KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGA

DAINIUS MOCKUS

TINKLALAPIO ATSAKO LAIKO PRIKLAUSOMYBĖS NUO XSS APSAUGOS TYRIMAS

Magistro baigiamasis darbas

Darbo vadovas
dr. J. Čeponis

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS IR INFORMACINIŲ TECHNOLOGIJŲ SAUGA

DAINIUS MOCKUS

TINKLALAPIO ATSAKO LAIKO PRIKLAUSOMYBĖS NUO XSS APSAUGOS TYRIMAS

Magistro baigiamasis darbas

Darbo vadovas
dr. J. Čeponis

Recenzentas
dr. D. Matulis

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013 - 05 - 22 d.
Kaunas

Autoriai, Dainius Mockus
(vardas, pavardė)

_____,

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis bakalauro (magistro) darbas (toliau vadinama – Kūrinys)_____
(kūrinio pavadinimas)
_„Tinklalapio atsako laiko priklausomybės nuo XSS apsaugos tyrimas“_____

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

Dainius Mockus (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)

SANTRAUKA

Išaugus internetinių sistemų naudojimo populiarumui, vis daugiau slaptų, asmeninių duomenų talpinama internete, atliekamos įvairios finansinės operacijos. Toks reiškinys tapo viena iš esminių priežasčių, lėmusių išaugusį nusikaltimų skaičių internetinėje erdvėje. Dažnai informacijos perėmimas ar kitos kenkėjiškos operacijos atliekamos įterpiant programinį kodą į vartotojo peržiūrimą tinklapį (XSS ataka). Šių atakų metu bandoma įterpti į svetainę kenkėjišką kodą, kurį naršyklė įvykdys vartotojui atsidarius svetainę. Kodo įterpimo atakos yra sunkiai sustabdomos, nes internete veikiančios programos tampa vis dinamiškesnės ir jos suteikia vartotojui vis daugiau laisvės įvairiausiems veiksmams. XSS atakoms dažniausiai yra naudojami JavaScript programavimo kalba parašyti kodai, tačiau gali būti panaudoti ir HTML, Flash ar bet kokio kito tipo kodai, kuriuos gali įvykdyti vartotojo naršyklė. Šio tipo atakos nukreiptos prieš vartotojus, o ne serverius. Užpuolikai dažniausiai renkasi patikimo turinio svetaines, kuriomis vartotojai pasitiki ir taip tampa lengviau pažeidžiami.

Siekiant apsaugoti nuo šio tipo atakų, realizuojami įvairūs apsaugos mechanizmai, tačiau jie sulėtina tinklalapio atsako laiką ir ne visi visuomet tinkamai saugo vartotojus. Norint tinkamai apsisaugoti nuo kodo įterpimo atakų, būtina apsauga tiek kliento, tiek ir serverio pusėje. Kadangi kliento pusėje veikiančios naršyklės turi programinio lygio ugniasienes ir kitokias saugumo priemones, šiame darbe daroma prielaida, kad kliento pusėje saugumo lygis yra pakankamas ir tyrimas buvo atliekamas serverio pusės apsaugai.

Darbe apžvelgtos ir ištirtos trimis skirtingomis technologijos sukurtų tinklalapių kodo filtravimo priemonės. Tinklalapiai sukurti pasirinkus PHP, ASP.NET ir Java internetinių sistemų kūrimo technologijas. Kiekvienos technologijos tinklalapiai ištirti taikant skirtingas HTML turinio filtravimo bibliotekas. Tiriama ne tik tinklalapio atsako laiko pokyčiai, bet ir saugumo lygis tinklalapiuose su kiekviena turinio filtravimo biblioteka. Remiantis tyrimo rezultatais padarytos išvados, kurios saugumo priemonės yra rekomenduojamos naudoti.

Sekančiame tyrimo etape buvo pabandyta optimizuoti geriausią PHP tinklalapių filtravimo biblioteką. Šiame etape buvo siekiama išsiaiškinti kaip ir kiek galima sumažinti atsako laiką naudojant šią saugumo priemonę. Tyrimo metu išsiaiškinta, kad atsisakant tam tikro funkcionalumo ir saugumo priemonę optimaliai sukonfigūravus galima sumažinti atsako laiką išlaikant aukštą saugumo lygį.

SUMMARY

With the increase of popularity in using online systems, more and more sensitive and personal data is stored on the Internet, and a variety of financial transactions are performed there. This phenomenon has become one of the essential reasons that have increased the number of crimes on the Internet. Often transferring information and performing some malicious operations include inserting program code to the website that the user is viewing (XSS attack). During these attacks, cyber criminals attempt to insert malicious code to the website, which the browser will perform when the user opens that website. Code insertion attacks are difficult to stop because the programs that work on the Internet are becoming more dynamic and allow the user to perform actions more freely. Usually, XSS attacks use codes created with JavaScript. However, cyber criminals can also use HTML, Flash or any other type of codes which user's browser can perform. This type of attacks is directed to the users and not servers. The attackers usually choose websites with reliable content which are trusted by users and this way these websites become more vulnerable.

There are various security mechanisms that are used in order to protect against this type of attacks. However, they slow down response time of a website and not all of them always protect users properly. A protection is necessary both for a client's side and a server's side in order protect against code insertion attacks. Since a client uses browsers that have firewalls and other security measures, this paper assumes that protection level of client's side is sufficient, so the research was carried for the protection of server's side.

The paper reviews and analyses code filtering measures of websites that have been created using three different technologies. The websites have been created using PHP, ASP.NET and Java technologies for developing web-based systems. Websites of each technology have been analysed applying different HTML content filtering libraries. Not only changes in response time of a websites have been analysed but also the security level on the websites with each content filtering library. According to the results of the research, the paper has concluded which security measures are recommended to use.

On the next stage of the research, the best PHP websites filtering library has been tried to be optimized. On this stage, it has been aimed to find out how and how much it is possible to reduce the response time using this security measure. The research has shown that it is possible to gain a sufficiently good response time that does not overload the system, by refusing certain functionalities and optimally configuring security measure.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

1. **XSS (angl.: *Cross-site scripting*)** – Informacinių technologijų sistemų pažeidžiamumas, dažniausiai aptinkamas tinklalapiuose, kurie leidžia įterpti papildomą programinį kodą į vartotojo peržiūrimą tinklalapį. Toks pažeidžiamumas atsiranda dėl nepakankamo įvedamos informacijos filtravimo.
2. **Tinklalapio atsako laikas** – tai laikas, kuris sugaištamas kai internetinė sistema (dažniausiai tinklalapis) išsiuntusi užklausą serveriui laukia atsakymo.
3. **Internetas (angl.: *internet*)** – tai yra pasaulinis kompiuterių tinklas, jungiantis visuotinius ir vietinius kompiuterių tinklus į vieną bendrą pasaulinį tinklą.
4. **SQL (angl.: *Structured Query Language*)** – struktūrizuota užklausų kalba. Tai yra populiariausia iš šiuo metu naudojamų kalbų, skirtų aprašyti duomenis ir manipuliuoti jais releacinių duomenų bazių valdymo sistemose.
5. **HTML (angl.: *Hyper text Markup Language*)** – hiperteksto žymėjimo kalba. Tai yra kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete.
6. **XML (angl.: *Extensible Markup Language*)** – bendrosios paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. Pagrindinė XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų internetu.
7. **CSS (angl.: *Cascading Style Sheets*)** – kalba, skirta nusakyti kita struktūrine kalba aprašyto dokumento vaizdavimą. Dažniausiai CSS aprašomas HTML dokumentų pateikimas, tačiau ją galima naudoti ir įvairiems kitiems XML dokumentams.
8. **JavaScript** – tai yra scenarijų programavimo kalba, besiremianti prototipų principu. Dažniausiai kalba naudojama internetinių puslapių interaktyvumo realizacijai, bet taip pat naudojama ir kaip galimybė scenarijais manipuliuoti tam tikromis programomis.
9. **Flash** - (anksčiau vadintas Shockwave Flash ir Macromedia Flash) yra komplektas daugialypės programinės įrangos, sukurtos „Macromedia“ kompanijos, o šiuo metu jis vystomas „Adobe“. Flash yra populiarus būdas pridėti animacijos ir interaktyvumo į interneto svetaines. Flash dažniausiai naudojamas sukurti animaciją, reklamas ir įvairius tinklalapio komponentus. Juo naudojantis galima integruoti vaizdo medžiagą į tinklalapius.
10. **Sesija** – kompiuterių kontekste sesija yra interaktyvūs informacijos mainai tarp dviejų ar daugiau tarpusavyje bendraujančių įrenginių.
11. **PHP (angl. *Hypertext Preprocessor*)** - plačiai paplitusi dinaminė interpretuojama programavimo kalba, sukurta 1995 m. ir specialiai pritaikyta interneto svetainių kūrimui.
12. **ASP.NET** – tai yra tinklalapio struktūros technologija kuriama Microsoft kompanijos, kurią programuotojai gali naudoti norėdami sukurti dinaminę internetinę svetainę, žiniatinklio konstrukciją arba paslaugą.
13. **Java** – tai yra objektiškai orientuota programavimo kalba, 1991 metais sukurta Džeimso Goslingo ir kitų „Sun Microsystems“ inžinierių.
14. **URL (angl. *Uniform resource locator*)** – tai yra universalus informacijos šaltinio adresas, kuris skirtas patogiai nurodyti tikslią informacijos vietą internete.
15. **Serveris** - specialios paskirties kompiuteris ar programa, skirta kitų kompiuterių (klientų) aptarnavimui (paprastai nėra skirta asmeniniam naudojimui).
16. **OWASP (angl. *Open Web Application Security Project*)** – tai yra nekomercinio tipo projektas orientuotas į taikomosios programinės įrangos saugumo gerinimą. Šiame projekte nagrinėjami saugumo pavojai ir analizuojama, bei pasiūlomi būdai kaip būtų galima įvertinti saugumo rizikas ir apsaugoti savo aplikacijas nuo kylančių pavojų.
17. **IE (angl. *Internet Explorer*)** – vienos iš populiariausių pasaulyje interneto naršyklių žymėjimas.
18. **jQuery** – tai yra JavaScript programavimo kalbos biblioteka palengvinanti JavaScript kodo rašymą.

19. **localhost** – vietinis kompiuteris. Tai dažniausiai naudojamas adresas pasiekti vietiniame kompiuteryje veikiančias svetaines.

20. **DOM (angl. *Document Object Model*)** – taip vadinamas dokumento objekto modelis.

21. **API (angl. *Application programming interface*)** – tai yra aplikacijų programavimo sąsaja. Sąsaja, kurią suteikia kompiuterinė Sistema, biblioteka ar programa tam, kad programuotojas per kitą programą galėtų pasiekti jos funkcionalumą ar apsikeistų su ja duomenimis.

ĮVADAS

Žmogaus privačiame gyvenime šiuolaikinės informacijos technologijos tapo svarbia ir neatsiejama jo gyvenimo dalimi. Todėl vis daugiau asmens ar ypač jautrių duomenų saugoma ir tvarkoma įvairių valstybinių ir verslo įmonių informacijos sistemose bei jais keičiamasi elektroninių ryšių tinklais. Dėl išaugusių internetinių sistemų naudojimo populiarumo vis daugiau slaptų, asmeninių duomenų talpinama internete. Toks reiškinys tapo viena iš esminių priežasčių, lėmusių išaugusį nusikaltimų skaičių internetinėje erdvėje. Augantis nusikaltimų skaičius reikalauja daugiau dėmesio skirti sistemų saugumo užtikrinimui. Bet kurioje kompiuterinėje sistemoje taikant įvairias saugumo priemones padidinti saugumą, tai reikalauja tam tikro kiekio resursų. Ta pati situacija ir su XSS tipo atakomis. Į sistemas sudiegus daug įvairių tikrintuvų, ugniasienių, filtrų ir kitų saugos priemonių, tinklalapio atsako laikas taptų labai ilgas ir tokia sistema taptų nepatogi naudoti. Todėl kiekvienos sistemos kūrėjai privalo išsamiai išanalizuoti galimas rizikas ir jas įvertinę pasirinkti geriausias apsaugos priemones ir jų naudojimo būdus. Šiame darbe tyrinėjamos populiariausių technologijų saugumo priemonės ir vertinamas jų poveikis tinklalapio atsako laikui. Be atsako laiko taip pat vertinamas ir saugumo užtikrinimo lygmuo. Šiuo tyrimu bandoma padėti programuotojams, dirbantiems su skirtingomis technologijomis, lengviau pasirinkti saugumo priemones jų kuriamoms sistemoms nuo programinio kodo įterpimo atakų.

Šio tyrimo objektas yra tinklalapio atsako laiko priklausomybė nuo XSS atakų apsaugos būdų.

Šio tyrimo tikslas yra ištestuoti ir patikrinti tinklalapio atsako laiko pokyčius taikant skirtingus apsaugos būdus ir jų kiekius, pasiūlyti geriausią tinklalapio apsaugos būdą nuo XSS atakų.

Tyrimo uždaviniai:

1. Išanalizuoti esamus tinklalapių apsaugos tipus ir metodus nuo XSS tipo atakų.
2. Sukurti tinklalapio atsako laiko testavimui naudojamus tinklalapius veikiančius Linux ir Windows serveriuose.
3. Ištestuoti nemokamus apsaugos metodus su sukurtais testavimo tinklalapiais skirtinguose platformose ir išmatuoti tinklalapių atsako laikus taikant skirtingus apsaugos metodus.
4. Ištirti kurie, kokiose platformose ir kaip taikomi apsaugos metodai mažiausiai didina tinklalapio atsako laiką.
5. Pasiūlyti tinklalapių apsaugos nuo XSS atakų variantus su mažiausiais atsako laikais.
6. Optimizuoti pasirinktą XSS apsaugos biblioteką, siekiant sumažinti atsako laiką.

Darbas sudarytas iš keturių pagrindinių skyrių: „Tinklalapiu apsaugos problemos“, „Atsako laiko įvertinimo ir jo minimizavimo tyrimo metodikos“, „Atsako laiko įvertinimo ir jo minimizavimo tyrimo“ bei rezultatų apibendrinimo. Pirmajame skyriuje pateikiamas išsami saugumo problemų analizė, apžvelgiamos saugumo problemos, pasirinktos temos aktualumas. Taip pat pirmajame skyriuje detaliai aprašomos XSS atakos, pateikiami jų pavyzdžiai ir nurodymai kaip galima apsisaugoti nuo tokio tipo įsilaužimų. Antrajame skyriuje aprašomas atliekamas tyrimas. Skyriuje išanalizuojamos pasirinktos technologijos kuriamiems testavimo tinklalapiams, jie apibūdinami, pasirenkamos saugumo priemonės ir jos detaliai aprašomos. Skyriuje „Atsako laiko įvertinimas ir jo minimizavimas“ aprašomas atliktas tyrimas, pateikiami rezultatai ir jie paaiškinami. Paskutiniame skyriuje pateikiami apibendrinimai ir gautų tyrimo rezultatų išvados.

1. TINKLALAPIŲ APSAUGOS PROBLEMOS

Internetas – tai pasaulinė kompiuterių tinklų sistema, jungianti dešimtis tūkstančių atskirų kompiuterių tinklų. Šiam tinklui negalioja valstybių sienos. Interneto pagrindą sudaro superkompiuteriai, tarpusavyje sujungti greitaeigėmis linijomis. Kiekvienas jų sujungtas su eile mažesnių tinklų, kurie jungiasi su dar mažesniais tinklais ir t.t. Visi mažesni tinklai tarpusavyje sujungti įvairiomis ryšio linijomis ir maršrutizatoriais [1].

Dabar, kai informacijos ir ryšių technologijos sudaro šiuolaikinės visuomenės ir ekonomikos pagrindą, tinklų ir informacijos saugumas tampa vis aktualesnis. Tinklų ir informacijos saugumą galima nagrinėti vertinant jų patikimumą ir funkcionalumą. Tinklų ir informacijos saugumui šiandien tenka daug iššūkių, o interneto saugumas tapo daugelio organizacijų ir įvairiausių sričių specialistų pagrindinė diskusijų tema. Anksčiau tinklų ir informacijos saugumas buvo išimtinai nacionaline veiklos sfera ir vyriausybių „galvos skausmas“. Šiuo metu tinklų ir informacijos saugumas tapo aštria Europos Sąjungoje vyraujančia politine veiklos kryptimi. Prieiga prie elektroninių tinklų dabar yra lengvai pasiekiama tiek iš namų, tiek ir viešojoje erdvėje, o taipogi vyrauja tendencija jungtis nešiojamais įrenginiais, skirtais duomenų įvedimui ar atvaizdavimui. Informacijai laisvai kertant valstybių sienas, plinta ir tinklų bei informacijos saugumo problemos.

1.1. Pavojai internete

Pagrindinis saugojimo objektas yra vertinga informacija. Pašaliniam asmeniui priėjus prie šių vertybių kyla didelis pavojus, kad su šia informacija bus atlikti nereglamentuoti veiksmai. Todėl pirmaeiliu uždaviniu visada išliks informacijos sistemų apsauga nuo nesankcionuoto jų panaudojimo, o žemesnį prioritetą turės pačios informacijos saugumo užtikrinimas šiose sistemose. Galimi pavojai turi būti vertinami pagal tai, kaip jie gali paveikti tris pagrindinius saugumo elementus - sistemos resursų vientisumą, konfidencialumą ir prieinamumą. Pavojai yra skirstomi į grėsmes bei pažeidžiamumus. Grėsmė yra suvokiama kaip galimybė pažeisti informacijos sistemos saugumą (galimybė atskleisti, pakeisti arba sunaikinti informaciją, trukdyti sistemos darbui) ir klasifikuojama pagal jos kilmę, motyvaciją, kelią, tikslą ir rezultatą [2], o jos kilmė gali būti:

- grėsmė iš vidaus (sistemos vartotojai, aptarnaujantis personalas, socialinė inžinerija).
- grėsmė iš išorės (kompiuteriniai nusikaltėliai, kiberteroristai).
- fizinė grėsmė (vagystė, gaisras ar kitoks žalingas aplinkos poveikis).

Informacinėse sistemose dažnai sutinkamos grėsmės yra:

- BotNets - prijungtų prie interneto kompiuterių kompromitavimas, realizuojamas per jų nuotolinę kontrolę arba sudarantis galimybę per juos vykdyti komandas inicijuojamas kitais kompiuteriais, puolant kitus pasirinktus taikinius.
- Programinio kodo įterpimo į vartotojo peržiūrimą tinklapį atakos - veiksmas kurio metu piktavalis vartotojas sugeba įterpti piktavališką kodą (angl. *malicious code*) per ryšio sesiją į tinklalapį arba aplikaciją [1, 3]. Kai yra spragtelėjama pele ant šios nuorodos, piktavališkas kodas priverčia dalį vartotojo svetainės užklausos tapti valdoma ir taip pažeisti ar kompromituoti kompiuterį ar duomenis. Plačiau šio tipo atakos aprašomos kitame skyriuje.
- Denial of service (DoS) - metodas kai užpuolikas neprileidžia prie sistemos prieigos teisėtų vartotojų, siekdamas kompromituoti taikiniu pasirinktą informacijos sistemą [1]. Ši ataka užvaldo tiktai vieną šaltinį t. y. vieną kompiuterį blokuojant jo pranešimus ir srautą kreipiamą į jį. Tuo pačiu tai gali būti naudojama ir apsaugai nuo informacijos apsikeitimo tarp sistemų ar jos apsaugai internete.

- Distributed Denial of service (DDoS) - toks pats metodas kaip ir Denial of service (DoS), tik veiksmas atliekamas koordinuotai daugeliu kompiuteriu siunčiant užklausas. Dažniausiai šiai realizacijai naudojamas kirminas (ang. *worm*) – savarankiškai veikianti piktavališka programa, kuri paskleidžiama daugelyje kompiuterių, o šie, vėliau gavę komandą, gali atakuoti taikinį.

- Piktnaudžiavimo įrankiai (angl. *exploit tools*) - viešai prieinami įrankiai, kurie leidžia įsiskverbti į įvairius sistemų lygius ir gali būti naudojami šių sistemų pažeidimui ar prieigos gavimui [1].

- Loginės bombos - sabotažo forma (atgalinė socialinė inžinerija), kai programuotojas įveda kodą, kuris sukelia neigiamą poveikį programai [1]. Tai gali sukelti tokias neigiamas pasekmes, kaip programos veikimo nutraukimą.

- Paketiniai šniukštinėtojai (angl. *sniffer*) – programa, kuri perima srauto duomenis ir juos analizuoja išskirstydama paketais, ieškant klasifikuotos informacijos, perduodamų slaptažodžių atvirame informacijos formate [1].

