KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Rokas Paškevičius

Žiniatinklio programų ugniasienės sudarymas ir tyrimas

WEB application firewall development and research

Magistro darbas

Darbo vadovas
Doc. Dr. A. Venčkauskas

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Rokas Paškevičius

Žiniatinklio programų ugniasienės sudarymas ir tyrimas

Magistro darbas

Recenzentas
Dr. I. Lagzdinytė
2010-05-

Vadovas
doc. dr. A.Venčkauskas
2010-05-

Atliko
IFN-8/3 gr. stud.
Rokas Paškevičius
20010-05-26

Kaunas, 2010

TURINYS

SUMMARY

The purpose of this work is to create and analyze Web application firewall. Nowadays via the Internet it is very easy to access applications and databases. Standard firewalls are designed to restrict access to certain ports, or services. Standard network firewalls are unable to protect against Web application attacks.

A Web application attack is the new type of threat. Web application firewall has to provide inbound and outbound access control of Web application. Before creating Web application firewall model it was important to analyze existing threats, protection methods and working solutions.

The majority of protection methods and solutions are based on Open Web Application Security Project (OWASP). OWASP Top Ten Most Critical Web Application Vulnerabilities – a high-level document to help focus on the most critical issues. Project offers many solutions how to avoid potential threats.

In this work was designed and tested Web embedded application firewall. The aim of creating Web application firewall model is to improve accuracy, performance and management of existing Web applications firewalls. Created Web application firewall model consist of modules, it helps to deploy each module separately.

1. ĮVADAS

Per pastarąjį dešimtmetį internete išsiplėtojo paslaugų teikimo Internetu sfera. Įmonės greitai suprato, kad nėra kito tokio lankstaus būdo bendrauti su klientais kaip Internetas. Pradėta diegti milijonai naujų sistemų užtikrinančių vis daugiau funkcionalumo. Pagal pasaulinę Interneto vartotojų statistiką [1] 2008 m. pasaulyje buvo 1.463 mlrd. interneto vartotojų. Internete ištisą parą vyksta prekyba valiutomis, akcijomis, veikia įvairūs aukcionai, elektroninės parduotuvės, kai kuriuose šalyse vyksta balsavimai Internetu. Kuriasi įvairūs socialiniai tinklai, atsiranda virtualių pinigų ir darbo sąvokos. Visur naudojamos žiniatinklio taikomosios programos, be kurių šie procesai negalėtų vykti. Internetas tapo interaktyvus. Interaktyvumą užtrikina tarnybinėse stotyse veikianti programinė įranga. Vartotojui paprastai tereikia turėti naršyklę, priklausomai nuo žiniatinklio programos realizavimo ypatumų, gali reikėti ir papildomų įskiepių (pvz. *Flash*) ir interneto prieigą.

Be to, kad reikia užtikrinti funkcionalumą yra ir kitas labai svarbus faktorius - informacijos saugumas. Žiniatinklio programos operuoja įvairaus lygio duomenimis, jie gali būti vieši, skirti uždaram vartojimui arba ypatingai slapti ir konfidencialūs. Kiekvienas programos vartotojas turi tam tikrą jautrią informaciją, kuri leidžia jam save identifikuoti. Pavogus, ar kitaip pasisavinus ir tinkamai pasinaudojus šia informacija galima perimti kontrolę nuo viso to žmogaus turto internete, tai tas pats kas gauti raktą nuo namų.

Duomenų bazės, kuriuose saugoma įvairi informacija taip pat tapo prieinamos internetiniais kanalais. Aptikus spragų programose, kurios atsakingos už duomenų pateikimą, galima prieiti prie duomenų bazių. Kiekviena informacija turi savo kainą. Gali būti pažeistas informacijos vientisumas arba konfidencialumas. Esant nesaugiai žiniatinklio programai, saugumo užtikrinimas tarnybinėje stotyje, ar vidaus politikoje nepagelbės.

Tapo būtina užtikrinti naudojamų Žiniatinklio programų saugumą. Žinoma, kad absoliutaus saugumo nėra, bet yra saugumas, kuris yra brangesnis nei jo apėjimas. Saugumas turėtų būti visapusiškas, ne tik iš programavimo pusės, bet ir iš dirbančiųjų saugos politikos.

Šiame darbe nagrinėjama taikomosios žiniatinklio programos aplinka ir pati programa iš saugumo pusės, kurią sudaro sąsajos su duomenų bazėmis, tarnybine stotimi ir galiniu vartotoju. Kuriant sistemas klaidų gali būti neišvengta visose grandyse. Prieš kuriant žiniatinklio programą svarbu nustatyti, koks saugumo lygis jai būtinas. Šiandienos programų kūrime daug maišaties. Nors saugumą užtikrinančios priemonės ir randasi, jos vis dar labai jaunos ir tobulintinos , o vadovų požiūris „Mums tas nenutiks“ vis dar nepasikeitė.

Darbo tikslas – sukurti ir ištirti taikomųjų žiniatinklio programų ugniasienės modelį. Norint sukurti ugniasienę reikia žinoti su kokiomis grėsmėmis ir kaip ji kovos. Darbe išnagrinėtos

aktualiausios grėsmės remiantis naujausiais duomenimis. Išanalizuoti laiko patikrinti ir patys naujausi metodai kovoti su žiniatinklio programų grėsmėmis. Atlikta auditui skirtų programų bei pačių ugniasienių analizė.

Įvertinus surinktą informaciją suprojektuota žiniatinklio programų ugniasienė. Žiniatinklio ugniasienė sudaryta iš atskirų modulių, kurie turi atskirus privalumus. Žiniatinklio programų spragų išnaudojimo atvejai tampa vis modernėsi, įsilaužimo priemonės nuolatos tobulinamos, todėl turi būti tobulinami ir apsaugos metodai. Realizuotas ugniasienės modulis, skirtas apsaugoti nuo SQL injekcijų, atliktas eksperimentas atskleidžiantis modulio teigiamas savybes. Remiantis sukurtu žiniatinklio ugniasienės projektu galima kurti realiai veikiančią žiniatinklio taikomųjų programų ugniasienę.

2. ŽINIATINKLIO PROGRAMŲ SAUGUMO ANALIZĖ

2.1 Tyrimo sritis objektas ir problema

Visas pasaulis migruoja į Internetą. Kompanijos priverstos atidaryti savo sistemų prieigas partneriams ir klientams. Paslaugų spektras teikiamas 80 prievadu išsiplėtė. Klasikinės ugniasienių architektūros šiandienos situacijose retai pagelbėja.

Kuriant programas veiksiančias Internete pirmiausia žiūrima į jų funkcionalumą, o saugumas lieka antraeiliu ar trečiaeiliu dalyku. Dėl funkcionalumo trūkumo galima nesulaukti vartotojų dėmesio, bet dėl saugumo spragų galima netekti reputacijos, sulaukti teisinės atsakomybės bei patirti realių finansinių nuostolių. Šiame darbe tiriamos saugumo grėsmės susijusios su žiniatinklio taikomosiomis programomis, bei saugumą užtikrinančios priemonės: metodai taikomi spragoms pašalinti bei aptikti, taip pat taikomųjų žiniatinklio programų ugniasienės. Nebus nagrinėjama tarnybinių stočių fizinė sauga ar saugumą užtikrinančių priemonių realizavimas jose. Taip pat nenagrinėjamos aparatūrinės apsaugos priemonės. Dėmesys koncentruojamas į pačią žiniatinklio programą, jos veikimo specifiką, vartotojo tyčinių ar netyčinių veiksmų įtaką, operuojamos informacijos klasifikaciją. Verta pažymėti, kad kai kurias žinomas saugumo spragas verčiau užtaisyti programiniame kode nei bandyti nuo jų apsisaugoti naudojant kokią nors papildomą programinę įrangą.

Saugios žiniatinklio programos kūrimo tikslai

Saugumas turi būti užtikrintas norint pelnyti vartotojo pasitikėjimą. Vartotojas nesinaudos elektronine bankininkystės sistema jeigu žinos, kad yra bent menkiausia galimybė pažeisti jo informacijos konfidencialumą ar vientisumą. Taip pat retas, kuris patikės savo kreditinės

kortelės duomenis elektroninei parduotuvei nematydamas tam tikrų saugumą garantuojančių požymių (pvz. Saugumo sertifikatų).

2.2 Žiniatinklio programos analizė

Norint žinoti, kokias apsaugos priemones taikyti ir kokią metodiką naudoti kuriant žiniatinklio programą, naudinga jas išanalizuoti ir suklasifikuoti. Reiktų vengti išleisti milijoną saugumui tam, kad apsaugoti vieną litą.

Žiniatinklių programų klasifikavimas

Webappsec straipsnyje *The Importance of Application Classification in Secure Application Development [2]* siūlo programas klasifikuoti pagal konfidencialumo, integralumo ir pasiekiamumo lygio svarbą. Lygis gali būti žemas, vidutinis ir aukštas.

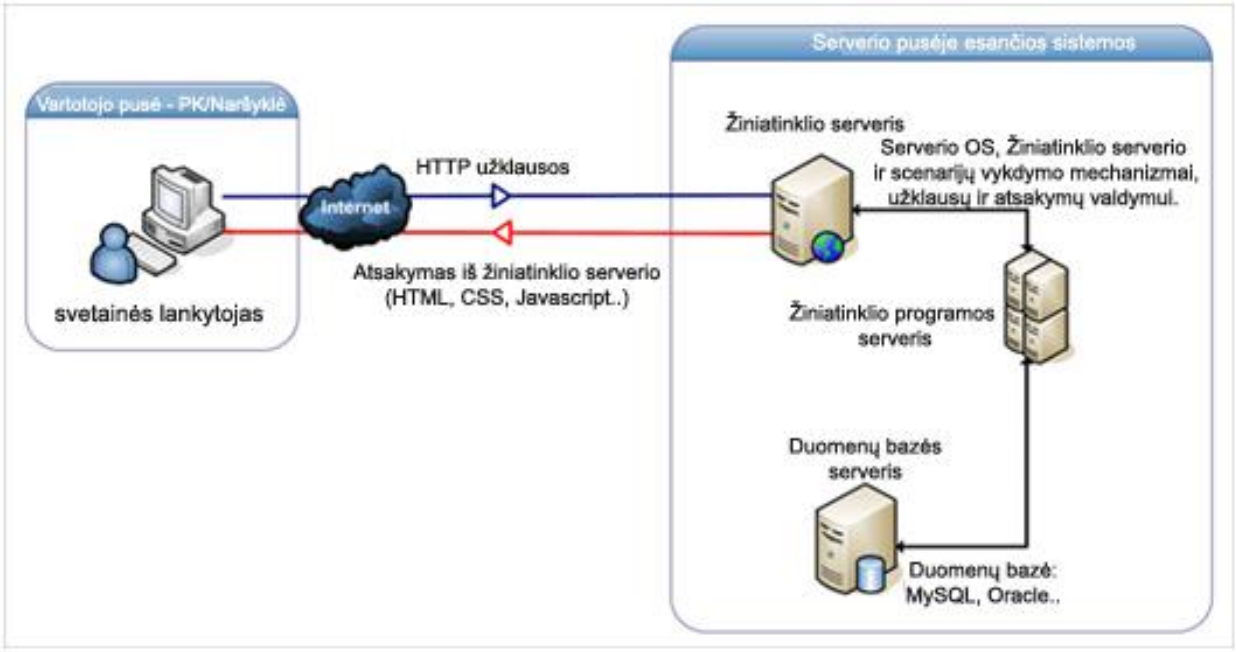
Konfidencialumas reiškia tai, kad dedant, keičiant ar siunčiant informaciją sistemoje ar tinkle, ji nebus nuskaityta ar pasisavinta kitų asmenų.

Integralumas reiškia tai, kad dedant, keičiant arba siunčiant informaciją sistemoje arba tinkle, ji nebus pakeista (modifikuota) kitų asmenų. Pavyzdžiui, siunčiant tam tikrus svarbius duomenis svarbu, kad jie nebūtų pakeisti ar iškraipyti.

Pasiekiamumas reiškia, svarbą prie išteklių priėjimo. Pavyzdžiui, naujienų portalui svarbu, kad programa pateiktų informaciją.

Žiniatinklio programos gali būti taikomos įvairiose srityse, kažkurios programos gali neoperuoti slaptais duomenimis, tačiau svarbus jų pasiekiamumas, pvz.: žemėlapių svetainės. Duomenų integralumas svarbus daugumoje žiniatinklio programų, kadangi niekas nenori kad informacija būtų pakeista ar pašalinta neautorizuotų asmenų.

1 paveiksle pateikiamas supaprastintas programos veikimo modelis remiantis saugumo kompanija **Acunetix [3]**



1 pav. Žiniatinklio programos veikimo modelis

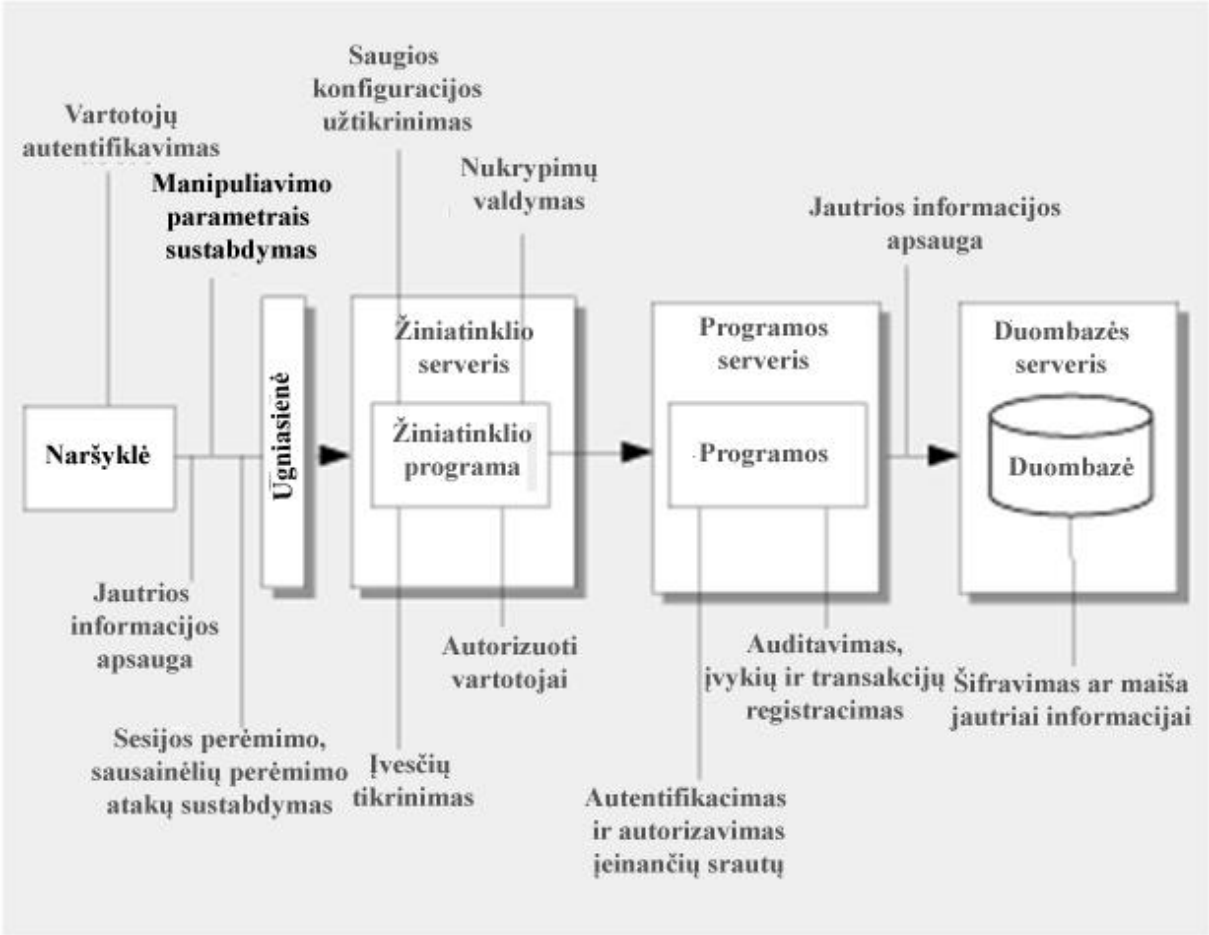
Vartotojo pusėje yra kompiuteris su interneto prieiga ir veikiančia naršykle. Naršyklės pagalba kreipiamasi tam tikru URL į pasirinktą sistemą. Tarnybinės stotys įvykdo susijungimą. Reikia pažymėti, kad žiniatinklio programa veikia ne vartotojo pusėje, bet tarnybinės stoties. Tarnybinė stotis perduoda į naršyklę sąsajos duomenis, naršyklė veikia kaip HTML, CSS, JavaScript ir t.t. interpretatorius. Vartotojas gali operuoti duomenimis, gali užklausti tam tikrų duomenų iš duomenų bazės, kuriuos žiniatinklio programa paims ir jam pateiks. Taip pat duomenys gali būti sukuriami ir naujai įtraukiami į duomenų bazę.

Taigi, žiniatinklio programa atlieka sąsajos funkciją tarp vartotojo ir duomenų bazės, kuri gali teikti įvairialypes paslaugas.

Tarnybinės stotys, arba serveriai, (angl. web servers) atlieka kompiliatoriaus ir internetinės grandies vaidmenį. Programos gali būti realizuotos įvairiomis programavimo kalbomis (PHP, ASP, JAVA). Priklausomai nuo pasirinktos technologijos turi būti sukonfigūruotos ir tarnybinės stotys.

Saugios žiniatinklio programos kūrimo metodologija

Microsoft straipsnyje *Architecture and Design Issues for Web Applications* [4] pažymimi saugios žiniatinklio programos architektūros aspektai.



2 pav. programos saugumo aspektai

Iš 2 pav. matyti, kad saugumo elementai turi būti realizuoti visuose segmentuose. Programa turi autentifikuoti ir autorizuoti kiekvieną identifikuotą dalyvį. Atlikti auditą bei registruoti veiksmus bei transakcijas.

Saugios žiniatinklio programos ypatumai:

- Atsparumas žinomoms saugumo spragoms;
- Įeinančių duomenų tikrinimas, nepasitikėti duomenimis iš vartotojo sąsajos, vartotojas gali apeiti įvesties filtravimo mechanizmus, jeigu jie veikia jo naršyklėje;
- Visų vartotojų autentifikavimas, nepasitikėjimas niekuo;
- Vartotojų elgsenos tikrinimas;
- Apsaugojimas išeinančios jautrios informacijos (saugus perdavimas ją apsaugantiems elementams).

Kas turi būti daroma:

- Stebėjimas – reikia nuolatos žinoti kas vyksta, nepalikti sukurtos programos tarsi į gatvę išmesto vaiko;

- Atakų aptikimas – žinoti kada žiniatinklio programą bandoma pažeisti;
- Atakų sustabdymas – atakos turi būti sustabdomos greičiau, negu kad joms pavyktų pasiekti tikslą;
- Vertinimas – sužinoti apie pažeidžiamumus greičiau, negu tą padaro galimi piktavaliai.

Žiniatinklio programų saugumo grėsmių klasifikavimas

Grėsmės saugumo programai gali būti skirstomos į dvi klases objektyviosios ir dirbtinės grėsmės.

Objektyviosios dažniausiai nenugalimos jėgos (*Force Majeure*) veiksniai, juos taip pat galima minimizuoti, bet tai daugiau fizinės priemonės. Nuo žmogaus šios grėsmės paprastai tiesiogiai nepriklauso.

Dirbtinės – tai tyčinės arba netyčinės veikos sukeltos grėsmės, kurios vienaip ar kitaip įtakoja normalų programos veikimo procesą.

Pagrindinių netyčinių grėsmių pavyzdžiai:

- Naudingos informacijos pašalinimas;
- Informacijos atskleidimas;
- Įvesties klaidos;
- Netinkamas darbo su programa užbaigimas.

Pagrindinės tyčinės (apgalvotos) grėsmės (SANS institutas [5]) žiniatinklio programose:

- Bandymas pasinaudoti SQL injekcijomis norint išgauti duomenis;
- Nesankcionuotas prisijungimas prie programos apeinant autentifikaciją;
- Informacijos sunaikinimas, suklastojimas ar pažeidimas;
- Pačios programos konfigūracinių failų pažeidimas.

Žiniatinklio programų vartotojų analizė

Žiniatinklio programos palengvina ne tik informacijos gavimą bet ir pakeitimą. Paprastai programos turi keletą vartotojų lygių, kurie autentifikuojami naudojant prisijungimo vardą ir slaptažodį. Priklausomai nuo to prisijungus suteikiamos atitinkamos teisės prie tam tikro lygio sistemos funkcionalumo. Rekomenduojama, kad lygių būtų kuo mažiau, ir kuo mažiau būtų vartotojų su perteklinėmis teisėmis.

Vartotojų tipai skirstomi į dvi pagrindines grupes administracija ir ribotos prieigos vartotojai. Administracija dirba su informacija, paprastai suvedinėja, redaguoja daro atsargines kopijas jeigu tai įmanoma vartotojo sąsajos pagalba. Administratoriai tvarko ir eilinius sistemoje esančius vartotojus. Pvz. E-parduotuvės atveju, administracija suvedinėja prekes, administruoja užsakymus taip pat palaiko atgalinį ryšį su pirkėjais.
Ribotos prieigos vartotojai yra tie žmonės, kuriems programa sukurta, jie turi gauti iš jos kažkokią naudą. Jeigu tai pvz. E-parduotuvė tai toks vartotojas yra pirkėjas, arba eilinis lankytojas, kuris naršo prekių katalogą.

Vartotojų tikslai ir problemos

Abi vartotojų grupės siekia gauti asmeninę naudą. Administracija suinteresuota informacijos saugumu, pateikimo efektyvumu, nepertraukiamu darbu. Administracija turi atidžiai saugotis netyčinių klaidų, nes jos gali įtakoti programų darbą, ir vėliau padaryti žalos. Pvz.: Netyčinis pirkėjo užsakymo ištrynimas, ar didelės prekių grupės pašalinimas.
Ribotos prieigos vartotojams visuomet svarbu, kad duomenys, kuriuos jie palieka sistemoje, būtų konfidencialūs. Kad negalėtų pašaliniai asmenys peržvelgti jų asmeninių failų (pvz. nuotraukų). Neretai vartotojai naudoja tą patį prisijungimo vardą ir slaptažodį keliose sistemose, todėl praradus vienoje vietoje galima susidurti su rūpesčiais ir kitur.
Programai svarbu, kad visi veiksmai būtų įrašyti. Svarbu fiksuoti įvykius tam, kad iškilus neaiškumams būtų galima sužinoti kas, kaip, ir kodėl buvo atlikta ir dėl ko sutriko žiniatinklio programos darbas. Programuotojai kurdami sistemą turi žiūrėti kritiškai į visus vartotojus nepriklausomai nuo jų lygio. Nepasitikėti pateikiamais duomenimis, nors jie ir buvo filtruojami grafinėje vartotojo sąsajoje, kadangi ten jas galima apeiti.
OWASP [6] rekomenduoja, kad privilegijuotų vartotojų būtų kuo mažiau, jeigu sistemos vartotojui nebūtinos administratoriaus teisės dirbti savo darbą, jis ir neturėtų jų turėti. Darbuotojai dirbantys su žiniatinklio programa turi žinoti, ką gali ir ko negali padaryti su sistema.

Žiniatinklio programų operuojamų duomenų analizė ir klasifikavimas

Paprastai visi tekstinio tipo duomenys saugomi duomenų bazėse (pvz.: MySQL). Nuotraukos, failai ir kitokia informacija fiziškai saugoma duomenų tarnybinėse stotyse, o duomenų bazėse indeksuojami jų vardai, tam kad būtų galima žinoti kur juos rasti.
Stanford‘o Universitas[7] pateikia populiarų duomenų klasifikavimo atvejį:

- Vieši – žemėlapiai, prekių katalogai, kainos - prieinami visiems be apribojimų arba su minimaliais;
- Vidinio vartojimo – failai skirti lokaliam naudojimui (pvz. įmonės dokumentai)
- Konfidencialūs – ypatingai jautri informacija (pvz.: vartotojų slaptažodžiai).

