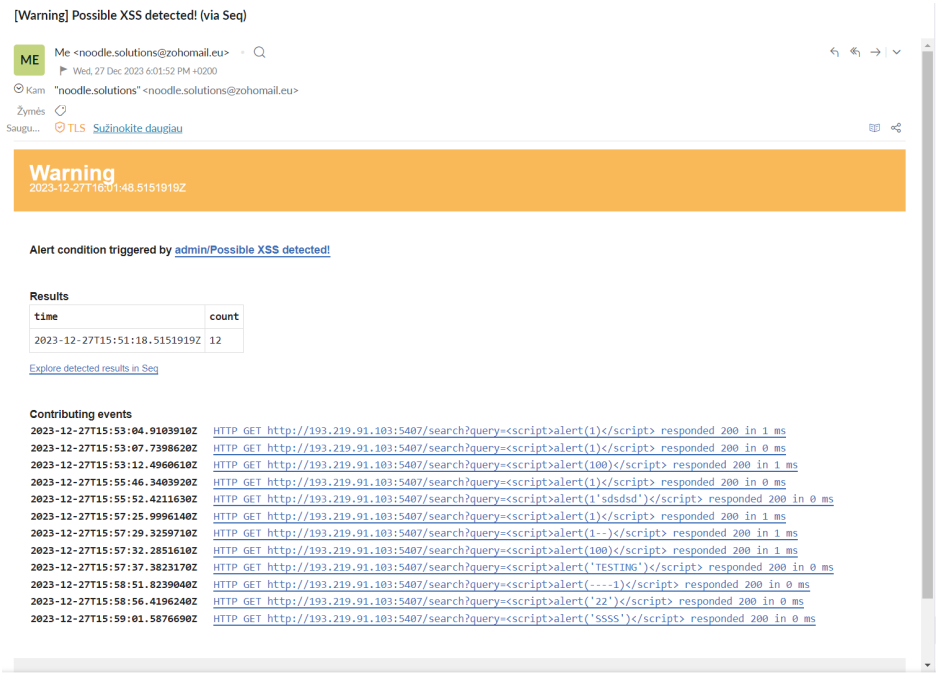
„Seq“ – žurnalizavimo, analizės ir perspėjimų serveris. **defending_vm** virtualioje mašinoje svetainė, kurioje yra integruotas neuroninio tinklo modelis, siunčia kiekvieną vartotojo užklausos informaciją į **seq_vm** virtualioje mašinoje paleistą „Seq“ serverį. Užklausos informacijoje yra – užklausos metodas, kodas, URL, trukmė, ar užklausa yra kenksminga ar ne.



21 pav. Užklausų informacija „Seq“ grafinėje sąsajoje.

21 pav. galima matyti „Seq“ grafinę sąsają, kurioje yra atvaizduojama užklausų informacija. Paspaudus ant išsaugotos užklausos, galima pamatyti detalesnę informaciją.

Jeigu neuroninio tinklo modelis aptiko, jog užklausoje yra kenksmingo kodo, bus atsiunčiamas el. laiškas su visomis kenksmingomis užklausomis. „Seq“ buvo sukonfigūruotas nusiųsti el. laiškus apie potencialius pavojus kas 10min.



22 pav. „Seq“ serverio išsiųstas el. laiškas, pranešantis apie potencialią ataką.

22 pav. yra pavaizduotas el. laiškas su sugrupuotomis potencialiai kenksmingomis užklausomis. Prie kiekvienos užklausos yra nurodyta užklausos data ir laikas, užklausos informacija.

„Seq“ kaupia užklausų duomenis ir gali juos atvaizduoti vartotojo pasirinktose diagramose.



23 pav. „Seq“ prietaisų skydelis

23 pav. yra pavaizduotas „Seq“ prietaisų skydelis, kurio struktūra gali būti modifikuojama pačio vartotojo. Renkama statistika gali būti naudinga norint, pavyzdžiui, sužinoti kokiu paros metu dažniausiai vyksta atakos, kokiame svetainės puslapyje ataka yra vykdoma dažniausiai ir t.t.

Išvados ir rekomendacijos

Bakalauro baigiamajame darbe buvo sukurtas konvoliucinių neuroninių tinklų modelis, skirtas aptikti įterptinių instrukcijų atakas. Modelį sudaro įterpimo sluoksnis, trys konvoliucijos sluoksniai, kurių svoriai gilėjat auga, maksimalaus sutelkimo sluoksnis, išmetimo sluoksnis ir galiausiai, pilnai sujungtasis sluoksnis. Tyrimo metu buvo atliekami eksperimentai, kurių metu buvo siekiama išsiaiškinti, su kokiomis parametrų reikšmėmis ir su kokiais tokenizacijos ir vektorizacijos metodais modelis gali pasiekti geriausią vidutinį tikslumą.