- SQL injekcijos - jei tinklalapio parametrai, kurie perduodami iš nepatikimų šaltinių (pavyzdžiui formos), dedami tiesiai į SQL duomenų bazę nepatikrinus SQL META simbolių, tai blogasis lankytojas gali „patobulinti“ jūsų SQL užklausą ir pakeisti duomenis arba perimti iš duomenų bazės slaptus duomenis [1, 3].

- Trojos arkliai - kompiuterinė programa, kuri atrodo naudinga, tačiau iš tiesų gali kenkti kompiuteriui. Trojos arkliai platinami, kai žmonės yra suviliojami atidaryti programą, nes jie tiki, kad ji buvo pateikta iš patikimo šaltinio, arba gali pasitaikyti programinėje įrangoje, kuri buvo atsiųsta nemokamai [1].

- Pasitikėjimo išnaudojimas - kompiuteriai ar informacijos sistemos, vykdydamos kai kurias programas su kitais kompiuteriais ar informacijos sistemomis, įgauna jų pasitikėjimą. Pavogus šį identitetą, įgyjama neautorizuotos prieigos galimybė prie pasitikėjimų deklaravusių sistemų.

- Virusai - infekuotos kompiuterinės bylos ar vykdomosios programos, bet su infekuotomis bylomis. Šias nukopijavus į atmintį yra užkrečiamos kitos bylos, bet tam yra būtinas žmogaus įsikišimas - įrašymo, kopijavimo, apsikeitimo komandų paleidimas.

- Prasiskverbimo kova – metodai, kuriais bandoma gauti prieigą prie bevielių tinklų, naudojantis nešiojamais kompiuteriais, specializuotomis antenomis ir bevielių tinklų adapteriais, ieškant neautorizuotos prieigos taško ar apeinant jo apsaugos reikalavimus.

- Kirminai (angl. *worms*) - nepriklausomos kompiuterinės programos, savarankiškai perduodančios duomenis per tinklus kompiuteriams ar informacijos sistemoms, atliekant duomenų apsikeitimą arba kopijuojant duomenis. Kitaip nei virusai, kirminai nereikalauja žmogaus įsitraukimo į platinimą.

- Zero-Day spragos - yra saugumo spragos, kurios išnaudoja kompiuterių taikomosios programinės įrangos pažeidžiamumus, vartotojui arba programinės įrangos kūrėjui to nežinant. Šio tipo atakomis galima, daugiau ar mažiau neatpažįstamai, įsilaužti į sistemas, bei manipuliuoti duomenimis arba juos pavogti.

Šių grėsmių sukeliami pažeidžiamumai vertinami kaip informacijos sistemų neatsparumas pažeidimams, nepakankamas saugumo priemonių taikymo lygis, neištaisytos saugumo klaidos ir kitos aplinkybės, kurios gali sąlygoti sistemų saugumo pažeidimą. Informacijos sistemų funkcionalumo ir žmogiškojo faktoriaus įtakos dėka pažeidžiamumai kyla iš suinteresuotų šaltinių:

- savamokslių kompiuterinių nusikaltėlių (programišių).
- organizuoto nusikalstamumo grupių.
- profesionalų, nesusijusių su valstybės institucijomis – kiberteroristų.
- politinių aktyvistų.
- konkuruojančių korporacijų ir valstybinių institucijų siekiančių kompetencinio pranašumo (žvalgybos institucijos).
- piktų ir nelojalių darbuotojų ir konsultantų.
- programinės ir aparatinės įrangos gamintojų, siekiančių finansinės naudos, ir neetiškų reklamos platintojų.

Programinės įrangos kūrėjai, paprastai ir po programos versijos išleidimo palikdami saugumo spragas savo produktuose, siūlo gerą finansinį atlygį už tokių klaidų aptikimą ir neviešinimą, kol yra užtaisomos šios spragos. O namų ūkių vartotojams į rinką teikiami produktai, kurie, kooperuojantis techninės ir programinės įrangos gamintojams, turi įdiegtas bent minimalias saugumo funkcijas. Bet informacijai įgyjant vertę, suteikiančią konkurencinį technologinį pranašumą, ypatingai agresyvios šioje srityje tampa tiek valstybių, kuriose išvystytas informacijos technologijų naudojimas, žvalgybos institucijos, tiek organizuoto nusikalstamumo ir internetinio terorizmo atstovų grupės. Kiekvienu konkrečios informacijos sistemos atveju galimi pavojai turėtų būti kuo tiksliau įvardinti, nes tai lemia efektyvią apsaugos priemonių jai parinkimą.

Numatant galimus pavojus reikia nepamiršti, kad pagrindinis pavojų šaltinis dažniausiai yra žmogus, vedamas skirtingos motyvacijos ir šios veiklos priežastingumo:

- Šnipinėjimas. Asmuo gali būti specialiai tam apmokytas, pasamdytas, nupirktas, priverstas pavogti vertingą informaciją.
- Įsilaužimas, atliekamas sąmoningai. Sąmoningas asmens bandymas užvaldyti sistemos resursus, prie kurių jam nėra suteiktos teisės prieigai, bet siekiant sužinoti daugiau nei jam leista, taip pat įvairiai motyvuojamas veiksmas: iš nuobodulio, norint išbandyti savo sugebėjimus, taikant įgytas teorines žinias praktiniame lygmenyje ir panašiai.
- Kerštas. Nepatenkintas, pažemintas pareigose ar kitaip įžeistas darbuotojas ar atleistas iš darbo asmuo keršto tikslais bando sugadinti ar sunaikinti duomenis, palikti logines bombas, sabotuoti informacijos sistemų funkcionalumą.
- Pavojinga nesąmoninga veikla. Tai gali būti bet kuris informacijos sistemos vartotojas, kuris dėl gebėjimų stokos gali pakenkti šiai sistemai (kompetentingi saugumo ekspertai nešiojamus FLASH atminties įrenginius priskiria prie pagrindinių informacijos sistemos saugumo pažeidimo įrankių) sustabdyti jos darbą, apkrėsti virusais ir kitais programiniais produktais, sunaikinti duomenis. Taip pat dėl sistemos konfigūravimo klaidų ar paliktų tinklo prieigos apsaugos silpnų vietų (intensyviau naudojant bevielę prieigą, prie sistemų atsiranda ir naujos jos apsaugos problemos) sistemos vartotojas atsitiktinai gali pažeisti sistemos saugumą.

Patikimiausiai sistemoje esančius pavojus nustatyti ir įvertinti gali tik saugumo specialistai, kurie nuolatos dirba šioje srityje, domisi naujovėmis ir turi pakankamai kompetencijos saugumo rizikų įvertinimui.

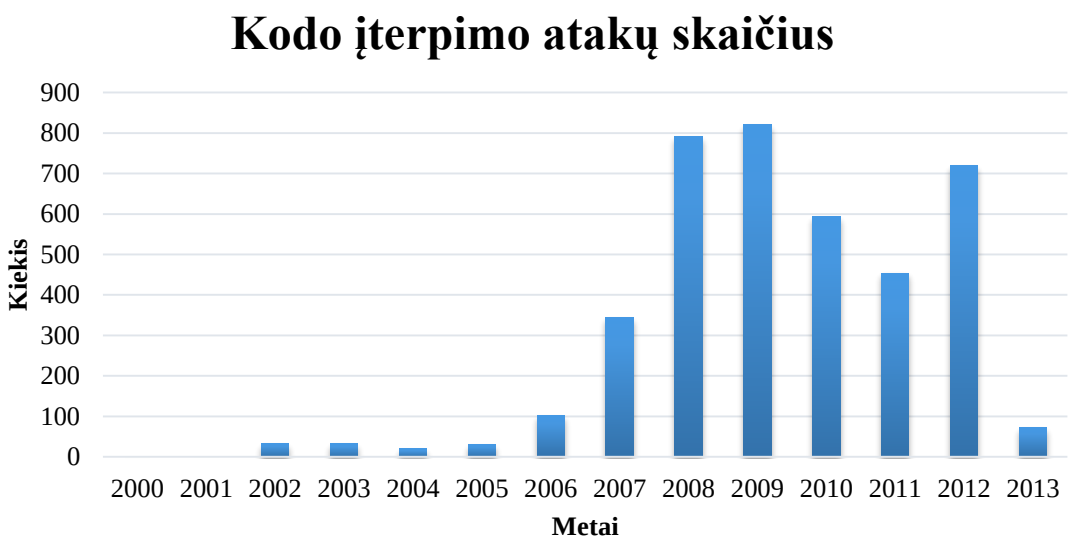
1.2. Programinio kodo įterpimas į vartotojo peržiūrimą tinklalapį (XSS)

Internete veikiančios taikomosios programos tampa vis populiaresnės ir į jas keliama vis daugiau svarbios informacijos, o jos saugumo užtikrinimas taip pat sudėtingėja. Dažniausios atakos nukreiptos prieš internete veikiančias programas yra programinio kodo įterpimo arba XSS tipo atakos, tai yra atakos, kurių metu yra bandoma įterpti į svetainę kenkėjišką kodą [1], kurį naršyklė įvykdys vartotojui atsidarius svetainę. Kartais šio tipo atakos dar žymimos sutrumpinimu CSS, tačiau, dėl kito kompiuterinio termino CSS žymėjimo tokiais pačiais ženklais, geriau naudoti XSS žymėjimą. Kodo įterpimo atakos yra sunkiai sustabdomos, nes internete veikiančios programos tampa vis dinamiškesnės ir jos suteikia vartotojui vis daugiau laisvės įvairiausiems veiksmams. Kodo įterpimo atakos į tinklalapį tampa įmanomomis tuomet kai:

1. Duomenys į sistemą yra įvedami nepatikimų šaltinių (neautorizuotų vartotojų), dažniausiai tai daroma sistemos reikalavimu.
2. Duomenys yra dinamiškai įvedinėjami vartotojų ir vėliau puslapio turinys rodomas kitiems vartotojams, neatliekant tinkamo įvedamų duomenų skenavimo. Tai gali būti komentarai, forumai, diskusijos ir pan.

XSS atakoms dažniausiai yra naudojami JavaScript programavimo kalbomis parašyti kodai, tačiau gali būti panaudoti ir HTML, Flash ar bet kokio kito tipo kodai, kuriuos gali įvykdyti vartotojo naršyklė [4]. Šio tipo atakos nukreiptos prieš vartotojus, o ne serverius. Užpuolikai dažniausiai renkasi patikimo turinio svetaines, kuriomis vartotojai pasitiki ir taip tampa lengviau pažeidžiami [1].

Pirmą kartą šio tipo atakos buvo pastebėtos 2000 metais ir nuo to laiko jos smarkiai išpopuliarėjo, tai aiškiai matoma pirmajame paveiksle (žr. 1 pav.).



1 pav. XSS atakų skaičius 2000-2013 m. kovas [5]

Vis populiarėjant internetui ir vis daugiau informacijos perkeliant į sistemas internete, išpopuliarėjo ir XSS atakos [1, 4]. Pagrindinė to priežastis buvo ta, kad pirmosios šio tipo atakos nebuvo labai žalingos ir niekas nemanė, kad į šiuo pažeidžiamumus reikia atkreipti dėmesį. Atakoms vis įžūlėjant ir pridarant vis daugiau nuostolių, buvo pradėtos kurti apsaugos priemonės, kurios padėjo sulėtinti atakų populiarumo didėjimą, tačiau 2012 metais atakų populiarumas vėl padidėjo. Šis užfiksuotų atakų statistiko padidėjimas leidžia manyti, kad atakų saugos priemonės vis dar nėra tokios veiksmingos arba žmonės vis dar vangiai saugosi nuo šio tipų pavojų.

Nors apie šio tipo atakas jau kalbama seniai ir yra įvairiausių pasiūlytų būdų, kaip apsisaugoti nuo jų, tačiau šio tipo atakų skaičiai ir padarytos žalos dydis yra stulbinantis. Atakos gali leisti apeiti saugumo mechanizmus, kurie integruoti į visas šiuolaikines naršykles, gali išgauti svarbių duomenų

ir juos panaudoti atliekant įvairius veiksmus kito asmens vardu, gali stebėti ir siųsti įsilaužėliui vartotojo atliekamus veiksmus, spaudžiamus klavišus ir t.t.

Apibendrinant, visi programinio kodo įterpimo į vartotojo peržiūrimus tinklalapius būdai yra skirstomi į tris grupes: nuolatinius, vienkartinius ir paremtus dokumento objektų modeliu (DOM) [6]. Toliau išsamiai apžvelgsime atskirus kodo įterpimo į tinklalapius atakų būdus.

1.2.1. Nuolatiniai XSS

Nuolatinis kodo įterpimas - tai yra atakos tipas, kai kodas yra įterpiamas į puslapį, kuriame jis yra išsaugomas ir nuolat vykdomas, puslapio vartotojams peržiūrint tinklalapį. Šio tipo atakos įmanomos tik tuomet, jeigu puslapyje, prieš saugant vartotojo įvestus pranešimus ar informaciją, nėra tinkamai patikrinami, įvedami duomenys [1, 7, 8]. Nuolatiniai arba išsaugoti XSS pažeidžiamumai leidžia atlikti pačias veiksmingiausias atakas, nes atakų skaičius priklausys nuo puslapio peržiūrėjimų skaičiaus, o jeigu tai yra labai populiarus tinklalapis, tai atakos gali pasiekti ir tūkstančius vartotojų. Tai yra vienas iš paprasčiausiai realizuojamų XSS atakos tipų. Nuolatinių XSS atakų realizacijos būdų yra sukurta labai daug ir įvairių, keletą iš jų ir pabandysime apžvelgti:

1. Paprastas programinio kodo įterpimas į vartotojo peržiūrimą tinklalapį (XSS) – tai yra paprasčiausia XSS ataka. Kaip jau buvo paminėta anksčiau, gali būti vykdoma tuo atveju, kai pagal vartotojo įvestus duomenis formuojamas puslapio turinys (pvz.: rašant komentarus, pasisakymus, netikrinami duomenys vartotojo varduose, pavardėse, atliekant paiešką ir pan.). Pavyzdžiui, atliekant įrašą tinklalapio svečių knygoje įvedamas tekstas:

```
<script>alert('Tu esi uzkrėstas!')</script>
```

Analogiškas tekstas bus rodomas formuojant HTML turinį, tuo pačiu įvykdant ir kodą. Analogiška situacija yra vykdant paiešką ir, pavyzdžiui, ieškant teksto su HTML žymomis. Jei paieškos rezultate parodoma ieškota frazė su visu HTML kodu, tokį puslapį galima panaudoti ir perimant vartotojų slapukus ar sesijas. Taip pat leidžiant įvesti kai kuriuos HTML elementus (pvz.: IMG, A, B, I...) galima pridėti atributą (pvz.: onmouseover) ir įvykdyti Javascript kodą.

2. UTF-7 XSS – ataka, vykdoma pasinaudojant naršyklės savybe "nuspėti" naudojamą simbolių rinkinį. UTF-7 leidžia bet kokį simbolį išreikšti ASCII simboliais, o naršyklė, bandydama nuspėti, kokia koduotė naudojama, UTF-7 užkoduotus simbolius gali paversti į reprezentuojamus simbolius, kurie leidžia įvykdyti ataką. Pavyzdžiui, UTF-7 koduotėje simbolis < žymimas +ADw, o simbolis > +AD4. Tokiu atveju formuojama antraštė (TITLE) iš simbolių:

```
+ADw-script+AD4-alert(document.location)+ADw-/script+AD4-
```

Naršyklei interpretuojant šiuos simbolius bus gaunamas toks kodas:

```
<script>alert(document.location)</script>
```

Ir tai taip pat suteikia galimybę įvykdyti JavaScript kodą.

3. Sausainiukų/sesijos perėmimas - šios rūšies atakos esmė - naudojant išorinį elementą gauti vartotojo slapuką, kuriame įrašytas sesijos ID. Pavyzdžiui, įterpus elementą:

```
<div onmouseover="document.getElementById(' div_id ').innerHTML='<img
src=http://puslapis.com/saugoti-cookie.php?'+document.cookie+' width=0
height=0/>'" style="height: 100px"></div>
<div id="div_id" ></div>
```

Užvedus pelyte ant pirmojo bloko, bus įvykdomas Javascript kodas, kuris užkraus kenkėjišką kodą ir perduos jam visus slapukų duomenis (pastarieji gali būti išsiunčiami el. paštu ar kitaip perduodami), kartu ir sesijos ID. Piktavaliui užtenka pačiam įdėti slapuką su sesijos duomenimis ir jis jau bus prisijungęs prie sistemos kaip kitas vartotojas. Sesijos ID gali būti perimamas ir tuo atveju, jei vartotojo naršyklė nepalaiko slapukų [9]. Tokiu atveju jeigu bus naudojamas PHP, jis automatiškai prideda sesijos ID prie kiekvienos nuorodos ir kaip HIDDEN lauką formose.

4. CSRF (Užklausų į svetaines klastojimas) - tarkime, žinoma, jog vartotojas yra prisijungęs prie kažkokio interneto tinklalapio. Taipogi žinoma, kokia nuoroda kokį veiksmą atlieka. Pagal turimas žinias suformuojama nuoroda, kuri atlieka atitinkamą veiksmą ir nematomai patalpinama kažkokiame tinklalapyje. Pavyzdžiui:

```
<img src="http://www.e-
parduotuve.com/pervesti?saskaita=LT300025644552112 21&amp;suma=1000" />
```

Tuomet vartotojui pasiūloma apsilankyti kenkėjišką kodą turinčiame tinklalapyje. Ir vartotojui nepastebint, įvykdomas atitinkamas veiksmas. POST metodas negelbsti, kadangi kenkėjiškame puslapyje gali būti ir automatiškai įvykdoma forma, ir IFRAME elementas. Tikrinti HTTP_REFERER taipogi neefektyvu, kadangi šiuos duomenis galima suklastoti.

5. XSS: katalogo pakeitimas/kenksmingo failo įterpimas - tokią ataką galima įvykdyti, kai sisteminiuose failuose atidaromi ar iškviečiami failai, kurių pavadinimas generuojamas iš vartotojo įvedamų duomenų. Pavyzdžiui:

URL: `http://www.pavyzdys.com/modulis=straipsnis`
PHP skriptas: `include($_GET['modulis'].'.php');`

Tokiu atveju URL pakeitus į http://www.pavyzdys.com/modulis=http://www.kenkejas.com/blogas-kodas bus įvykdomas skriptas iš http://www. kenkejas.com/blogas-kodas.php. Analogiškai galima iškviesti bet kokią programą ar įtraukti bet kokį failą, esantį tame pačiame serveryje (pavyzdžiui, iškviečiant /etc/passwd). Taip pat galima užkrėsti ir vartotojo slapukus, todėl negalima aklai pasitikėti ir juose saugomų duomenų tikrumu.

6. Kenkėjiškų failų įkėlimas į tinklalapį – tai atliekama pasinaudojus failo įkėlimo forma. Įkeliamas programinis kodas, kuris įvykdomas per naršyklę. Toks kodas sistemoje gali atlikti bet kokį darbą - nuo failų bei duomenų bazės informacijos nuskaitymo, įrašymo, modifikavimo iki visiško sunaikinimo.
7. HTTP nukreipimo išnaudojimas - siunčiant HTTP antraštę "Location", naršyklė atveria nurodytą tinklalapį, vartotojui nerodydama HTML kodo, net jei jis perduodamas. Visgi, jei HTML kodas yra perduodamas, jį galima gauti naudojant CURL biblioteką, taip gaunant papildomos informacijos (paprastam vartotojui nematomas nuorodas, tekstą ir pan.), kas gali palengvinti įsilaužimą į sistemą.
8. SQL injekcijos - formuojant SQL užklausą dažnai tenka į ją įtraukti duomenis, kuriuos perduoda vartotojas. Pavyzdžiui, kviečiant užklausą `http://www.pavyzdys.com/gautiVartotojoDuomenis.php?id=59` turi būti formuojama tokia SQL užklausa: `SELECT * FROM users WHERE id=59`. Tiesiogiai paimant duomenis su PHP, eilutė atrodytų taip:

```
$result = mysql_query('SELECT * FROM users WHERE id='.$_GET['id']);
```

Vietoje parametro "id", perduodant kenksmingą kodą, galima suformuoti bet kokią užklausą, pvz.:

```
http:// www.pavyzdys.com/ gautiVartotojoDuomenis.php?user=59 OR 1=1
```

tokiu atveju bus sugeneruota užklausa, kuri gražins visus įrašus iš duomenų lentelės. Panaudojant SQL injekcijos atakas galima ne tik gauti, bet ir modifikuoti duomenis, bei nuskaityti duomenis, esančius failuose.