Priklausomai nuo informacijos klasės programa taiko atitinkamas saugumo priemones. Tačiau turi stebėti ar vartotojas imituojantis bandymą naudotis viešais duomenimis, bando galbūt ir ne visai sėkmingai gauti kitos klasės informaciją. Vartotojų veiksmams stebėti turi būti naudojamos atitinkamos stebėjimo priemonės. Svarbu kad nesėkmingų bandymų gauti tam tikrą informaciją skaičius būtų ribotas.

2.3 Žiniatinklio programoms kylančių grėsmių analizė

Nekomercinė organizacija OWASP [6] (angl. Open Web Application Security Project) yra sudariusi dešimties populiariausių žiniatinklio taikomųjų programų saugumo grėsmių dešimtuką [8]. Sąrašas sudaromas ir atnaujinamas reguliariai (2004 m., 2007 m., 2010 m.).
Didžiausias dėmesys šiame darbe skiriamas 2010 m. sudarytam grėsmių sąrašui, todėl pirmiausia pateikiamos populiariausios 2010 m. sąrašo grėsmės. Galiausiai aprašytos tos grėsmės, kurios šiemet į dešimtuką nebepapuolė, jų naudojimo atvejų ženkliai sumažėjo arba populiarumas per mažas. Nors kažkurios gerai žinomos grėsmės čia nebus paminėtos būtina žinoti apie papildomų grėsmių egzistavimą.

Injekcijos

Injekcijų grėsmė kyla, kai žiniatinklio programų naudotojai siunčia nepatikimą informaciją į programą įskaitant išorinius bei vidinius vartotojus ir administratorius.
Autorių Hope Paco ir Walther Ben knygoje apie žiniatinklio saugumą [9] įvardijamos klaidingos programų kūrėjų prielaidos, dėl kurių sudaromos sąlygos sėkmingai įvykdyti injekcijas:

Programuotojų tikimasi kad:

- įvestyje niekada nepaklius meta simboliai;
- bus įvedami pvz. tik skaitmenys;
- įvestis niekuomet neviršys tam tikro ilgio;

- skaitmenys niekuomet nebus didesni ar bent jau prilygs aukščiausiai arba žemiausiai ribai;
- vartotojai nepastebės ir negalės keisti paslėptų laukų kurie skirti programai ar tarnybinei stočiai;
- nenutiks nieko bloga ir galima paimti rodykles ar masyvų indeksus iš įvesties;

SQL injekcijos

SQL injekcijos yra populiariausia injekcijų rūšis, jos gali būti įvykdomos tose programose, kurios sąveikauja su duomenų bazėmis, tokių programų internete yra absoliuti dauguma [10]. Spragos lengvai aptinkamos ir išnaudojamos. Dažniausiai šios atakos įvykdomos, kuomet įvedami papildomi simboliai į duomenų įvesties formų laukus. Iš laukų paimamų duomenų sukonstruojama SQL užklausa į duomenų bazę. Ši ataka nepriklauso nuo platformos, kurioje veikia žiniatinklio programa.

SQL injekcijos įvyksta kai:

1. Duomenys patenka į programą iš nepatikimų šaltinių;
2. Duomenys panaudojami dinamiškai konstruojant SQL užklausas.

Pagrindinės SQL injekcijų pasekmės:

Informacijos konfidencialumas: SQL duomenų bazėse laikoma jautri informacija, todėl kyla grėsmė jos atskleidimui;

Autentifikacija: Jeigu prastos SQL komandos naudojamos tikrinti vartotojų vardams ir slaptažodžiams, sudaroma galimybė prisijungti prie sistemos kito vartotojo vardu;

Informacijos vientisumas: Jeigu sudaroma galimybė jautriai informacijai nuskaityti taip pat atsiranda galimybė informaciją modifikuoti arba išvis ištrinti su viena iš SQL injekcijų ataka. SQL injekcijų atakos taip pat žinomos kaip SQL įterpimo atakos [11].

SQL atakos vykdymo pavyzdys

Pavyzdžiui, kviečiant užklausą http://www.auka.lt/VardotojoDuomenys.php?id=119 suformuojama SQL užklausa:

*SELECT * FROM vartotojai WHERE id=119*

PHP programavimo kalboje užklausa:

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

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

http://www.auka.lt/VardotojoDuomenys.php?id=119 OR 1=1 - bus sugeneruota užklausa, kuri grąžins visus įrašus iš duomenų lentelės „vartotojai“.

XSS

XSS (angl. *Cross Site Scripting*) atakos yra viena iš injekcijų rūšių. Tai viena ir dažniausiai kylančių grėsmių šiuolaikinėse žiniatinklio programose. XSS išnaudoja trūkumą, kuomet programa priima piktavalio įrašomą programinį kodą ir nesugeba tinkamai jo apdoroti ir nukenksminti. Pasinaudodamas XSS piktavalis gali perimti sesijas, autentifikavimo duomenis, platinti papildomą kenkėjišką programinę įrangą (virusus), galiausiai suniokoti svetainę, taip pakenkiant jos funkcionalumui.

XSS atakų tipai ir pavyzdžiai

- **Nepastovios**

Nepastovios arba neišsaugomos XSS atakos yra pačios populiariausios [12]. Šios spragos pasirodo tuomet kai nepatikimi duomenys gauti iš žiniatinklio vartotojų, dažniausiai HTTP užklausų parametrų arba iš HTML formų įvesčių, iš karto panaudojami dinamiškai sugeneruojant puslapį. [13]

Klasikinis nepastovios XSS atakos pavyzdys: jeigu svetainėje yra paieškos sistema, vartotojui įvedus ieškomo objektą jis matomas įvestų rezultatų viršuje (ko buvo ieškota). Jeigu įvestis tinkamai nenukenksminama, gali būti ieškoma kokio nors žalingo kodo, kuris atvaizduojant būtų įvykdomas.

- **Pastovios**

Pastovios arba išsaugomos XSS atakos pasitaiko tuomet kai įvedami duomenys užsaugomi į duomenų bazę ir vėliau nuolatos atvaizduojami.

Klasikinis tokių atakų pavyzdys, yra įvairios komentarų formos, kuomet prie straipsnio matomi vartotojų įvesti komentarai, XSS atakos atveju tai galėtų būti kiekvieną kartą aktyvuojami žalingi kodai [13].

Pagrindinės XSS atakų pasekmės [14]:

- Identiteto vagystė;
- Priėjimas prie jautrios ar apribotos informacijos;
- Gavimas prieigos prie neautorizuotos informacijos;
- Kitų vartotojų šnipinėjimas;
- Naršyklių funkcionalumo iškreipimas;

- Vieša interneto programos (svetainės) savininko diskreditacija;
- Žiniatinklio programos išdarkymas, sugadinimas;
- Atsisakymo aptarnauti atakos.

Pažeista autentifikacija ir sesijos valdymas

Pažeistos autentifikacijos ir sesijos valdymo (angl. Broken Authentication and Session Management) spragos atsiranda dėl netinkamo paskyrų įgaliojimų ir sesijos duomenų saugojimo. Piktavaliai nuolat stengiasi gauti duomenis, kuriais pasinaudodami galėtų prieiti prie neautorizuotų duomenų. Žiniatinklio programų funkcijos surištos su autentifikacija ir sesijos valdymu ne visuomet teisingai įgyvendinamos. Tai suteikia galimybę piktavaliams perimti slaptažodžius, sesijos raktus ir kitus duomenis, kurių praradimas sukelia grėsmę kitų vartotojų identitetui.

Galimi pažeistos autentifikacijos ar sesijos valdymo pavyzdžiai:

1. Sesijos *id* rodomas URL kaip ***jsessionid*** parametro reikšmė,
   *http://orolinijos.lt/paradvimas/bilietai;**jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV**?tikslas=Vilnius*
   Vartotojas A būdamas prisijungęs, nusiunčia šią nuorodą *Vartotojui B*, *Vartotojas B* atidaręs nuorodą automatiškai perima sesiją, gali naudotis visomis *Vartotojo A* privilegijomis.
2. Netinkamai nustatytas laikas, per kurį atsijungiama nuo sistemos. Pavyzdžiui vartotojas pasinaudojęs paslauga viešos prieigos kompiuteryje, vietoj „išsiregistruoti“ uždaro naršyklę. Kitas vartotojas atidaręs naršyklėje tą patį URL adresą toliau gali naudotis paslauga, prieš tai buvusio vartotojo teisėmis.
3. Nutekinami duomenys su slaptažodžių duomenų baze. Jeigu slaptažodžiai saugomi atviru tekstu, jais bet kas gali greitai pasinaudoti.

Pažeidus autentifikaciją arba sesiją kyla grėsmė duomenų vientisumui ir konfidencialumui. Jeigu pavyksta perimti administratoriaus teises, kyla grėsmė visai žiniatinklio programai.

Nesaugios tiesioginės nuorodos į objektus

Nesaugios tiesioginės nuorodos į objektus (angl. Insecure Direct Object Reference) naudojamos kaip URL parametrai, tai gali būti failų ar katalogų vardai, duomenų bazių įrašai arba raktai. Piktavalis gali pradėti manipuliuoti URL parametrais ir prieiti prie tų objektų, kurie turėti būti apsaugoti. Taip sudaroma galimybė prieiti prie neautorizuotų duomenų.

Atviri nukreipimai ir direktorijų pakeitimas yra du klasikiniai nesaugių tiesioginių nuorodų į objektus pažeidžiamumų tipai.

Atvirų nukreipimų pavyzdys, kuomet programa nepatikrindama įterpia puslapį į kurią nuorodą gauna iš URL.
Pvz.:
Piktavalis pastebėjęs, kad *http://auka.lt/puslapis.php?url=http://auka2.lt* įterpinėja šaltinio URL kaip parametrą be jokio tikrinimo, modifikuoja užklausą *http://auka.lt/puslapis.php?url=http://piktavalis.lt* ir priverčia programą įvykdyti žalingą kodą. Tai gali būti koks nors scenarijus atskleidžiantis pirminio kodo šaltinį su konfigūraciniais duomenimis.

Direktorijos pakeitimai, kuomet piktavalis pastebėjęs URL *http://auka.lt/rodyti.php?faila=ataskaita.txt* pakeičia jį į *http://auka.lt/rodyti.php?faila=**../../../etc/shadow*** taip priversdamas žiniatinklio programą atvaizduoti */etc/shadow* ar bet kurį kitą failą su jautria informacija.
Šie pažeidžiamumai kelia grėsmę duomenų konfidencialumui.

CSRF

CSRF (angl. Cross-site request forgery) tipo atakos įvykdomos tuomet, kai prisijungusios aukos naršyklė priverčiama išsiųsti HTTP užklausą panaudojant aukos sesijos raktus žiniatinklio programai, kuri turi spragų. Tai leidžia piktavaliui priversti aukos naršyklę vykdyti užklausas, o žiniatinklio programa jas priima kaip tikras.

CRSF atakos įvykdymo pavyzdys:
Piktavalio svetainėje patalpintas HTML kodas:
<img
src="http://bankas.lt/pavedimas?paskyra=vartotojasA&suma=1000000&kam=Piktavalis">
Jeigu *Vartotojas A* savo prisijungimo duomenis laiko slapuke, kurio galiojimo laikas nesibaigęs, tuomet naršyklė kraudama paveiksliuką atliks transakciją, *Vartotojui A* apie tai nieko nežinant. Piktavaliui tiesiog pakanka prisivilioti *Vartotoją A* į svetainę su žalingu kodu.
Pasinaudodamas CRSF pažeidžiamumu atakuotojas gali priversti auką pakeisti, bet kokią informaciją prie kurios jis gali prieiti. Taip pat gali atlikti bet kokį veiksmą, kuris reikalauja atitinkamos autorizacijos.

Netinkama saugumo konfigūracija

Norint užtikrinti tinkamą saugumą, reikia tinkamai sukonfigūruoti žiniatinklio programą, karkasą; programos, žiniatinklio, duomenų bazės serverį ir pačią platformą. Neretai diegiant naujas sistemas paliekami galioti nustatymai pagal nutylėjimą, kurie gali neužtikrinti tinkamo saugumo lygio arba įdiegta sistema niekada nebebūna atnaujinama.

Dažniausi netinkamos saugumo konfigūracijos išnaudojimo pavyzdžiai:

- Pasinaudojama neatnaujintų sistemų trūkumais apie kuriuos paskelbia vystytojai;
- Nepakeičiami pradiniai slaptažodžiai ir prisijungimo, vardai kurie lieka įdiegus sistemą, nepašalinami diegimo katalogai;
- Neuždraudžiama matyti katalogų turinio, taip piktavaliai gali sužinoti katalogų vardus su sisteminiais failais ir juos parsisiųsti;
- Suteikiama per daug informacijos apie klaidas eiliniams vartotojams.

Pasinaudodami šiais pažeidžiamumais piktavaliai paprastai gauna neautorizuotą priėjimą prie sisteminių failų arba funkcionalumo. Paprastai tai gali sukelti grėsmę visai žiniatinklio programos sistemai.

Nesaugi kriptografija

Žiniatinklio programos susiduria su nesaugios kriptografijos grėsmėmis kai netinkamai naudoja kriptografijos funkcijas duomenų šifravimui. Piktavaliai stengiasi išgauti duomenis iš silpnai apsaugotų duomenų bazių, dažniausiai dėl to nukenčia jautri vartotojų informacija.
Piktavaliai paprastai nebando iššifruoti užkoduotų duomenų, o bando pasinaudoti iššifruojančiais kanalais arba prieiti prie duomenų kur jie nešifruojami.

Nesaugios kriptografijos pavyzdžiai:

- Žiniatinklio programa užšifruoja jautrią informaciją duomenų bazėje siekiant apsaugoti atskleidimą galutiniams vartotojams. Tačiau duomenų bazė nustatyta automatiškai iššifruoti jautrios informacijos stulpelį kuomet gauna užklausą, taip sudarant galimybę gauti visus tame stulpelyje esančius duomenis atviru tekstu pasinaudojant SQL injekcija.
- Atsarginės kopijos duomenys užšifruoti, tačiau šifravimo raktas saugomas toje pačioje atsarginėje kopijoje;

- Slaptažodžiai saugomi panaudojant paprasčiausias maišos funkcijas, kurios per trumpą laiką gali būti dešifruotos brutalios jėgos įrankiais. Tuo tarpu panaudojant papildomas maišos funkcijas iššifravimas naudojant esamas technologijas gali trukti ilgiau nei 3000 metų.

Šis pažeidžiamumas kelią grėsmę visiems konfidencialiems duomenims.

Netinkamas URL prieigų apribojimas

Netinkamas URL prieigų apribojimas (angl. Failure to Restrict URL Access), dažniausiai pasitaiko kai žiniatinklio programos apsaugo funkcionalumą su jautria informacija tik nuo neautorizuotų vartotojų nerodydama jiems slapto URL. Piktavaliai gali išnaudoti šias silpnybes bandydami prieiti prie URL tiesiogiai ir įvykdyti neautorizuotas operacijas. Taip nutinka kai netikrinama vartotojo autentifikacija kiekviename žingsnyje. Nemaža dalis žiniatinklio programų patikrina URL prieigos teises prieš suformuodamos apsaugotas nuorodas ar elementus. Tačiau programoms reikia atlikinėti tokią pačią prieigos kontrolę kiekvieną kartą, kuomet puslapiai prieinami, kitaip piktavaliams suteikiama galimybė prieiti prie slaptų puslapių bet kuriuo atveju.

Esant šiam pažeidžiamumui piktavaliai gali daryti tą patį ką gali daryti autorizuoti vartotojai. Kyla grėsmė duomenų vientisumui ir konfidencialumui.

Nepakankama transporto lygmens apsauga, nesaugios komunikacijos

Žiniatinklio programos dažnai klysta autentifikuodamos, šifruodamos ar apsaugodamos konfidencialumą bei vientisumą jautrios informacijos tinklo sraute. Tai dažniausiai nutinka dėl naudojamų silpnų algoritmų, pasibaigusio galiojimo ar neteisingai naudojamų sertifikatų.

Šis pažeidžiamoms kelią grėsmę duomenų konfidencialumui.

Kenksmingų failų vykdymas

Kenksmingų failų vykdymas (angl. Malicious File Execution) ypač aktualus populiariausia žiniatinklio programų programavimo kalba PHP sukurtoms programoms. Įterpus nuotolinį kodą galima perimti netgi tarnybinės stoties kontrolę. Tokios klaidos neretai pasitaiko ten kur karkasai leidžia vartotojams įkelti duomenų failus. Įkeliami failai turi būti patikrinti ne tik pagal plėtinį, bet ir pagal jų turinį, meta duomenis.

Nepatikrinti nukreipimai ir persiuntimai

Žiniatinklio programoms dažnai tenka nukreipinėti ar persiuntinėti vartotojus į kitus puslapius ar interneto svetaines, naudojant nepatikimą informaciją norint apibrėžti tikslo puslapius. Be tinkamo tikrinimo, piktavaliai gali nukreipti aukas į suklastotas ar kitokias žalingas svetaines.

Galimi nepatikrintų persiuntimų išnaudojimo pavyzdžiai:

- Žiniatinklio programa turi puslapį ***nukreipimas.php***, šis puslapis priima gautą parametrą *url* ir nukreipia vartotojus tuo adresu. Piktavalis gali pakeisti URL į blogą pvz.: *http://www.auka.lt/nukreipimas.php?url=blogis.lt*
- Kitos programos atlikus vieną ar kitą sėkmingą transakciją gali naudoti persiuntimų funkcijas į kitą svetainės vietą, piktavalis gali modifikuoti persiuntimo adresą ir patekti ten kur reiktų papildomos autorizacijos pvz.: *http://www.auka.lt/mokejimai.php?PersiuntimoVieta=admin.php*

Šie pažeidžiamumai suteikia galimybę perimti vartotojus ir priversi atkleisti jautrią informaciją, taip pat apeiti prieigos kontrolės mechanizmus. Kyla grėsmė duomenų konfidencialumui ir vientisumui.

Informacijos nutekinimas ir netinkamas klaidų valdymas

Žiniatinklio programos gali nejučiomis atskleisti informaciją apie jų konfigūraciją, vidinius darbus ar kitaip pažeisti privatumą per įvairiausias iškilusias problemas. Piktavaliai išnaudoja šias silpnybes gauti jautrios informacijos arba ruošiantis rimtesnėms atakoms.

Paspaudimų perėmimas

Paspaudimų perėmimas (angl. Clickjacking) yra pavojinga technika nukreipta prieš žiniatinklio programų vartotojus, priverčianti juos atskleisti konfidencialią informaciją [15]. Taip pat gali perimti kompiuterių valdymą. Taip nutinka paspaudus ant mygtuko formoje, kuris įvykdo funkcijas apie kurias vartotojui nėra žinoma.

Automatizuotos užklausos

Automatizuotų užklausų programos vykdo užklausas ieškodamos galimų pažeidžiamumų. Išnaudodamos paieškos sistemas iš anksto suprogramuotos procedūros ieško galimų taikinių.

Aptikę pvz.: prisijungimo formą jos gali akimirksniu ištestuoti didžiąją dalį pažeidžiamumų. Tokios programos yra pavojingos, kadangi jos išnaudoja pačius naujausius pažeidžiamumus. Gunter Ollmann [16] suklasifikuotų automatizuotų programų sąraše pavojingiausia vieta priskirta pažeidžiamumų ieškikliams. Tokios programos naudoja įprastas puslapių aptikimo technologijas pagrindinį dėmesį skiriant įvesties formų paieškai. Aptikusios įvesties formą bando vykdyti gerai žinomus netinkamus simbolius, pagal gautą atsakymą klasifikuoja galimas žiniatinklio spragas.

Taikomosioms žiniatinklio programoms kylančių grėsmių apibendrinimas

Populiariausių grėsmių sąrašas, sudarytas 2010 m. OWASP organizacijos, pateiktas lentelėje Iš 1 lentelės matyti, kad pagrindinė žiniatinklio programų grėsmė yra injekcijos ir XSS tipo atakos.

1 lentelė. *Populiariausių grėsmių sąrašas, sudarytas 2010 m. OWASP organizacijos*

Injekcijos	Injekcijų keliamos grėsmės, dažniausiai **SQL injekcijos**, taip pat OS ir LDAP injekcijos atsiranda kuomet nepatikimi duomenys siunčiami į interpretatorių kaip dalis komandos ar užklausos. Piktavaliai priverčia veikti interpretatorius taip kad galėtų įvykdyti piktavališkas užklausas ir prieiti prie kitų duomenų.
XSS	XSS pažeidžiamumai įvyksta kuomet žiniatinklio programa priima nepatikimus duomenis ir siunčia juos tiesiai naršyklei neatlikdama tinkamos validacijos ir pavojingų simbolių nukenksminimo. XSS piktavaliui suteikia galimybę aukos naršyklę priversti vykdyti žalingus skriptus kurie gali perimti vartotojo sesiją, iškreipti vaizduojamą svetainę, ar nukreipti į žalingus puslapius.
Pažeista autentifikacija ir sesijos valdymas	Žiniatinklio programų funkcijos surištos su autentifikacija ir sesijos valdymu ne visuomet teisingai implementuojamos. Tai suteikia galimybę piktavaliams perimti slaptažodžius, sesijos raktus ir kitus duomenis kurie sukelia grėsmę kitų vartotojų identitetui.
Nesaugios tiesioginės nuorodos į objektus	Tiesioginės nuorodos į objektus kuriamos kai programų kūrėjai viešai parodo vidines nuorodas į objektus, pvz. failus, direktorijas ar duomenų bazių raktus, be atitinkamos prieigos kontrolės tikrinimo ar kitokios apsaugos. Piktavaliai gali manipuliuoti šiomis nuorodomis ir gauti prieigą prie neautorizuotos informacijos.

CSRF	CSRF tipo atakos nutinka tuomet, kai prisijungusios aukos naršyklė priverčiama išsiųsti HTTP užklausą panaudojant aukos sesijos raktus žalingai žiniatinklio programai. Tai leidžia piktavaliui priversti aukos naršyklę vykdyti užklausas, o žiniatinklio programa jas priima kaip tikras.
Netinkama saugumo konfigūracija	Norint užtikrinti tinkamą saugumą, reikia tinkamai sukonfigūruoti žiniatinklio programą, karkasą, programos, žiniatinklio, duomenų bazės serverį ir pačią platformą. Neretai vietomis paliekami galioti nustatymai pagal nutylėjimą, kurie gali neužtikrinti tinkamo saugumo lygio esančioms aplinkybėms. Taip turi būti rūpinamasi konfigūracijos valdymu, atnaujinimu.
Nesaugi kriptografija	Daug žiniatinklio programų netinkamai apsaugo jautrią informaciją, tokią kaip kreditinių kortelių duomenys,socialinio draudimo nr. ar prisijungimo duomenys. Naudojant netinkamą kriptografiją, kyla grėsmę tokios informacijos vientisumui ir konfidencialumui,
URL prieigos apribojimų klaidos	Nemaža dalis žiniatinklio programų patikrina URL prieigos teises prieš suformuodamos apsaugotas nuorodas ir mygtukus. Tačiau programoms reikia atlikinėti tokius pačią prieigos kontrolę kiekvieną kartą kuomet puslapiai prieinami kitaip piktavaliams suteikiama galimybė prieiti prie slaptų puslapių bet kuriuo atveju.
Nepakankama transporto lygmens apsauga	Žiniatinklio programos dažnai klysta autentifikuodamos, šifruodamos ar apsaugodamos konfidencialumą bei vientisumą jautrios informacijos tinklo sraute. Tai dažniausiai nutinka dėl naudojamų silpnų algoritmų, pasibaigusio galiojimo ar neteisingai naudojamų sertifikatų.
Nepatikrinti nukreipimai ir persiuntimai	Žiniatinklio programoms dažnai tenka nukreipinėti ar persiuntinėti vartotojus į kitus puslapius ar interneto svetaines, naudojant nepatikimą informaciją norint apibrėžti tikslo puslapius. Be tinkamo tikrinimo, piktavaliai gali nukreipti aukas į Phishing ar kitokias žalingas svetaines, arba panaudoti persisiuntimus patekimui prie neautorizuotos informacijos.