Visų pirma, tyrimas buvo pradėtas optimalaus epochų kiekio nustatymu. Naudojant 25 epochas šiam eksperimentui, buvo nustatyta, jog su 9 epochomis, modelis pasiekia didžiausią vidutinį tikslumą. Tolimesni bandymai buvo atliekami su 9 epochomis. Kaip ir su epochomis, taip ir su kitomis parametrų reikšmėmis, po kiekvieno eksperimento nustačius geriausią parametro reikšmę, jos buvo naudojamos kitiems eksperimentams.

Toliau, buvo išsiaiškinta, jog su rinkinio dydžiu – 16, modelis rodo geriausius rezultatus, o rinkinio dydžiui didėjant, krenta ir vidutinis tikslumas. Mokymosi koeficiento nustatymo bandyme, aukščiausias vidutinis tikslumas buvo pasiektas su 10^{-3} mokymosi koeficiento reikšme. Svorio nykimo koeficiento nustatyme, optimali reikšmė buvo 10^{-2}. Optimalaus išmetimo koeficiento bandyme, modelis demonstravo maksimalų vidutinį tikslumą su 0.8 išmetimo koeficientu.

Išsiaiškinus optimalias parametrų reikšmes, buvo tiriama skirtingų vektorizacijos ir tokenizacijos metodų įtaka modelio vidutiniam tikslumui. Buvo nustatyta, jog su GloVe vektorizacijos metodu, neuroninis tinklas rodo geriausius rezultatus, o prasčiausius – su simbolių vektorizacija. Tiriant tokenizacijos metodus, buvo išsiaiškinta, jog simbolių tokenizacija yra pranašesnė už n-gram ir žodžio tokenizacijos metodus.

Galiausiai, nustačius optimalias parametrų reikšmes ir vektorizacijos, ir tokenizacijos metodus, buvo sužinoti modelio rezultatai. Tikrai teigiami rezultatai – 92.8%, klaidingai teigiami – 2%, klaidingai neigiami – 7.1%, tikrai neigiami – 97.9%.

Norint toliau pratęsti, plėsti ir tobulinti šį tyrimą, būtų rekomenduojama:

- Naudoti didesnę duomenų aibę neuroninio tinklo modelio treniravimui.
- Atlikti pakeitimus neuroninio tinklo modelio architektūroje, pvz. pridėti daugiau konvoliucinių sluoksnių, pritaikyti normalizacijos metodus ir stebėti pokyčius.

Ateities tyrimų planas

Neuroninių tinklų potencialas įterptinių instrukcijų atakos kontekste yra itin aukštas. Tačiau, kaip ir su kiekvienu sprendimo būdu tinklų saugume, atsiranda tam tikri iššūkiai. Vienas iš tokių iššūkių – atakos, specializuotos neuroninių tinklų modelio suklaidinimui. Neuroniniams tinklams populiarėjant ir vis dažniau jų pritaikymą aptinkant tinklų saugume, didėja neuroninių tinklų patikimumo svarba.

Ateities tyrimuose, būtų galima palyginti skirtingus apsisaugojimo būdus prieš specializuotas atakas, išsiaiškinti jų efektyvumą. Yra keli apsisaugojimo būdai [30]:

- Neuroninio tinklo apmokymas, pridedant specializuotas atakas į treniravimo duomenų aibę. Vienas iš būdų išgauti tokias atakas, yra sukurti atskirą neuroninio tinklo modelį, generuojantį atakas prieš apsaugai skirtą neuroninį tinklą.
- Atskiro neuroninio tinklo modelio sūkurimas, skirtas stebėti ir atskirti specializuotas atakas.
- Neuroninio tinklo modelyje implementuotas filtravimas, patobulinantis apsaugą prieš tokio tipo atakas.
- Kiti algoritmai, skirti sumažinti specializuotų atakų įtaką.

Literatūros šaltiniai

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533--1545, 2014.

[2] Ajith Abraham. Artificial neural networks. *Handbook of measuring system design*, 2005.

[3] Qasem Abu Al-Haija. Cost-effective detection system of cross-site scripting attacks using hybrid learning approach. *Results in Engineering*, 19:101266, 2023.

[4] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1--6. Ieee, 2017.

[5] Jon Ezeiza Alvarez and Hannah Bast. A review of word embedding and document similarity algorithms applied to academic text. *Bachelor thesis*, 1, 2017.

[6] Domantas Baliukynas. Atvirojo kodo programinės įrangos, aptinkančios įterptinių instrukcijų atakų (XSS) pažeidžiamumus, analizė. https://www.overleaf.com/read/xsvxkxgnykhv#43b47e, 2023.

[7] Li Chen, Cong Tang, Junjiang He, Hui Zhao, Xiaolong Lan, and Tao Li. XSS adversarial example attacks based on deep reinforcement learning. *Computers & Security*, 120:102831, 2022.

[8] Ahmed Elsobky. Unleashing an Ultimate XSS Polyglot. https://github.com/0xsobky/HackVault/wiki/Unleashing-an-Ultimate-XSS-Polyglot, 2018.

[9] Eric. XSS classification using Textcnn with pytorch. https://www.kaggle.com/code/xeric7/xss-classification-using-textcnn-with-pytorch, Oct 2022.