1.2.2. Vienkartiniai XSS

Vienkartinės XSS atakos būdingos dinamiškai generuojamiems tinklalapių puslapiams. Ši ataka dažniausiai atliekama vartotojui pateikiant įvedimo formą, į kurią suvesti duomenys panaudojami visai kitais tikslais nei numatyta vartotojo [8]. Tai gali būti įvairiausi iššokantys langai, suklastoti puslapiai, kuriuose vartotojo prašoma ar net bandoma jį įtikinti, kad žmogui toje formoje reikia suvesti savo duomenis (dažnai tai būna slaptažodžiai ar kita slapta informacija). Pasitelkiant socialinę inžineriją ir tinkamai išnaudojant žmogiškąjį faktorių, galima priversti auką paspausti ant norimos nuorodos.

Vienas iš paprasčiausių šios atakos realizavimo scenarijų:

1. Žmogus lankosi kažkokiame tinklalapyje (pvz.: http://www.puslapis.lt/) ir jame yra reikalaujama prisijungimo duomenų;
2. Prisijungus tinklalapio turinys su tam vartotojui skirta informacija atvaizduojamas naršyklėje;
3. Įsilaužėlis, apsimetęs to puslapio administracijos vardu, nusiunčia suklastotą nuorodą vartotojui ir vienokiais ar kitokiais būdais bando įtikinti vartotoją atidaryti tą įsibrovėlio atsiustą nuorodą. Tokios nuorodos dažniausiai siuntinėjamus naudojantis brukalais (angl. *spam*);
4. Auka įtikinta, kad jai reikia atidaryti įsilaužėlio pateiktą nuorodą, apsilanko tinklalapyje, kai tuo metu yra prisijungusi savo vardu;
5. Tuomet suveikia suklastotojo nuorodoje įterptas skriptas, kuris iš sistemos išgauna prisijungusio vartotojo informaciją ir nusiunčia ją įsilaužėliui.

Yra ir daug kitokių įvairiausių galimų scenarijų šioms atakoms realizuoti, tačiau kadangi šios atakos yra nukreiptos prieš vartotojus, kurie yra patiklūs, jų žala dažniausiai nėra tokia didelė kaip nuolatinių arba DOM XSS atakų.

1.2.3. DOM XSS

DOM XSS pažeidžiamumas paremtas dokumento objektų modeliu. Šio tipo ataka paprastai naudojasi JavaScript [10] ar kita programavimo kalba ir keičia tinklalapio puslapio atvaizdą naršyklėje. Šis pažeidžiamumas įvyksta vartotojo kompiuteryje, o ne serveryje. Kadangi interneto naršyklės operacinėje sistemoje turi nemažai teisių, o su tokios rūšies ataka įmanoma paleisti visus procesus su tomis pačiomis teisėmis kaip ir naršyklės proceso teisės, įvykdytos atakos gali padaryti daug žalos [11]. Pasekmės bus dar blogesnės, jeigu vartotojas naudojasi operacine sistema prisijungęs kaip sistemos administratorius.

DOM XSS atakos realizacija gali būti labai panaši kaip ir vienkartinio XSS, bet jos pasekmės gali būti daug sudėtingesnės:

1. Įsilaužėlis pateikia nuorodą, kuri yra nukreipta į atakuotojo sukurtą puslapį;
2. Vartotojas atidaro užkrėstą tinklalapį;
3. Tinklalapis parodomas vartotojui ir įvykdomi kenkėjiški skriptai;
4. Kenkėjiški skriptai paleidžia vartotojo kompiuteryje įsilaužėlio suformuotas komandas.

Šio tipo atakas atpažinti ir jos išvengti paprastam vartotojui gali būti labai sudėtinga, nes piktavaliai gali sukurti identiškas sistemas, kurios atrodo lygiai taip pat kaip tikroji sistema, tačiau nežymiai skiriasi URL adresas į ką vartotojas gali ir neatkreipti dėmesio.

1.3. Bendrieji internetinių sistemų saugos reikalavimai

Internetinių sistemų visiško saugumo užtikrinimas yra beveik neįmanomas. Rašoma daugybė straipsnių, mokslinių darbų, o taip pat atliekami tyrimai, kurių tikslai yra išsiaiškinti kaip apsaugoti internetines sistemas, tačiau vienų bendrų rekomendacijų visoms sistemoms, kurias būtų galima taikyti praktikoje nėra. Taip yra dėl galimų saugumo spragų įvairovės. Saugumo spragų gali turėti tiek pati sistema, tiek ir serveryje veikianti kita programinė įranga, kuri įsilaužėliui suteikia galimybę pasiekti internetinės sistemos duomenis. Prieš tai pateiktuose skyriuose daugiausia susikoncentruota ties pažeidžiamumais, kurie kyla dėl programuotojų kaltės, kai jie neužtikrina patikimos internetinės sistemos saugos. Viena iš dažniausiai pasitaikančių saugumo spragų yra dinaminio turinio filtravimas. Dauguma šiuolaikinių internetinių svetainių yra dinamiškos, taigi vartotojui naršymas tampa ženkliai malonesnis. Dinamiškumas įgyvendinamas serverio procesais kurie parsiųsti į vartotojo kompiuterį yra vykdomi ir priklauso nuo vartotojo poreikių ir nustatymų. Visi sistemoje vartotojams suteikiami dinamiškumai yra ir potencialios sistemos saugumo spragos. Dinamiškam funkcionalumui ypač pavojingos XSS atakos.

2. APSAUGA NUO KODO ĮTERPIMO (XSS) ATAKŲ

Beveik kiekvienas dabartinis tinklalapis turi dinamiško turinio formavimo dalis [10], kuriuose rašomi komentarai, bendraujama su kitais vartotojais arba tiesiog galima atlikti paiešką tame puslapyje ir po paieškos yra parodoma paieškos frazė. Visos šios funkcijos piktavaliams vartotojams suteikia galimybę bandyti įvykdyti kodo įterpimo atakas. Tai reiškia, kad kiekviename puslapyje programuotojai turi imtis apsaugos priemonių, kuriomis tinklalapio vartotojai būtų apsaugomi nuo XSS atakų [12]. Apsaugos būdų nuo kodo įterpimo atakų yra trijų rūšių ir jie skirstomi pagal tai, kurioje vietoje yra realizuojami: serveryje, vartotojo kompiuteryje (naršyklėje) ir hibridinė apsauga [4].

1. **Serveryje vykdoma apsauga.** Vykdant apsaugą nuo kodo įterpimo atakų, serverio pusėje dažniausiai yra diegiama programinio lygio ugniasienė, kuri atliktų visus saugumo patikrinimus, perduodant informaciją tarp vartotojo kompiuterio ir serverio. Programinio lygio ugniasienės gali tikrinti užklausas einančias iš vartotojo kompiuterio į serverį, tikrinti serverio atsakus į vartotojo užklausas arba gali atlikti šiuos abu veiksmus vienu metu. Efektyviausią apsaugą užtikrina visos, tiek įeinančios, tiek išeinančios informacijos srauto tikrinimas, tačiau tai gali labai apkrauti serverį ir sulėtinti tinklalapio atsako laiką.
2. **Vartotojo kompiuteryje/naršyklėje realizuota apsauga.** Apsauga nuo kodo įterpimo atakų gali būti vykdoma ir vartotojo kompiuteryje. Tokiai apsaugai realizuoti reikalingi vartotojo kompiuteryje įdiegti tam tikri įrankiai. Tai gali būti vartotojo naršyklėje įdiegtas JavaScript kodo tikrinimo komponentas ir įsibrovimo aptikimo programinė įranga. Taip pat kaip ir serveryje, vartotojo kompiuteryje gali būti įdiegiama ir programų lygio ugniasienė, tačiau tokių priemonių įdiegimas į kompiuterį reikalauja, kad vartotojas atliktų apsaugos priemonių įdiegimą ir konfigūravimą sistemoje.
3. **Hibridinė apsauga.** Hibridinė XSS apsauga – tai tokia apsauga, kai serveryje yra įdiegiama tam tikra programinė įranga ir taip pat yra papildoma ar modifikuojama vartotojo kompiuteryje veikianti naršyklės apsauga. Tokiu atveju dažniausiai serveryje veikianti programinė įranga atsakinga už turinio pradinį patikrinimą ir saugumo taisyklių nustatymą, o naršyklė privalo reaguoti į nusakytas taisykles ir atlikti galutinį turinio patikrinimą.

2.1. Bendrieji tinklalapių apsaugos reikalavimai

Nepriklausomai nuo to, kur bus realizuota apsauga nuo kodo įterpimo atakų, tinklalapis turi būti apsaugotas pagal kelis esminius kriterijus. Pirmiausia, norint puslapį apsaugoti, visuomet reikalingas išsamus įvestų duomenų tikrinimas. Prieš bet kur ir bet kada naudojant vartotojo įvestus duomenis reikia juos patikrinti ar jie atitinka nustatytus duomenų tipus ir struktūras. Šis veiksmas turi būti atliekamas tiek vartotojo naršyklėje, prieš perduodant duomenis serveriui, tiek serverio pusėje, prieš saugant duomenis į duomenų bazę ar kitaip juos panaudojant. Patikrinant įvestas reikšmes, turi būti įsitikinama, ar įvestos reikšmės telpa į galimas reikšmių kitimo ribas, taip pat reikia išvalyti įvestą informaciją nuo galimų kodo įterpimo variantų ir užtikrinti, kad vartotojo įvedami duomenys atitinka naudojamą saugumo politiką. Vartotojo įkeliamus failus būtina tikrinti pagal nustatytus dydžių ir formatų apribojimus. Norint užtikrinti, kad vartotojas negalėtų įterpti ir įvykdyti savo kodą į tinklalapį, yra būtina filtruoti neleistinus simbolius, kurie gali pakeisti tinklalapio HTML kodą [2]. Visi šie kritiniai simboliai pateikiami pirmoje lentelėje (žr. 1 lentelė).

1 lentelė. Simboliai, kurie dažniausiai naudojami XSS atakose

Simbolis	Unikodas
"	U+0022 (34)
&	U+0026 (38)
’	U+0027 (39)
<	U+003C (60)
>	U+003E (62)

Lentelėje pateikiami tik tie simboliai, kurie tiesiogiai atvaizduojami gali sugadinti arba išformatuoti HTML kodą. Tačiau yra ir daug kitokių simbolių, kurie gaunami keičiant simbolių koduotes, kurias atvaizduojant naršyklė automatiškai pakeičia į atvaizduojamojo tinklalapio koduotę ir taip vis tiek įterpia į HTML kodą neleistinus simbolius, kurie sugadina HTML kodą arba įterpia kenkėjišką turinį.

2.2. OWASP rekomendacijos apsaugai nuo kodo įterpimo atakų

OWASP nekomercinio tipo projektas orientuotas į taikomosios programinės įrangos saugumo gerinimą. Šiame projekte nagrinėjami saugumo pavojai ir analizuojama, bei pasiūlomi būdai kaip būtų galima įvertinti saugumo rizikas ir apsaugoti savo aplikacijas nuo kylančių pavojų. „XSS“ pažeidžiamumams šio projekto autoriai skiria pirmą vietą tarp rimčiausių ir dažniausiai pasitaikančių pažeidžiamumų tinklo aplikacijose [4].

Apsaugai serverio pusėje OWASP projekto kūrėjai siūlo naudoti priklausomai nuo platformos vieną iš įrankių „ESAPI“ arba „Microsoft Anti-Cross Site Scripting Library“.

- „ESAPI“ – tai yra nemokama, atviro kodo internetinių aplikacijų saugumo kontrolės biblioteka, kuri palengvina programuotojams saugesnės programos rašymą. Jis yra kuriamas ir vystomas OWASP projekto kūrėjų. Šią biblioteką savo internetinėse aplikacijose naudoja ne viena žinoma kompanija: American Express, Apache Foundation, Booz Allen Hamilton, Aspect Security, Foundstone(McAfee) ir t.t. Ši biblioteka yra gerai dokumentuota ir yra išleista visoms populiariausioms platformos Java, PHP, ASP ir pan. [13].
- „Microsoft Anti-Cross Site Scripting Library“ - yra kompanijos Microsoft sukurta kodavimo biblioteka, skirta padėti programuotojams apsaugoti nuo XSS atakų savo ASP.NET platformoje parašytas internete veikiančias taikomąsias programas [14].

OWASP projekto kūrėjai, norėdami visiškai apsisaugoti nuo kodo įterpimo atakų, išskiria aštuonias esmines taisykles ir pataria kiekvienu atveju rinktis ir taikyti nebūtinai visas pateiktas taisykles, tačiau rinktis geriausius ir tinkamiausius saugumo sprendimus kiekvienam atveju. Toliau pateikiamos visos šiame projekte siūlomos saugumo taisyklės [4]:

1. Nepatikimi duomenys negali būti talpinami puslapyje, išskyrus tam tikrose vietose.
   Pirmoji taisyklė draudžia talpinti duomenis visame HTML dokumente, nebent jie tenkina antrą ir šeštą taisyklę, o tai reiškia yra tinkamai patikrinti.

```
<script>...NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ...</script>          skriptuose
<!--... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ...-->                  HTML komentaruose
<div ... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ...=test />            atributo pavadinime
< NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ... href="/test" />           žymės pavadinime
<style>... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ...</style>          CSS failuose
```

Labiausiai OWSAP akcentuoja JavaScript kodo įterpimo draudimą, nes tai yra pati pavojingiausia vieta ir, kūrėjų nuomone, niekada jokie JavaScript kodo fragmentai negali būti įterpti ir vykdomi iš nepatikimų šaltinių.

2. Prieš talpinant duomenis i HTML elementą, jie privalo būti patikrinti ir suformatuoti. Antroji taisyklė taikoma tuomet, kai norima įterpti nepatikimus duomenis tiesiai į kažkurį HTML elementą, tai gali būti , <p>, <b>, <td> ir kiti HTML elementai. Taikyti vien šią taisyklę apsaugai nuo kodo įterpimo atakos nepakanka ir papildomai reikia taikyti kitas taisykles.

```
<body>... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS...</body>
 <div>... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS...</div>
```

….. kiti HTML elementai

Duomenys tikrinami dėl to, kad įterpiant tekstus nebūtų keičiama tinklalapių HTML elementų struktūra ir taip sumažinamas sistemos saugumas. Į HTML kodą įterpiamus elementus reikia patikrinti ir neleisti įterpti penkių pagrindinių simbolių, kurie yra naudojami XML kalboje (&, <, >, “, ‘) , taip pat draudžiamas simbolis yra pasvirasis brūkšnys (/), nes jis padeda uždaryti HTML elementą. Tekste, kurį yra bandoma įterpti į tinklalapį, atradus šiuos simbolius, juos reikia arba pašalinti iš teksto, arba pakeisti į šešioliktainius simbolius:

& → &
< → <
> → >
" → "
' → ' ' naudoti nerekomenduojama
/ → / pasvirusus brūkšnys leidžia uždaryti HTML elementus

3. HTML atributų reikšmės privalo būti patikrintos ir tinkamai suformatuotos. Ši taisyklė apibrėžia duomenų įterpimą į tipinių atributų reikšmes, tokias kaip: „width“, „name“, „value“ ir pan. Ji nėra taikoma sudėtingiems atributams tokiems kaip: „href“, „src“, „style“ ir visiems įvykių atributams, tokiems kaip, pavyzdžiui, „onmouseover“. Labai rekomenduojama įvykių atributams naudoti ketvirtąją taisyklę.

```
<div attr=... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS...>turinys</div>
```

atributo reikšmė įterpiama be kabučių

```
<div attr='... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS...'> turinys </div>
```

atributo reikšmė tarp apostrofų ženklų

```
<div attr="... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS..."> turinys
</div>
```

atributo reikšmė tarp dvigubų kabučių ženklų

Išskyrus raides ir skaičius neleisti jokių kitokių simbolių įterpti į atributų reikšmes, kurios aprašomos be kabučių. Į tokius atributus įterpiant nepatikimus duomenis, HTML kodą gali sugadinti simboliai: [tarpas] % * + - / ; < = > ^ ir |. Atributus tarp kabučių gali sugadinti tik tokios pačios kabutės.

4. JavaScript kintamųjų ar naudojamų atributų reikšmės privalo būti patikrintos ir tinkamai suformatuotos. Taisyklė pabrėžia tai, kad labai svarbu atkreipti dėmesį į tai, kad reikia labai kruopščiai tikrinti dinamiškai generuojamą JavaScript kodą. Tai yra ir kodo blokai, ir įvykių atributai. Vienintelė saugi vieta įterpti dinamiškai formuojamą turinį į kodą yra reikšmė tarp kabučių ženklų. Kitur įterpti nepatikimus duomenis į JavaScript kodų blokus yra labai

pavojinga, kadangi labai paprastai galima pakeisti kodo struktūra į įterpto nepatikimo teksto-kodo vykdymą.

```
<script>alert('... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ NEPATIKRINUS...')</script>
```
teksto eilutės įterpimas tarp kabučių ženklų į kodą

```
<script>x='... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ...'</script>
```
kintamųjų reikšmių priskyrimas

```
<div onmouseover="x='... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ
                    NEPATIKRINUS...'"</div>
```
teksto įterpimas į įvykio reikšmę tarp kabučių

Atkreipkite dėmesį, kad yra JavaScript funkcijų į kurias niekuomet negalima saugiai įterpti duomenų, net jeigu kodas yra ir patikrintas. Pavyzdžiui:

```
<script>
window.setInterval('... NIEKUOMET SAUGIAI NEGALIMA ĮTERPTI NEPATIKIMŲ
DUOMENŲ...');
 </script>
```

5. HTML stiliaus savybių reikšmės privalo būti patikrintos ir tinkamai suformatuotos. Penktoji taisyklė nurodo, kad įterpiant nepatikimą tekstą į stilių žymes ar stilių blokus taip pat reikia išsamiai patikrinti tekstus. CSS yra stebėtinai galingas ir gali būti naudojamas atliekant įvairius išpuolius. Vietos, kur įterpiant duomenis, svarbiausia tikrinti:

```
<style> selektorius { parametras : ... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ
                      NEPATIKRINUS...; } </style>
```
parametro reikšmė

```
<style>selektorius { parametras : "... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ
                      NEPATIKRINUS..."; } </style>
```
parametro reikšmė

```
<span style=" parametras : ... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ
                    NEPATIKRINUS...">text</style>
```
parametro reikšmė

Yra ir tokių vietų, kur negalima saugiai įterpti ir patikrintų duomenų:

```
{ background-url : "javascript:alert(1)"; }
```
// ir kituose adresų nustatymuose

```
{ text-size: "expression(alert('XSS'))"; }
```
// tik IE naršyklėje

Nuo pavyzdyje parodytų atakų būdų galima apsisaugoti patikrinant ar adresų reikšmės prasideda „http", bet ne „javascript", o kitos CSS nurodytų parametrų reikšmės niekuomet neprasidės „expression".

6. HTML URL parametrų reikšmės privalo būti patikrintos ir tinkamai suformatuotos. Ši taisyklė taikoma tuomet, kai nepatikimi duomenys yra perduodami naudojantis HTTP GET metodo parametrais.

```
<a href="http://www.tinklalapis.com?test=... NEGALIMA ĮTERPTI NEPATIKIMŲ DUOMENŲ
                    NEPATIKRINUS...">nuoroda</a >
```

Nuorodos su parametrais visuomet HTML kode turi būti rašomos tarp kabučių ženklų, nors ir rašant tarp kabučių parametruose yra negalimų simbolių: [tarpas], %, *, +, -, /, ;, <, >, =, ^ ir |. Reikia atkreipti dėmesį į tai, kad adreso kodavimas šiuo atveju nėra naudingas. Kodavimas gali būti

naudingas tik tokiu atveju, jeigu duomenis prieš koduojant yra patikrinami ar neturi negalimų simbolių.

7. Naudoti HTML saugumo politikos tikrinimo įrankius. OWASP siūlo įrankius, kurie, veikdami API principu, tikrina HTML kodą ir aptinka kode galimus pažeidimus bei juos panaikina. .NET platformai OWASP siūlo savo sukurtą įrankį „OWASP AntiSamy“. Jo panaudojimo pavyzdys:

```
import org.owasp.validator.html.*;
Policy policy = Policy.getInstance(POLICY_FILE_LOCATION);
AntiSamy as = new AntiSamy();
CleanResults cr = as.scan(dirtyInput, policy);
MyUserDAO.storeUserProfile(cr.getCleanHTML());
```

.NET platformoje tam pačiam tikslui galima rinktis ir kitą įrankį, kuris jau anksčiau yra paminėtas šiame darbe – „Microsoft AntiXSS“ biblioteka. Java platformai OWASP taip pat siūlo savo įrankį „OWASP Java HTML Sanitizer“. Pavyzdys:

```
import org.owasp.html.Sanitizers;
import org.owasp.html.PolicyFactory;
PolicyFactory sanitizer =
Sanitizers.FORMATTING.and(Sanitizers.BLOCKS);
String cleanResults = sanitizer.sanitize("<p>Labas, <b>pasauli!</b>");
```

Tiems, kas dirba su PHP platforma, OWASP siūlo naudoti „HTML Purifier”. „HTML Purifier” - tai yra standartinė biblioteka, kuri atlieka HTML kodo filtravimą. Ji parašyta PHP programavimo kalba. Ši biblioteka ne tik kad suranda ir pašalina iš HTML kodo kenksmingas vietas (XSS), bet ir užtikrina, kad kodas atitiks W3C standartus.