2 lentelė atskleidžia žiniatinklio programų grėsmių rizikos veiksnius. Matyti, kurias grėsmes paprasčiausia išnaudoti. Grėsmių paplitimas leidžia įvertinti, kokia tikimybė, kad rizika

aktuali. Aptikimas atspindi grėsmės problematiškumą. Poveikis šiuo atveju yra grėsmės techninis poveikis pačiai programai. Norint įvertinti koks poveikis verslui ar turtui, reiktų apsibrėžti, kurios grėsmės yra aktualios ir kurias bus galima išnaudoti pagal programos klasifikaciją. Iš lentelės taip pat matyti, kodėl injekcijos yra pirmojoje grėsmių sąrašo vietoje, jas lengva panaudoti, jų paplitimas dažnas, aptikti nėra lengva, o poveikis žiniatinklio programai - stiprus.

2 lentelė. *Žiniatinklio programų rizikos veiksniai*

Grėsmė	Išnaudojamumas	Paplitimas	Aptikimas	Poveikis
Injekcijos	PAPRASTAS	DAŽNAS	VIDUTINIS	STIPRUS
XSS	VIDUTINIS	LABAI DAŽNAS	PAPRASTAS	VIDUTINIS
Pažeista autentifikacija ir sesijos valdymas	VIDUTINIS	DAŽNAS	VIDUTINIS	STIPRUS
Nesaugios tiesioginės nuorodos į objektus	PAPRASTAS	DAŽNAS	PAPRASTAS	VIDUTINIS
CSRF	VIDUTINIS	PLAČIAI PAPLITĘS	PAPRASTAS	VIDUTINIS
Netinkama saugumo konfigūracija	PAPRASTAS	DAŽNAS	PAPRASTAS	VIDUTINIS
Nesaugi kriptografija	SUDĖTINGAS	RETAS	SUDĖTINGAS	STIPRUS
URL prieigos apribojimų klaidos	PAPRASTAS	RETAS	VIDUTINIS	VIDUTINIS
Nepakankama transporto lygmens apsauga	SUDĖTINGAS	DAŽNAS	PAPRASTAS	VIDUTINIS
Nepatikrinti nukreipimai ir persiuntimai	VIDUTINIS	RETAS	PAPRASTAS	VIDUTINIS

2.4 Žiniatinklio programų apsaugos metodų analizė

Atsižvelgiant į ankstesniame skyriuje nagrinėtas grėsmes, šiame skyriuje nagrinėjami apsaugos metodai kylančioms grėsmėms dėl vieno ar kito pažeidžiamumo aptikti ir pašalinti. Atsižvelgiant į grėsmių dešimtuką matyti, kad daugiausiai dėmesio skirti reikia apsaugai nuo SQL injekcijų ir XSS kodo įterpimų. Šie du pažeidžiamumai šiandien yra patys aktualiausi. Didžioji dalis metodų pateikti remiantis OWASP siūlomais sprendimais.

Injekcijos

Labiausiai paplitusi injekcijų forma yra SQL injekcijos. **SQL Injekcija** - tai, galimybė puslapyje, naudojančiam SQL duomenų bazę, įvykdyti norimą užklausą, kurios pagalba galima išgauti neautorizuotą informaciją esančią duomenų bazėje.

Pagrindinė apsauga:

- Naudoti iš anksto sukonstruotas užklausas sudarytas iš tam tikrų parametrų. Naudojant iš anksto sukonstruotas užklausas kintamieji gaunami iš vartotojo įrašomi į jiems skirtas vietas. Kai užklausos generuojamos dinamiškai, sudaroma galimybė pakreipti seką blogąja linkme. Statinės užklausos leidžia duomenų bazei atskirti kodą nuo duomenų, nepriklausomai nuo to ką įvedė vartotojas.
- Naudoti iš anksto paruoštas procedūras. Iš anksto paruoštos procedūros turi tokį patį saugumo efektą kaip ir iš anksto paruoštos užklausos, jos neleidžia atlikti veiksmų nesusijusių su tos funkcijos paskirtimi.
- Neutralizuoti visas vartotojų įvestis, PHP programavimo kalboje tam naudojama funkcija *mysql_real_escape_string()[17]*.
- Programai, kuri vykdo kreipimąsi į duomenų bazės serverį suteikti tik reikalingas privilegijas. Suteikti programai įgaliojimus vykdyti tik reikiamas užklausas. Nebūtina leisti ištrinti lentelę, perkrauti DB, ir pan..
- Teigiamo arba „baltųjų sąrašų“ įvesčių validacija su atitinkama kanonizacija taip pat padeda apsisaugoti nuo injekcijų. Ten kur galima vesti tik raides arba, tikrinti ar įvestos tik raidės arba skaičiai, nebandyti vykdyti klaidingos užklausos.

Justin Clarke, SQL Injection Attacks and Defense [10] plačiai aprašoma teigiamo modelio validacija. Priimami tik tie duomenys kurie žinomi kad yra geri. Pagrindiniai kriterijai: duomenų tipas, duomenų ilgis, duomenų kategorija, duomenų turinys. Tikrinimui atlikti siūloma naudoti reguliariąsias išraiškas (angl. regular expression).

SANS paskelbtame straipsnyje SQL Injection: Modes of Attack, Defence, and Why It Matters [18] pagrindinis siūlymas norint išvengti SQL injekcijų išvalyti ir nukenksminti bet kokius priimamus duomenis, nesvarbu jie laikomi saugiais ar ne. Taip pat apriboti ir nuolat tikrinti kiekvieno įvedamo lauko ilgio rėmus, įvestis negali būti nei per trumpa nei per ilga.

XSS

Norint išvengti XSS tipo atakų turi būti filtruojami visi iš vartotojo priimami duomenys, išvedami duomenys turi būti užkoduoti taip, kad specialūs kodo atributai nebūtų atvaizduojami ir interpretuojami kaip programinis kodas.

OWASP atsižvelgdama į geriausiai žinomas praktikas siūlo XSS apsaugos taisykles [19]:

- Niekada neįterpinėti nepatikimų duomenų išskyrus į reikiamas vietas
- Nukenksminti HTML kodą prieš įterpiant nepatikimą informaciją kaip HTML kodo komponentą;
- Nukenksminti atributus prieš tai kai jie įterpiami kaip HTML atributai;
- Nukenksminti JavaScript prieš įterpiantį į HTML kaip JavaScript duomenų reikšmes;
- Nukenksminti CSS prieš įterpiant kaip naudojamas reikšmes iš nepatikimų šaltinių;
- Nukenksminti URL jeigu jis sudaromas iš nepatikimų parametrų;
- Naudoti HTML politikos variklį kuris tikrintų arba išvalytų HTML kodą;
- Sustabdyti DOM technologija paremtus XSS;

Knygoje apie XSS atakų apsaugą siūlomi sprendimai. [20]

Specialių simbolių atpažinimas:

Identifikuojami specialūs simboliai, kuriais prasideda kodas. Vėliau galima daryti įvairius šablonus ir juos naudoti filtravime.

Turinyje:

"<" yra specialus, nes pradeda žymą;

"&" specialus, nes naudojamas išlaikyti simbolių vientisumui;

">" naršyklės gali interpretuoti kaip žymos pabaigą manydamos kad "<", turi būti , bet praleista per klaidą.

Atributuose:

Reikšmės apgaubtos kabučių, kabutės simbolizuoja atributo savybės pradžią ir pabaigą;

Viengubos kabutės;

"&", kuris naudojamas reikšmių sujungimui

URL:

Klavišai „space“, „tab“ ir nauja eilutė yra specialūs nes žymi URL pabaigą;

"&" specialus nes sujungia arba atskiria CGI parametrus;

Ne ASCII simboliai (viskas virš 128 ISO-8859-1 koduotėje) neleidžiami URL, todėl jie yra specialūs;

"%" įvedime turi būti filtruojamas.

HTML *Body* dalis:

<script></script> iš karto reiškia kad pradedamas ir baigiamas programinis kodas, reikia filtruoti.

Serverio pusėje veikiančios paprogramės kurios konvertuoja bet kurį šauktuką (!) į dvigubas kabutes (") išvedime reikalauja papildomo filtravimo.

Filtravimas

Vienas sprendimas yra įvedime ir išvedime filtruoti duomenis pašalinant specialiuosius simbolius.

Antrasis būdas specialiuosius simbolius užkoduoti. Specialieji simboliai užkoduojami taip, kad interpretatoriam jie nebeteikia prasmės. Šis būdas yra geresnis už pirmąjį sprendimą, kadangi apsaugo nuo informacijos praradimo. Paprastai pvz.: simbolis ">" tampa simbolių junginiu '>', '&' tampa '&', o "" - '"'.

PHP programavimo kalboje tam daryti naudojamos specialios funkcijos *htmlspecialchars()*, bei *strip_tags()*.

MSDN [21] siūlomos ASP.NET gairės straipsnyje (How To: Prevent Cross-Site Scripting in ASP.NET) panašios į prieš tai siūlytas.

Tik sprendimai kiti.

Siūloma naudoti užklausų validatorių:

```
<system.web>
  <pages buffer="true" validateRequest="true" />
</system.web>
```

Įvardijamos potencialiai pavojingos HTML žymės:

<applet>,<body> ,<embed> ,<frame> ,<script> ,<frameset> ,<html>, <iframe> ,<img> ,<style>,<layer> <link> <ilayer> <meta> <object>

Išvedimo HTML žymoms koduoti siūlomas pavyzdys:

Response.Write(HttpUtility.HtmlEncode(Request.Form["name"]));

URL kodavimo pavyzdys:

Response.Write(HttpUtility.UrlEncode(urlString));

MSDN rekomenduoja nebandyti filtruoti specialius simbolius, kadangi piktavaliai galiausiai ras kaip kurį simbolį pavaizduoti kitaip. Vietoje to, siūloma tikrinti ar įvedami duomenys atitinka saugios įvesties šabloną.

3 lentelė. *Simbolių išreiškimas kitomis koduotėmis*

Simbolis	Dešimtainėje	Šešioliktainėje	HTML simboliai	Unicode
" (dviguba kabutė)	&#34	&#x22	"	\u0022
' (viena kabutė)	&#39	&#x27	'	\u0027
& (=ir)	&#38	&#x26	&	\u0026
< (mažiau)	&#60	&#x3C	<	\u003c
> (daugiau)	&#62	&#x3E	>	\u003e

3 lentelėje pateikiamas pavyzdys kaip vieni simboliai gali būti išreiškiami kitais.

Pasiūlymai remiantis [17] šaltiniu kaip sulaikyti XSS tipo atakas.

Pasiūlyti du papildomi sprendimai:

1. Konvertuoti visus ne alfabetinius simbolius į HTML simbolius prieš rodant vartotojui.
2. Tam kad sistema atskirtų kurią paprogramę vykdyti, sukurti paprastą parašų sistemą su viešais ir privačiais raktais. Programa patikrintų ar vykdoma paprograme autentifikavusi, jei ne tuomet jos nevykdo.

Autentifikacijos ir sesijos valdymo saugumo užtikrinimas

Remiamasi [22], [23], [24] šaltinių pasiūlymais suformuotos gairės, kurių reikia vadovautis norint užtikrinti autentifikacijos ir sesijos valdymo saugumą.

- Naudoti laiko nustatymus, kuomet sesija nutraukiama automatiškai po tam tikro neveiksnumo;
- Riboti įvedamų slaptažodžių skaičių;
- Jeigu naudojama SSL, neleisti pradėti autentifikavimo iš neužkoduotų puslapių;
- Naudoti stiprių slaptažodžių politiką;
- Nesiuntinėti slaptažodžių paprastu tekstu, naudoti maišos funkcijas;

- Sistemos viduje veikiantys komponentai turi išvengti besąlygiško paklusnumo kai bendrauja tarpusavyje, geriausiu atveju jie taip pat turi save identifikuoti;
- Saugoti sesijos ID, jeigu įmanoma SSL;
- Sesijos ID turėtų būti kompleksinis;
- Neleisti sesijos ID atvaizduoti URL viduje;
- Nepriimti vartotojų pasirinktų, nustatytų sesijos ID;
- Neleisti kopijuoti puslapių į laikinąją atmintį.

Nesaugios tiesioginės nuorodos į objektus

Norint išvengti nesaugių tiesioginių nuorodų į objektus atakų grėsmių reikia apsaugoti kiekvieną prieinamą objektą ir direktoriją [25]:

- Nenaudoti tiesioginių nuorodų, naudoti jų indeksus, kurie sugeneruojami su kiekvieno vartotojo sesija;
- Prieigos kontrolė: kiekvienas bandymas jungtis prie objektų iš nepatikimų šaltinių turi neišvengiamai praeiti pro prieigos kontrolės patikrinimo funkcijas, tam kad būtų patikrinta ar vartotojas tinkamai autorizuotas;
- Kai tik įmanoma vengti atskleisti privačių objektų nuorodas tokias kaip pirminiai raktai ar failų vardai;
- Ten kur įmanoma naudoti raktų ir nuorodų šifravimą;
- Tinkamai nustatyti failų prieigos teises tarnybinėse stotyse, itin apgalvotai rinktis kam priskirti *chmod 777* reikšmę;
- Apriboti direktorijų turinio skaitymą *robots.txt* failo pagalba;
- Apriboti direktorijų pateikimą *.htaccess* falo pagalba

Naudoti netiesioginius ryšius su objektais, failų vardus koduoti maišos funkcijomis:

Objektų vardų šifravimo pavyzdys:

Vietoje įprastai naudojamo:

<select name=”language”>

<option value=”English”>English</option>

</select>

<input type="button" value="Priimti" name="acceptData">

Naudoti:

<select name=”language”>

<option value=”78463a384a5aa4fad5fa73e2f506ecfc”>English</option>

</select>

<input type="button" value="Priimti" name="DpeJycJLnKIfEdKEdKEiFGTsKbmLGMhE">

Šiuo atveju objekto vardas šifruojamas ir kiekvienu atveju skiriasi, todėl piktavaliams būtų sudėtinga atlikti ataką nežinant tikrų vardų.

Vidinių užklausų (CSRF) klastojimo prevencija

OWASP rekomendacijos [26]:

- Autentifikavimas naudojant GET ir POST parametrus, ne tik slapukai, naudoti slaptus sesijos unikalius numerius (angl. token), vengti GET naudojimo, tuomet piktavaliams bus sunkiau manipuliuoti užklausomis;
- HTTP referer antraštės tikrinimas. Iš šios antraštės matyti iš kur atėjo vartotojas, jeigu URL yra iš neautentifikuoto puslapio jį reiktų atmesti; Tačiau čia reikia atsižvelgti į galimą HTTP referer antraštės klastojimą.
- Ribojamas autentifikavimui skirtų slapukų galiojimo terminas;
- Naudoti slaptus laukus formose susietus su sausainėliais, ir priimant naujus duomenis tikrinti ar priimti duomenys atėjo iš tos pačios formos;
- Svetainės naudojančios AJAX technologiją, gali kiekvienai *XMLHttpRequest* priskirti atskirą antraštę ir tikrinti jos tikrumą.

Autoriaus Chris Shiflett knygoje Essential PHP security[27] rekomenduojama naudoti POST metodą vietoje GET veiksmams HTML formose, o PHP kalboje įgyvendinant programinę funkcijų logiką vietoje *$_REQUEST* naudoti *$_POST*.

Apsauga panaudojant slaptą skaičių derinį tinkamą tik tai transakcijai:

*http://bankas.lt/pervedimas?paskyra=vartotojasA&suma=1000000&kam=Piktavalis**&token=5asdkj45***

Sesijos raktas susietas su tuo metu vykdoma užklausa, be sesijos rakto transakcija negalėtų būti įvykdyta, antrą kartą to pačio sesijos rakto panaudoti taip pat negalima. Be to sesijos raktas turi ribotą gyvavimo laiką [28].

Informacijos nutekinimas ir netinkamas klaidų valdymas

OWASP [29] ir webappsec [30] teikia panašias rekomendacijas:

- Svarbu, kad atitinkami pranešimai apie klaidas pasiektų žmones pagal jų kompetencija. Detalus pranešimas apie klaidą turi būti užsaugomas į žurnalą ir rodomas programuotojams, vartotojui pranešamas lakoniškas klaidos apibūdinimas.
- Klaidos gali būti iš skirtingų sluoksnių tarnybinės stoties, duomenų bazės, programos, visos jos turi būti surenkamos ir atpažįstamos. Rekomenduojama vartotojui gražinti kokį nors unikalų klaidos kodą, kurį jis pranešęs administracijai galėtų gauti daugiau informacijos (jeigu ji būtina).
- Periodiškai žiūrėti žurnalą ir ieškoti anomalijų

Steven McElwee straipsnių serijoje apie saugų programimą [31] siūlymai atkreipti dėmesį kaip programa elgiasi susidūrusi su klaida, ar bando vykdyti toliau savo operaciją, ar nutraukiamas visas tolimesnis darbas.

Kenksmingų failų vykdymo prevencija

Informit straipsnyje ***Preventing RFI Attacks*** [32] rašoma, kad saugiausias būdas apsisaugoti nuo nuotolinių failų įterpimo yra užtikrini, kad kodas neturi įterpimo funkcijų kurios naudojasi neapsaugotais kintamaisiais. Taip pat yra tam tikri nustatymai serveryje, kurie gali blokuoti nutolusių failų įterpimo atakas.

PHP turi būti išjungta *register_globals* reikšmė.Tai apsaugo nuo nuotolinių failų, bet neapsaugo jeigu pavyks įkelti failą į serverį, tuomet jį bus galima vykdyti.

Joel Scambray [33] Siūloma išjungti funkcijos *allow_url_fopen* naudojimą tam, kad nebūtų galima atidarinėti nutolusių failų.

Naudoti detalius tikrinimo mechanizmus:

$hostile = &$_POST; // rodyklė į POST kintamąjį, bet ne į $_REQUEST

$safe['filename']= validate_file_name($hostile['unsafe_filename']); // padaroma saugiai

Toliau naudoti taip:

Netinkamas variantas: *require_once($_POST['unsafe_filename'] . 'inc.php');*

Saugus variantas: *require_once($safe['filename'] . 'inc.php');*

Naudoti ugniasienės taisykles, kurios uždraustų tarnybinei stočiai sudaryti naujus susijungimus su išorinėmis tarnybinėmis stotimis ar vidinėm sistemomis. Aukšto saugumo reikalaujančiose sistemose izoliuoti tarnybinę stotį nuo VLAN ar privataus potinklio.

Tikrinti visus vartotojų naudojamus failus ir jų vardus. Vartotojų naudojami failai neturi pažeisti nustatytų taisyklių.

Kaip galima labiau izoliuoti vidines programas vieną nuo kitos, gali būti naudojama virtuali izoliacija, testinės vietos ir pan.

Nesaugi kriptografija, nesaugios komunikacijos

Žiniatinklio programos retai tinkamai naudoja kriptografijos funkcijas, kad apsaugotų duomenis. Siūloma naudoti tik patikrintus šifravimo algoritmus AES, RSA viešo rakto kriptografiją, maišai naudoti SHA-256 [34].

Plačiai naudojamus MD5/SHA1 keisti į SHA-256. Neperdavinėti privačių raktų nesaugiais kanalais.

Žiniatinklio programos ne visada naudoja šifruotą tinklo srautą, kai tai yra būtina norint apsisaugoti nuo jautrios informacijos nutekėjimo.

Visiems autentifikuotiems ryšiams ir jautrių duomenų perdavimui naudoti TLS/SSL šifruotiems ryšiams skirtus protokolus.

Netinkamas URL prieigų apribojimas

OWASP rekomendacijos [34]:

- Užtikrinti, kad bet koks URL turi patikrinti vartotojo teises, ar gali prie jo prieiti;
- Neleisti bibliotekų failų atidaryti iš tos pačios direktorijos kaip ir programa;
- Nebandyti slėpti slaptų duomenų su specialiais ar paslėptais URL, būtina visuomet patikrinti privilegijas, kad sužinojus adresą negalėtų prieiti prie neautorizuotų duomenų;
- Blokuoti bandymus prieiti prie failų tipų kurių programa niekuomet neturės. Tai padės apsisaugoti nuo spėliojimų ir blokuos bet kokius bandymus prieiti prie žurnalų failų, xml ar kitų, kurių niekuomet neperduodami tiesiogiai.

Automatizuotų užklausų atpažinimas

Atakas dažniausiai sukelia ne mechaniniai veiksmai, bet įvairios iš anksto paruoštos programos, kurios automatizuotai tikrina žiniatinklio programas internete ieškodamos gerai žinomų spragų. Svarbu atskirti tokias programas nuo paprastų vartotojų ir kuo greičiau apriboti jiems prieigą. Programas galima atpažinti vykdant aktyvų auditą, atliekant Apache

žiniatinklio serverio analizę, iš pauzių tarp užklausų galima atskirti ar tai žmogus. Tačiau reikia sudaryti ir leidžiamų paprogramių sąrašą.

Saugumo kompanija Acuentix straipsnyje [35] siūlo būdus kaip atpažinti automatizuotas programas:

- Naudoti spąstus robotams, patalpinant nuorodas kurias aptikti gali tik automatizuotos programos;
- Naudoti paslėptą įvesties lauką, paprastas vartotojas jo niekada nemodifikuos, o robotai automatiškai užpildys;
- Norint išvengti brutalios jėgos robotų reikia apriboti maksimalų neteisingų bandymų skaičių;
- Tikrinti kada buvo paskutinis pažeidimas iš to pačio IP;
- Pasitelkiant *robots.txt* naudoti medaus puodynės metodą, nukreipiant į paprastiems vartotojams nežinomas vietas;
- Apriboti užklausų skaičių;
- Įvairūs žmogui suprantami patikrinimai, įvedimas simbolių kombinacijų iš paveiksliuko ir pan.

Gunter Ollmann [36] rekomenduoja:

- Blokuoti HEAD tipo užklausas, kurių tikslas gauti informaciją apie turinį;
- Pervadinti tarnybinės stoties naudojamos programinės įrangos informaciją;
- Tikrinti REFERER antraštę patikrinant iš kur atėjo vartotojas;

Apsaugos metodų apibendrinimas

4 lentelė rodo kokios priemonės naudojamos norint apsisaugoti nuo kiekvienos grėsmės.

4 lentelė. *Apsaugos metodai pagal populiariausių grėsmių sąrašas, sudarytą 2010 m. OWASP organizacijos*

Injekcijos	Atskirti nepatikimus duomenis nuo komandų ir užklausų. Naudoti parametrituotas užklausas iš anksto paruoštose procedūrose. Nukenksminti specialiuosius simbolius. Įvedamų laukų tikrinimas, taisyklių,baltųjų sąrašų naudojimas.
XSS	Tinkamai išskirti visą nepatikimą informaciją paremta HTML turiniu (body, JavaSceipt, CSS).