[10] Damiano Esposito, Marc Rennhard, Lukas Ruf, and Arno Wagner. Exploiting the potential of web application vulnerability scanning. In *ICIMP 2018 the Thirteenth International Conference on Internet Monitoring and Protection, Barcelona, Spain, 22-26 July 2018*, pages 22--29. IARIA, 2018.

[11] Bernhard Garn, Daniel Sebastian Lang, Manuel Leithner, D Richard Kuhn, Raghu Kacker, and Dimitris E Simos. Combinatorially xssing web application firewalls. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 85--94. IEEE, 2021.

[12] Mario Heiderich, Jörg Schwenk, Tilman Frosch, Jonas Magazinius, and Edward Z Yang. mxss attacks: Attacking well-secured web-applications by using innerhtml mutations. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 777--788, 2013.

[13] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2515--2528, 2022.

[14] Dr M Jayamsakthi Shanmugam. Cross Site Scripting-Latest developments and solutions: A survey. *Int. J. Open Problems Compt. Math*, 1(2), 2008.

[15] Jasleen Kaur and Urvashi Garg. A Detailed Survey on Recent XSS Web-Attacks Machine Learning Detection Techniques. In *2021 2nd Global Conference for Advancement in Technology (GCAT)*, pages 1--6. IEEE, 2021.

[16] Jasleen Kaur, Urvashi Garg, and Gourav Bathla. Detection of cross-site scripting (XSS) attacks using machine learning techniques: a review. *Artificial Intelligence Review*, pages 1--45, 2023.

[17] Md Abu Kausar, VS Dhaka, and Sanjeev Kumar Singh. Web crawler: a review. *International Journal of Computer Applications*, 63(2), 2013.

[18] Emma Lavens, Pieter Philippaerts, and Wouter Joosen. A Quantitative Assessment of the Detection Performance of Web Vulnerability Scanners. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1--10, 2022.

[19] Manuel Leithner, Bernhard Garn, and Dimitris E Simos. HYDRA: Feedback-driven black-box exploitation of injection vulnerabilities. *Information and Software Technology*, 140:106703, 2021.

[20] Miao Liu, Boyu Zhang, Wenbin Chen, and Xunlai Zhang. A survey of exploitation and detection methods of XSS vulnerabilities. *IEEE access*, 7:182004--182016, 2019.

[21] Jalen Mack, Yen-Hung Frank Hu, and Mary Ann Hoppa. A study of existing cross-site scripting detection and prevention techniques using XAMPP and VirtualBox. *Virginia Journal of Science*, 70(3):1, 2019.

[22] Mahmoud Mohammadi, Bill Chu, Heather Richter Lipford, and Emerson Murphy-Hill. Automatic web security unit testing: XSS vulnerability detection. In *2016 IEEE/ACM 11th International Workshop in Automation of Software Test (AST)*, pages 78--84. IEEE, 2016.

[23] Tomasz Andrzej Nidecki. Mutation XSS in Google Search. https://www.acunetix.com/blog/web-security-zone/mutation-xss-in-google-search/, 2019.

[24] Yassine Sadqi and Yassine Maleh. A systematic review and taxonomy of web applications threats. *Information Security Journal: A Global Perspective*, 31(1):1--27, 2022.

[25] Mike Samuel, Prateek Saxena, and Dawn Song. Context-sensitive auto-sanitization in web templating languages using type qualifiers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 587--600, 2011.

[26] Kenji Suzuki. *Artificial neural networks: methodological advances and biomedical applications*. BoD--Books on Demand, 2011.

[27] Isam Kareem Thajeel Thajeel, Khairulmizam Samsudin, Shaiful Jahari Hashim, and Fazirulhisyam Hashim. Machine and Deep Learning-based XSS Detection Approaches: A Systematic Literature Review. *Journal of King Saud University-Computer and Information Sciences*, page 101628, 2023.

[28] Bin Wang, Lu Liu, Feng Li, Jianye Zhang, Tao Chen, and Zhenwan Zou. Research on web application security vulnerability scanning technology. In *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 1, pages 1524--1528. IEEE, 2019.

[29] Pei Wang, Julian Bangert, and Christoph Kern. If it's not secure, it should not compile: Preventing DOM-based XSS in large-scale web development with API hardening. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1360--1372. IEEE, 2021.

[30] Yulong Wang, Tong Sun, Shenghong Li, Xin Yuan, Wei Ni, Ekram Hossain, and H Vincent Poor. Adversarial attacks and defenses in machine learning-empowered communication systems and networks: A contemporary survey. *IEEE Communications Surveys & Tutorials*, 2023.

[31] M Arif Wani, Farooq Ahmad Bhat, Saduf Afzal, Asif Iqbal Khan, M Arif Wani, Farooq Ahmad Bhat, Saduf Afzal, and Asif Iqbal Khan. Basics of supervised deep learning. *Advances in Deep Learning*, pages 13--29, 2020.

[32] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.