8. Vengti DOM lygio XSS pažeidžiamumų. DOM lygio atakos gali būti vykdomos praktiškai visais anksčiau aprašytais būdais (įterpiant kenkėjišką kodą į atributų reikšmes, spausdinant pranešimus su neleistinais simboliais, kurie gali sugadinti HTML kodą ir pan.), tik jos vykdomos naudojantis JavaScript kodais, jų pagalba keičiant HTML turinį ir paleisti vartotojo kompiuteryje procesus, kurie gali šnipinėti vartotoją, apkrauti vartotojo kompiuterį, ištrinti naudingą informaciją ir pan. Norint apsisaugoti nuo šio tipo pažeidžiamumų, apsauga taikoma tik kliento pusėje, nes šios atakos yra vykdomos vartotojų kompiuteriuose. Kad apsisaugoti nuo DOM XSS, OWASP siūlo naudoti duomenų kodavimą kliento pusėje vykdomą JavaScript pagalba. Šiam tikslui siūloma naudoti jau parengtas bibliotekas su kodavimo funkcijomis: jQuery duomenų kodavimo biblioteka (angl.: *jQuery Encoder*; adresas: http://yet-another-dev.blogspot.com/2011/02/client-side-contextual-encoding-for.html) ir ESAPI4JS (randama adresu: http://bit.ly/9hRTLH).

Papildoma taisyklė. Kaip matote, visiškai apsisaugoti nuo XSS atakų yra labai sunku. Siekiant kuo labiau sušvelninti XSS spragos poveikį, OWASP rekomenduoja tinklalapio sesijose naudoti papildomą pridėtą žymę HTTPOnly. Ši žymė pagal nutylėjimą yra naudojama .NET platformoje, bet kituose platformose tai reikia daryti puslapio kūrėjams.

2.3. Bendrosios tinklalapių saugumo taisyklės

Išsamiai išanalizavus saugumo problemas internete matome, kad XSS tipo atakos yra labai populiarios. Išanalizavus šio tipo atakų variantus galima spręsti, kad apsauga nuo šių atakų yra labai sudėtinga ir norint užtikrinti savo tinklalapio duomenų ir jame besilankančių vartotojų saugumą būtina imtis saugumo priemonių. Darbe surinktos ir aprašytos dažniausiai aptinkamos atakos. Kadangi XSS tipo atakų yra nuolat sukuriama vis naujų ir nutaikytų į vis skirtingus tinklalapių

pažeidžiamumus, nuo šių atakų apsisaugoti yra ypač sunku. Literatūroje ir internete yra siūlomos bendros saugumo taisyklės, kuriomis vadovaujantis galima sumažinti tikimybę nukentėti nuo šio tipo atakų. Apsauga gali ir turi būti vykdoma ne tik tinklalapyje, bet ir vartotojo kompiuteryje.

3. ATSAKO LAIKO ĮVERTINIMO IR MINIMIZAVIMO TYRIMO METODIKA

Programinio kodo įterpimas į vartotojo peržiūrimą tinklalapį (XSS) - tai pažeidžiamumas, kuris dažniausiai aptinkamas tinklapiuose, leidžiančiuose įterpti papildomą programinį kodą į vartotojų peržiūrimą puslapį. Toks pažeidžiamumas atsiranda dėl nepakankamo įvedamos informacijos filtravimo [15]. Radęs būdą įterpti kenksmingą kodą, piktavalis gali gauti prieigą prie tinklapio turinio, sesijos slapukų ar kitos informacijos. Beveik visas XSS galimybes sukuria tos programos, kurios nesugeba saugiai valdyti HTML įvesties ir išvesties - ypač HTML žymių laužtiniuose skliaustuose (< ir >) ir tarp kelių kitų simbolių [2]. Beveik kiekvienai aptiktai XSS pažeidžiamai vietai būdingas nesugebėjimas pašalinti laužtinių skliaustų iš išvesties arba nesugebėjimas koduoti tokių skliaustus išvestyje. XSS atakų tikslas dažniausiai yra vartotojo duomenų vagystė, siekiant gauti prieigą prie konfidencialios informacijos, gauti mokamą informaciją nemokamai, sekti vartotojų lankomus puslapius, pakeisti vartotojo naršyklės nustatymus ir pan. Todėl labai svarbu užtikrinti efektyvų tinklalapio saugumą.

Tinklalapių turinio saugumo užtikrinimui yra sukurta įvairių apsaugos priemonių. Norint visiškai apsisaugoti nuo kodo įterpimo atakų, vartotojo kompiuteryje galima tiesiog išjungti programinio kodo vykdymą vartotojo kompiuteryje, tačiau tai labai apriboja puslapių funkcionalumą, kas yra nepatogu vartotojui. Naršyklių kūrėjai kuria vis labiau apsaugotas naršykles, kurios apriboja įtartinų puslapio kodų vykdymą, taip pat leidžia įvairiausius įskiepius, kurie padidina saugumą. Naršyklėmis kiekvienas žmogus naudojasi pagal savo individualius įpročius ir poreikius, todėl įtakoti žmogaus pasirinkimą motyvuojant tuo, kad kažkuri naršyklė yra saugesnė, beveik neįmanoma. Be to visos modernios naršyklės jau turi bent minimalias apsaugas nuo kodo įterpimo atakų. Norint, kad naršant internete vartotojas jaustųsi saugus, jis būtinai turi naudoti naujausią savo naudojamos naršyklės versiją.

Vien vartotojo pastangų apsisaugoti nuo kodo įterpimo atakų nepakanka. Nemaža dalis saugumo problemų kyla serverio pusėje, todėl kiekvieno tinklalapio saugumą serverio pusėje privalo užtikrinti puslapio kūrėjai [2]. Kiekviename dinaminio turinio tinklalapyje privaloma naudoti išsamų vartotojo įvedamų duomenų filtravimą, taip pat tikrinti rodomą informaciją, kartais net būtina naudoti turinio šifravimą [1]. Serverio pusėje tinklalapio apsaugai programuotojai, įvertinę visas rizikas ir pavojus, privalo pasirinkti tinkamus apsaugos lygius ir būdus. Tai nėra lengvas uždavinys, nes apsaugos būdų yra siūloma labai daug: nuo paprasčiausių programavimo kalbos komandų (pavyzdžiui PHP kalboje naudojamos komandos: htmlentities(), strip_tags(), utf8_decode()) iki serverio pusėje API principu veikiančių tinklalapio tekstų šifravimo arba kodavimo bibliotekų. Tokių bibliotekų internete gausu. Jos kuriamos įvairioms programavimo kalboms tiek mokamos, tiek ir nemokamos. Todėl pagrindinis šio tyrimo tikslas bus ištirti, kurios iš nemokamų turinio filtravimo bibliotekų pakankamai apsaugo tinklalapį ir kaip tai paveikia tinklalapio atsako laiką.

3.1. Testavimo priemonės

Norint atlikti testavimus, pirmiausia buvo sukurti trys testavimui naudojami tinklalapiai. Tinklalapiai buvo kuriami trijuose skirtinguose platformose, skirtingomis technologijomis:

1. Apache 2.2.22 serveryje veikiantis PHP 5.3.13 programavimo kalba parašytas tinklalapis. Duomenų saugojimui tinklalapis naudoja MySQL 5.5.24 duomenų bazę. Sistema veikia vietiniame kompiuteryje (localhost).
2. ASP.NET (.NET Framework 4) serveryje veikiantis C# programavimo kalba parašytas tinklalapis. Duomenų saugojimui jis naudoja MS SQL duomenų bazę. Sistema veikia vietiniame kompiuteryje (localhost).
3. Java programavimo kalba parašytas tinklalapis (naudojamas Spring Framework 3.0.2 karkasas). Sistema veikia vietiniame kompiuteryje (localhost) ir naudoja MySQL 5.5.24 duomenų bazę.

Siekiant atlikti tinklalapių testavimą, buvo pasirinktos trys pagrindinės tinklalapio kūrimo technologijos, t.y. PHP, ASP.NET, Java. Šios tinklalapio kūrimo technologijos pasirinktos dėl

didelio jų populiarumo kuriant internetines programas. Kaip matome antroje lentelėje, pasirinktos technologijos yra plačiau naudojamos. 2 lentelėje yra pateikta, kurias technologijas naudoja didžiosios informacijų technologijų kompanijos.

2 lentelė. Žinomiausių pasaulyje IT kompanijų naudojamos technologijos

Tinklalapis	Serveris (platforma)	Programavimo kalbos
Google.com	Linux	C, Java, C++, PHP & MySQL
Facebook.com	Linux	PHP, MySQL and C++
YouTube.com	Linux	C, Java and MySQL
Yahoo.com	Linux	C++, C, Java, PHP & MySQL
MSN.com (sukurta Microsoft)	Windows	ASP.net
Live.com (sukurta Microsoft)	Windows	ASP.net
Wikipedia	Linux	PHP & MySQL
Amazon.com	Linux & Solaris	C++, Java, J2EE
WordPress.com	Linux	PHP & MySQL

Visi trys tinklalapiai turi pavyzdinę vartotojams skirtą komentavimo formą ir jau įvestų komentarų sąrašą. Testuojant įvestų komentarų sąrašą naudojami jau turimi XSS atakų pavyzdžiai [16]. Šie pavyzdžiai, kurie yra pateikti OWASP projekto tinklalapyje, naudojami testuojant komentarų atvaizdavimą. Testuojant komentarų įvedimo formą buvo naudojamas vienas sukurtas pavyzdinis komentaras, kuriame įrašyta daug įvairiausių simbolių. Komentarų forma pasirinkta, nes simboliai aptinkami beveik kiekviename dinaminio turinio puslapyje ir dažnai viešai prieinami, todėl dažnai tampa piktavalių taikiniu. Komentarai saugomi duomenų bazėje. Tinklalapių testavimui naudojama Google Chrome naršyklės 24.0.1312.57 versijos programuotojams skirtas įrankis. Testuojant, buvo matuojamas tinklalapio atsako laikas be jokios apsaugos ir rezultatai fiksuojami. Po to tinklalapiai buvo bandomi su integruotomis saugos priemonėmis, kurios kiekvienai technologijai yra savitos.

Eksperimentų metu buvo fiksuojamas tinklalapių atsako laikas. Atsako laikas buvo matuojamas duomenis nuskaitant iš duomenų bazės ir apdorojant juos prieš parodant vartotojui ekrane, ir saugant naują vartotojo įvestą komentarą, prieš tai iš jo išfiltravus duomenis su parinktomis saugumo priemonėmis. Abu eksperimentai atliekami atskirai, kiekvieną kartą suvienodinant duomenų bazėje esančios informacijos kiekį (ištrinamas po kiekvieno eksperimento išsaugotas įrašas iš duomenų bazės, jeigu tai yra įrašo išsaugojimo laiko matavimas). Informacijos kiekis duomenų bazėje turi būti vienodas, nes apdorojamos informacijos kiekis turi įtakos atsako laikui [17]. Kuo daugiau informacijos bus bandoma atvaizduoti ekrane, prieš tai ją patikrinus, tuo daugiau laiko užims informacijos tikrinimas. Informacijos atvaizdavimo bandymams buvo naudojama OWASP projekto tinklalapyje pateikiami visų dažniausiai aptinkamų tipų XSS atakų pavyzdžiai [16], kurie iš anksto, nenaudojant jokios apsaugos, surašomi į duomenų bazę ir tuomet vykdomi atidarant testavimo tinklalapį. Su kiekvienu iš apsaugos variantų testavimai atliekami po 10 kartų ir apskaičiuojamas aritmetinis laiko verčių vidurkis, o tai bus laikoma to apsaugos būdo atsako laiku. Gauti rezultatai bus palyginti, taip nustatant, kaip bibliotekos keičia tinklalapio atsako laiką.

Matuoti tik atsako laiką tam, kad galėtume spręsti, jog tai yra geriausia apsauga, būtų klaidinga. Bibliotekos, kurios atlieka tik neleistinų simbolių paiešką ir jų pašalinimą arba konvertavimą į leistinus HTML simbolius, visada veiks greičiau už sudėtingus tekstų filtravimus atliekančias bibliotekas, tačiau jos negali tinkamai užtikrinti saugumo tinklalapiuose. Galima lengvai nuspėti, kad programa, kurioje nenaudojama jokia papildoma biblioteka ir apsaugai panaudotos tik programavimo kalbos komandos, veiks greičiausiai. Būtent dėl visų šių priežasčių būtina įvertinti kiekvienos bibliotekos patikimumą saugumo kontekste. Pats paprasčiausias būdas patikrinti bibliotekos

patikimumą yra apžvelgti jos funkcionalumą, taip įvertinant, nuo kurio tipo atakų padeda apsisaugoti analizuojama biblioteka. Tokiam įvertinimui buvo sudaromos lentelės, kuriuose buvo išskirtos penkios jautrios zonos saugumo prasme HTML kode ir surašomos visos bibliotekos. Lentelė užpildoma įvertinant bibliotekos funkcionalumą, ir tuomet galima aiškiai pastebėti, kuri biblioteka sugeba apsaugoti daugiausia pažeidžiamų vietų. Lentelės pavyzdys pateikiamas trečioje lentelėje.

3 lentelė. HTML kodo tikrinimo bibliotekų funkcionalumo įvertinimas

Galimi pažeidimai / **Biblioteka**	**Koduotės simbolių filtravimas**	**HTML kodo formatavimas (užtikrinimas, kad žymės uždarytos ir pan.)**	**Atributų filtravimas**	**Kodo įterpimo skenavimas**	**Standartų užtikrinimas**
Biblioteka1	taip	ne	ne	taip	ne
....					
BibliotekaN	ne	dalinai	ne	taip	ne

Trečioje lentelėje pateikti trys galimi saugumo įrankių įvertinimai. Reikšmės gali būti: taip, ne ir dalinai. Taip – reiškia, kad saugumo biblioteka pilnai palaiko nurodytą funkcionalumą, ne – reiškia, kad saugumo priemonė neturi realizuotos nurodyto tipo apsaugos ar funkcionalumo, o įvertinimas dalinai nurodo, kuri biblioteka turi realizuotą dalį nurodyto funkcionalumo, bet pilnai negali apsaugoti nuo nurodyto pažeidžiamumo.

Atliekant testavimus atsako laikas matuojamas naudojantis Google Chrome naršyklėje integruotas tinklalapių programuotojų priemonėmis. Matavimai atliekami nuosekliai, vienas po kito. Atliekant testavimus, kompiuteris papildomų uždavinių neatliko. Testavimui naudojamo kompiuterio aparatūrinės ir programinės įrangos sąrašai pateikti ketvirtoje ir penktoje lentelėse.

4 lentelė. Testavimui naudojamo kompiuterio aparatinė įranga

Procesorius:	Intel(R) Core(TM) i7-2600 3,4 GHz
Darbinė atmintis:	DDR3 8,00 GB
Vaizdo plokštė:	NVIDIA GeForce GTX 560
Kietasis diskas:	SAMSUNG SATA3, Talpa: 1TB

5 lentelė. Testavimui naudojamo kompiuterio programinė įranga

Operacinė sistema:	Windows 7 Professional 64-bit
Interneto naršyklė:	Google Chrome 24.0.1312.57 versija
Antivirusinė programa:	(nėra)
Serverio programinė įranga:	Apache Tomcat v7.0 (paleidžiamas ir veikiantis kartu su Eclipse Java EE IDE for Web Developers. Versija: Indigo Service Release 2) WampServer Version 2.2 pakete naudojama: Duomenų bazė: MySQL 5.5.24 Projekte naudojamas: Spring Framework 3.0.2 karkasas

Testavimo tinklalapiams talpinti ir publikuoti naudojamas tas pats kompiuteris, su kuriuo bus atliekami ir testavimai, t.y. paprastas asmeninis kompiuteris. Visa testavimams reikalinga programinė įranga yra naudojama nemokama arba laikinai veikiančios bandomosios versijos.

3.2. PHP tinklalapis

PHP tinklalapis testuojamas su šiomis tinklalapio kodo filtravimo priemonėmis:

- Nenaudojant jokio filtravimo.
- Filtruojant spausdinamą turinį PHP programavimo kalbos komandomis (htmlentities, strip_tags, utf8_decode) [18].
- HTMLPurifier – biblioteka filtruojanti HTML kodą ir pašalinanti neleistinus simbolius, be to ji sugeba ištaisyti HTML kodo elementarias klaidas [19].
- SafeHTMLChecker - HTML kodo filtravimo biblioteka [20].
- htmLawed - HTML kodo filtravimo biblioteka. Turi įskiepius į populiariausias turinio valdymo sistemas, todėl plačiai naudojama juose [21].
- Kses – PHP HTML/XHTML kodo filtras. Jis ištrina iš tikrinamo bloko nepageidaujamus elementus ir atributus [22].

Šio tinklalapio testavimui buvo pasirinktos plačiausiai naudojamos bibliotekos, tokios kaip HTMLPurifier saugumo įrankis, kurį pataria naudoti OWASP projekto kūrėjai [4]. HTMLPurifier, SafeHTMLChecker saugumo įrankis turi įskiepius, kurie yra pritaikyti naudoti populiariausiuose turinio valdymo sistemose (Joomla!, Wordpress, Drupal ir kt.). Taip pat šie įrankiai turi sukurtas integravimo priemones su populiariais PHP programavimo kalbos karkasais: Symfony, CakePHP, CodeIgniter ir kitais.

3.3. ASP.NET (C#) tinklalapis

Sistema testuojama su šiomis tinklalapio kodo filtravimo priemonėmis:

- Nenaudojant jokio filtravimo.
- Filtruojant spausdinamą turinį ASP.NET programavimo kalbos komandomis (HttpUtility.HtmlEncode) [23].
- Microsoft Anti-Cross Site Scripting LibrarySafe HTML Checker – Microsoft programuotojų išleista biblioteka HTML kodo filtravimui ir neleistinų simbolių aptikimui bei pašalinimui [14].
- OWASP AntiSamy – OWASP išleista biblioteka skirta išeities kodo filtravimui [24].
- TidyManaged – mažiau populiarus ir rečiau naudojamas HTML kodo analizatorius [25].

Microsoft Anti-Cross Site Scripting LibrarySafe HTML Checker biblioteka buvo pasirinkta testavimams todėl, nes ji yra kuriama ASP.NET technologijos kūrėjų (Microsoft) ir rekomenduojamas kaip patikimas apsaugos įrankis [14]. Šį įrankį taip pat rekomenduoja ir OWASP projekto kūrėjai [4]. OWASP projekto tinklalapyje taip pat siūlomas ir jų kuriamas įrankis OWASP AntiSamy. Jis taip pat buvo įtrauktas į eksperimentą.

Atliekant bandymus, kaip ir PHP tinklalapio testavimo atveju, atsako laikas buvo matuojamas naudojantis Google Chrome naršyklėje integruotomis priemonėmis. Matavimų metu kompiuteris papildomų užduočių neatliko.

3.4. Java tinklalapis

Java programavimo kalba parašytas tinklalapis buvo testuojamas su šiomis tinklalapio kodo filtravimo priemonėmis:

- Nenaudojant jokio filtravimo.
- Filtruojant spausdinamą turinį Java programavimo kalbos komandomis (escapeXml, replaceAll, regex) [26].

- ESAPI [13].
- OWASP Java HTML Sanitizer – OWASP projekto sukurtas HTML kodo analizatorius parašytas Java programavimo kalba [4].

Testavimui buvo pasirinktos OWASP projekto kuriamos ir plėtojamos bibliotekos. Šios bibliotekos kuriamos ir patariamos naudoti OWASP projekto dalyvių ir kūrėjų, kurie aktyviai dalyvauja kovojant su kodo įterpimo atakomis [4, 13]. Apsauga realizuota programavimo kalbos komandomis escapeXml, replaceAll ir regex, o į eksperimentą įtraukta siekiant pademonstruoti, kad ir paprastais metodais realizuota minimali apsauga mažai keičia atsako laiką, bet ji nėra pakankama.