	Įvedamų laukų tikrinimas, dekodavimas įvedamų duomenų, apribojimas ilgiu, simbolių formatų pagal tam tikras taisykles, reikalingas tai atitinkamai programai.
Pažeista autentifikacija ir sesijos valdymas	Naudoti stiprius autentifikavimo ir sesijos valdymo kontrolės įrankius
Nesaugios tiesioginės nuorodos į objektus	Generuoti užklausas priklausomai nuo kiekvieno vartotojo. Tikrinti prieigas, kiekvienas bandymas prieiti prie objekto turi būti patikrinamas prieigos kontrolę užtikrinančio mechanizmo
CSRF	Naudoti unikalų žymenį paslėptame lauke. Žymuo kiekvieną kartą skiriasi ir yra patikrinamas, todėl negalima sugeneruoti iš anksto paruoštos nuorodos.
Netinkama saugumo konfigūracija	Konfigūracijos auditavimas bei valdymas, pritaikymas saugumo politikai.
Nesaugi kriptografija	Užtikrinti atitinkamą duomenų šifravimą. Identifikuoti duomenis kurie turi būti saugomi šifruotu pavidalu (pvz. atsarginės kopijos). Slaptažodžiai saugomi naudojant maišos algoritmus.
URL prieigos apribojimų klaidos	Tinkami autentifikacijos ir autorizacijos mechanizmai užtikrinantys kiekvieno puslapio prieigos kontrolę.
Nepakankama transporto lygmens apsauga	SSL naudojimas puslapiuose kuriuose operuojama jautria informacija; Saugumo žymens naudojimas jautrios informacijos sausainėliuose; Konfigūruoti SSL palaikymą stipriems algoritmams; Užtikrinti tinkamą sertifikato naudojimą;
Nepatikrinti nukreipimai ir persiuntimai	Vengti naudoti nukreipimus ir persiuntimus; Neįtraukti vartotojų paramtytų į tikslo puslapius; Jeigu naudojami parametrai, tuomet užtikrinti parametrų validumą ir autorizaciją atitinkamam vartotojui.

Kai kurių spragų ugniasienė apsaugoti negali. Ugniasienė nesusijusi su duomenų šifravimu, komunikacijomis. Pagrindinės ugniasienės savybės apsaugoti nuo XSS, SQL injekcijų, komandinio kodo filtravimas, automatizuotų atakų aptikimo ir sulaikymo, CSRF galimų loginių klaidų, pažeistos autentifikacijos arba sesijos.

Kai kurias spragas daug lengviau užtaisyti programos kode, nei bandyti nuo jų apsisaugoti naudojant ugniasienes. Tačiau sunku analizuoti trečiųjų šalių kodą, todėl keliams tikslas kad ugniasienė padėtų kuo geriau apsisaugoti nuo bet kokių pažeidžiamumų netrikdydama žiniatinklio programos tiesioginio darbo. Taip pat svarbu, kad ugniasienė rūšiuotų galimus pažeidžiamumus, pateiktų kuo mažiau pranešimų apie netikrus pavojus (pažeidžiamumų klasifikacija).

2.5 Žiniatinklio programų saugumą užtikrinančių priemonių analizė

2.5.1 Apsaugos priemonės pasirinkimo gairės

OWASP [9] siūlomi ugniasienės pasirinkimo kriterijai:

- Apsauga nuo OWASP sudarytų grėsmių dešimtuko;
- Kuo mažiau klaidingų rezultatų (netikrų pavojų);
- Stiprios saugumo savybės: įvesties ir išvesties tikrinimas;
- Paprastumas naudoti: sistemą apsaugančias priemones vartotojams padaryti kuo suprantamesnes, neverta tikėtis, kad vartotojas tikrai norės įvedinėti dvyliką slaptažodžių;
- Tipai pažeidžiamumų nuo kurių gali apsaugoti;
- Galimybė išlaikyti individualius vartotojus nenutraukiant jų sesijos: Jeigu programa nesugeba įvykdyti užduoties ji turėtų nutraukti savo darbą, bet neleisti pamatyti sisteminių klaidų pranešimų. Analogiškai elgiasi ugniasienės, jeigu kažkokių paketų nepavyksta apdoroti jos juos išmeta;
- Naudoti jau esamus apgalvotus saugumą užtikrinančius komponentus, nebandyti sugalvoti iš esmės kažką naujo;
- Komponentai turėtų pastebėti vienas kito netinkamą darbą;
- Žinoti, kad programa yra saugi tiek kiek saugi visos sistemos silpniausia grandis.
- Galimybė sukonfigūruoti apsaugai nuo bet kokios specifinės problemos (lankstumas);
- Programinės ir aparatinės įrangos priešpriešos (pirmenybė teikiama aparatinei).

Webappsec siūlo kitus žiniatinklio programų ugniasienių vertinimo kriterijus [10]:

- Išdėstymo architektūra;
- HTTP palaikymas;

- Apsaugos technologijos;
- Įvykių rašymas į žurnalus (angl. Logging);
- Atskaitingumas;
- Valdymas;
- Veikimo greitis;
- Susidorojimas su XML.

Amol Sarwate rašydamas apie ugniasienių kriterijus [11] iškėlė dar vieną svarbų kriterijų: galimybę dirbti su SSL srautais

Papildomas kriterijus dar galėtų būti kaina, ir tipas atviro kodo ar uždaro. Ar galima pačiam daryti pakeitimus.

2.5.2 Žiniatinklio programų saugą užtikrinančios priemonės

Kompiuterių tinkluose naudojamos įsibrovimų aptikimo sistemos (angl. Intrusion detection system (IDS)) bei prevencinės priemonės bendrai vadinamos ugniasienėmis. Žiniatinklio programose taip pat naudojami panašūs įrankiai stebėti ir užfiksuoti pažeidimus realiu laiku. Tokie stebėtojai nesunkiai aptinka atakų požymių turinčius vartotojų veiksmus, reaguoja į anomalijas priklausomai nuo konfigūracijos. Apie smulkų pažeidimą gali tik užrašyti į žurnalą, o labai svarbiu atveju nedelsiant siųsti pranešimą administracijai.

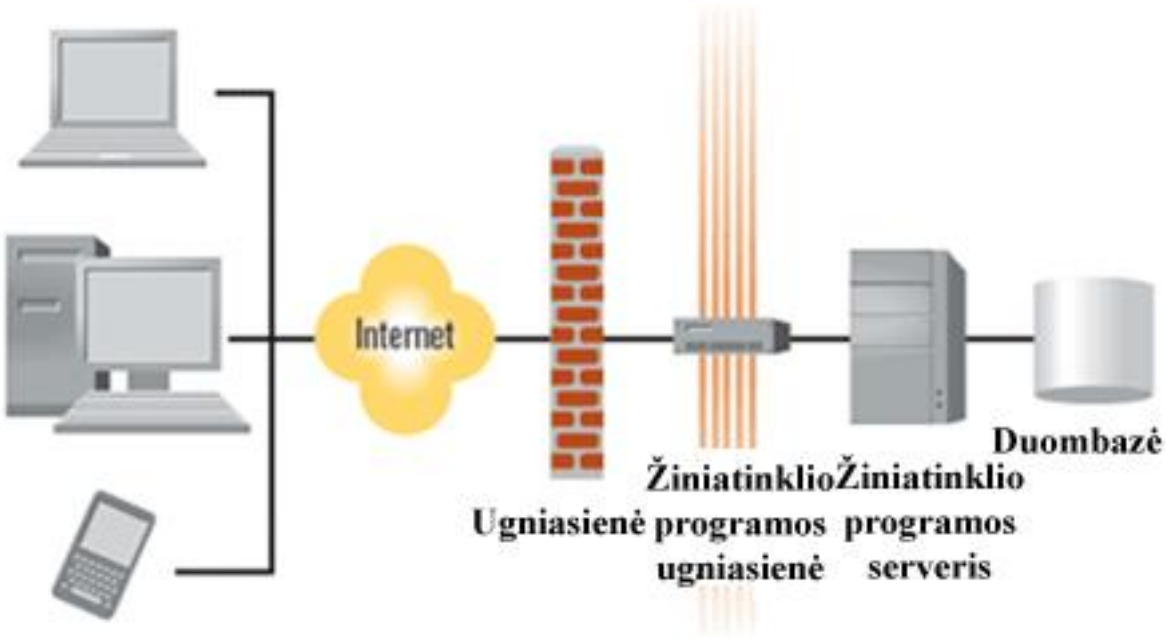
Priemones galima suskaidyti į:

- Aptikimo sistemos – (angl. Intrusion detection systems (IDS)) įsibrovimus aptinka realiu laiku, juos užrašo ir informuoja;
- Prevencinės sistemos – (angl. Intrusion prevention systems (IPS)) stebi ir reaguoja realiu laiku į vykstančias atakas;
- Universalios – ugniasienės (angl. Web application firewall (WAF)) laikoma, kad ugniasienės turi ir aptikimo ir prevencijos funkcijas.

Kompiuterių tinkluose naudojamos tinklų įsibrovimų stebėjimo sistemos (NIDS) žiniatinklio programoms nėra tinkamos dėl šių priežasčių:

- Per dideli srautai, kuriuos per daug sudėtinga sekti;
- Šifravimas (SSL) padaro informaciją nematoma;
- Duomenų suspaudimo mechanizmai apsunkina stebėjimą;

- Sukurta veikti TCSP/IP lygmenyje todėl ne taip gerai tinka HTTP;
- Svarbiausia NIDS nėra tinkama programų lygio apsaugai.



3 pav. ugniasienių išsidėstymas

3 paveiksle parodyta žiniatinklio programos ugniasienės vieta. Teoriškai ji yra tarp programos ir tinklo ugniasienės. Kuo programos ugniasienė yra arčiau programos tuo geriau.

Žiniatinklio programų ugniasienių ypatumai

- Gerai supranta HTTP;
- Gali būti naudojamos pasirinktinoms srauto dalims;
- Dirba tuomet kai srautas iššifruojamas;
- Galima prevencija.

Žiniatinklio įsibrovimų aptikimo-prevencijos (IDS/IPS) veikimo principai (strategijos):

Paprasta bazinė apsauga – palaiko tam tikras iš anksto apibrėžtas apsaugos nuostatas;

Paremta taisyklėmis – naudoja taisykles ieškoti pažeidžiamumų, arba taisyklėse ieškotų tam tikrų atakų klasių;

Paremta nukrypimais – bando koncentruotis į normalią operaciją, visa kita laikoma nukrypimu;

Neigiamo saugumo modelis – atmeta viską, kas pasirodo pavojinga, bet niekada negali žinoti visko kas galėtų būti pavojinga (Uždrausta viskas kas nėra leistina.);

Teigiamo saugumo modelis – leidžia viską kas yra pripažinta saugu, šis modelis yra geresnis už neigiamo saugumo modelį (Leistina viskas, kas nėra uždrausta).

2.5.3 Žiniatinklio programų ugniasienių palyginimas

REAL-WORL LABS [39] paskelbė keturių plačiai paplitusių žiniatinklio programų ugniasienių išsamų testavimą:

Testavimas buvo atliktas pakankamai realiomis sąlygomis. Buvo pakviesti atstovai iš kiekvienos ugniasienės kompanijų. Jie sukonfigūravo ugniasienes taip kaip jiems atrodė geriausia. Programos realizuotos ASP.NET, tarnybinėse stotyse veikė (IIS 6.0) ir SQL duomenų bazės. Kai kurioms ugniasienėms reikėjo laiko adaptuotis, realaus pasaulio funkcijoms imituoti buvo paleistos papildomos paprogramės. Pažeidžiamumams testuoti naudota AppScan 6.0 programinė įranga. Tai pakankamai universali auditui skirta programa, kuris apima visus pagrindinius ir nemažai papildomų pažeidžiamumų patikrinimų.

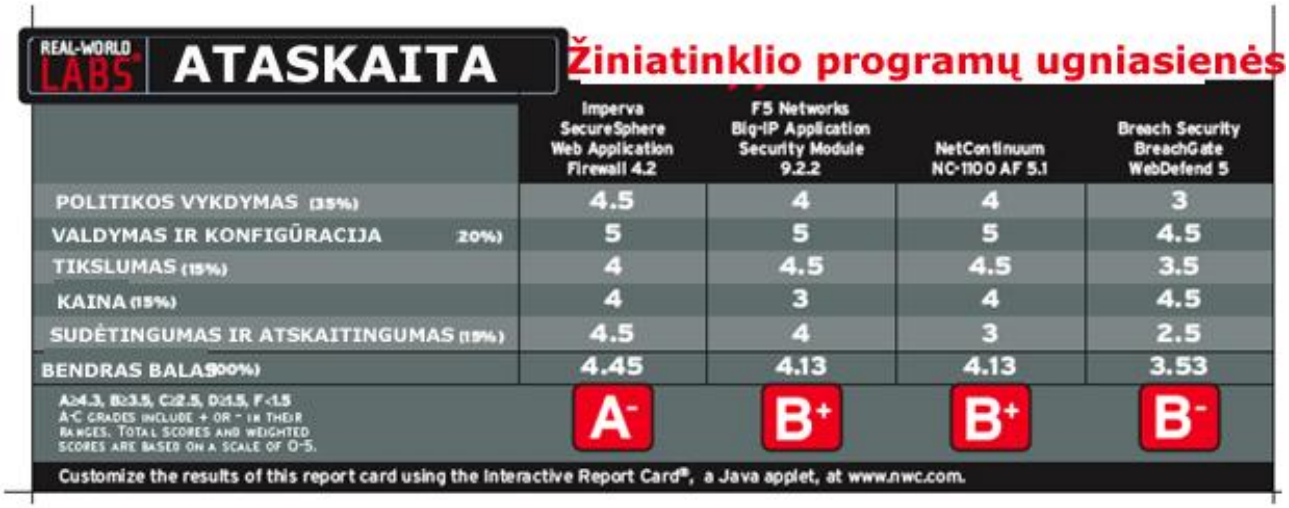
Eksperimento pradžioje buvo testuojama programa pavadinimu *AcmeHackme Web site* patikrinta be ugniasienių. Buvo rasta apie 8000 pažeidžiamumų.

Eksperimente nebuvo tikrinamas apkrova tarnybinei stočiai, kadangi dvi iš ugniasienių veikė kitu principu. Išskirti kriterijai ir įtaka balui procentais, vertinant penkių balų sistema.

Testuojamos ugniasienės:

- Imperva SecureSphere Web Application Firewall [40] 4.2 versija
- F5 Networks Big-IP Application Security Module [41] 9.22 versija
- NetContinuum NC-1100-AF [42] 5.1 versija
- Breach Security BreachGate WebDefend [43] 5 versija

4 paveiksle pateikti eksperimento rezultatai originalioje lentelėje, geriausią įvertinimą gavo SecureSphere produktas, prasčiausią Brench Security.



4 pav. ugniasienių testavimo rezultatai

Kitos dažnai naudojamos ugniasienės

PHPIDS [44**]** PHP programavimo kalba sukurtų programų stebėtojas, naujesnės versijos gali užtikrinti ir prevenciją.

PHPIDS funkcijos:

- Aptinka visų rūšių injekcijas;
- Bandymus prieiti prie neleistinų failų keičiant direktoriją (angl. directory traversal);
- Aptinka atsisakymo aptarnauti DoS ir LDAP atakas;
- Aptinka ir itin stipriai užmaskuotas atakas;
- Aptinka dar nežinomas atakas pagal tam tikrus šablonus.

PHPIDS pasirodė 2007 metų viduryje, kol kas yra tik trys stabilios versijos. Ši apsaugos priemonė yra nemokama.

ModSecurity [45] – atviro kodo žiniatinklio programų ugniasienė turi ir nekomercinę licenciją.

Veikia su JAVA, .NET, PHP.

Modsecurity funkcijos ir savybės:

- Auditų rašymas į žurnalą (angl. audit logging);
- Užtikrina priėjimą į bet kurią užklausos ir atsakymo dalį bei turinį;
- Lanksti regexp (angl. Regular expression-based) paremta taisyklių sistema;
- Galima taikyti išorinės apsaugos logiką;

- Taisyklės gali būti kombinuotos;
- Palaiko neribotą kiekį skirtingų politikų;
- Palaiko failo įkėlimo nutraukimą ir realų tikrinimą (antivirusinės sistemos integracija);
- Laukų tikrinimai;
- Buferio perpildymo apsauga.

Neigiama savybė:
Tai yra Apache modulis, kuris kiekvieną kartą turi būti užkraunamas, todėl veikimo sparta nėra tokia gera, kuria pasižymi alternatyvūs produktai pvz. PHPIDS.

Egzistuoja įvairaus tipo ugniasienių ar tik stebėjimo sistemų. Pagrindiniai keliami kriterijai: apsaugoti nuo gerai žinomų pažeidžiamumų, kuo tikslesnis veikimas, atskaitingumas ir kuo mažesnė apkrova – mažesnis vėlinimas. Kaip ir su bet kuriais produktais jie gali būti atviro kodo ir platinami nemokamai arba mokami sprendimai. Daugumos apsaugos sistemų trūkumas, kad juos saugo nuo pažeidimų apie kurių apsaugojimą įsibrovėliai žino ir net nebando vykdyti. Svarbu, kad būtų užfiksuotos bet kokio tipo užmaskuotos atakos, svarbu, kad nebūtų grąžinami pranešimai apie sistemos klaidas, nebent jie specialiai suformuluoti žiniatinklio programos. Dar vienas svarbus aspektas, suderinamumas su saugos politika ir atskaitingumas.

2.6 Žiniatinklio programų pažeidžiamumų aptikimo įrankių analizė

Norint patikrinti realizuotų žiniatinklio programų saugumą reikia atlikti auditą. Auditui atlikti naudojami specialūs įrankiai, kurie ieško jiems žinomų spragų požymių, dažniausiai tai būna speciali programinė įranga. Programų įvairovė gali būti labai didelė, nuo realizacijos skirtumų iki tarnybinių stočių ir naudojamų duomenų bazių ypatumų, todėl vieno universalaus įrankio nėra.

Auditą atliekančios programos ieškančios pažeidžiamumų gali:

- Aptikti pažeidžiamumą;
- Neaptikti pažeidžiamumo;
- Generuoti pranešimą apie netikrą pavojų;

Žiniatinklio programų pažeidžiamumų aptikimo įrankių palyginimas

Pateikiu trijų gerai žinomų audito programų palyginimą:

- Acunetix Web Security Scanner + Acusensor;
- IBM Rational AppScan;
- HP WebInspect.

Visų trijų kompanijų produktai atstovauja žinomus programinės įrangos gamintojus. Buvo testuojama įvairūs Javascript vykdomo kodo scenarijai bei dvylika skirtingų žiniatinklio gerai žinomų ir plačiai naudojamų žiniatinklio programų, ieškota gerai žinomų pažeidžiamumų. Vertinimas apskaičiuojamas: 5 balai už pažeidžiamumo aptikimą, -5 balai už neaptiktą pažeidžiamumą ir -1 balas už netikro pavojaus pranešimą.

Galutiniai audito įrankių testavimo rezultatai 5 lentelėje atskleidė, kurie įrankiai turėjo didžiausią pranašumą.

5 lentelė. *Žiniatinklio programų saugumo audito programų palyginimas*

Nr.	Testuota programa	Platforma	App Scan	Web Inspect	Acunetix	Acunetix + AcuSensor
1	Javascript testai	N/A		+		
2	Vanilla-1.1.4	PHP				+
3	VivvoCMS-3.4	PHP				+
4	fttss-2.0	PHP				+
5	Wordpress-2.6.5	PHP	Nebuvo aiškaus nugalėtojo			
6	vbulletin_v3.6.8	PHP	Nebuvo aiškaus nugalėtojo			
7	riotpix v0.61	PHP				+
8	javabb_v0.99	Java			+	
9	Yazd Discussion Forum_v3.0	Java	+	+		
10	pebble_v2.3.1	Java	Nebuvo aiškaus nugalėtojo			
11	TriptychBlog_v.9.0	ASP.NET	Nebuvo aiškaus nugalėtojo			
12	DMG Forums_v3.1	ASP.NET				+
13	Dave's CMS_v2.0.2	ASP.NET	Nebuvo aiškaus nugalėtojo			
14	Acunetix Demo Application - Acunetix Acuart	PHP				+
15	AppScan Demo Application - Altoro Mutual	ASP.NET	+			
16	WebInspect Demo Application - free Bank online	ASP		+		
	Viso (+)		**2**	**3**	**7**	

Iš 5 lentelės matyti, kad geriausiai su užduotimis susidorojo Acuentics + Acusensor produktas. Visos pažeidžiamumų tikrinimui skirtos programos turi trūkumų, šis palyginimas parodė jog negalima pasitikėti tik viena auditavimo programa.

2.7 Analizės išvados

Nagrinėjant žiniatinklio taikomąsias programas tapo aišku, kad tipiškos tinklo lygmenyje veikiančios ugniasienės žiniatinklio programų apsaugoti negali. Norint įgyvendinti saugumo reikalavimus žiniatinklio programose reikia naudoti atitinkamas saugumą užtikrinančias priemones. Pro tinklo ugniasienes einantis šifruotas srautas negali būti tinkamai filtruojamas. Tą padaryti turi žiniatinklio programų lygmenyje veikiančios ugniasienės. Dažniausiai žiniatinklio programa yra tarpinė grandis, tarp vartotojo ir duomenų bazės veikiančios įmonės tinklo ribose. Problema ta, kad toje pačioje duomenų bazėje saugomi įvairaus tipo duomenys, kurie klasifikuojami pagal tris kriterijus. Priklausomai nuo duomenų klasės, su jais gali dirbti tam tikros grupės žmonės. Čia ir iškyla pagrindinė problema, vartotojai iš kitos grupės siekia gauti informaciją, kuri jiems turėtų būti nepasiekiama. Piktavaliai bando pasinaudoti žiniatinklio programų spragomis, bandydami įterpti ar vykdyti neleistinas komandas. Norint apsisaugoti programose naudojamos įvairaus tipo stebėjimo ir prevencinės priemonės. Praktiškiausias būdas naudoti žiniatinklio programų ugniasienę. Rekomenduojama, kad pirmiausia būtų imtasi ne apsaugoti spragą, bet bandyti ją užtaisyti pačioje programoje. Norint aptikti saugumo spragas programose naudojamos įvairios audito priemonės. Šioje darbo dalyje ištirtos svarbiausios žiniatinklio taikomųjų programų grėsmės, jų aptikimo būdai, apsaugos metodai ir priemonės. Tiriant saugumo grėsmes pagrindinis dėmesys skirtas grėsmių aktualumui laike. Analizuojant informaciją paaiškėjo, kad populiariausių grėsmių sąrašas kinta. Atsižvelgiant į pasikeitimus išanalizuoti gerai žinomi iš anksčiau ir naujausi apsaugos metodai.

Išanalizavus kylančias grėsmes ir apsaugos priemones nuspręsta pasiūlyti žiniatinklio taikomųjų programų ugniasienės modelį, kuris išnaudotų naujausius ir iki šiol gerai žinomus apsaugos metodus. Ypatingas dėmesys skiriamas svarbiausiems kokybės kriterijams bei kovai su pačiais aktualiausiais pažeidžiamumais.

3. ŽINIATINKLIO UGNIASIENĖS PROJEKTAS

3.1 Darbo tikslas ir keliami reikalavimai

Darbo tikslas – remiantis žiniatinklio programų grėsmių, apsaugos priemonių analizės duomenimis sukurti žiniatinklio programų ugniasienės modelį.

Pagrindinis šioje darbo dalyje sprendžiamas uždavinys – žiniatinklio taikomųjų programų ugniasienės modelio sudarymas. Pagrindinis žiniatinklio programų ugniasienei keliamas uždavinys – susidoroti su analizės dalyje ištirtomis grėsmėmis panaudojant geriausius apsaugos metodus.

Ugniasienės tikslas nėra vien tik nukenksminti galimą ataką, svarbu kad būtų sustabdyti tolimesni atakuotojo ketinimai, apie pažeidimus informuotos atitinkamos institucijos.

3.1.1 Reikalavimai duomenims

Duomenys priimami iš HTTP užklausų GET, POST, REQUEST, COOKIE. Priimami duomenys nešifruoti.

3.1.2 Funkciniai ir nefunkciniai reikalavimai

Funkciniai reikalavimai keliami žiniatinklio ugniasienei

- Įeinančių ir išeinančių duomenų tikrinimas nuo gerai žinomų pažeidžiamumų;
- Operavimas XML duomenimis;
- Pažeidžiamumų duomenų bazės reguliarus atnaujinimas;
- Užklausų filtravimas;
- Tam tikros prevencinės galimybės;
- Pažeidžiamumų klasifikavimas;
- Reagavimas atsižvelgiant į pažeidžiamumo klasę (sesijos nutraukimas, įrašymas į žurnalą, informavimas elektroniniu paštu, informavimas trumpąja SMS žinute);
- Galimybė keisti taisykles;
- Įeinančių ir išeinančių duomenų tikrinimas nuo gerai žinomų pažeidžiamumų;
- Automatizuotų užklausų aptikimas;
- Teigiamas ir neigiamas saugumo modelis.

Nefunkciniai reikalavimai

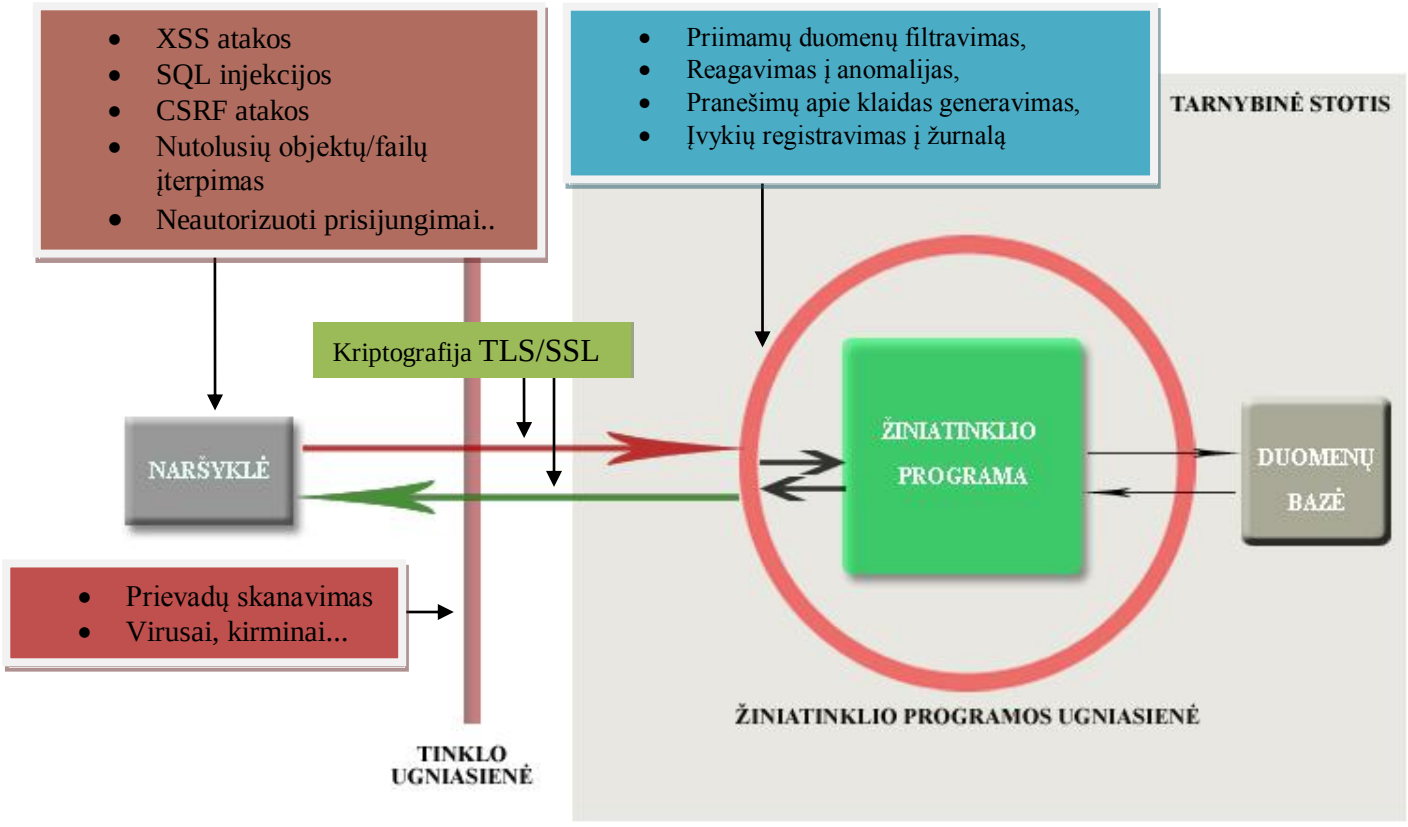
- Ugniasienė atviro kodo;
- UTF-8 koduotės palaikymas;
- Grafinė vartotojo sąsaja;
- Galimybė konfigūruoti ir valdyti visus elementus;
- Suderinamas su PHP, Apache, MySQL.

3.1.3 Rezultato kokybės kriterijai

Ugniasienės pagrindiniai teigiami kokybės kriterijai:

- Platus pažeidžiamumų aptikimo spektras;
- Kuo mažiau klaidingų pranešimų;
- Maža apkrova tarnybinei stočiai;
- Mažas žiniatinklio programos atsakymų vėlinimas – veikimo sparta;

3.2 Sistemos architektūra - statinės struktūros modelis



5 pav. Žiniatinklio programos ugniasienės architektūra

Statinės struktūros modelyje 5 pav. matyti, kad grėsmės su kuriomis kovoja ugniasienė ateina iš vartotojo naršyklės. Šiuo atveju naršyklė apibendrintai apima ne tik interneto naršyklių programas, bet ir įvairius srautų generatorius, kurie imituoja naršykles.

Žiniatinklio programų ugniasienė yra įterptinė programa, kuri naudojama kaip sudėtinė žiniatinklio programos dalis. Žiniatinklio programų ugniasienė apsupa pačią programą, kadangi pirmiausiai visi gaunami parametrai praeina pro ugniasienę. Ugniasienė, priklausomai nuo konfigūracijos gali nepasitikėti ir filtruoti visus įeinančius duomenų srautus taip apsaugodama žiniatinklio programą, tarnybinę stotį ir duomenų bazę nuo galimų grėsmių.

Ugniasienės architektūra yra trijų lygių:
Vartotojo – jame pateikiama ugniasienės valdymo sąsaja vartotojui. Vartotojas turi galimybę valdyti modulius, keisti jų parametrus, valdyti taisykles ir veikimą.
Logikos – šiame lygmenyje veikia visos funkcijos, tai yra modulio šerdis.
Duomenų – duomenų lygmenyje saugomi konfigūraciniai parametrai, taisyklės ir kita susijusi informacija.

3.3 Žiniatinklio programų ugniasienės sudedamosios dalys – moduliai

Žiniatinklio ugniasienė sudaryta iš atskirtų modulių. Kiekvieno modulio paskirtis atlikti tam tikras su saugumu susijusias užduotis. Modulius galima suskirstyti į dvi grupes:

1. **Prevenciniai moduliai** – moduliai skirti aptikti ir pašalinti grėsmes susijusias su pažeidžiamumais žiniatinklio taikomajai programai.

Pagrindiniai prevenciniai moduliai:

- SQL injekcijų prevencinis modulis;
- XSS atakų prevencinis modulis;
- CSRF tipo atakų prevencinis modulis;

Galima reakcija į pažeidimą:

- Blokuoti HTTP užklausą;
- Blokuoti susijungimą;
- Blokuoti IP adresą;
- Blokuoti žiniatinklio programos sesiją;
- Blokuoti žiniatinklio programos vartotoją.

2. **Valdymo moduliai** – vartotojo sąsajos, įvykių registravimo, ataskaitų teikimo, duomenų bazių filtrų atnaujinimų ir kiti moduliai.

Ugniasienės išskaidymas į atskirus modulius supaprastina kiekvieno modulio vystymą, leidžia geriau suprasti ugniasienės darbą.

SQL injekcijų prevencinis modulis

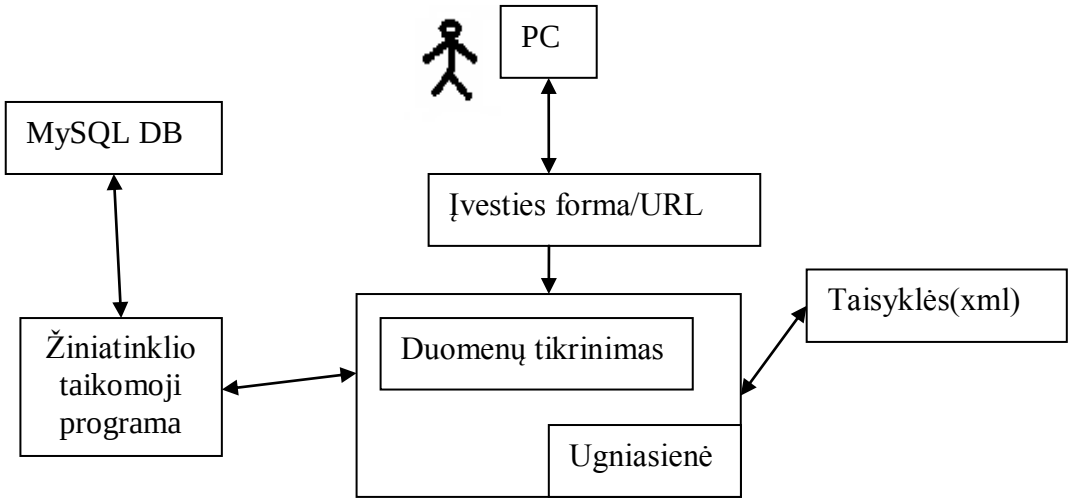
SQL injekcijų prevencinis modulis atlieka apsaugos nuo galimų SQL injekcijų funkciją. Modulis veikia ten kur priimami duomenys dalyvauja SQL užklausų į duomenų bazę formavime.

Iš įvesčių ar URL priimtos GET, POST parametrų reikšmės analizuojamos, esant SQL injekcijos požymiui generuojamas pranešimas apie klaidą į klaidų valdymo modulį. Jeigu SQL modulis gautuose parametruose neaptinka SQL injekcijų požymių atlieka pavojingų simbolių nukenksminimo funkcijas ir perduoda reikšmes žiniatinklio programai.

Modulis įgyvendina teigiamo ir neigiamo tipo saugumo politiką. Naudojant XML taisykles galima įgyvendinti „baltųjų sąrašų“ politiką.

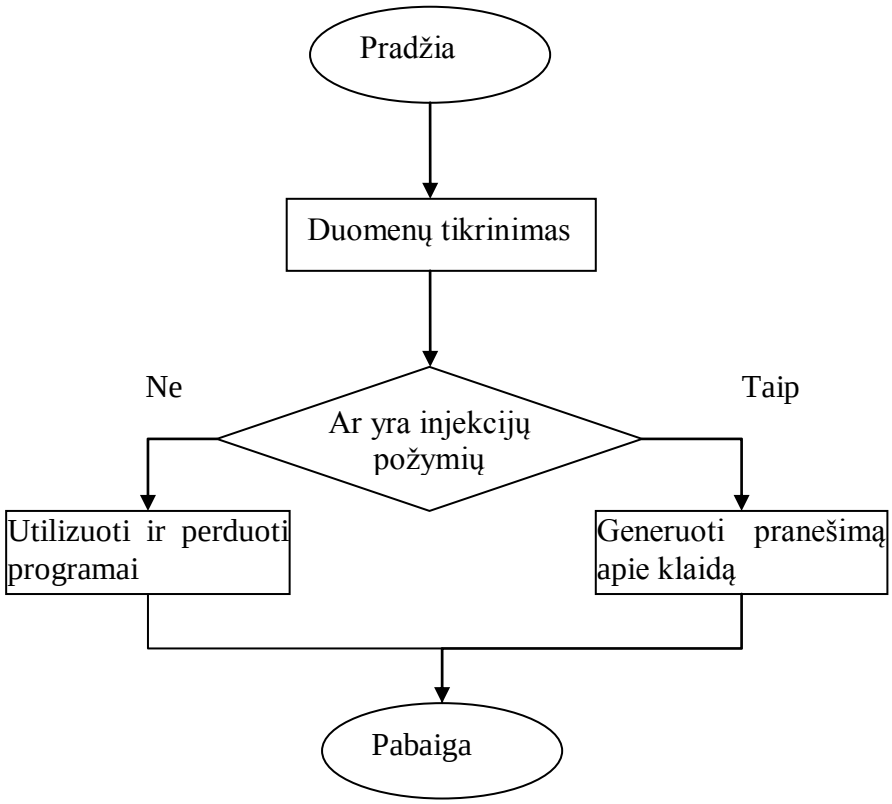
Kabučių išvalymo funkcija nukenksmina kabutes, kurios paprastai gali atskirti SQL užklausas ar jų reikšmes. Ši išvalymo funkcija gali būti implementuota programoje, todėl jos veikimą galima išjungti.

SQL injekcijų prevencinio modulio veikimo modelis pavaizduotas 6 pav. Iš vartotojo atkeliaujantys nepatikimi duomenys tikrinami ugniasienės naudojant taisykles xml formatu. Jeigu modulis pažeidimų neužfiksuoja, toliau gautus duomenis perduoda žiniatinklio programai.



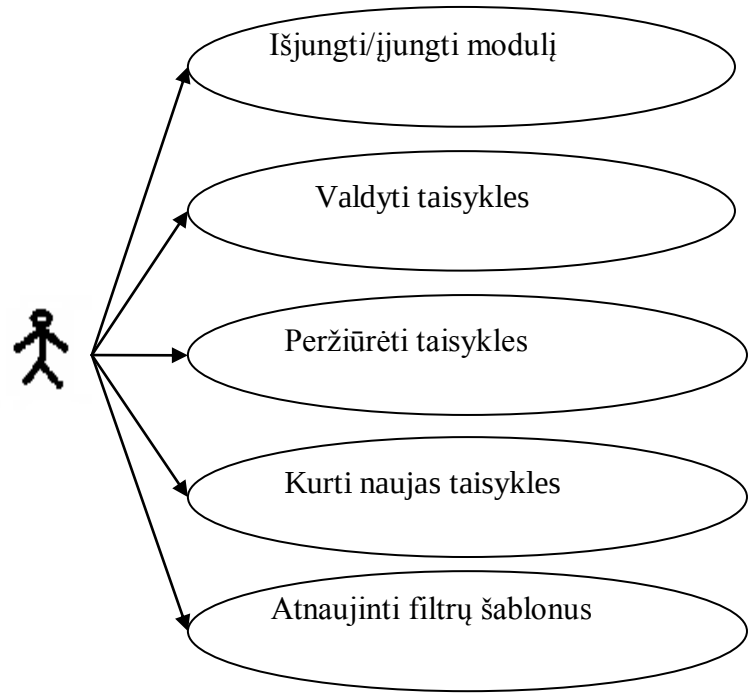
6 pav. SQL injekcijų prevencinio modulio veikimo modelis

Supaprastinta modulio veikimo schema 7 pav. parodo modulio veikimo principą. Paprastai nesant tinkamam klaidų valdymui, žiniatinklio programa susidūrusi su SQL užklausos problemomis grąžina klaidą apie nepavykusią užklausą į duomenų bazę. Naudojant ugniasienės modulį aptikus injekcijų požymių, generuojamas specialus pranešimas apie klaidą. Jeigu duomenys ir neatitinka injekcijų požymių, jie papildomai utilizuojami norint visiškai apsidrausti nuo galimų klaidų.



7 pav. Supaprastinta modulio veikimo schema

SQL injekcijų prevencinio modulio panaudojimo atvejų diagramoje 8 pav. parodyti veiksmai, kuriuos galima atlikti grafinės vartotojo sąsajos pagalba. Pagrindiniai atliekami veiksmai susiję su taisyklių valdymu. Panaudojant taisykles galima įgyvendinti teigiamo ir neigiamo saugumo modelio filtravimą.



8 pav Modulio konfigūracijos valdymo panaudojimo atvejų diagrama

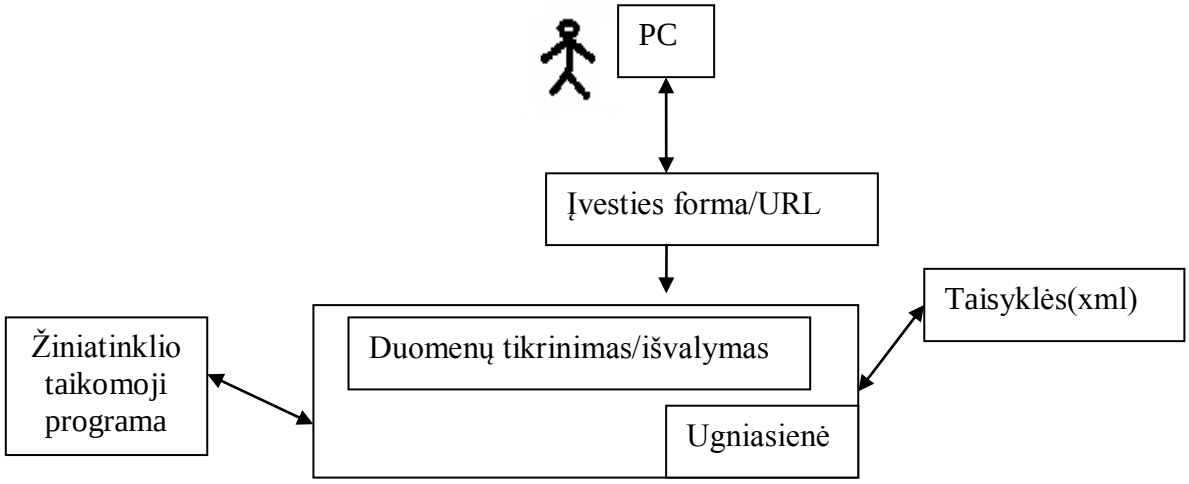
XSS prevencinis modulis

XSS atakos iš esmės yra specialus injekcijų atvejis, todėl kovai su XSS priemonės panašios į SQL injekcijų prevencinio modulio funkcijas.

- Priimamų duomenų filtravimas pagal šablonus ieškant galimų XSS atakų grėsmių požymių;
- Specialių simbolių filtravimas;
- Duomenų gaunamų iš nepatikimų šaltinių nukenksminimas ir išvalymas;

XSS turi taisyklių šablonų duomenų bazę, kuri saugoma XML formatu kartu su kitomis taisyklėmis. Priimdamas XSS modulis duomenis patikrina ar jie neatitinka tam tikrų požymių. Taisyklės sugrupuotos pagal grėsmę. XSS duomenų bazė atnaujinama reguliariai arba rankiniu būdu, naujas taisykles sukurti gali ir ugniasienės administratoriai.

XSS prevencinio modulio veikimo modelis 9 pav. parodo kad veikimo principas toks pats kaip ir SQL prevencinio modulio.



9 pav. XSS prevencinio modulio veikimo modelis

Jeigu modulis aptinka XSS pažeidžiamumą priimamuose iš vartotojo duomenyse, jis jį nukenksmina ir užregistruoja bandymą pažeisti sistemą, priklausomai nuo konfigūracijos bandymų skaičius ribotas, pažeidėjo IP adresas gali būti įtraukiamas į blokuotųjų IP sąrašą, o vėliau žurnale administratorius pamatęs apie vykusius pažeidimus galės imtis papildomų veiksmų.

CSRF tipo atakų prevencinis modulis

- HTTP referer antraštės tikrinimas

Jeigu HTTP referer antraštė neatitinka modulio konfigūracijoje nurodyto adreso programai neleidžiama veikti toliau.

- Slaptų raktų naudojimas

Gavus informaciją iš GET arba POST patikrinamas raktas. Taip įsitikinama, kad duomenys ateina tikrai iš formos svetainėje kurioje yra programa, o ne sugeneruotos dinamiškai.
Slaptas raktas susietas su slapuku naudojamas atpažinti duomenims, patikrinama kad duomenys atėjo iš tos pačios formos svetainėje, o ne iš kitur dinamiškai sugeneruojant URL.
Slaptas raktas naudojamas tarp programos ir ugniasienės. Programai leidžiama aptarnauti tik ugniasienės patikrintus vartotojų duomenų srautus.

Nutolusių failų vykdymo prevencinis modulis

Modulis tikrina įkeliamų duomenų failų tipą, galima būtų vykdyti dvejopą failų leidimų nustatymą. Galimybė leisti naudoti tik vienos rūšies failus pvz. *.doc, .xls* ar *.ppt.*

Tačiau jeigu failų plėtiniai praktiškai neribojami, tuomet uždraudžiama tik keletas pavojingų vardų tokių kaip *.php, .pl* ar *.js*. Kadangi juos įkėlus į serverį būtų galima vykdyti komandas ir pakenkti visos programos, o gal net ir sistemos darbui.
Modulis failą filtruoja ne tik, pagal jo plėtinį bet ir pagal turinį skaitydamas meta informaciją. Radus pažeidimų generuojamas nustatytas pranešimas apie klaidą, atliekami kiti prevenciniai veiksmai.

Aktyvaus audito modulis

Modulis analizuoja Alache *error_log* žurnalo failą ir įvykių registracijų žurnalą. Ieškodamas anomalijų priima veiksmus, blokuoti IP adresus, bei nusiunčia pranešimą Klaidų valdymo moduliui.

Konfigūraciniai modulio parametrai:

- Maksimalus užklausų skaičius per sekundę/minutę;
- IP blokavimo laikas;
- Analizės pauzės (sekundės);
- Leidžiamų IP sąrašas (pvz. paieškos sistemos).

Šis modulis gali aptikti automatizuotas užklausas įvertindamas pauzes tarp užklausų, konkurencinių užklausų skaičių ir pan.

Klaidų valdymo ir registravimo moduliai

Šis modulis atsakingas už klaidų susijusių su ugniasienę valdymą. Duomenų bazėje saugo pranešimus apie klaidas. Gavęs klaidos kodą jis grąžina specialų pranešimą apie klaidą galutiniam vartotojui, taip pat tą pranešimą įrašo į žurnalą priskirdamas tam tikrai grupei. Klaida reiškia galimą pažeidimą ar netinkamą programos darbą, administratorius vėliau peržiūrės įrašą, ir galės priimti sprendimą kaip į jį reaguoti. Galima informuoti CERT, arba paskelbti tokių klaidų pranešimus nepavojingais. Tai padės sumažinti netikrų pavojų skaičių ateityje.
Įvykių registravimo modulio ypatumai:

- Įvykių žurnalą galima eksportuoti kaip failą įvairiais formatais (*MS Word, PDF, HTML, xml*);
- Įvykių žurnalo renkami laukai gali būti keičiami;

- Įvykių žurnalas gali būti persiųstas į įvykių registravimo tarnybinę stotį pasitelkiant *Syslog*;
- Įvykių žurnalas periodiškai gali būti persiunčiamas į kitą failų serverį pasinaudojant FTP, SFTP ar kitais protokolais;
- Įvykiai kaupiami XML formatu.

Pranešimo apie klaidas modulio ypatumai:

- Pranešimas el. paštu;
- Pranešimas trumpąja SMS žinute;
- Syslog;
- SNMP Trap.

Konfigūracijos valdymo modulis

Šis modulis analizuoja naudojamos programinės įrangos versijas. Reguliariai tikrinama ar yra išleistos naujos naudojamos programinės įrangos versijos, jeigu naudojama versija nesutampa su naujausia išleista versija, apie tai informuojama ugniasienės administracija, kuri savo ruožtu gali informuoti tarnybinės stoties administraciją apie tai jog naudojama pasenusi programinė įranga, kuri galimai nesaugi. Modulis taip pat atsakingas už pažeidžiamumų filtrų duomenų bazės atnaujinimą.