Atliekant bandymus kaip ir prieš tai aprašytais atvejais, atsako laikas matuojamas naudojantis Google Chrome naršyklėje integruotomis priemonėmis. Matavimų metu kompiuteris papildomų užduočių neatliko, kad rezultatai būtų kuo tikslesni.

3.5. Bendrieji testavimų nurodymai

Tyrimui pasirinktos trys plačiausiai vartojamos internetinių sistemų kūrimo technologijas: PHP, ASP.NET, Java. Kiekvienos technologijos tinklalapiai bus tiriami taikant skirtingas HTML turinio filtravimo bibliotekas. Tiriama ne tik tinklalapio atsako laiko pokyčiai, bet ir saugumo lygis tinklalapiuose su kiekviena turinio filtravimo biblioteka. Pagal aprašyto tyrimo rezultatas bus galima daryti išvadas, kurios saugumo priemonės pakankamai apsaugo tinklalapį ir labai nesulėtina atsako laiko.

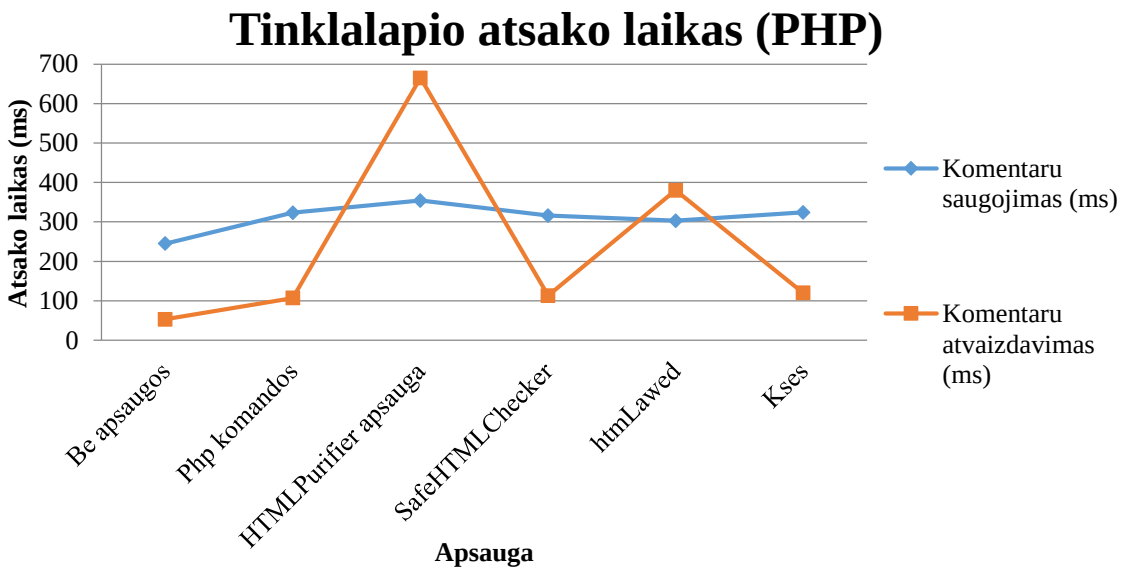
4. ATSAKO LAIKO ĮVERTINIMAS IR JO MINIMIZAVIMAS

Pagal aprašytas tyrimo metodikas buvo atliekami tyrimai, siekiant išsiaiškinti analizuojamų HTML kodo analizatorių greitaveiką ir saugumo užtikrinimo efektyvumą. Buvo testuojami atskirų technologijų tinklalapiai su jiems skirtais įrankiais. Pirmiausia buvo testuojamas tinklalapių atsako laikas, tuomet palyginamas bibliotekų funkcionalumas, kad būtų galima nuspręsti, kurios bibliotekos užtikrina pakankamą saugumo lygį. Šiame skyriuje pristatomi tyrimo rezultatai. Testavimai buvo atliekami vienodomis sąlygomis, tuo pačiu atvaizduojamų įrašų skaičiumi visuose eksperimentuose ir vienodais saugomais duomenimis, kuriant naują įrašą. Visos trys platformos buvo testuojamos atskirai.

4.1. PHP tinklalapio atsako laiko ir apsaugos nuo XSS atakų tyrimas.

PHP tinklalapis buvo testuojamas su šešiomis skirtingomis tinklalapio kodo filtravimo priemonėmis: nenaudojant jokio filtravimo, filtruojant spausdinamą turinį PHP programavimo kalbos komandomis, naudojant filtravimo bibliotekas HTMLPurifier, SafeHTMLChecker, htmLawed ir Kses.

Atliekant testavimus, atsako laikas buvo matuojamas naudojantis Google Chrome naršyklėje integruotomis tinklalapių programuotojų priemonėmis. Matavimai buvo atliekami nuosekliai, vienas po kito. Atliekant testavimus kompiuteris papildomų uždavinių neatliko, todėl testavimo rezultatai nebuvo vėlinami pašalinių veiksnių. Atlikus atsako laiko testavimus su visomis anksčiau išvardintomis priemonėmis buvo gauti rezultatai, kurie pateikiami 2 paveiksle.



2 pav. Tinklalapio atsako laiko matavimų rezultatai PHP tinklalapyje

Iš gautų rezultatų galima aiškiai pastebėti kurie apsaugos metodai labiausiai vėlina tinklalapio atsako laiką. Į testavimą taip pat buvo įtrauktas matavimas kai netaikoma jokia apsauga. Tai buvo atliekama tam, kad būtų galima pastebėti, kad bet koks menkiausias apsaugos būdas didina tinklalapio atsako laiką. Antrajame paveiksle pateiktame grafike matyti, kad HTMLPurifier biblioteka veikė lėčiausiai (saugant duomenis: 354 ms; atvaizduojant: 665 ms), o tik PHP komandų naudojimas buvo sparčiausias (saugant duomenis: 323 ms; atvaizduojant: 107 ms), tačiau jos neužtikrina pakankamos tinklalapio apsaugos ir norint nustatyti, kuris metodas veikia pakankamai greitai ir užtikrina saugumą, bus atliekamas sekantis eksperimentas. Pagal saugumo laiką iš testuotų bibliotekų mažiausiai atsako laiką didino SafeHTMLChecker biblioteka.

Tinklalapio saugojimo atsako laikas kito daug mažiau negu atvaizdavimo. Saugant komentarą skenavimai atliekami tik vienam tam komentarui, o kadangi vienas komentaras neturi daug teksto,

tikrinimas, su daug įvairių skenavimų, atsako laiko labai nepablogino. Du testavimo atvejai, kai komentarų atvaizdavimo atsako laikas buvo ilgesnis negu saugojimo parodo kaip smarkiai įtakoja atsako laiką įvairūs atliekami veiksmai, kai nuskaičius informaciją iš duomenų bazės sistema ją detaliai analizuoja ir apdoroja.

Vien pagal atsako laiką negalima spręsti, kad ištirtos apsaugos priemonės yra tinkamiausios apsaugai nuo XSS atakų. Kadangi šio darbo tikslas yra išsiaiškinti, kuri biblioteka užtikrina saugumą labai nesuvėlindama tinklalapio atsako laiko, būtina išsiaiškinti, kuri biblioteka pakankamai gerai saugo nuo kodo įterpimo atakų. Šiam tikslui pasiekti sudaryta lentelė (6 lentelė), kurioje sužymėtos kiekvienos bibliotekos palaikomos funkcijos, leidžiančios aiškiai nuspręsti, kuri biblioteka gali užtikrinti saugumą tinklalapyje.

6 lentelė. PHP programavimo kalbos HTML kodo tikrinimo bibliotekų funkcionalumo įvertinimas

Galimi pažeidimai / **Biblioteka**	**Koduotės simbolių filtravimas**	**HTML kodo formatavimas (užtikrinimas, kad žymos uždarytos ir pan.)**	**Atributų filtravimas**	**Kodo įterpimo skenavimas**	**Standartų užtikrinimas**
PHP komandos	Dalinai	Ne	Ne	Dalinai	Ne
HTMLPurifier	Taip	Taip	Taip	Taip	Taip
SafeHTMLChecker	Ne	Ne	Taip	Taip	Ne
htmlLawed	Dalinai	Ne	Taip	Taip	Taip
Kses	Dalinai	Ne	Dalinai	Taip	Ne

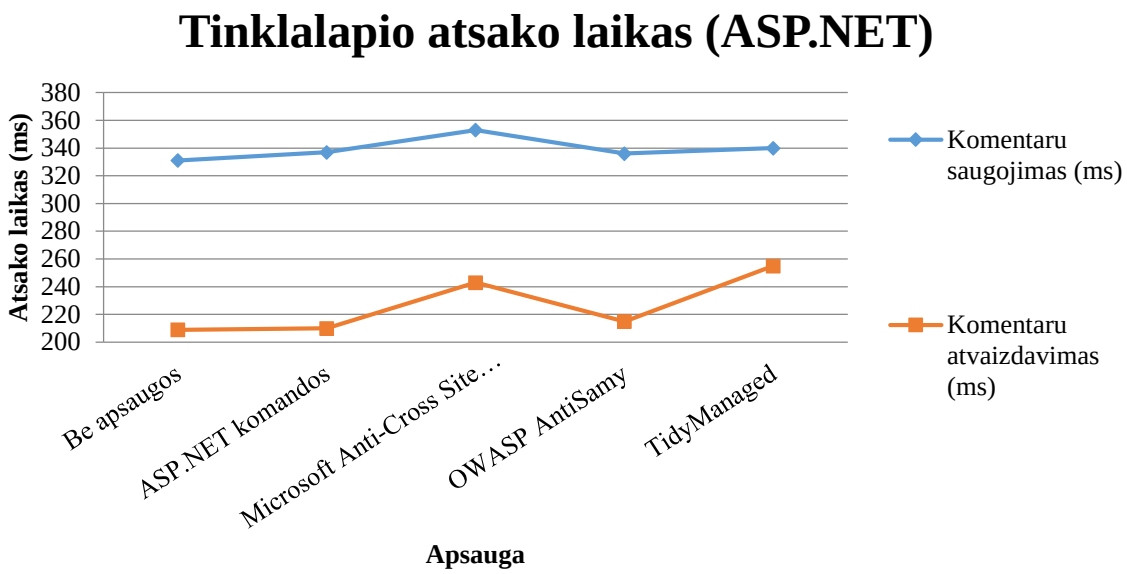
Pagal pateiktą ketvirtąją lentelę matyti, kad geriausią atsako laiką palaikanti filtravimo biblioteka SafeHTMLChecker negali užtikrinti saugaus HTML kodo. Ši biblioteka neturi realizuotos apsaugos nuo dviejų tipų pažeidimų: neatlieka patikrinimo ar HTML kode nėra kitos koduotės elementų; neturi HTML kodo sintaksės tikrinimo funkcijos. Koduotės elementų tikrintuvas būtinas, nes naršyklė geba keisti koduotes ir taip gali atvaizduoti tinklalapyje neleistinus simbolius, kuriuose yra užkoduotas piktavališkas kodas. O taip pat didžiulis šios bibliotekos trūkumas yra ir tai, kad ji neturi funkcijos, kuri patikrina HTML kode ar visos gairės yra uždarytos ir HTML kodas nėra klaidingas.

Kadangi šio darbo uždavinys yra pasiūlyti apsaugą, kuri užtikrina pakankamą saugumo lygį tinklalapyje, iš pirmojo eksperimento rezultatų galima teigti, kad pakankamą saugumo lygį gali užtikrinti tik biblioteka, kuri labiausiai vėlina tinklalapio atsako laiką: HTMLPurifier.

4.2. ASP.NET (C#) tinklalapio atsako laiko ir apsaugos nuo XSS atakų tyrimas.

Sistema ištestuota su šiomis tinklalapio kodo filtravimo priemonėmis: nenaudojant jokio filtravimo, filtruojant spausdinamą turinį ASP.NET programavimo kalbos komandomis, Microsoft Anti-Cross Site Scripting, OWASP AntiSamy, TidyManaged.

Atliekant bandymus, kaip ir PHP tinklalapio testavimo atveju, atsako laikas buvo matuojamas naudojantis Google Chrome naršyklėje integruotomis priemonėmis. Matavimų metu kompiuteris papildomų užduočių neatliko.



3 pav. Tinklalapio atsako laiko matavimų rezuyltatai ASP.NET tinklalapyje

Iš gautų rezultatų galima pastebėti, kurie apsaugos metodai labiausiai vėlina tinklalapio atsako laiką (3 pav.). Daugiausiai tinklalapio atsako laiką didino Microsoft Anti-Cross Site Scripting LibrarySafe HTML Checker, o mažiausiai, kaip ir buvo galima tikėtis, ASP.NET komandos. Testuojant šios technologijos tinklalapį nebuvo pastebėtas toks didelis atsako laiko vėlinimo skirtumas kaip testuojant PHP tinklalapį. Programavimo kalbos komandos saugumo užtikrinti negali, todėl, kaip geriausią apsaugos biblioteką pagal atsako laiko didinimą, galima įvardinti OWASP AntiSamy. Kaip matyti pateiktame trečio paveikslo grafike OWASP AntiSamy biblioteka labai nedaug padidino atsako laiką nuo paprastos HTML kodo tikrinimo komandos panaudojimo. Siekiant išsiaiškinti ar pakankamą saugumo lygį galima užtikrinti naudojantis OWASP ASP.NET technologija sukurtu įrankiu buvo sudaryta septintoji lentelė.

7 lentelė. ASP.NET programavimo kalbos HTML kodo tikrinimo bibliotekų funkcionalumo įvertinimas

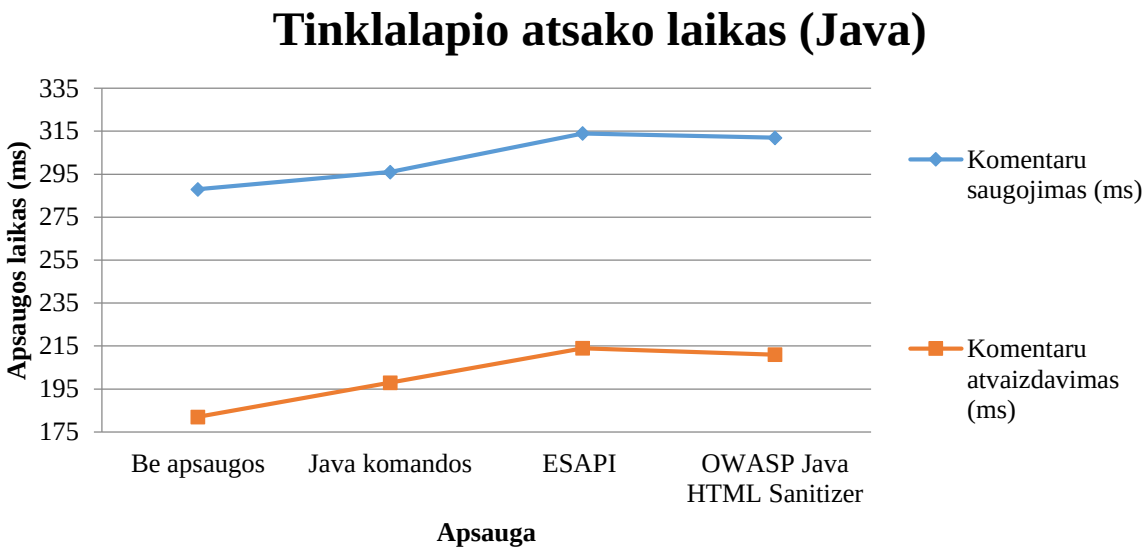
Galimi pažeidimai / Biblioteka	Koduotės simbolių filtravimas	HTML kodo formatavimas (užtikrinimas, kad žymos uždarytos ir pan.)	Atributų filtravimas	Kodo įterpimo skenavimas	Standartų užtikrinimas
ASP.NET komandos	Dalinai	Ne	Ne	Dalinai	Ne
Microsoft Anti-Cross Site Scripting Library	Taip	Taip	Taip	Taip	Ne
OWASP AntiSamy	Taip	Taip	Taip	Taip	Ne
TidyManaged	Taip	Ne	Taip	Taip	Ne

Septintoje lentelėje matyti, kad Microsoft Anti-Cross Site Scripting Library ir OWASP AntiSamy kodo filtravimo bibliotekos gali atlikti tokias pačias funkcijas, tačiau atsako laiko rodiklis yra geresnis OWASP AntiSamy ir dėl to galima teigti, kad pakankamą saugumo lygį užtikrina ir pakankamai gerą atsako laiką demonstruoja OWASP AntiSamy biblioteka.

4.3. Java tinklalapio atsako laiko ir apsaugos nuo XSS atakų tyrimas.

Sistema ištestuota su šiomis tinklalapio kodo filtravimo priemonėmis: nenaudojant jokio filtravimo, filtruojant spausdinamą turinį Java programavimo kalbos komandomis, ESAPI, OWASP Java HTML Sanitizer.

Atliekant bandymus, kaip ir prieš tai aprašytais atvejais, atsako laikas buvo matuojamas naudojantis Google Chrome naršyklėje integruotomis priemonėmis. Matavimų metu kompiuteris papildomų užduočių neatliko, kad rezultatai būtų kuo tikslesni. Naudojamas tas pats asmeninis kompiuteris.



4 pav. Tinklalapio atsako laiko matavimų rezultatai Java tinklalapyje

Iš gautų rezultatų galima pastebėti, kad tinklalapio atsako laiką labiausia didino ESAPI biblioteka, o mažiausiai – programavimo kalbos komandos. Iš nagrinėtų filtravimo bibliotekų greičiausiai veikė OWASP Java HTML Sanitizer.

Gavus atsako laikų testavimo rezultatus būtina įvertinti bibliotekų saugumo užtikrinimo lygius. Pagal saugumo funkcionalumą sudaryta aštunta lentelė, iš kurios bus galima įvertinti priemonių užtikrinamą saugumo lygmenį.

8 lentelė. Java programavimo kalbos HTML kodo tikrinimo bibliotekų funkcionalumo įvertinimas

Galimi pažeidimai / Biblioteka	**Koduotės simbolių filtravimas**	**HTML kodo formatavimas (užtikrinimas, kad žymos uždarytos ir pan.)**	**Atributų filtravimas**	**Kodo įterpimo skenavimas**	**Standartų užtikrinimas**
Java komandos	Dalinai	Ne	Ne	Dalinai	Ne
ESAPI	Taip	Ne	Taip	Taip	Ne
OWASP Java HTML Sanitizer	Taip	Taip	Taip	Taip	Ne

Pateiktoje Java HTML kodo filtravimo priemonių saugumo įvertinimo lentelėje matyti, kad geriausią saugumo lygmenį užtikrina OWASP Java HTML Sanitizer. Ši biblioteka taip pat yra geriausia ir atsako laiko prasme, atmetant Java programavimo kalbos komandas, kurios neužtikrina patikimo saugumo lygio. Remiantis gautais rezultatais galima teigti, kad Java programavimo kalba kuriamuose tinklalapiuose geriausia naudoti OWASP projekto sukurtą priemonę OWASP Java HTML Sanitizer.

4.4. PHP tinklalapio atsako laiko mažinimo tyrimas

Atlikus PHP tinklalapio filtravimo bibliotekų atsako laikų ir saugumo lygio užtikrinimo matavimus buvo nustatyta, kad geriausią saugumo lygį PHP tinklalapio atveju gali užtikrinti tik HTMLPurifier apsaugos įrankis. Jis yra labiausiai tinklalapio atsako laiką įtakojantis apsaugos įrankis, nes labiausiai iš visų testuotų įrankių vėlina tinklalapio atsako laiką. Todėl buvo atliekamas išsamesnis šio įrankio tyrimas, kurio metu bandoma sumažinti tinklalapio atsako laiką naudojant HTMLPurifier filtravimo biblioteką.