Pagrindinė modulio funkcija analizuoja konfigūracinį failą. Naudojant PHP platformą reikiamą informaciją pateikia *phpinfo* funkcija. Iš šios informacijos galima sužinoti tarnybinės stoties programinės įrangos (pvz. *Apache*) versiją taip pat PHP ir MySQL programinės įrangos versijas.

3.4 Grafinė vartotojo sąsaja

Ugniasienės valdymo supaprastinimui reikalinga grafinė vartotojo sąsaja.

Pagrindinės grafinės sąsajos sudedamosios dalys:

- **Vartotojų valdymas**

Informacija apie prisijungusį administratorių, prisijungimų istorija, nesėkmingi bandymai, jungimosi IP adresai. Galimybė valdyti administratorius, kurti naujus ir šalinti esamus.

- **Modulių valdymas**

Grafinės vartotojo sąsajos pagalba ugniasienės administratoriai gauna prieigą prie kiekvieno atskiro modulio valdymo. Kiekvieno modulio veikimas gali būti išjungtas, modifikuojama modulių elgsena, pažymima ką laikyti netikru pavoju.

- **Įvykių žurnalas**

Įvykių žurnalas pateikia agreguotą informaciją apie pažeidimus, su reakcijų į pažeidimą parinktimis:

Užfiksuoti pažeidimai [Data – IP – URL – informacija apie pažeidimą]

Galimi papildomi pasirinkimai:

Pranešti CERT arba pažymėti pažeidimą kaip netikrą pavojų.

Taip pat yra galimybė išjungti klaidų registravimo įrankį. Keisti registravimo įrankio taisykles.

- **Blokuotų vartotojų sąrašo valdymas**

Blokuotų vartotojų sąraše matoma informacija apie tuos vartotojus kuriems buvo apribota prisijungimo prie žiniatinklio programos teisė. Į blokuotų vartotojų sąrašą patenka tie vartotojai, kurių elgsena atitiko nustatytus pažeidimų požymius.

Ugniasienės administratorius išanalizavęs matomą informaciją gali atblokuoti vartotoją, pašalindamas jo IP iš sąrašo arba informuoti apie pažeidimą CERT arba pranešti IP adresą administruojančiai bendrovei.

- **Pranešimo apie incidentą siuntimas**

CERT-LT [46] yra Lietuvos Respublikos nacionalinis elektroninių ryšių tinklų ir informacijos saugumo incidentų tyrimo padalinys, kurio misija yra užtikrinti elektroninių ryšių tinklų ir informacijos saugumo incidentų tyrimus, koordinuoti veiksmus stabdant incidentų plitimą ir vykdyti incidentų prevenciją.

Pranešimo sandara:

- Vardas, pavardė;
- El. paštas;
- Telefonas;
- Organizacija, pareigos;
- Incidento tipas;
- Incidento data;
- Incidento aprašymas.

Speciali funkcija pagal iš anksto nustatytą konfigūraciją automatiškai sugeneruoja pranešimo statinius laukus, administratoriui tereikia įvesti detalesnį incidento aprašymą, kuriame iš anksto būna pateikiama informacija iš žurnalo.

Galimo pranešimo pavyzdys

Jonas Jonaitis

jonas@ofisas.lt

+37011102304

UAB „Ofisa“, IT administratorius

Elektroninės paslaugos trikdymo ataka

[11/Jun/2009:00:17:13 +0300]

IP: 78.57.4.152 "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; InfoPath.1)"

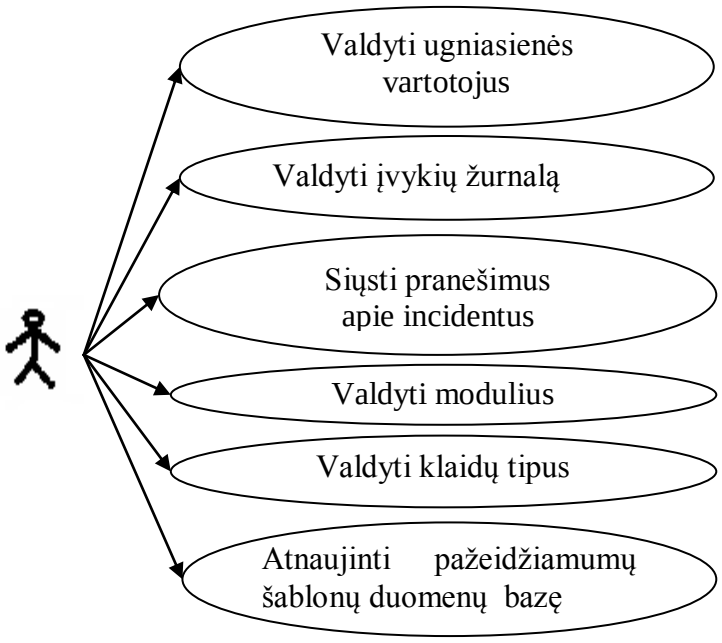
Bandyta naudoti XSS ataką: perl -e 'print "<SCR\0IPT>alert(\"XSS\")</SCR\0IPT>";' > out

Šis tekstas siunčiamas nustatytu el. paštu į *cert@cert.lt* automatiškai iš sistemos vos tik administratorius priima sprendimą informuoti apie pažeidimą. Priklausomai nuo aplinkybių el. pašto adresas gali būti keičiamas.

Šis pavyzdys skirtas Lietuvai, tačiau kiekvienoje šalyje už saugumą Internete atsakingų institucijų.

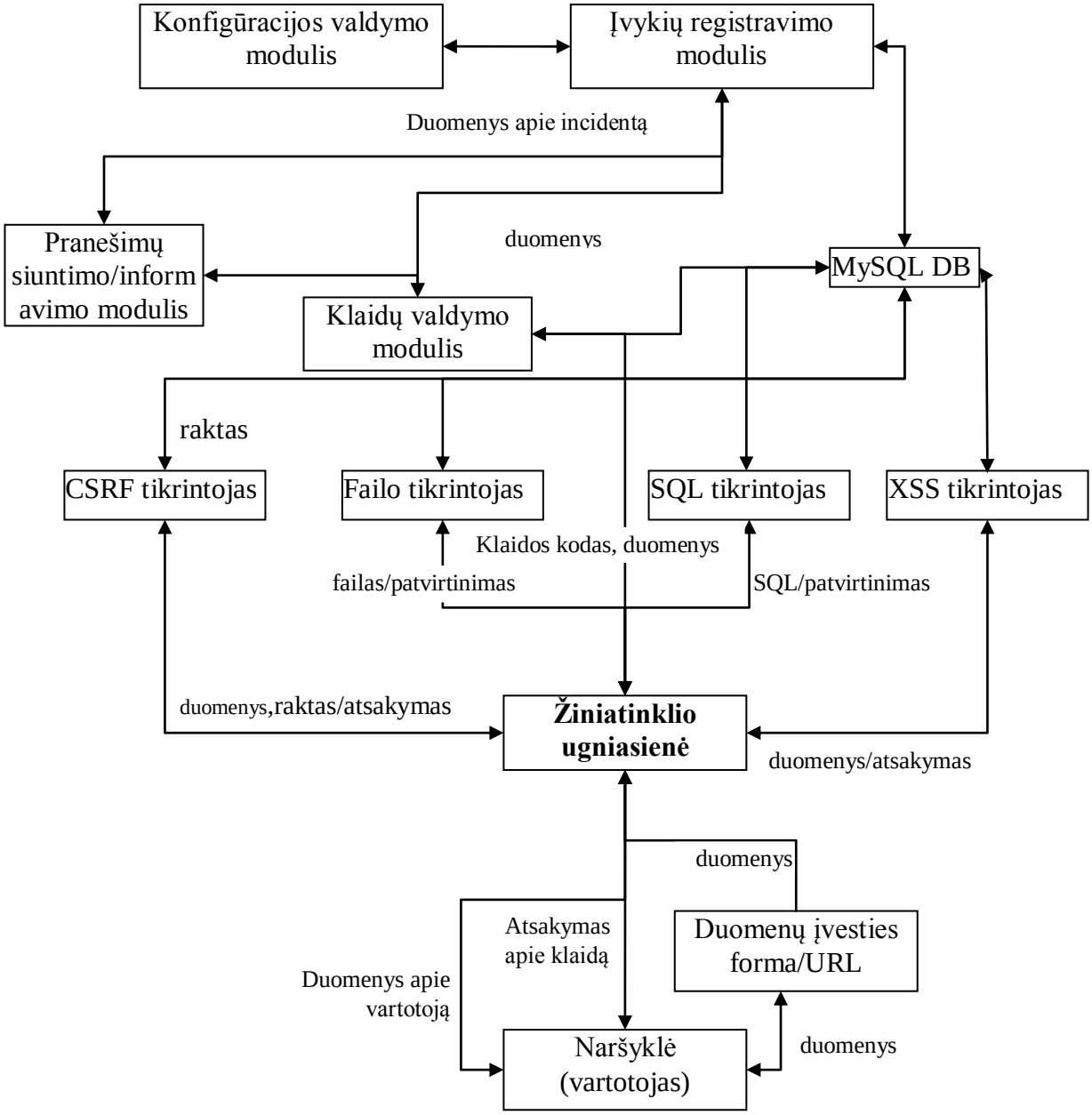
3.5 Konfigūracijos valdymo panaudos atvejų diagrama

10 pav. Pavaizduotoje konfigūracijos valdymo panaudos atvejų diagramoje matyti pagrindiniai žiniatinklio ugniasienės panaudojimo atvejai.



10 Pav. Konfigūracijos valdymo panaudojimo atvejų diagrama

3.6 Funkcinė ugniasienės diagrama



11 Pav. Žiniatinklio ugniasienės funkcinė diagrama

11 paveiksle pavaizduota funkcinė žiniatinklio ugniasienės diagrama. Žiniatinklio ugniasienė atsižvelgdama į konfigūraciją nukreipia srautus tam tikriems moduliams. Tikrintojai – prevenciniai ugniasienės moduliai. Aptikę klaidų perduoda pranešimą klaidų generavimo moduliui. Klaidų generavimo modulis atsižvelgdamas į klaidą perduoda reikiamus duomenis įvykių registravimo moduliui, bei pranešimų siuntimo/informavimo moduliui, jeigu apie tam tikrą pažeidimą reikia informuoti. Jeigu neaptinkama jokių pažeidimų, toliau perduodami

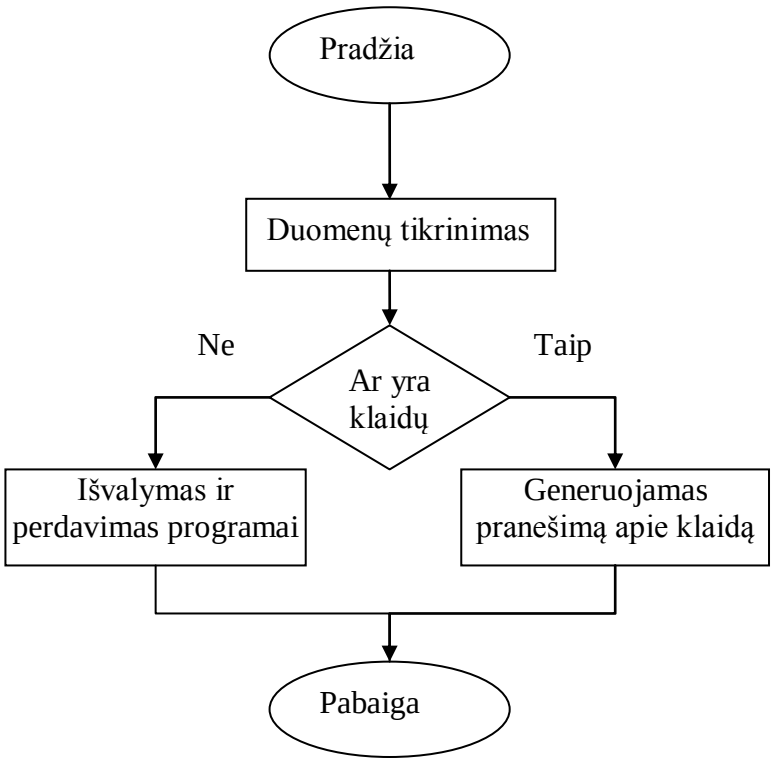
duomenys žiniatinklio programai. Detalesnis veikimas aprašytas ugniasienės veikimo aprašyme.

3.7 Ugniasienės veikimo aprašymas

Aprašoma seka kaip veikia žiniatinklio programų ugniasienė, kokie parametrai priimami apie programos naudotoją taip pat kokie galimi veiksmai.

1. Priimami duomenys iš vartotojo naršyklės: HTTP referer, formos raktas, IP, Reikšmės iš formų (GET, POST reikšmės), failas ir kt.
2. Kreipiamasi į atitinkamus modulius perduodant jiems turimą informaciją;
3. Moduliai priima duomenis sau suprantamu formatu ir siunčia atsakymą;
4. Priklausomai nuo modulio ugniasienė įsirašo klaidos kodą;
5. Jeigu klaidų nėra leidžiama veikti programai;
6. Jeigu klaidų yra kreipiamasi į klaidų valdymo modulį, priklausomai nuo klaidos kodo jis grąžina klaidos pranešimą, įrašo į žurnalą ir imasi papildomų nustatytų veiksmų.

Supaprastinta ugniasienės veikimo schema



12 Pav. Supaprastinta ugniasienės veikimo schema

12 paveiksle pateikta supaprastinta ugniasienės veikimo schema, kai priimami duomenys iš formų. Atliekamas įvestų duomenų filtravimas, jeigu klaidų nerandama, įvesti duomenys išvalomi ir perduodami toliau žiniatinklio programai. Priešingu atveju vartotojui siunčiamas pranešimas apie klaidą, kurį sugeneruoja žiniatinklio programos ugniasienė.

3.8 Išvados

Sukurtas žiniatinklio programos ugniasienės modelis. Ugniasienė sudaryta iš atskirų komponentų – modulių, grafinės vartotojo sąsajos ir pažeidžiamumų duomenų bazės. Ugniasienė naudoja naujausius metodus kovai su populiariausiais žiniatinklio programų pažeidžiamumais. Didelis dėmesys skirtas ugniasienės atskaitingumui, operatyviai reakcijai į galimas grėsmes. Ugniasienės architektūra yra trijų lygių: vartotojo, logikos ir duomenų. Pagrindiniai ugniasienės moduliai yra prevenciniai, jie užtikrina aktyvią žiniatinklio programos apsaugą nuo pačių svarbiausių grėsmių. Ugniasienės valdymo moduliai stebi aplinką kurioje veikia pati ugniasienė, tikrinama ar naudojama naujausia programinė įranga. Svarbus elementas, galimybę operatyviai reaguoti į incidentus modulis. Administracijai supaprastinamas pranešimo siuntimas už saugą Interneto erdvėje atsakingoms institucijoms.

4. ŽINIATINKLIO UGNIASIENĖS REALIZACIJA

Šioje darbo dalyje pateikiamos realizuojamo SQL prevencinio modulio detalės. Parengiamos SQL prevencinio modulio pagrindinės naudojamos funkcijos, realizuojamas grafinė vartotojo sąsaja, modulis parengiamas eksperimentui: parenkami testavimo įrankiai ir iškeliami kokybės kriterijai.

4.1 SQL injekcijų prevencinis modulis

SQL injekcijų prevencinio modulio vaidmuo žiniatinklio ugniasienėje

Atsižvelgiant į tai jog SQL ir kitokios injekcijos užima pirmą vietą tarp didžiausių grėsmių žiniatinklio programoms, SQL injekcijų prevencinis modulis turi būti neatsiejama žiniatinklio programos ugniasienės dalis. Šis modulis turi būti tarpinė stotis tarp duomenų priimamų iš vartotojo užklausai į duomenų bazę suformuoti.

Įprasta žiniatinklio programos veikimo seka be ugniasienės

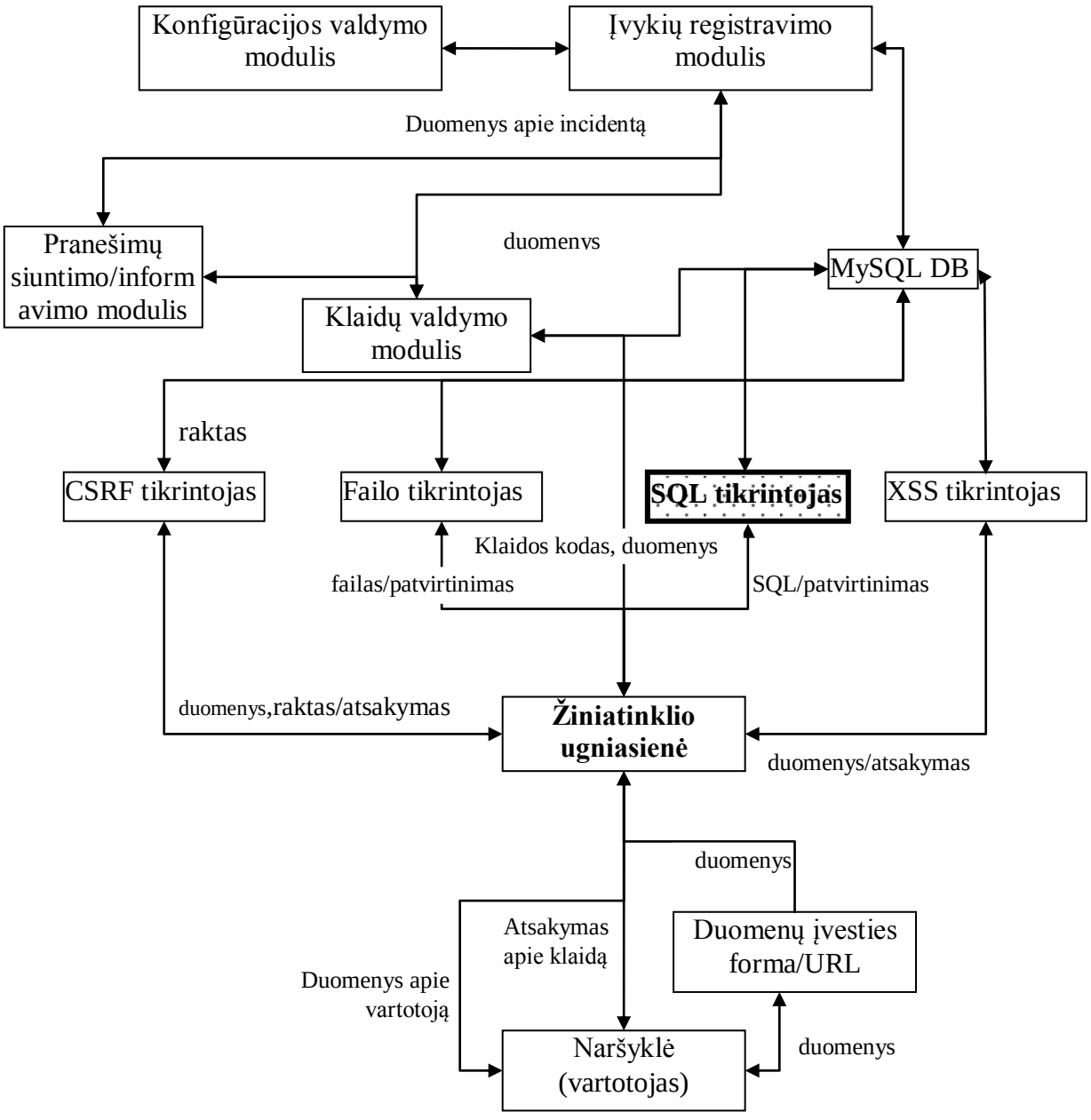
Vartotojas užpildo formą žiniatinklio programoje: dažniausiai tai autentifikavimo formos, paieškos laukai ar pan.. Kitu atveju gali būti tiesioginiai įterpimai į URL kuomet parametrai atvaizduojami adrese. Programa priima užduotus parametrus paprastai GET arba POST metodu.Pateiktus parametrus sustato į iš anksto paruoštas SQL užklausas ir siunčia jas į duomenų bazę, gautą atsakymą pateikia vartotojui į naršyklės langą.

Vykdant tokį besąlygišką pasitikėjimą vartotoju gali būti išnaudotos SQL injekcijos, kuomet programai pateikiami duomenys suteikia galimybė prieiti prie neautorizuotų duomenų. Tokiu atveju užklausa prieinanti prie visos duomenų bazės gali atskleisti, modifikuoti ar visiškai pašalinti konfidencialius arba svarbius duomenis. Tam jog tai nenutiktų būtinas įvesčių

tikrinimas. Pagrindinis SQL injekcijų prevencinio modulio uždavinys tikrinti priimamus duomenis iš kurių formuojamos SQL užklausos. Radus pažeidimų kreipimosi į SQL duomenų bazę nevykdyti, tačiau bet kokiu atveju vykdyti priimtų duomenų išvalymą ir nukenksminimą tam kad galima grėsmė būtų sumažinta iki minimalaus lygio.

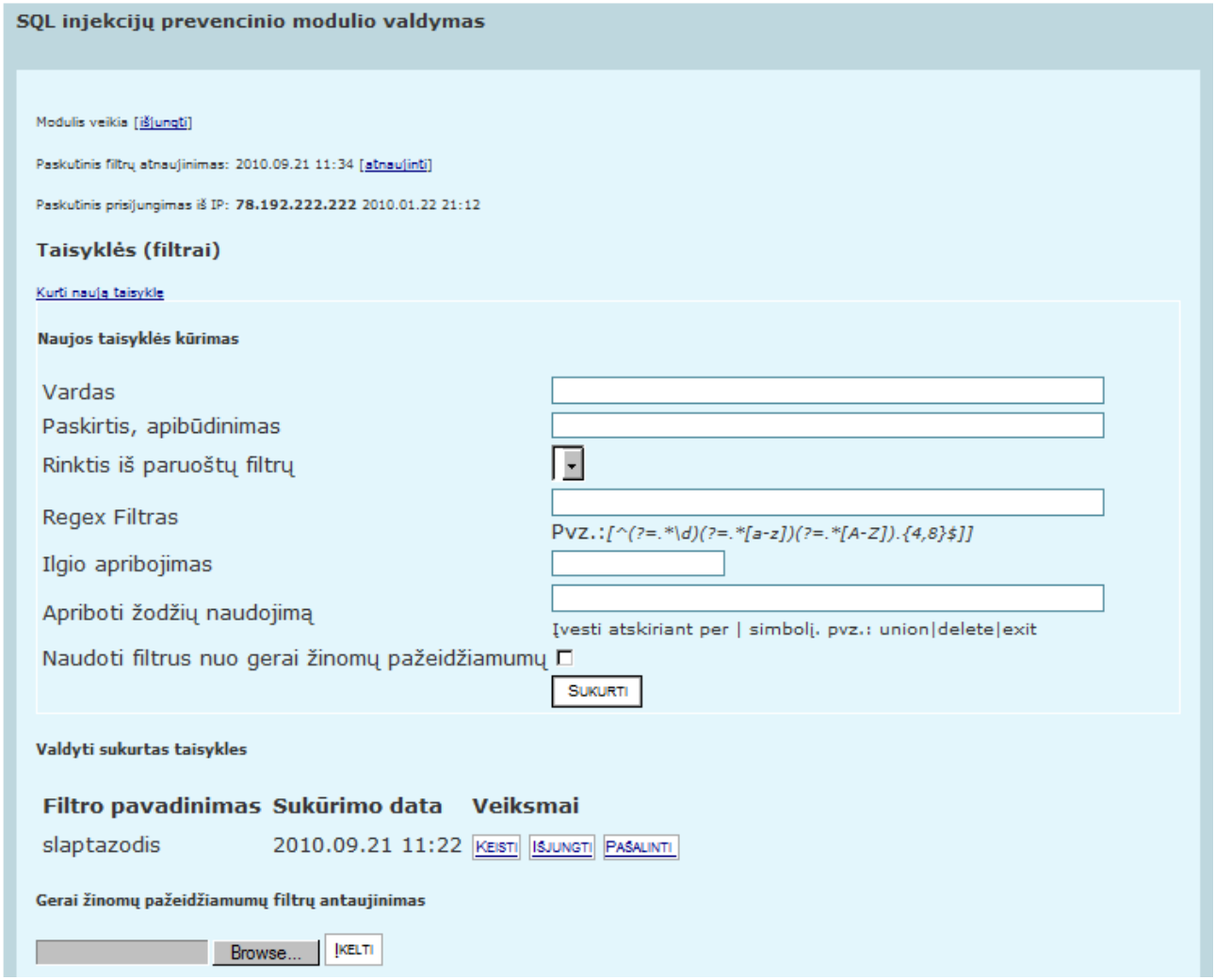
4.2 Modulio vieta žiniatinklio ugniasienės modelyje

13 paveiksle vaizduojamas SQL tikrintojas yra SQL injekcijų prevencinis modulis.



13 Pav. Modulio vieta žiniatinklio ugniasienės modelyje

4.3 Grafinė vartotojo sąsaja



14 pav. SQL prevencinio modulio vartotojo sąsaja

14 paveiksle pavaizduota žiniatinklio programos ugniasienės SQL prevencinio modulio valdymo grafinė vartotojo sąsaja. Iš paveikslo matyti kad informacija apie vartotoją, naujos taisyklės kūrimo pavyzdys, viena jau sukurta taisyklė, taip pat gerai žinomų filtrų šablonų atnaujinimo sąsaja. Grafinė vartotojo sąsaja pasiekiama Interneto naršyklės pagalba.

4.4 Realizuoto modulio naudojamos funkcijos

Iš suformuotos konfigūracijos sukuriamos reguliariųjų išraiškų taisyklės skirtos duomenims ateinantiems iš įvesties laukų tikrinimui. Atakoms aptikti naudojamos specialios šių duomenų tikrinimo funkcijos. Duomenys priklausomai nuo jų tipo gali būti tikrinami iš kokių simbolių jie sudaryti ar tai raidės ir skaičiai ar tik raidės, ilgis ir pan..

Jei tikrinimo funkcijoms ir nepavyktų aptikti iš tikrųjų žalingo kodo, jo nukenksminimą atliks išvalymo funkcijos. Tačiau priešingu atveju neleis žiniatinklio programai vykdyti SQL užklausos, priklausomai nuo klaidų ribojimo ir dažnumo nustatymų vartotojui gali būti apribotas klaidingų įvedimų skaičius ar visiškai apribota tolesnė teisė kreiptis į žiniatinklio programą. Tikrinimas pagal iš anksto nustatytus šablonus vykdomas tik tuomet, kai įvestis sudaryta ne tik iš skaičių ir raidžių. Tokie įvesties laukai gali būti paieškose pvz.: ieškant prekių pavadinimų.

Pagrindinės modulio naudojamos funkcijos:

Filtro sukūrimo funkcija

Atsižvelgiant į įvestus parametrus kuriant naują taisyklę sukuriamas atitinkamas filtras. Informacija apie filtrą įrašoma į xml failą.

Tikrinimo pagal šablonus funkcija

Gauta POST arba GET reikšmė sulyginama REGEXP funkcija ieškant atitikmenų su visais šablonais iš xml failo.

Utilizavimo funkcija

Reikšmėse neutralizuoja pavojingus simbolius naudojant rekomenduojamą PHP funkciją. Funkcija *mysql_real_escape_string()* iškviečia MySQL bibliotekos funkciją mysql_real_escape_string, kuri prideda pasvirus brūkšnius prie šių simbolių: \x00, \n, \r, \, ', ",\x1a taip panaikindama jų įtaką SQL užklausoms.

4.5 Filtrų pavyzdys

Baltųjų sąrašų funkcijos bei filtrai ar atakų požymių šablonai saugomi xml failuose.

Pvz.:

```
<regex>
        <name>el. paštas</name>
        <pattern><![CDATA[^[\w\-\+\&\*]+(?:\.[\w\-\_\+\&\*]+)*@(?:[\w-]+\.)+[a-zA-Z]{2,7}$]]></pattern>
        <description>Tikrinamas ar teisingas el. pašto formatas</description>
</regex>
```

Injekcijų šablonai saugomi xml faile fragmentas:

<filters>

<filter>

<id>1</id>

<rule><![CDATA[(?:"\s(?:#|--*

|{))|(?:\/!\s?\d+)|(?:ch(?:a)?r\s*\(\s*\d)|(?:(?:(n?and|x?or|not)\s+|\|\||\&\&)\s*\w+\()]]></*

rule>

</filter>

...

</filters>

Realizuotame modulyje taisyklės naudojamos iš PHPIDS [2] kurios platinami pagal LGPL licenciją.

4.6 Realizuoto modulio ypatumai

- SQL prevencinis modulis ne tik aptinka, bet ir neutralizuoja galimą pažeidimą;
- Turi iš anksto paruoštas specialias laukų tikrinimo funkcijas, kurias galima išnaudoti tikrinimui baltųjų sąrašų principu taip padidinant efektyvumą;
- Galima adaptuoti modulį skirtingiems uždaviniams ir saugumo lygiams užtikrinti.

Modulio trūkumai

Norint kurti taisyklę reikia išmanyti reguliariųjų išraiškų struktūrą ir savybes.

4.7 Panaudojimo atvejai – diegimas

Tarkime yra sukurta slaptažodžio tikrinimo funkcija vardu „*slaptazodis*“, tuomet tokios funkcijos panaudojimas būtų toks:

include('sqlstart.php'); // iškviečiamas modulis

$slaptazodis = slaptazodis($_POST['email']); reikšmei $slaptazodis priskiriama funkcijos *slaptazodis()* reikšmė. Ši funkcija patikrins užduotus parametrus ir grąžins išvalytą reikšmę, tačiau jeigu ras neatitikimų reaguos pagal iš anksto nustatytą eigą.

Jeigu laukų daug tuomet galima naudoti tik tikrinimo pagal filtrus funkciją. Tokiu atveju visos priimamos reikšmės pirmiausia patikrinamos pagal filtrus, neradus klaidų leidžiama tęsti darbą. Jeigu aptinkama pažeidžiamumų tolimesnis vykdymas nutraukiamas ir nusiunčiamas pranešimas apie klaidą į klaidų valdymo modulį.

Tuomet naudojama $_REQUEST kuris apima reikšmes iš metodų $_GET, $_POST, ir $_COOKIE

include('sqlstart.php'); // iškviečiamas modulis

include('filtrai.php'); // iškviečiamas filtravimo funkcijų rinkinys

4.8 Pasirengimas eksperimentui

Kokybės kriterijai

Iškelti žiniatinklio programų ugniasienės SQL prevencinio modulio kokybės kriterijai. Sudėtingumas naudoti yra vienas iš kokybės kriterijų tačiau jo negalima tiksliai įvertinti.

- Netikrų pavojų fiksavimas;
- Tikrų pavojų neužfiksavimas;
- Apkrova serveriui;
- Vėlinimas;
- Sudėtingumas naudoti

Testavimo įrankių pasirinkimas

Pasirinkti testavimo įrankiai tinkantys testuoti ugniasienės tikslumui ir apkrovai serveriui simuliuojant konkurencinius vartotojus. Vertinant testų rezultatus bus atsižvelgiama į kokybės kriterijus.

Remiantis analizės dalyje pristatytu testavimo įrankių palyginimu, pasirenkamas Acunetix Web Security Scanner + Acusensor produktas. Apkrovai matuoti naudojamos specialios PHP funkcijos ir NeoLoad programa, o papildomam tikslumui nustatyti iš anksto paruoštas gerai žinomų pažeidžiamumų sąrašas.

4.9 Išvados

Realizuotas SQL injekcijų prevencinis modulis. Pagrindinė modulio užduotis aptikti ir neutralizuoti gaunamus duomenis su SQL injekcijų požymiais. Modulis yra integruojamoji žiniatinklio ugniasienės dalis. Modulis turi grafinę vartotojo sąsają, kuri pasiekiama naršyklės pagalba. Modulis galėtų būti naudojamas taip kaip jis sukonfigūruotas, kadangi modulis turi pažeidžiamumų požymių duomenų bazę xml formatu, kurią galima bet kada atnaujinti. SQL prevenciniu moduliu galima įgyvendinti teigiamo ir neigiamo saugumo modelio filtravimą. Taip pat aptarti modulio diegimo scenarijai, bei kokybės vertinimo kriterijai. Vien tik šio

modulio apsaugoti žiniatinklio programai nepakaktų, būtina kitų modulių realizacija. Kitas žingsnis realizuoto ugniasienės modulio kokybės parametrų tyrimas.

5. REALIZUOTO UGNIASIENĖS MODELIO TYRIMAS

Šioje darbo dalyje atliekamas realizuoto SQL injekcijų prevencinio modulio tyrimas. Patikrinta kaip modulis tenkina ankstesniuose skyriuose iškeltus teigiamus kokybės kriterijus. Atlikti skirtingų tipų testavimai realiai veikiančiame dedikuotame serveryje. Šioje dalyje ugniasiene laikomas SQL prevencinis modulis.

5.1 Testavimo scenarijai

Norint patikrinti atskirus modulio kokybės kriterijus naudojami naudojamus scenarijus galima skirstyti į dvi dalis:

1. Ugniasienės tikslumo testavimas
2. Vėlinimo ir apkrovos tarnybinei stočiai testavimas

Pirmuoju atveju tikrinama kaip ugniasienės modulis susidoroja su pagrindine užduotimi SQL injekcijų aptikimu. Tikrinamas ir netikrų pavojų rodiklis. Antruoju atveju testuojama modulio pajėgumai, kokias maksimalias apkrovas gali atlaikyti tarnybinė stotis su šiuo moduliu, tam pasitelkiami streso testai bei žiniatinklio programos vėlinimas naudojant SQL prevencinį modelį. Vėlinimui matuoti pasitelkiama NeoLoad programinė įranga galinti simuliuoti konkurencinius vartotojus.

5.2 Ugniasienės tikslumo matavimas

Testavimui naudojama:

- Iš anksto paruoštas failas su gerai žinomų atakų šablonais
- Žiniatinklio programų audito programa - Acunetix Web Security Scanner + Acusensor

Tikslumo testavimas naudojant gerai žinomų atakų šablonus

Testavimo eiga:

- Parengiamas gerai žinomų SQL injekcijų pažeidžiamumų failas;
- Visus failo pažeidžiamumus nuosekliai patikriname su ugniasienės taisyklėmis;
- Užfiksuojami atakų požymių aptikimo rezultatai.

Testavimo failas

Pagal OWASP rekomendacijas naudojamas kompiuterių saugumo specialistų kompanijos YEGH paruoštas pažeidžiamumų požymių ir pažeidžiamumų failas [47].

Testavimo vykdymas

Vykdomas ciklas, kurio metu nuosekliai imama eilutė iš SQL injekcijų požymių failo ir lyginama su visomis taisyklėmis, radus atitikimą užfiksuojama.

Testavimo rezultatai

Aptikta atakų požymių: 144

Atlikta tikrinimų: 8568

Tikrinimas truko: 0.1821 s.

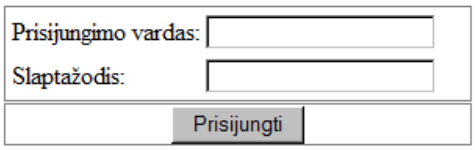
Tikrinimas atliktas pagal 69 taisykles.

Pastebėjimai:

Kai kurios eilutės tebuvo tik atakų požymiai, bet ne pačios atakos pvz.: **‘** , **!**. MySQL tipo duomenų bazėms šie simboliai pavojaus nekelia todėl nebuvo užskaityti kaip atakos. Kai kurios rimtesnės tikrosios atakos pvz.: **' or 1=1 or ''='** atitiko keletą požymių.

Tikslumo testavimas naudojant Acuentix Web Vulnerability Scanner

Testuojama 15 pav. pateikta prisijungimo forma, kurioje reikia įvesti prisijungimo vardą ir slaptažodį.

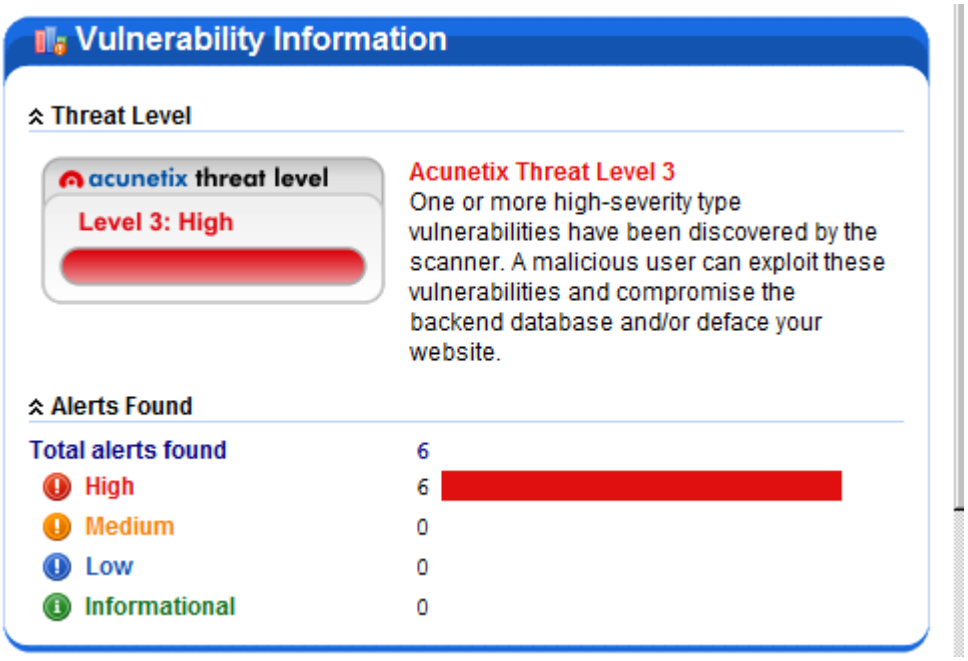


15 pav. prisijungimo forma

Įjungiama ugniasienė, testuoti pagal SQL injekcijų šablonus. Jeigu priimti duomenys neatitinka atakos požymių vykdomas kreipimasis į duomenų bazę. Įvedus teisingus duomenis užrašoma „Prisijungimas pavyko“, priešingu atveju „Prisijungimas nepavyko“.

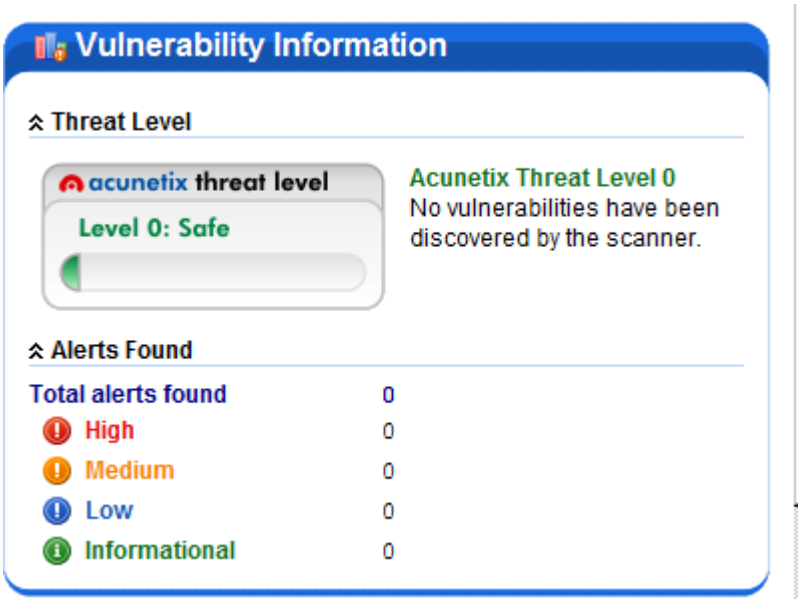
Testavimo rezultatas naudojant Acuentics programą

Atlikus testą nenaudojant ugniasienės gauti 16 paveiksle pateikti rezultatai. Aptikta, kad prisijungimo forma turi 6 ypatingai dideles saugumo spragas, susijusias su SQL injekcijomis.



16 pav. pažeidžiamumų informacija nenaudojant jokios apsaugos

Atlikus specialų SQL injekcijų patikrinimą naudojant SQL injekcijų prevencinį modulį gauti 17 pav. matomi rezultatai, rodo kad įjungus ugniasienę neaptikta jokių pažeidžiamumų, susijusių su SQL injekcijomis.



17 pav. pažeidžiamumų informacija naudojant ugniasienę

Testavimas nuo netikrų pavojų

Testuojama kaip ugniasienė atskiria netikrus pavojus.

Testavimui naudojamas iš anksto paruoštas failas su gerai žinomais klaidingais požymiais. Remiantis SQL injekcijų ir galimų įvesčių analize suformuotas netikrų pavojų pavyzdžių sąrašas.

Pvz.: *Mr. Brown, Orreili's, UNION, Mike AND Partners, HAVING, 80`ts*. Šie ir panašūs žodžiai nors ir turi atakų požymių bet pasitaiko natūraliame naudojime. Vienas iš ugniasienės kokybės kriterijų yra kuo mažesnis netikrų pavojų fiksavimas.

Testavimo eiga

Kiekvienas pavyzdys sulyginamas su visomis taisyklėmis, jeigu kuris nors elementas laikomas pavojingu fiksuojama ataka.

Testavimo nuo netikrų pavojų rezultatai

Atlikus tikrinimą ugniasienė parodė:

Aptikta atakų požymių: 0

Atlikta tikrinimų: 408

Tikrinimas truko: 0.012 s.

Ugniasienė neužfiksavo netikrų pavojų.

Tikslumo testavimo rezultatų apibendrinimas

Atlikus tikslumo testavimus paaiškėjo, kad ugniasienės SQL injekcijų aptikimo modulis veikia preciziškai. Ugniasienė aptiko visus tikrus pažeidimus iš gerai žinomų SQL injekcijų šablonų failo. Acuentix audito programa parodė, kad be ugniasienės žiniatinklio programa turi didelių saugumo spragų susijusių su SQL pažeidžiamumais, o aktyvavus ugniasienę tokių spragų nebuvo rasta. Tikrinant netikrų pavojų pranešimus nebuvo aptikta nei vieno netikro pavojaus atliekant 408 patikrinimus iš šablonų failo su potencialiais netikrais pavojais.

5.3 Tarnybinės stoties apkrovos testavimas

Testuojama prisijungimo forma, imituojant jungimųsi skaičių priklausomai nuo prisijungimų skaičiaus nuo 100 iki 10000 naudojamas iš anksto paruoštas atsitiktinai sugeneruotų prisijungimo duomenų failas. Kiekvienu atveju bandoma kreiptis į duomenų bazę, kadangi ugniasienė nėra naudojama. Pvz. 100 bandymų reiškia, kad bus iškviečiamas failas su šimto

vartotojų prisijungimo duomenim, kurie iš anksto atsitiktinai sugeneruojami. Tokiu atveju būtų vykdoma 100 atskirų susijungimų su duomenų baze ieškant galimo vartotojo.

NeoLoad programinė įranga [48], naudojama testuoti žiniatinklio programoms. NeoLoad pagalba atliekama vienu metu veikiančių vartotojų simuliacija, kurie atlieka nurodytą skaičių iteracijų. Vykdomas programos darbas nenaudojant ugniasienės, vėliau naudojant. Gauti rezultatai sulyginami.

Testuojama be ugniasienės

Testavimo eiga:

Iš pradžių įrašomas elgsenos scenarijus *test1.php*

Nuosekliai atlikta 100, 1000, 5000, 10000 patikrinimų.

NeoLoad šį scenarijų įsimena ir vėliau pakartoja nustatytą kartų skaičių.

Testavimo rezultatai pateikiami 6 lentelėje. Lentelėje matyti laiko priklausomybė nuo konkurencinių vartotojų kiekio. Kuo vartotojų skaičius didesnis, tuo testavimo trukmė ilgesnė.

6 lentelė. *NeoLoad testavimo rezultatai be ugniasienės*

Simuliuota konkurencinių vartotojų	4	5	6	7	8	9	10
Vidutinis atsakymo į užklausą laikas(s)	0,49	0,638	0,738	0,894	0,971	1,12	1,25
Užklausų į serverį per sekundę	7,1	6,9	7,1	7,2	7,2	7,3	7,1
Užklausų į duombazę per sekundę	7273	7042	7143	7292	7273	7317	7194
Testavimo trukmė	00:00:55	00:01:11	00:01:24	00:01:36	00:01:50	00:02:03	00:02:19

Testuojama su ugniasiene

Atliekant testavimą su ugniasienę matuojama užklausų per sekundę priklausomybė nuo ugniasienės modulio vidurkiai pateikiami 7 lentelėje.

7 lentelė. *NeoLoad testavimo rezultatai su ugniasiene*

Simuliuota konkurencinių vartotojų	4	5	6	7	8	9	10
Vidutinis atsakymo į užklausą laikas(s)	0,531	0,687	0,774	0,921	1,06	1,18	1,34
Užklausų į serverį per sekundę	6,7	6,8	6,7	6,7	6,7	6,9	6,8
Užklausų į duombazę per sekundę	6780	6849	6742	6796	6957	6923	6849
Testavimo trukmė	00:00:59	00:01:13	00:01:29	00:01:43	00:01:55	00:02:10	00:02:26

Testavimo rezultatų palyginimas

8 lentelėje pateikiamas rezultatų testuojant su NeoLoad programa palyginimas.

Vidutinė atsakymui į užklausą laiko padidėjimo įtaka veikiant ugniasienei - 6,5%. Įtaka svyruoja nuo 3% iki 9.2 %. Testavimai buvo daromi realiai veikiančiame serveryje, todėl jo apkrova įvairiuose laiko tarpuose nebuvo vienoda.

8 lentelė. *Vidutinio atsakymo į užklausą laiko pokytis*

Simuliuota konkurencinių vartotojų	4	5	6	7	8	9	10
Be ugniasienės	0,49	0,638	0,738	0,894	0,971	1,12	1,25
Su ugniasiene	0,531	0,687	0,774	0,921	1,06	1,18	1,34
Pokytis	8,4%	7,7%	4,9%	3,0%	9,2%	5,4%	7,2%

Iš 9 lentelėje pateikiamų duomenų matyti, kad vidutinė įtaka užklausų į duomenų bazę kiekiui veikiant ugniasienei 5.2%. Įtaka svyruoja nuo 2.7% iki 6.8 %. Testavimai buvo daromi realiai veikiančiame virtualiame privačiame dedikuotame serveryje, todėl jo apkrova įvairiuose laiko tarpuose nebuvo vienoda.