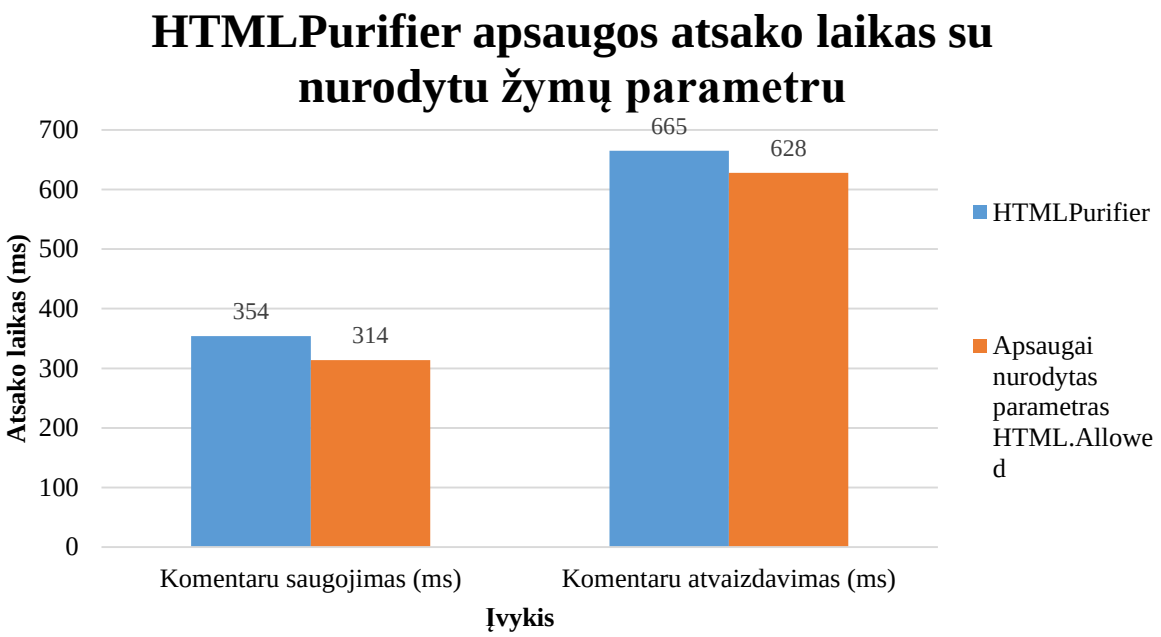
HTMLPurifier yra labai funkcionalus įrankis, kuris turi įvairių skenavimo priemonių. Skenavimo priemonės padeda sustiprinti saugumą tinklalapiuose, tačiau didina operacijų skaičių ir kartu lėtina atsako laiką. Norint pagerinti atsako laiką individualiais atvejais galima atsisakyti tam tikrų įrankio dalių vykdymo, kurių, programuotojų nuomone, tikrinti naudojamoje sistemoje ar tinklalapyje nebūtina. Visų sistemų kūrėjai turi įvertinti rizikas, kurios sistemos vietos yra pažeidžiamos, o koriuose atakas įvykdyti yra beveik neįmanoma. Įvertinus rizikas ir nusprendus, kad tam tikro pažeidimo skenavimas sistemoje nebūtinas, galima HTMLPurifier įrankį sukonfigūruoti taip, kad jis tam tikro skenavimo ar filtravimo nevykdytų ir taip sutaupytų dalį tikrinimams ir skenavimams sugaištamo laiko.

Tyrimo metu bus siekiama išsiaiškinti, kaip PHP tinklalapio, kuriame apsaugai nuo XSS atakų naudojama HTMLPurifier biblioteka, atsako laiką įtakoja tam tikrų konfigūracinių parametrų keitimas. Bibliotekos parametrais galima patikslinti tam tikrų filtravimų taisykles, kurias privalo apsibrėžti sistemos kūrėjai. Biblioteka turi numatytuosius nustatymus, kuriuos ji naudoja tuo atveju, kai parametrai nėra nurodomi.

Pirmojo šios dalies eksperimento metu buvo atliekami testavimai saugumo įrankiui, nurodant tikslias leistinas HTML kodo žymes ir jų atributus. Kaip šis parametras atrodo PHP kode pateikta pirmajame kodo fragmente.

```
$config->set('HTML.Allowed', 'div,p,b,a[href],i,span,strong');          (1)
```

Šis parametras nurodo kokios žymės ar jų atributai yra leistini skenuojamajame HTML kodo fragmente. Visos žymės, kurios nėra išvardintos šiame parametre bus atfiltruojamos kaip neleistinos. Žymės parametrui išvardijamos skiriant kableliais. Pavyzdyje, kuris pateiktas pirmoje formulėje, matyti parametras div, kuris reiškia, kad tikrinamame kode gali būti žymės ir jos yra leistinos. O, pavyzdžiui, viena iš nurodytų žymių a[href] reiškia tai, kad nuorodų žymės <a> yra leistinos, bet jos gali turėti tik vieno tipo atributą: href. Visi kiti atributai nuorodoms negalimi ir jie bus atfiltruoti [20]. Šis žymų sąrašas buvo pasirinktas darant prielaidą, kad tik šios žymės yra saugios testuojamame tinklalapyje.



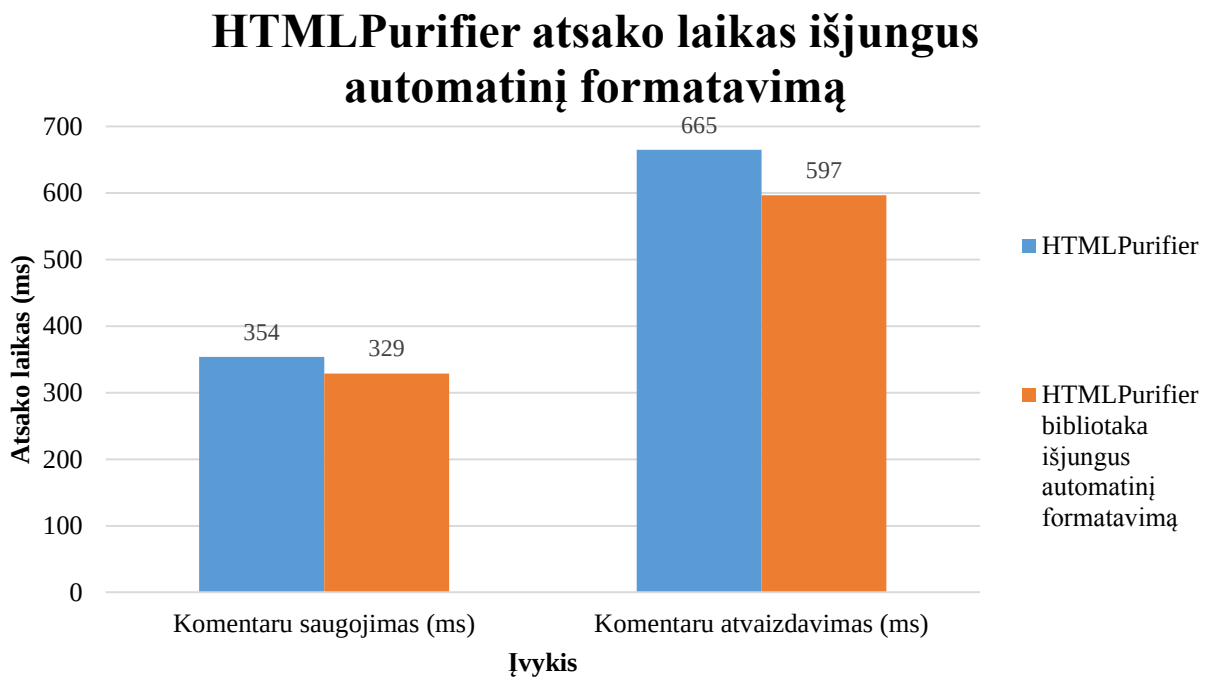
5 pav. Eksperimento rezultatas su nurodytu žymių sąrašu HTMLPurifier saugumo bibliotekoje

Ištestavus tyrime naudojamą tinklalapį, kai HTMLPurifier apsaugoje yra nustatomos konkrečios žymų reikšmės (1 formulė) buvo gauti rezultatai kurie pateikiami penktajame paveiksle. Paveiksle matomas atsako laikas, kuris buvo gautas filtravimo bibliotekai nurodžius parametrą ir prieš tai gauti testavimo rezultatai, kai buvo naudojami saugumo įrankio gamintojo įvesti bazinei nustatymai. Paveiksle aiškiai matoma, kad komentarų saugojimo ir atvaizdavimo tinklalapio atsako laiką nurodytas parametras pagerina. Atsako laikas komentarų atvaizdavimui sumažėja 37 ms, o saugojimas 40 ms. Taip yra todėl, kad nurodžius konkrečias leistinas žymes filtravimo biblioteka nebeturi tikrinti visų žymų kiek jai yra numatę kūrėjai (tai yra visos galimos HTML žymos), o tikrinti turi tik tas kurios nurodytos parametre. Visos kitos žymos, priklausomai nuo kitų parametrų, yra arba panaikinamos iš tikrinamo teksto, arba konvertuojamos į specialius simbolius ir atvaizduojamos kaip simboliai. Šis eksperimentas parodo, kad parametrai gali stipriai įtakoti filtravimo bibliotekos veikimo greitaveiką, todėl būtina ištirti ar ir kiti parametrai gali panašiai įtakoti įrankio veikimo laiką.

Toliau nagrinėjamas kitas parametras, kuriuo pasinaudojant galima išjungti arba įjungti HTML kodo žymių automatinį formatavimą į paragrafus. Ši funkcija atlieka dvigubų naujos eilutės simbolių paiešką ir pastraipas, atskirtas tokiais simboliais, suskirsto į HTML kodo paragrafus [19]. Paragrafams atskirti įrankis nenaudoja
 žymų, kas yra semantiškai neteisinga, o naudojamos <p> žymės. Kitaip tariant suformatuoja HTML kodą taip, kad jis būtų teisingai suformatuotas pagal HTML kodo standartus. Šis funkcionalumas naudingas, nes jis sumažina klaidų riziką HTML kode. Pagal nutylėjimą šis funkcionalumas filtravimo bibliotekoje yra išjungtas, o jį įjungti galima pasinaudojant komanda, kuri pateikiama antrajame kodo fragmente.

```
$config->set('AutoFormat.AutoParagraph', true);
```
(2)

Testavimo tinklalapyje šis funkcionalumas buvo įjungtas, nes mūsų pradinis tikslas buvo surasti saugumo įrankius kurie gali užtikrinti kuo patikimesnį saugumo lygį tinklalapiuose. Tačiau kai kuriuose sistemose šis funkcionalumas gali būti ir nenaudojamas, todėl buvo atlikti testavimai kaip keičiasi tinklalapio atsako laikas sistemoje išjungus šį funkcionalumą.



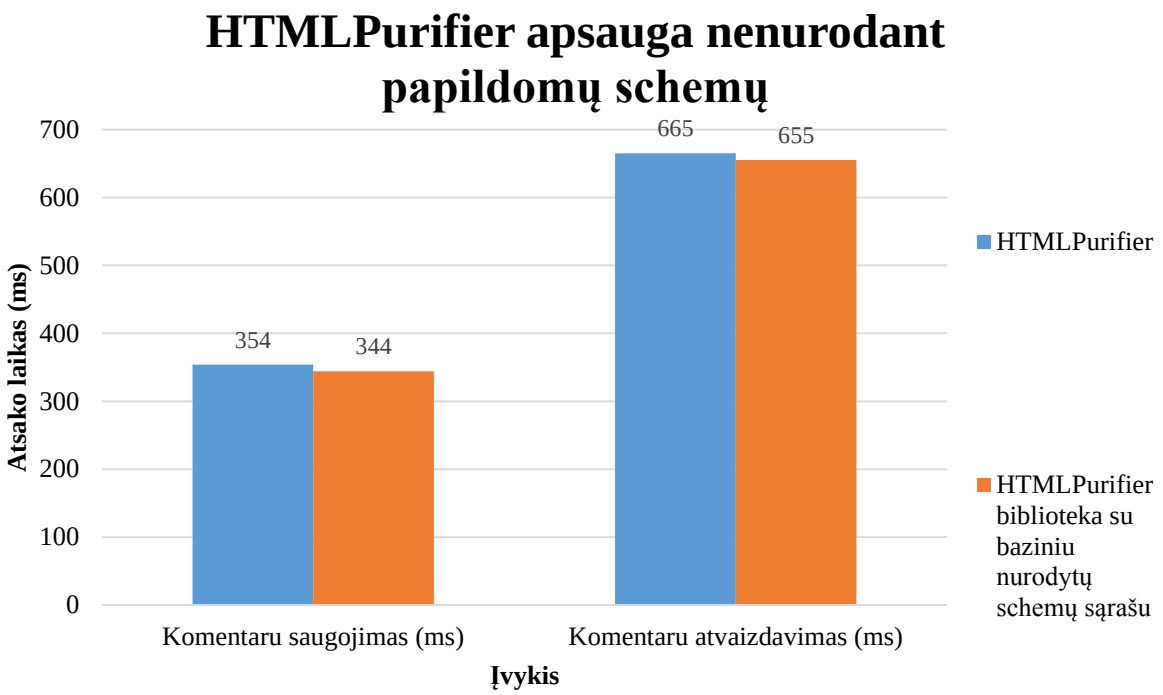
6 pav. Eksperimento rezultatas išjungus automatinį formatavimą

Kaip matyti 6 paveiksle (6 pav.), HTMLPurifier bibliotekos funkcionalumas reikalauja nemažai resursų. Testavimui naudojamo tinklalapio atveju komentarų atvaizdavimas, atsisakius automatinio formatavimo, pagreitėjo 68 ms, o saugojimas 25 ms. Esant tokiam ženkliam atsako laiko pokyčiui naudojant šią funkciją ir jos atsisakius, programuotojams privalu įvertinti ar jų kuriamoje sistemoje šis funkcionalumas tikrai būtinas ir yra vertas sugaištamo laiko.

Kitas nagrinėjamas parametras skirtas nustatyti schemas adresų, kurie gali būti talpinami skenuojamame HTML kode. Schemų tikrinimas reikalingas tam, kad dažnai bandant aptikti atakas į kodą bandoma įterpti nuorodas, kurios į vartotojo kompiuterį atsiunčia viruso failą, atidaro piktavališką kodą, nukreipia vartotoją į užkrėstą tinklalapį ar pan. Parametro panaudojimo pavyzdys pateiktas trečioje kodo eilutėje.

```
$config->set('URI.AllowedSchemes',array('http'=>true,'https'=>true));
```
(3)

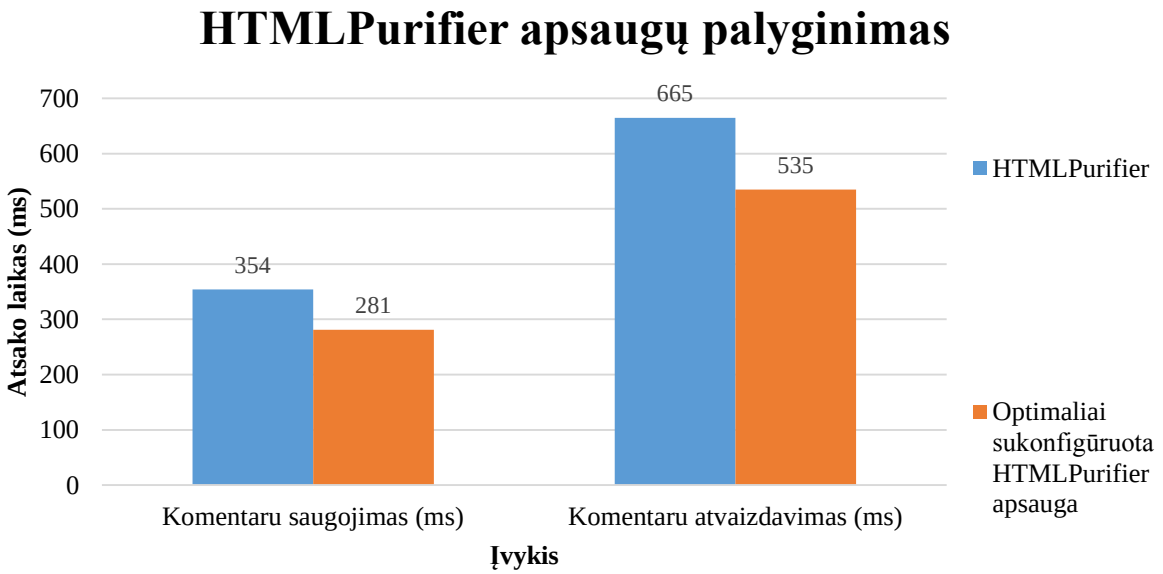
Pavyzdyje pateiktas parametras URI.AllowedSchemes, kuriam priskiriamas masyvas su schemų sąrašu ir pažymėjimu, ar nurodyta schema yra leidžiama skenuojamam kodui. Pavyzdyje nurodytos schemos http ir https yra leistinos ir gali būti naudojamos nuorodose tikrinamame kode. Tai reiškia, kad kode nuorodos ... ir ... yra leistinos, o kitokio tipo nuorodos, pavyzdžiui, ... bus atfiltruotos kaip nepatikimos. Tyrimo metu tinklalapis buvo testuojamas nurodant, kad leistinos schemos yra šios: http, https, mailto, ftp, nntp, news, callto. Pagal nutylėjimą biblioteka kaip leistinas turi nurodytas tik šias schemas: http, https, malto, ftp, nntp, news. Kaip šio parametro papildymas didesniu kiekiu schemų įtakoja atsako laiką galima pamatyti sekančiame paveiksle (7 pav.).



7 pav. HTMLPurifier apsauga nenurodant papildomų schemų

Grafike matomi rezultatai, kaip keičiasi tinklalapio atsako laikas nustatymuose sumažinant schemų sąrašą ir naudojant tik pagal nutylėjimą nustatomas schemas. Galima pastebėti, kad papildomos schemos uždraudimas tinklalapio atsako laiką sumažino vienodai tiek ir duomenų saugojimui, tiek ir atvaizdavimui - vidutiniškai po 10 ms. Taip yra todėl, kad sistemai reikia atlikti mažiau operacijų tikrinant ar tekste yra leistinos schemos nuorodų.

Ištestavus keletą variantų ir išsiaiškinus, kaip parametrai lemia bibliotekos veikimo greitaveiką galima daryti prielaidą, kad kiekvienai sistemai yra būtina nustatyti tinkamiausius parametrus. Kaip visi prieš tai eksperimentuose aprašyti optimalūs veikimo greitaveikai parametrai kartu įtakoja testuojamą tinklalapį pateikiama sekančiame paveiksle (8 pav.).



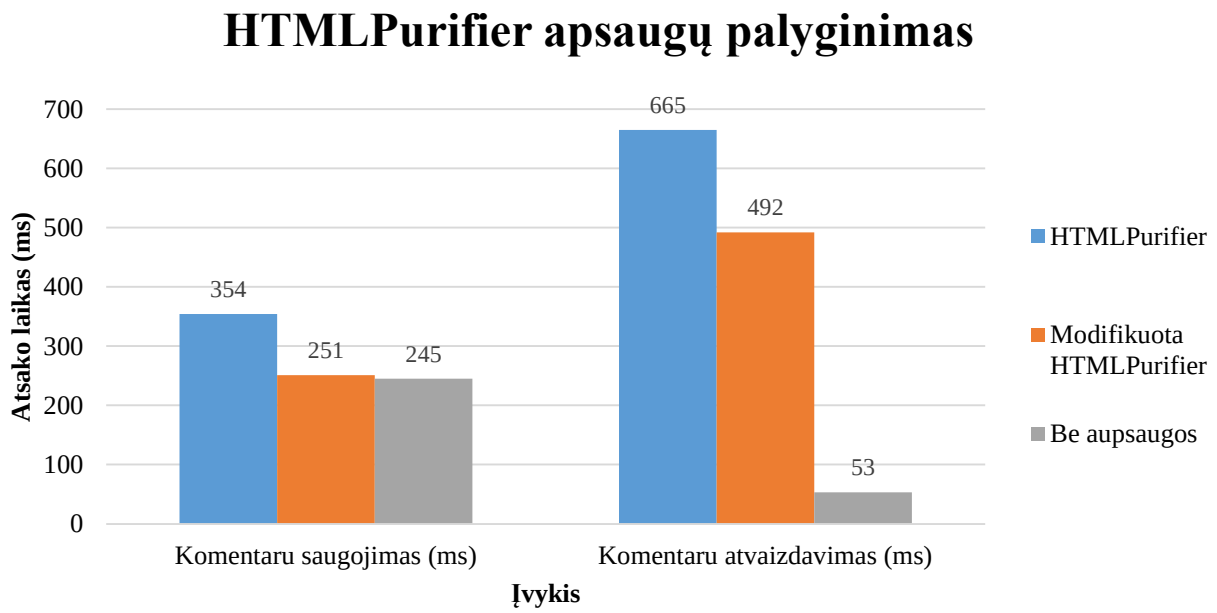
8 pav. HTMLPurifier apsauga nenurodant papildomų schemų

Visi parametrai, kurie sumažino atsako laiką, kartu pritaikyti testuojamam tinklalapiui, ženkliai sumažino atsako laiko vėlinimą. Buvo nustatyta, kad komentarų saugojimui pakoregavus nustatymus atsako laikas sumažėjo 73 ms, o atvaizdavimas buvo sumažintas dar labiau – 130 ms. Atlikus korekcijas nustatymuose, saugumo laikai nedaug nusileidžia kitų anksčiau darbe nagrinėtų filtravimo bibliotekų atsako laikams.

Kadangi filtravimo biblioteka HTMLPurifier yra atvirojo kodo ir jos išeities kodas yra visiems viešai prieinamas, kodą galima laisvai naudoti ir koreguoti. Būtent dėl šios priežasties prieš atliekant sekantį eksperimentą buvo atliekama išsami HTMLPurifier kodo analizė. Kodas analizuojamas siekiant išsiaiškinti jo veikimą ir kiek įmanoma optimizuoti jo greitaveiką. Atlikus analizę sistemoje buvo atliekami tokie kodo pakeitimai:

- klaidų pranešimų ir informacijos surinkimas ir kaupimas masyvuose;
- klasių perkėlimas į vieną failą, kad nereikėtų sistemai surinkinėti ir užkrauti daug atskirų klasių failų.

Atlikus kodo analizę buvo daroma prielaida, kad testuojamoje sistemoje galima atsisakyti tekste aptiktų klaidų sąrašo rinkimo ir saugojimo masyvuose. Kadangi šioje pavyzdinėje sistemoje nėra sukurtas ir naudojamas išsamus sistemos žurnalų pildymas, ši saugumo priemonės dalis yra nenaudojama, todėl ją galima pašalinti iš programinio išeities kodo ir taip sumažinti sistemos sunaudojamų resursų kiekį. Antrasis kodo pakeitimas susijęs su klasių įkėlimu. Sistemai kraunant daug atskirų failų tam yra sugaištama daugiau laiko, negu visas klases užkraunant iš vieno bendro failo, kuriame patalpintos visos bibliotekos naudojamos klasės. Atlikus šiuos smulkius sistemos pakeitimus buvo atliekamas eksperimentas, kurio metu siekiama dar labiau sumažinti filtravimo bibliotekos atsako laiko įtaką testavimo tinklalapiui.



9 pav. HTMLPurifier apsaugos su optimaliais parametrais ir modifikuotos filtravimo bibliotekos palyginimas

Pateiktame 9 paveiksle (9 pav.) nurodomi tinklalapio testavimo rezultatai, kurie buvo gauti atlikus eksperimentą po kodo analizės ir numatytų pakeitimų. Sistema buvo testuojama su jau anksčiau išbandytais ankstesnių eksperimentų optimaliais atsako laikui nustatymais. Taigi, po visų eksperimentų ir pakeitimų komentarų saugojimo metu tinklalapio atsako laikas buvo gautas 251 ms., o atvaizdavimo 492 ms. Grafike vaizdingumui padidinti pridėta ir atsako laikas, kuris buvo gautas testuojant tinklalapį be apsaugos.

Visi šiame skyriuje nagrinėti nustatymai ir pakeitimai pagreitina tinklalapio atsako laiką, tačiau tai nereiškia, kad juos visus galima naudoti kiekvienoje sistemoje, siekiant optimizuoti atsako laiką. HTMLPurifier yra labai funkcionali filtravimo biblioteka, tačiau parametrais išjungus atskiras

funkcionalumo dalis ar sumažinus skenavimų kiekį nukenčia sistemos saugumas. Bet koks nustatymas, kuris išjungia ar sumažina tam tikrų skenavimų kiekį, paprastai pagreitina atsako laiką, bet sumažina saugumo lygį. Kiekvienai sistemai būtinas išsamus rizikų įvertinimas ir individualus bibliotekos tinkamiausių nustatymų parinkimas.

5. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Išanalizavus saugumo problemas internete buvo pastebėta, kad XSS tipo atakos yra labai populiarios. Apsauga nuo šio tipo atakų yra labai sudėtinga ir norint užtikrinti savo tinklalapio duomenų ir jame besilankančių vartotojų saugumą būtina imtis įvairių saugumo priemonių. XSS tipo atakų yra nuolat sukuriama vis naujų ir nutaikytų į vis skirtingus tinklalapių pažeidžiamumus, todėl apsisaugoti nuo šių atakų yra ypač sunku. Literatūroje ir internete yra siūlomos bendros saugumo taisyklės, kuriomis vadovaujantis galima sumažinti tikimybę nukentėti nuo šio tipo atakų. Apsauga gali ir turi būti vykdoma ne tik tinklalapyje, bet ir vartotojo kompiuteryje.

2. Apsaugos nuo XSS atakų tyrimui pasirinktos trys vienos iš plačiausiai vartojamų internetinių sistemų kūrimo technologijų: PHP, ASP.NET ir Java. Kiekvienos technologijos tinklalapiai buvo tiriami naudojant skirtingas HTML turinio filtravimo bibliotekas.

3. Naudojantis sukurtais testavimo tinklalapius buvo testuojamos apsaugos priemonės, taikomos tinklalapiuose, ir matuojamas tinklalapių atsako laikas. Atliekami stebėjimai, kaip keičiasi atsako laikas taikant skirtingas saugumo priemones. Be atsako laiko matavimų buvo atliekamas ir kiekvienos filtravimo bibliotekos saugumo lygio testavimas tinklalapiuose.

4. Ištestavus trijų technologijų tinklalapius buvo nustatytos bibliotekos, kurios mažiausiai ir kurios daugiausiai didina tinklalapių atsako laikus. Gauti rezultatai parodė, kad mažiausiai atsako laikus didina programavimo kalbos komandos, tačiau jos negali užtikrinti pakankamo tinklalapių saugumo. Atrenkant geriausius rezultatus demonstravusias apsaugos priemones, buvo nustatyta, kad mažiausiai atsako laikus įtakoja:

PHP tinklalapyje – SafeHTMLChecker;
ASP.NET tinklalapyje - OWASP AntiSamy;
Java - OWASP Java HTML Sanitizer.

5. Ištestavus tinklalapių atsako laikus su skirtingomis priemonėmis buvo būtina išsiaiškinti, ar apsaugos priemonės, kurios veikia greičiausiai, tikrai gali užtikrinti tinkamą apsaugos lygį. Išanalizavus saugumo priemones ir išsiaiškinus jų funkcijas, buvo nustatyta, kad geriausią saugumo lygį užtikrina šios priemonės:

PHP tinklalapyje – HTMLPurifier;
ASP.NET tinklalapyje - OWASP AntiSamy;
Java - OWASP Java HTML Sanitizer.

6. Remiantis tyrimo rezultatais galima teigti, kad Java ir ASP.NET tinklalapiams pakankamą saugumo lygį užtikrina priemonės, kurios, palyginus su konkurentais, mažiausiai padidina atsako laiką ir tai yra:

ASP.NET tinklalapyje - OWASP AntiSamy;
Java - OWASP Java HTML Sanitizer.

7. PHP tinklalapio atveju pakankamą saugumo lygmenį užtikrina vienintelė biblioteka, kuri daugiausiai įtakojo tinklalapio atsako laiką: HTMLPurifier. Dėl šios priežasties buvo tęsiamas PHP tinklalapio apsaugos tyrimas, siekiant sumažinti atsako laiką. Atlikus HTMLPurifier kodo analizę ir pakoregavus jos kodą, buvo pagerintas jos atsako laikas.

8. HTMLPurifier optimizavimo metu nagrinėti nustatymai ir pakeitimai pagreitina tinklalapio atsako laiką, tačiau tai nereiškia kad juos visus galima naudoti kiekvienoje sistemoje. HTMLPurifier yra labai funkcionali filtravimo biblioteka, tačiau išjungus atskiras

funkcionalumo dalis ar sumažinus skenavimų kiekį nukenčia sistemos saugumas. Bet koks nustatymas, kuris išjungia ar sumažina tam tikrų skenavimų kiekį, paprastai pagreitins atsako laiką, bet sumažins saugumo lygį. Kiekvienai sistemai būtinas išsamus rizikų įvertinimas ir individualus bibliotekos tinkamiausių nustatymų parinkimas.

9. Pagal gautus tyrimų duomenis buvo parengtas ir publikuotas straipsnis „Tinklalapių apsaugos priemonių įtaka atsako laikui“ tarpuniversitetinėje konferencijoje „Informacinė Visuomenė ir Universitetinės Studijos“ Kaune (Data: 2013.04.25). Straipsnis pateikiamas pirmajame šio darbo priede.

6. LITERATŪRA

[1] J. Grossman, R. Hansen, P. D. Petkov, A. Rager ir S. Fogie, XSS Attacks. Cross site scripting exploits and defense, United State of America, 2007.

[2] L. K. Shar ir H. B. Kuan Tan, „Automated removal of cross site scripting vulnerabilities in web applications," Nanyang Avenue, Singapore, 2011.

[3] A. Kiezun, P. J. Guo, K. Jayaraman ir M. D. Ernst, „Automatic Creation of SQL Injection and Cross-Site Scripting Attacks," 2009.

[4] „OWASP atvirojo kodo internetinių aplikacijų saugumo tyrimo projektas. Apsaugos būdai ir priemonės nuo XSS tipo atakų.," [Tinkle]. Available: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet. [Kreiptasi 26 Vasario 2013].

[5] „Nacionalinis standartų ir technologijų institutas (angl. National Institute of Standarts and Technology - NIST) statistiniai duomenys apie aptiktas XSS atakas.," NIST, [Tinkle]. Available: http://web.nvd.nist.gov/. [Kreiptasi 26 Vasario 2013].

[6] A. E. Nunan, E. Souto, E. M. dos Santos ir E. Feitosa, „Automatic Classification of Cross-Site Scripting in Web Pages Using Document-based and URL-based Features," Manaus, Brazil.

[7] Y. Wang, Z. Li ir T. Guo, „Program Slicing Stored XSS Bugs in Web Application," Beijing, China, 2011.

[8] R. Putthacharoen ir P. Bunyatnoparat, „Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique," Technology Ladkrabang, Bankok Thailand, 2011.

[9] M. Heiderich, M. Niemietz, F. Schuster, T. Holz ir J. Schwenk, „Scriptless attacks: stealing the pie without touching the sill," įtraukta *In Proceedings of the ACM conference on Computer and communications security (CCS '12)*, 2012.

[10] A. E. Nunan, E. Souto ir E. Feitosa, „Automatic classification of cross-site scripting in web pages using document-based and URL-based features," pp. ISCC (p. 702-707). IEEE. ISBN: 978-1-4673-2712-1, 2012.

[11] A. Klein, „DOM Based Cross Site Scripting or XSS of the Third Kind," 2005. [Tinkle]. Available: http://www.webappsec.org/projects/articles/071105.shtml. [Kreiptasi 20 Balandžio 2013].

[12] P. Hope ir B. Walther, „Web Security Testing Cookbook: Systematic Techniques to Find Problems Fast," O'Reilly Media, Inc., 2008.

[13] „OWSAP ESAPI bibliotekos skirtos apsaugai nuo XSS atakų internetinis tinklalapis.," [Tinkle]. Available: https://www.owasp.org/index.php/ESAPI. [Kreiptasi 26 Vasario 2013].

[14] „Microsoft sukurto įrankio Microsoft Anti-Cross Site Scripting apsaugai nuo XSS atakų bibliotekos tinklalapis," [Tinkle]. Available: http://msdn.microsoft.com/en-us/security/aa973814.aspx. [Kreiptasi 17 Balandžio 2013].

[15] T. Scholte, W. Robertson, D. Balzarotti ir E. Kirda, „Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis," Scholte, 2012.

[16] „OWASP XSS atakų pavyzdžiai kuriamų tinklalapių testavimui," [Tinkle]. Available: https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet#Tests. [Kreiptasi 7 Balandžio 2013q].

[17] L. K. Shar ir H. B. Kuan Tan, „Defending against Cross-Site Scripting Attacks," Nanyang Avenue, Singapore, 2012.

[18] „PHP programavimo kalbos funkcijų aprašymas," [Tinkle]. Available: http://php.net/manual/en/. [Kreiptasi 17 Balandžio 2013].

[19] „HTML Purifier filtravimo bibliotekos gamintojų tinklalapis," [Tinkle]. Available: http://htmlpurifier.org/. [Kreiptasi 17 Balandžio 2013].

[20] „Svetainė, kurioje aprašyta ir talpinama htmlchecker biblioteka," [Tinkle]. Available: http://doc.b2evo.net/v-1-9/evocore/_blogs---inc---_misc---_htmlchecker.class.php.html. [Kreiptasi 17 Balandžio 2013].

[21] „htmlLawed bibliotekos kūrėjo tinklalapis,“ [Tinkle]. Available: http://www.bioinformatics.org/phplabware/internal_utilities/htmLawed/htmLawed_README.htm. [Kreiptasi 17 Balandžio 2013].

[22] „Kses kūrėjų tinklalapis,“ [Tinkle]. Available: http://kses.putnamschools.org/. [Kreiptasi 20 Vasario 2013].

[23] „.NET Framework 4.5 programavimo kalbos bibliotekų aprašymas ir dokumentacija,“ [Tinkle]. Available: http://msdn.microsoft.com/en-us/library/90d18ktz.aspx. [Kreiptasi 17 Balandžio 2013].

[24] „OWSAP AntiSamy projekto tinklalapis,“ [Tinkle]. Available: https://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project. [Kreiptasi 17 Balandžio 2013].

[25] „HTML Tidy .NET kodo tikrinimo įrankio svetainė,“ [Tinkle]. Available: http://www.w3.org/People/Raggett/tidy/. [Kreiptasi 17 Balandžio 2013].

[26] „Java programavimo kalbos bibliotekų aprašymas ir dokumentacija,“ Oracle, [Tinkle]. Available: http://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/fn/escapeXml.fn.html. [Kreiptasi 26 Vasario 2012].

7. PRIEDAI

7.1. priedas. Straipsnis: „Tinklalapių apsaugos priemonių įtaka atsako laikui“

Tinklalapių apsaugos priemonių įtaka atsako laikui

Dainius Mockus
Kompiuterių katedra
Kauno technologijos universitetas
Kaunas, Lietuva
dainius.mockus1987@gmail.com

Jonas Čeponis
Kompiuterių katedra
Kauno technologijos universitetas
Kaunas, Lietuva
Jonas.ceponis@ktu.lt

***Santrauka*—Išaugus internetinių sistemų naudojimo populiarumui, vis daugiau slaptų, asmeninių duomenų talpinama internete. Toks reiškinys tapo viena iš esminių priežasčių, lėmusių išaugusį nusikaltimų skaičių internetinėje erdvėje. Dažnai informacijos perėmimas ar kitos kenkėjiškos operacijos atliekamos įterpiant programinį kodą į vartotojo peržiūrimą tinklapį. Siekiant apsaugoti nuo šio tipo atakų, realizuojami įvairūs apsaugos mechanizmai, tačiau jie sulėtina tinklalapio atsako laiką. Straipsnyje pristatomas tinklalapių atsako laiko tyrimas taikant skirtingus apsaugos būdus.**

Reikšminiai žodžiai—XSS, tinklalapių apsauga, atsako laikas

I. ĮVADAS

Dabar, kai informacijos ir ryšių technologijos sudaro šiuolaikinės visuomenės ir ekonomikos pagrindą, tinklų ir informacijos saugumas tampa vis aktualesnis. Prieiga prie informacinių sistemų dabar yra įmanoma iš bet kur ir bet kuriuo metu. Informacijai laisvai kertant valstybių sienas, plinta ir tinklų bei informacijos saugumo problemos.

Pagrindinis saugojimo objektas yra vertinga informacija. Pašaliniam asmeniui priėjus prie šių vertybių, kyla didelis pavojus, kad su šia informacija bus atlikti nereglamentuoti veiksmai. Todėl pirmaeiliu uždaviniu visada išliks informacijos sistemų apsauga nuo nesankcionuoto jų panaudojimo, o žemesnį prioritetą turės pačios informacijos saugumo užtikrinimas šiose sistemose.

Šiandien vyraujančios ir aktualiausios grėsmės, su kuriomis susiduria informacinės sistemos: programinio kodo įterpimas į vartotojo peržiūrimą tinklapį (angl. Cross-Site Scripting XSS) [1, 4], užgrobti kompiuteriniai tinklai [1], piktnaudžiavimo įrankiai (angl. exploit tools) [1], loginės bombos [1], paketiniai šniukštinėtojai (angl. sniffer) [1], SQL injekcijos [1, 4], Trojos arkliai [1] ir kitos kenkėjiškos priemonės. Jau daug metų iš eilės nekomercinės organizacijos OWASP (angl. Open Web Application Project) kiekvienais metais sudaromame dažniausiai internete aptinkamų atakų sąraše kodo įterpimo atakos yra aukščiausiuose pozicijose. 2013 metų plačiausiai paplitusių pažeidžiamumų dešimtuke pasaulyje XSS atakos yra trečioje vietoje [3].

Internete pradėjus saugoti daug informacijos ir teikti įvairias paslaugas, pradėjo gausėti įvairiausių kenkėjiškų atakų, naudojant įvairius būdus ir priemones. Kadangi XSS tipo atakų yra nuolat sukuriama vis naujų ir nutaikytų į vis skirtingus tinklalapių pažeidžiamumus, nuo šių atakų apsisaugoti yra ypač sunku [14]. Apsauga gali ir turi būti realizuojama ne tik tinklalapyje, bet ir vartotojo kompiuteryje [6].

II. PROGRAMINIO KODO ĮTERPIMAS Į VARTOTOJO PERŽIŪRIMĄ TINKLALAPĮ

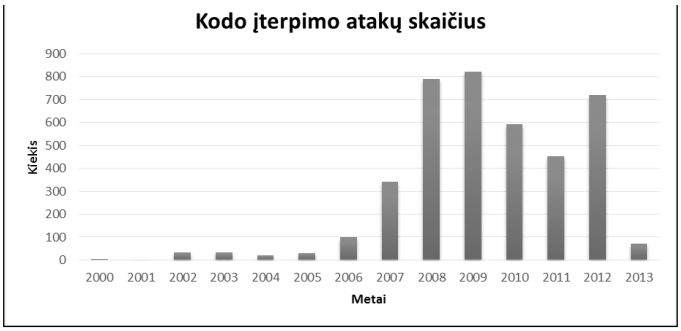
Internete veikiančios taikomosios programos tampa vis populiaresnės ir į jas keliama vis daugiau svarbios informacijos. Dažniausios atakos nukreiptos prieš internete veikiančias programas yra programinio kodo įterpimo, arba XSS tipo, atakos. Šių atakų metu bandoma įterpti į svetainę kenkėjišką kodą [1], kurį naršyklė įspraus vartotojui atsidarius svetainę. Kodo įterpimo atakos yra sunkiai sustabdomos, nes internete veikiančios programos tampa vis dinamiškesnės ir jos suteikia vartotojui vis daugiau laisvės įvairiausiems veiksmams. Kodo įterpimo atakos į tinklalapį tampa įmanomomis tuomet, kai:

1. Duomenys į sistemą yra įvedami nepatikimų šaltinių (neautorizuotų vartotojų), dažniausiai tai daroma sistemos reikalavimu.

2. Duomenys yra dinamiškai įvedinėjami vartotojų ir vėliau puslapio turinys rodomas kitiems vartotojams, neatliekant tinkamo įvedamų duomenų skenavimo. Tai gali būti komentarai, forumai, diskusijos ir pan.

XSS atakoms dažniausiai yra naudojami JavaScript programavimo kalba parašyti kodai, tačiau gali būti panaudoti ir HTML, Flash ar bet kokio kito tipo kodai, kuriuos gali įvykdyti vartotojo naršyklė [3]. Šio tipo atakos nukreiptos prieš vartotojus, o ne serverius. Užpuolikai dažniausiai renkasi patikimo turinio svetaines, kuriomis vartotojai pasitiki ir taip tampa lengviau pažeidžiami [1].

Pirmą kartą šio tipo atakos buvo pastebėtos 2000 metais ir nuo to laiko jos smarkiai išpopuliarėjo (1 pav.).



1 pav. Kodo įterpimo atakų skaičius 2000-2013 m. kovas [2]

Vis populiarėjant internetui ir vis daugiau informacijos perkeliant į sistemas internete, išpopuliarėjo ir XSS atakos [1, 3]. Pagrindinė to priežastis buvo ta, kad pirmosios šio tipo atakos nebuvo labai žalingos ir niekas nemanė, kad į šiuos pažeidžiamumus reikia atkreipti dėmesį. Atakoms vis įžūlėjant ir pridarant vis daugiau nuostolių, buvo pradėtos kurti apsaugos priemonės.

XSS atakos dažniausiai realizuojamos naudojant 3 pagrindinius būdus – nuolatinis, vienkartinis ir paremtas dokumento objektų modeliu (DOM) [1]. Nuolatinis kodo įterpimas - tai yra atakos tipas, kai kodas yra įterpiamas į puslapį, kuriame jis yra išsaugomas ir nuolat vykdomas, puslapio vartotojams peržiūrint tinklalapį. Šio tipo atakos įmanomos tik tuomet, jeigu puslapyje, prieš saugant vartotojo įvestus pranešimus ar informaciją, nėra tinkamai patikrinami įvedami duomenys [1, 5, 10]. Vienkartinės XSS atakos būdingos dinamiškai generuojamiems tinklalapių puslapiams. Ši ataka dažniausiai atliekama vartotojui pateikiant įvedimo formą, į kurią suvesti duomenys panaudojami visai kitais tikslais nei numatyta vartotojo [10]. DOM XSS pažeidžiamumas paremtas dokumento objektų modeliu. Šio tipo ataka paprastai naudojasi JavaScript [6] ar kita programavimo kalba ir keičia tinklalapio puslapio atvaizdą naršyklėje. Šis pažeidžiamumas įvyksta vartotojo kompiuteryje, o ne serveryje. Kadangi interneto naršyklės operacinėje sistemoje turi nemažai teisių, o su tokios rūšies ataka įmanoma paleisti visus procesus su tomis pačiomis teisėmis kaip ir naršyklės proceso teisės, įvykdytos atakos gali padaryti daug žalos [11].

III. APSAUGA NUO KODO ĮTERPIMO (XSS) ATAKŲ

Beveik kiekvienas dabartinis tinklalapis turi dinamiško turinio formavimo dalis [6], kuriuose rašomi komentarai, bendraujama su kitais vartotojais arba tiesiog galima atlikti paiešką tame puslapyje ir po paieškos yra parodoma paieškos frazė. Visos šios funkcijos piktavaliams vartotojams suteikia galimybę bandyti įvykdyti kodo įterpimo atakas. Tai reiškia, kad kiekviename puslapyje programuotojai turi imtis apsaugos priemonių, kuriomis tinklalapio vartotojai būtų apsaugomi nuo XSS atakų. Apsaugos būdų nuo kodo įterpimo atakų yra trijų rūšių ir jie skirstomi pagal tai, kurioje vietoje yra realizuojami: serveryje, vartotojo kompiuteryje (naršyklėje) ir hibridinėje apsaugoje [3].

Vykdant apsaugą nuo kodo įterpimo atakų, serverio pusėje dažniausiai yra diegiama programinio lygio ugniasienė, kuri atliktų visus saugumo patikrinimus, perduodant informaciją tarp vartotojo kompiuterio ir serverio. Programinio lygio ugniasienės gali tikrinti užklausas, einančias iš vartotojo kompiuterio į serverį, tikrinti serverio atsakus į vartotojo užklausas arba gali atlikti šiuos abu veiksmus vienu metu. Efektyviausią apsaugą užtikrina visos, tiek įeinančios, tiek išeinančios informacijos srauto tikrinimas, tačiau tai gali labai apkrauti serverį ir sulėtinti tinklalapio atsako laiką.

Apsauga nuo kodo įterpimo atakų gali būti vykdoma ir vartotojo kompiuteryje. Tokiai apsaugai realizuoti reikalingi vartotojo kompiuteryje įdiegti tam tikri įrankiai. Tai gali būti vartotojo naršyklėje įdiegtas JavaScript kodo tikrinimo komponentas ir įsibrovimo aptikimo programinė įranga. Taip pat kaip ir serveryje, vartotojo kompiuteryje gali būti įdiegiama ir programų lygio ugniasienė, tačiau tokių priemonių įdiegimas į kompiuterį reikalauja, kad vartotojas atliktų apsaugos priemonių įdiegimą ir konfigūravimą sistemoje.

Hibridinė XSS apsauga – tai tokia apsauga, kai serveryje yra įdiegiama tam tikra programinė įranga ir taip pat yra papildoma ar modifikuojama vartotojo kompiuteryje veikianti naršyklės apsauga. Tokiu atveju dažniausiai serveryje veikianti programinė įranga atsakinga už turinio pradinį patikrinimą ir saugumo taisyklių nustatymą, o naršyklė privalo reaguoti į nusakytas taisykles ir atlikti galutinį turinio patikrinimą.

XSS pažeidžiamumas dažniausiai atsiranda dėl nepakankamo įvedamos informacijos filtravimo [15]. Radęs būdą įterpti kenksmingą kodą, piktavalis gali gauti prieigą prie tinklapio turinio, sesijos slapukų ar kitos informacijos. Beveik visas XSS galimybes sukuria tos programos, kurios nesugeba saugiai valdyti HTML įvesties ir išvesties. Prieš bet kur ir bet kada naudojant vartotojo įvestus duomenis reikia juos patikrinti, ar jie atitinka nustatytus duomenų tipus ir struktūras. Ypač dažnai pažeidimai yra susiję su HTML žymėmis laužtiniuose skliaustuose (< ir >) ir tarp kelių kitų simbolių, kurie pateikiami pirmojoje lentelėje [7].

LENTELĖ 1. SIMBOLIAI, KURIE DAŽNIAUSIAI NAUDOJAMI XSS ATAKOSE

Simbolis	Unikodas
"	U+0022 (34)
&	U+0026 (38)
’	U+0027 (39)
<	U+003C (60)
>	U+003E (62)

Tinklalapių turinio saugumo užtikrinimui yra sukurta įvairių apsaugos priemonių. Norint visiškai apsisaugoti nuo kodo įterpimo atakų, vartotojo kompiuteryje galima tiesiog išjungti programinio kodo vykdymą, tačiau tai labai apriboja puslapių funkcionalumą, kas yra nepatogu vartotojui.

Vien vartotojo pastangų apsisaugoti nuo kodo įterpimo atakų nepakanka. Nemaža dalis saugumo problemų kyla serverio pusėje, todėl kiekvieno tinklalapio saugumą serverio pusėje privalo užtikrinti puslapio kūrėjai [7]. Kiekviename dinaminio turinio tinklalapyje privaloma naudoti išsamų vartotojo įvedamų duomenų filtravimą, taip pat tikrinti rodomą informaciją, kartais net būtina naudoti turinio šifravimą [1]. Serverio pusėje tinklalapio apsaugai programuotojai, įvertinę visas rizikas ir pavojus, privalo pasirinkti tinkamus apsaugos lygius ir būdus. Tai nėra lengvas uždavinys, nes apsaugos būdų yra siūloma labai daug. Nuo paprasčiausių programavimo kalbos komandų (pavyzdžiui Java programavimo kalboje naudojamos funkcijos: replaceAll(), escapeXml(), regex()) iki serverio pusėje API (angl. Application programming interface) principu veikiančių tinklalapio tekstų šifravimo arba kodavimo bibliotekų. Tokių bibliotekų internete gausu. Jos kuriamos įvairioms programavimo kalboms tiek mokamos, tiek ir nemokamos. Pagrindinis šio tyrimo tikslas ir bus ištirti, kurios iš populiariausių nemokamų turinio filtravimo bibliotekų pakankamai apsaugo tinklalapį ir per daug nesulėtina tinklalapio atsako laiko.

IV. TINKLALAPIO ATSAKO LAIKO TYRIMAS

Norint atlikti testavimus pirmiausia sukurtas tinklalapis testavimui - Java programavimo kalba parašytas tinklalapis (naudojamas Spring Framework 3.0.2 karkasas). Sistema veikia vietiniame kompiuteryje (localhost).

Tinklalapis turi pavyzdinę vartotojams skirtą komentavimo formą ir jau įvestų komentarų sąrašą. Komentarų forma pasirinkta todėl, kad jie aptinkami beveik kiekviename dinaminio turinio puslapyje ir yra visiems viešai prieinami, todėl dažnai tampa piktavalių taikiniu. Komentarai saugomi duomenų bazėse. Tinklalapių testavimui naudojama Google Chrome naršyklės (versija 24.0.1312.57) programuotojams skirtas įrankis. Testuojant buvo matuojamas tinklalapio atsako laikas be jokios apsaugos ir rezultatai fiksuojami. Po to tinklalapis buvo bandomas su integruotomis saugos priemonėmis: Java programavimo kalbos komandomis, kurios aptinka ir pakeičia neleistinus atvaizduojamojo HTML kodo simbolius (escapeXML, replaceAll, regex); ESAPI ir OWASP Java HTML Sanitizer kodo tikrinimo bibliotekomis.

Eksperimentų metu buvo fiksuojamas tinklalapių atsako laikas. Atsako laikas matuojamas tiek duomenis nuskaitant iš duomenų bazės ir apdorojant juos prieš parodant vartotojui ekrane, tiek ir saugant naują vartotojo įvestą komentarą, prieš tai iš jo išfiltravus duomenis su parinktomis saugumo priemonėmis. Abu eksperimentai atliekami atskirai, kiekvieną kartą suvienodinant duomenų bazėje esančios informacijos kiekį (ištrinamas po kiekvieno eksperimento išsaugotas įrašas iš duomenų bazės, jeigu tai yra įrašo išsaugojimo laiko matavimas). Informacijos kiekis duomenų bazėje turi būti vienodas, nes apdorojamos informacijos kiekis turi įtakos atsako laikui [13]. Kuo daugiau informacijos bus bandoma atvaizduoti ekrane, prieš tai ją patikrinus, tuo daugiau laiko užims informacijos tikrinimas. Informacijos atvaizdavimo bandymams buvo naudojama OWASP projekto tinklalapyje pateikiami visų dažniausiai aptinkamų tipų XSS atakų pavyzdžiai [12], kurie, iš anksto nenaudojant jokios apsaugos, surašomi į duomenų bazę ir tuomet vykdomi atidarant testavimo tinklalapį. Su kiekvienu iš apsaugos variantų testavimai atliekami po 10 kartų ir apskaičiuojamas aritmetinis laiko verčių vidurkis, kas ir bus laikomas to apsaugos būdo atsako laikas. Gautus rezultatus palyginame, taip nustatant, kurios bibliotekos ir kaip keičia tinklalapio atsako laiką.

Matuoti tik atsako laiką, kad galėtume spręsti, jog tai yra geriausia apsauga, būtų klaidinga. Bibliotekos, kurios atlieka tik neleistinų simbolių paiešką ir jų pašalinimą arba konvertavimą į leistinus HTML simbolius, visada veiks greičiau už sudėtingus tekstų filtravimus atliekančias bibliotekas, tačiau jos negali tinkamai užtikrinti saugumo tinklalapiuose. Galima lengvai nuspėti, kad programa, kurioje nenaudojama jokia papildoma biblioteka ir apsaugai panaudotos tik programavimo kalbos komandos, veiks greičiausiai. Būtent dėl visų šių priežasčių būtina įvertinti kiekvienos bibliotekos patikimumą saugumo prasme. Pats paprasčiausias būdas patikrinti bibliotekos patikimumą yra apžvelgti jos funkcionalumą ir taip bus galima įvertinti, nuo kurio tipo atakų padeda apsisaugoti analizuojama biblioteka.

Java programavimo kalba parašytas tinklalapis buvo testuojamas su šiomis tinklalapio kodo filtravimo priemonėmis:

Nenaudojant jokio filtravimo.

Filtruojant Java programavimo kalbos komandomis (escapeXml, replaceAll, regex) [9].

ESAPI [5].

OWASP Java HTML Sanitizer – OWASP projekto sukurtas HTML kodo analizatorius, parašytas Java programavimo kalba [3].

Testavimui buvo pasirinktos OWASP projekto kuriamos ir plėtojamos bibliotekos. Šios bibliotekos kuriamos ir patariamos naudoti OWASP projekto dalyvių ir kūrėjų, kurie aktyviai dalyvauja kovojant su kodo įterpimo atakomis [3, 5].

Apsauga realizuota programavimo kalbos komandomis escapeXml, replaceAll ir regex į eksperimentą yra įtraukta siekiant pademonstruoti, kad nors vien paprastais metodais realizuota minimali apsauga mažai keičia atsako laiką, tačiau ji nėra pakankama.

Testavimo metu matavimai atliekami nuosekliai, vienas po kito. Atliekant testavimus, kompiuteris papildomų uždavinių neatliko. Testavimui naudojamo kompiuterio aparatūrinės ir programinės įrangos sąrašai pateikti žemiau esančiose lentelėse.

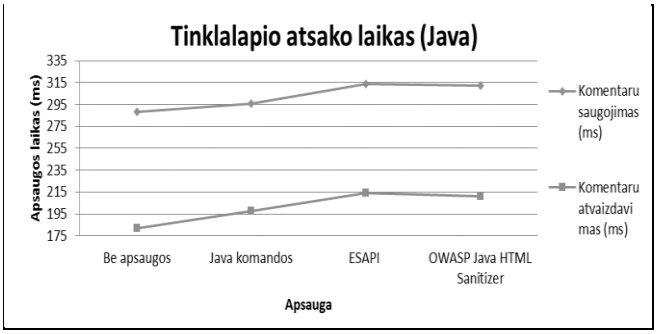
LENTELĖ 2. TESTAVIMUI NAUDOJAMO KOMPIUTERIO APARATINĖ ĮRANGA

Procesorius:	Intel(R) Core(TM) i7-2600 3,4 GHz
Spartinančioji atmintis:	DDR3 8,00 GB
Vaizdo plokštė:	NVIDIA GeForce GTX 560
Kietasis diskas:	SAMSUNG SATA3, Talpa: 1TB

LENTELĖ 3. TESTAVIMUI NAUDOJAMO KOMPIUTERIO PROGRAMINĖ ĮRANGA

Operacinė sistema:	Windows 7 Professional 64-bit
Interneto naršyklė:	Google Chrome 24.0.1312.57 versija
Antivirusinė programa:	(nėra)
Serverio programinė įranga:	Apache Tomcat v7.0 (paleidžiamas ir veikiantis kartu su Eclipse Java EE IDE for Web Developers. Versija: Indigo Service Release 2) WampServer Version 2.2 pakete naudojama duomenų bazė: MySQL 5.5.24 projekte naudojamas Spring Framework 3.0.2 karkasas

Atliekant testavimus atsako laikas buvo matuojamas, naudojantis Google Chrome naršyklėje integruotomis tinklalapių programuotojų priemonėmis. Matavimai buvo atliekami nuosekliai, vienas po kito. Testavimo metu kompiuteris papildomų uždavinių neatliko, kad testavimo rezultatai nebūtų vėlinami pašalinių veiksnių. Atlikus atsako laiko testavimus su visomis anksčiau išvardintomis priemonėmis, buvo gauti rezultatai, kurie pateikiami 2 paveiksle.



2 pav. Tinklalapio atsako laiko matavimų rezultatai Java tinklalapyje

Iš gautų rezultatų galima pastebėti, kad tinklalapio atsako laiką labiausia didino ESAPI biblioteka, o mažiausiai – programavimo kalbos komandos. Iš nagrinėtų saugumo bibliotekų greičiausiai veikė OWASP Java HTML Sanitizer.

Gavus atsako laikų testavimo rezultatus, būtina įvertinti bibliotekų saugumo užtikrinimo lygius. Pagal saugumo funkcionalumą sudaryta ketvirta lentelė, iš kurios bus galima įvertinti priemonių užtikrinamą saugumo lygmenį.

LENTELĖ 4. JAVA PROGRAMAVIMO KALBOS HTML KODO TIKRINIMO BIBLIOTEKŲ FUNKCIONALUMO ĮVERTINIMAS

Galimi pažeidimai / Biblioteka	Koduotės simbolių filtravimas	HTML kodo formatavimas (užtikrinimas, kad žymos uždarytos ir pan.)	Atributų filtravimas	Kodo įterpimo skenavimas
Java komandos	Dalinai	Ne	Ne	Dalinai
ESAPI	Taip	Ne	Taip	Taip
OWASP Java HTML Sanitizer	Taip	Taip	Taip	Taip

Pateiktoje Java HTML kodo filtravimo priemonių saugumo įvertinimo lentelėje matome, kad geriausią saugumo lygmenį užtikrina OWASP Java HTML Sanitizer. Ši biblioteka taip pat yra geriausia ir atsako laiko prasme, atmetant Java programavimo kalbos komandas, kurios neužtikrina patikimo saugumo lygio. Pagal gautus rezultatus galima teigti, kad Java programavimo kalba kuriamuose tinklalapiuose geriausia naudoti OWASP projekto sukurtą priemonę OWASP Java HTML Sanitizer.

V. IŠVADOS

Tyrimo metu buvo sukurtas Java programavimo technologijas naudojantis tinklalapis. Naudojantis šiuo tinklalapiu, ištestuotos pasirinktos apsaugos priemonės ir išmatuotas tinklalapio atsako laikas. Buvo stebima, kaip keičiasi atsako laikas taikant skirtingas saugumo priemones sukurtame tinklalapyje.

Testavimo rezultatai parodė, kad mažiausiai atsako laikus didina programavimo kalbos komandos, tačiau jos negali užtikrinti pakankamo tinklalapių saugumo. Atsirenkant geriausius rezultatus demonstravusias apsaugos priemones buvo nustatyta, kad mažiausiai atsako laikus įtakoja: OWASP Java HTML Sanitizer.

Ištestavus tinklalapių atsako laikus su skirtingomis priemonėmis, buvo būtina išsiaiškinti, ar tos apsaugos priemonės, kurios veikia greičiausiai, tikrai gali užtikrinti tinkamą apsaugos lygį. Išanalizavus saugumo priemones, išsiaiškinus jų funkcijas, buvo nustatyta, kad geriausią saugumo lygį užtikrina OWASP Java HTML Sanitizer.

Pagal turimus tyrimo rezultatus galima teigti, kad Java tinklalapiams pakankamą saugumo lygį užtikrinanti priemonė, kuri, palyginus su kitomis nagrinėtomis saugumo priemonėmis, mažiausiai padidina atsako laiką - OWASP Java HTML Sanitizer.

Tolimesniuose tyrimuose numatoma atlikti XSS atakų apsaugos priemonių tyrimą tinklalapiuose, sukurtuose naudojant PHP ir ASP.NET programavimo technologijas.

VI. LITERATŪRA

[1] J. Grossman, R. Hansen, P. D. Petkov, A. Rager, S. Fogie, XSS Attacks. Cross site scripting exploits and defense, United State of America, 2007.

[2] Nacionalinis standartų ir technologijų instituto (angl. National Institute of Standarts and Technology - NIST) statistiniai duomenys apie aptiktas XSS atakas. Paskutinis apsilankymas: 2013.02.26. http://nvd.nist.gov/.

[3] OWASP atvirojo kodo internetinių aplikacijų saugumo tyrimo projektas. Apsaugos būdai ir priemonės nuo XSS tipo atakų. Dažniausiai pasitaikančių atakų reitingavimas ir aprašymas. Paskutinis apsilankymas: 2013.02.26. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

[4] Adam Kiezun, Philip J. Guo, Karthick Jayaraman, Michael D. Ernst, Automatic Creation of SQL Injection and Cross-Site Scripting Attacks, 2009.

[5] Yi Wang, Zhoujun Li, Tao Guo, Program Slicing Stored XSS Bugs in Web Application, Beijing, China, 2011.

[6] E. Nunan, E. Souto, E. Feitosa. Automatic classification of cross-site scripting in web pages using document-based and URL-based features. ISCC (p. 702-707). IEEE. ISBN: 978-1-4673-2712-1, 2012.

[7] Lwin Khin Shar, Hee Beng Kuan Tan, Automated removal of cross site scripting vulnerabilities in web applications, Nanyang Avenue, Singapore, 2011.

[8] OWSAP ESAPI bibliotekos, skirtos apsaugai nuo XSS atakų, internetinis tinklalapis. Paskutinis apsilankymas: 2013.02.26. https://www.owasp.org/index.php/ESAPI

[9] Java programavimo kalbos bibliotekų aprašymas ir dokumentacija. Paskutinis apsilankymas: 2012.02.26. http://docs.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/fn/escapeXml.fn.html

[10] R. Putthacharoen, P. Bunyatnoparat, Protecting Cookies from Cross Site Script Attacks Using Dynamic Cookies Rewriting Technique, Technology Ladkrabang, Bankok Thailand, 2011.

[11] Klein, DOM Based Cross Site Scripting or XSS of the Third Kind, 2005. Straipsnis internete: http://www.webappsec.org/projects/articles/071105.shtml.

[12] OWASP XSS atakų pavyzdžiai kuriamų tinklalapių testavimui. Paskutinis apsilankymas: 2013.04.07. https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet#Tests

[13] Lwin Khin Shar, Hee Beng Kuan Tan, Defending against Cross-Site Scripting Attacks, Nanyang Avenue, Singapore, 2012.

[14] Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, Jörg Schwenk. Scriptless attacks: stealing the pie without touching the sill. In Proceedings of the ACM conference on Computer and communications security (CCS '12). 2012.

[15] Theodoor Scholte, William Robertson, Davide Balzarotti, Engin Kirda. Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis. In Proceedings of the IEEE 36th Annual Computer Software and Applications Conference (COMPSAC '12). 2012.