9 lentelė. *Užklausų į duomenų bazę per sekundę pokytis*

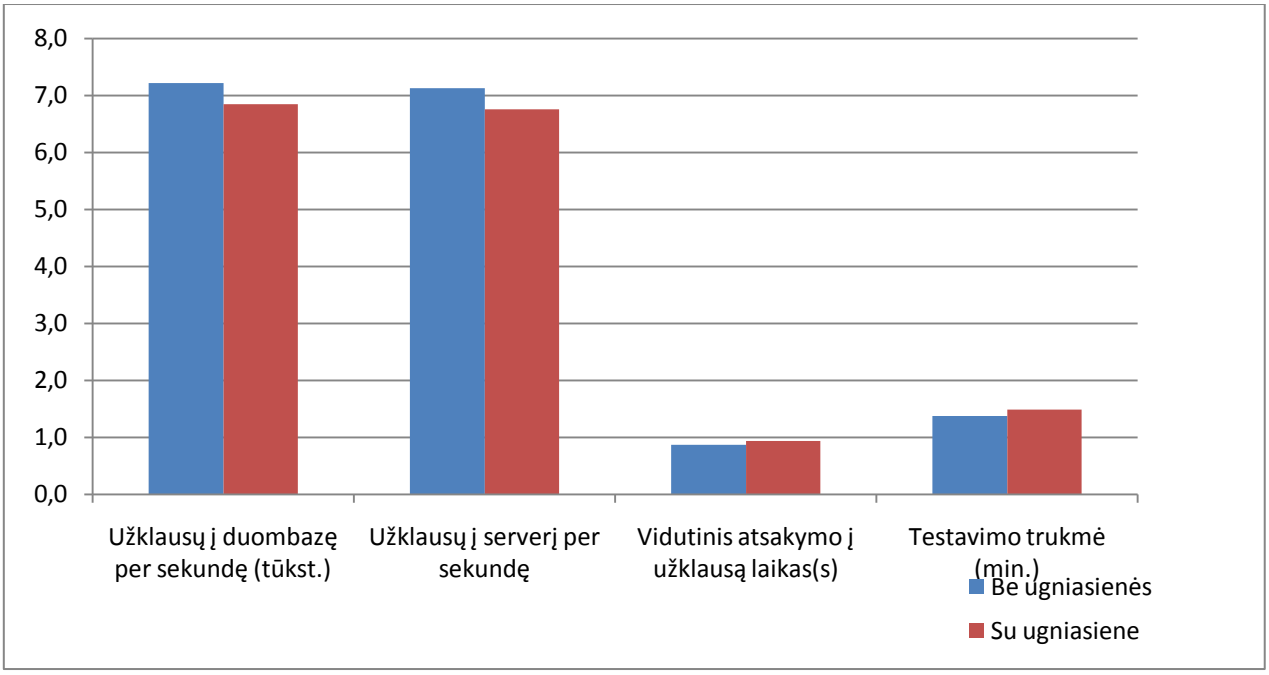
Simuliuota konkurencinių vartotojų	4	5	6	7	8	9	10
Be ugniasienės	7273	7042	7143	7292	7273	7317	7194
Su ugniasiene	6780	6849	6742	6796	6957	6923	6849
Pokytis (sumažėjimas)	6,8%	2,7%	5,6%	6,8%	4,3%	5,4%	4,8%

Iš 10 lentelėje pateiktų duomenų matyti, kad vidutinė įtaka užklausų į duomenų bazę kiekiui veikiant ugniasienei 5.5%. Įtaka svyruoja nuo 2.8% iki 7.3 %. Testavimai buvo daromi realiai veikiančiame serveryje, todėl jo apkrova įvairiuose laiko tarpuose nebuvo vienoda.

10 lentelė. *Testavimo trukmės pokytis naudojant ugniasienę*

Simuliuota konkurencinių vartotojų	4	5	6	7	8	9	10
Be ugniasienės	00:00:55	00:01:11	00:01:24	00:01:36	00:01:50	00:02:03	00:02:19
Su ugniasiene	00:00:59	00:01:13	00:01:29	00:01:43	00:01:55	00:02:10	00:02:26
Pokytis	7,3%	2,8%	6,0%	7,3%	4,5%	5,7%	5,0%

18 paveiksle pateikiamas aukščiau pateiktų lentelių vizualinis vidurkių pokytis.



18 pav. Vizualinis vidurkių pokytis

Atlikus įtakos vėlinimui testavimus, gauta vidutinė atsakymui į užklausą laiko padidėjimo įtaka veikiant ugniasienei - 6,5%, o vidutinė įtaka užklausų į duomenų bazę kiekiui veikiant ugniasienei 5.2%. Reikalingas laikas kinta priklausomai nuo simuliuotų vartotojų skaičiaus, tačiau tai įtakoja pačios tarnybinės stoties pajėgumai.

5.4 Streso testas

Norint ištirti modulio ir tarnybinės stoties veikimo galimybių ribas reikalingi streso testai imituojantys situaciją, kuri realiomis aplinkybėmis nepasitaiko. Streso testo metu taip pat

tiriama „baltųjų sąrašų“ įtaka. Streso testo vykdymo trukmę apriboja nustatytas leidžiamas programos sesijos veikimo laikas, kuris yra trisdešimt sekundžių.

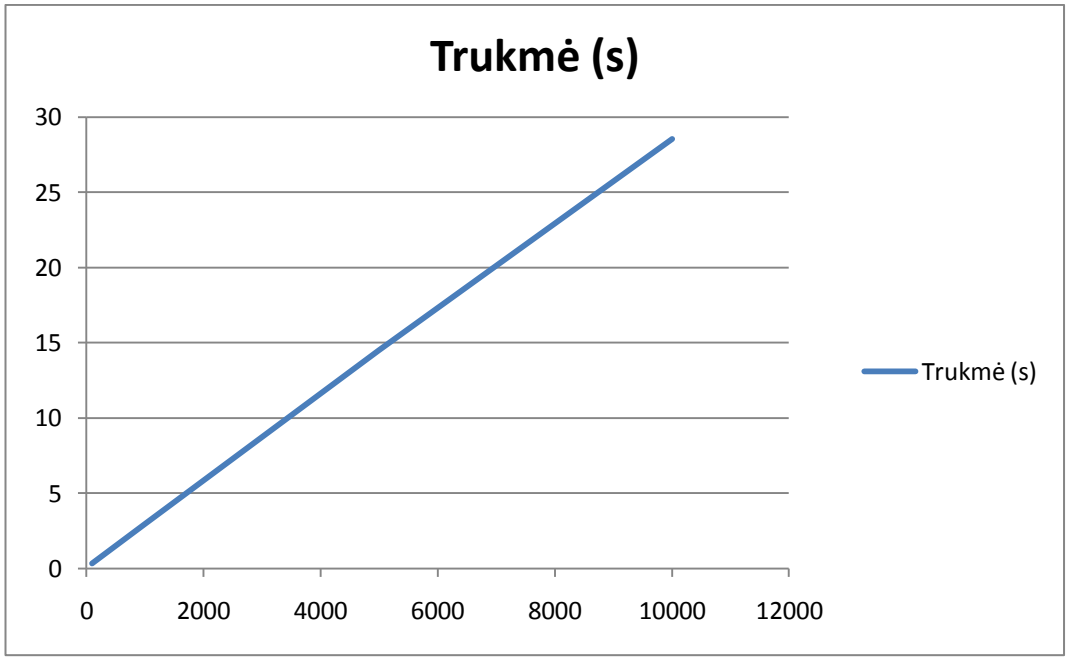
5.4.1 Streso testas tikrinant pagal taisykles

Atliekama nuo 100 iki 10000 užklausų. Laikoma kad kiekvienu atveju vykdoma ataka ir kiekviena užklausa tikrinama pagal taisykles. Norima parodyti kiek užklausų ugniasienė gali atlaikyti.

11 lentelė. *Streso testo tikrininant pagal taisykles rezultatai*

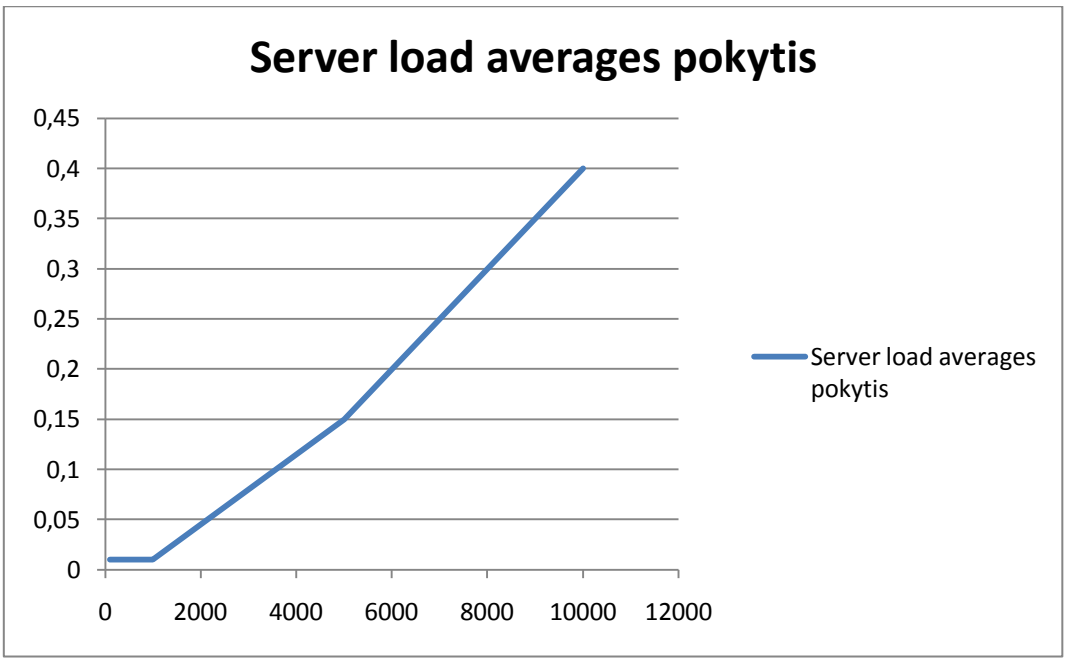
Atlikta tikrinimų	**100**	**1000**	**5000**	**10000**
Trukmė (s)	0,33	2,96	14,51	28,53
Server load averages pokytis	0,01	0,01	0,15	0,4
Užklausų per sekundę	303	338	345	351

Iš 11 lentelės matyti, kad traktuojant kiekvieną užklausą kaip pavojingą, programa per sekundę gali atlikti vidutiniškai 334 užklausas į duomenų bazę. 10000 tikrinimų yra riba naudojant ugniasienę, kadangi trukmė artima maksimaliam sesijos laikui.



19 pav. laiko priklausomybė nuo užklausų kiekio į duomenų bazę grafikas

19 pav. vaizduojamame grafike matyti, kad laiko priklausomybė nuo užklausų kiekio į duomenų bazę artima tiesinei.



20 pav. Serverio apkrova

20 pav. matyti, kad tarnybinės stoties apkrova pradeda žymiai kisti, kai užklausų skaičius pasiekia daugiau nei 1000.

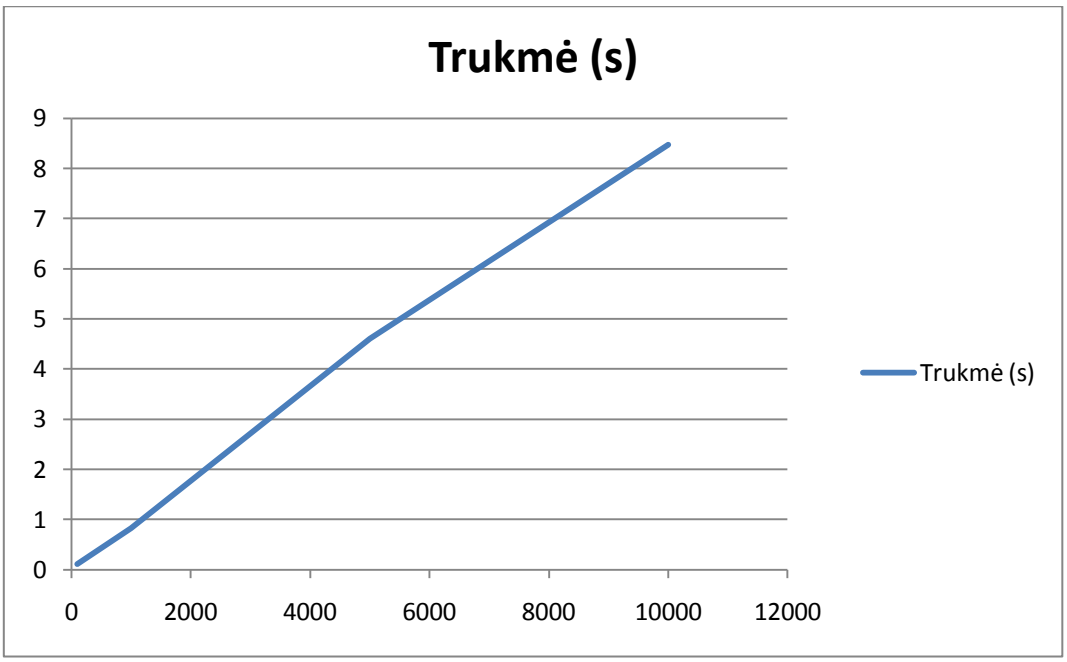
5.4.2 Streso testas naudojant baltuosius sąrašus

Atliekama nuo 100 iki 10000 užklausų. Įvedimo laukai tikrinami panaudojant baltuosius sąrašus, „ar el. paštas“ ir „ar slaptažodis iš tinkamų simbolių“.

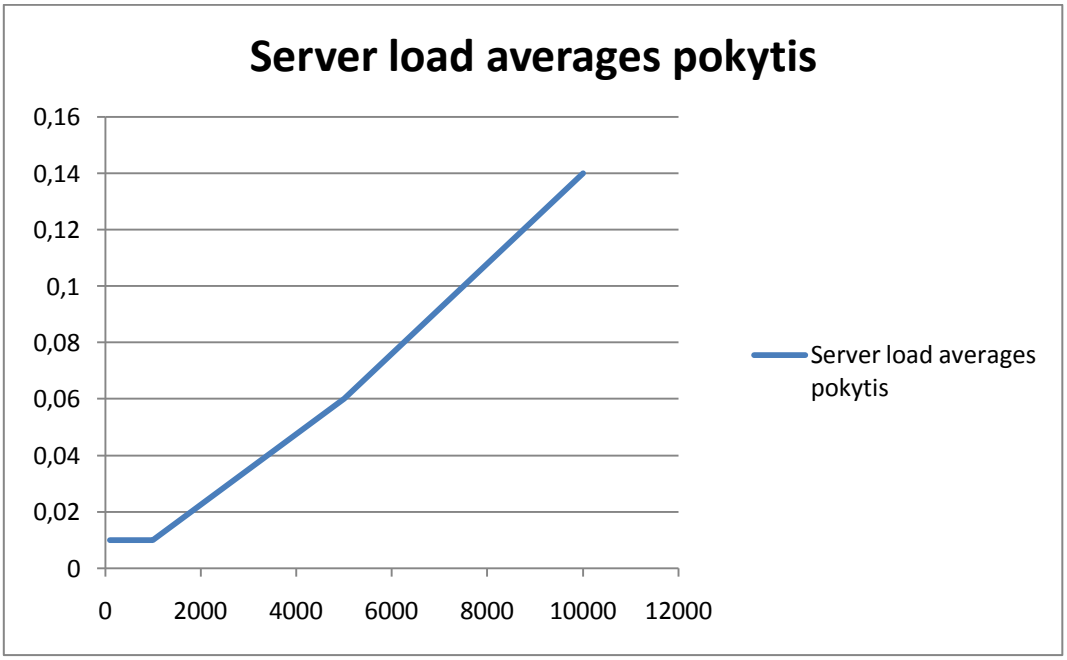
12 lentelė. *Streso testas naudojant baltuosius sąrašus rezultatai*

Atlikta tikrinimų	**100**	**1000**	**5000**	**10000**
Trukmė (s)	0,11	0,83	4,61	8,48
Server load averages pokytis	0,01	0,01	0,06	0,14
Užklausų per sekundę	909	1205	1085	1179

21 ir 22 paveiksluose pateikiami grafikai atspindintys laiko priklausomybė nuo užklausų kiekio bei serverio apkrovos priklausomybė nuo užklausų kiekio. Abu grafikai rodo artimas tiesinei priklausomybes. Teoriškai taip ir turi būti, kadangi kuo daugiau užklausų tuo tarnybinė stotis gauna didesnį krūvį. Tiesės kilimas gali būti didesnis arba mažesnis priklausomai nuo tarnybinės stoties pajėgumų arba tuometinio užimtumo.



21 pav. Laiko priklausomybė nuo užklausų kiekio



22 pav. Serverio apkrovos priklausomybė nuo užklausų kiekio

5.4.3 Gautų streso testavimo rezultatų palyginimas

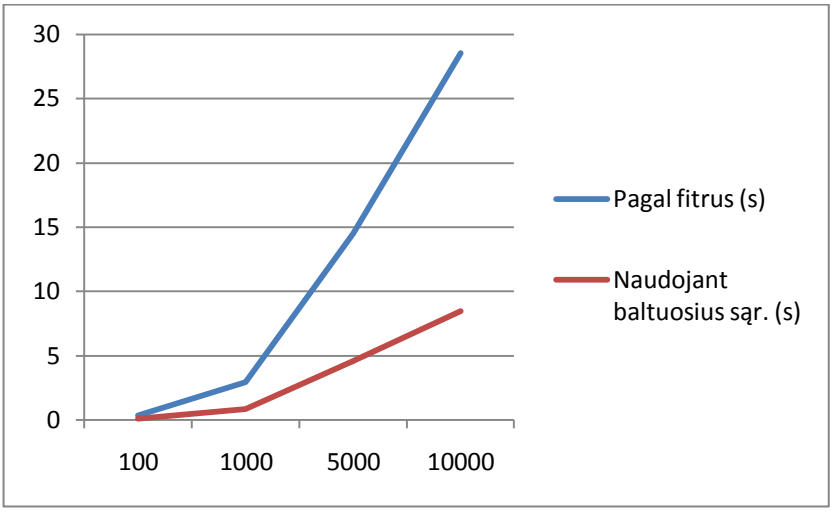
13 lentelė. *Palyginimas tikrinimo pagal visus filtrus su tikrinimu pagal baltuosius sąrašus*

Atlikta tikrinimų	**100**	**1000**	**5000**	**10000**
Pagal filtrus (s)	0,33	2,96	14,51	28,53
Naudojant baltuosius sąr. (s)	0,11	0,83	4,61	8,48
Pokytis (%)	66,67%	71,96%	68,23%	70,28%

Naudojant baltuosius sąrašus efektyvumas laiko atžvilgiu vidutiniškai pagerėja 69,28% 13 lentelė.

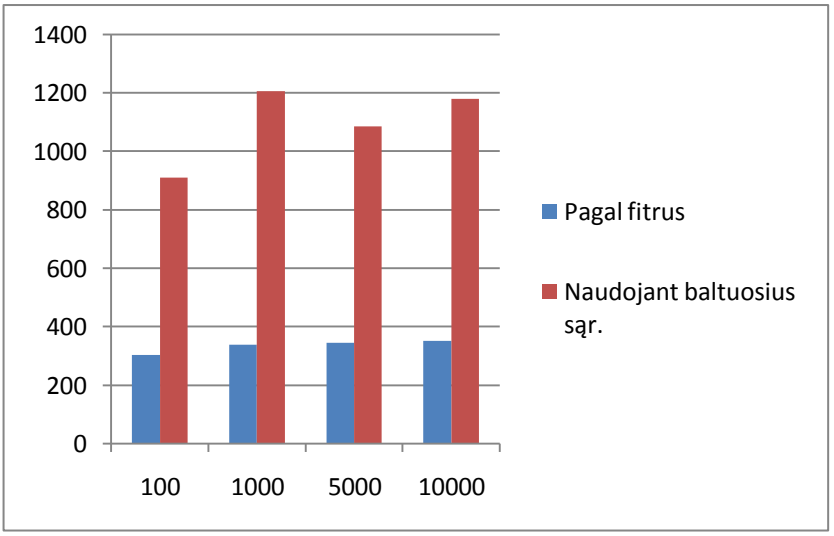
Tikrinimo trukmė kintant užklausų skaičiui

23 pav. matoma tikrinimo trukmės priklausomybė nuo užklausų skaičiaus naudojant filtrus ir baltuosius sąrašus.



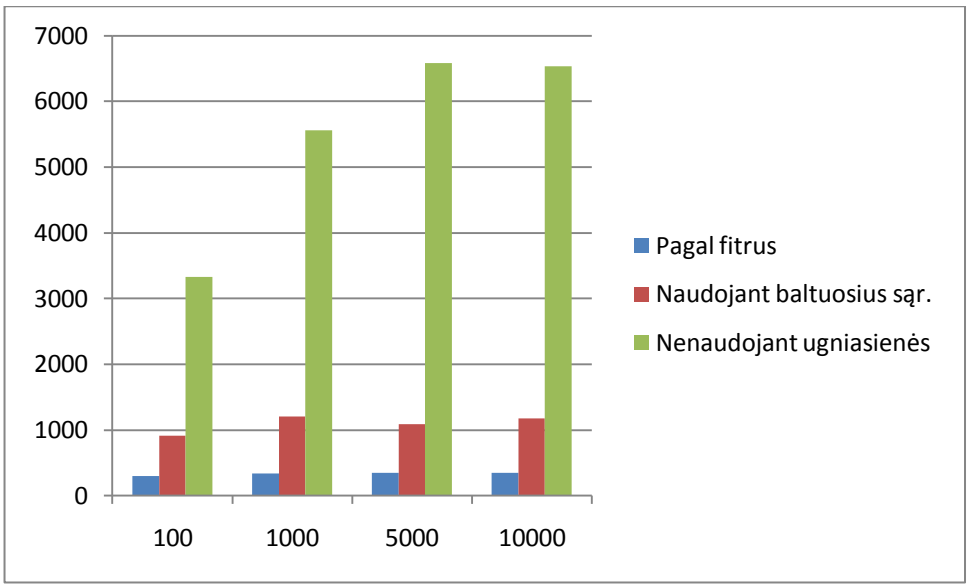
23 pav. Tikrinimo trukmė kintant užklausų skaičiui

24 pav. Vaizduojama kreipimųsi į duombazę skaičius per sekundę kintant užklausų skaičiui. Matyti akivaizdi baltųjų sąrašų persvara.



24 pav. Kreipimųsi į duomenų bazę skaičius per sekundę kintant užklausų skaičiui

Užklausų per sekundę palyginimas 25 pav. suformuotas iš 14 lentelės duomenų. lyginami visi trys variantai: nenaudojant ugniasienės, pagal filtrus ir naudojant baltuosius sąrašus. Testavimai daryti laikant, kad kiekviena užklausa yra pažeidimas.



25 pav. Užklausų per sekundę palyginimas

14 lentelė. *Užklausų per sekundę palyginimas*

Atlikta tikrinimų	**100**	**1000**	**5000**	**10000**
Pagal filtrus	303	338	345	351
Naudojant baltuosius sąr.	909	1205	1085	1179
Nenaudojant ugniasienės	3333	5556	6579	6536

5.5 SQL prevencinio modulio taikymo ir tobulinimo rekomendacijos

Atlikus tyrimą paaiškėjo baltųjų sąrašų naudojimo privalumai. Baltuosius sąrašus paranku naudoti ten kur nėra daug laukų o jų formatą galima iš anksto apibrėžti, pvz. prisijungimo ar registracijos forma. Išnaudojant baltuosius sąrašus žiniatinklio programa tampa atsparesnė didesnėms apkrovoms.

Žiniatinklio ugniasienė gali būti realizuota naudojant pačias efektyviausias programavimo patirtis, taip dar labiau sumažinant įtaką žiniatinklio programos veikimo spartai. Tyrimas parodė, jog žiniatinklio programa veikia preciziškai, tačiau programinis kodas yra labiausiai tobulintina ugniasienės dalis.

5.6 Išvados

Eksperimento metu ištirtas ugniasienės tikslumas, netikrų pavojų fiksavimas taip pat apkrova serveriui bei užduočių vykdymo laiko priklausomybė nuo ugniasienės.

Patikrinus ugniasienės tikslumą su pripažinta audito programa Acuentix bei iš anksto paruoštų pažeidžiamumų šablonų tikrinimo funkcija paaiškėjo jog ugniasienės SQL prevencinis modulis veikia preciziškai. Ugniasienė nefiksavo netikrų pavojų.

Tikrinant ugniasienės naudojimą nustatyta, jog vidutinė įtaka užklausų į duomenų bazę kiekiui veikiant ugniasienei 5.5%, lyginant su žiniatinklio programos veikimu be apsaugos.

Norint įsitikinti kiek užklausų apskritai gali įveikti ugniasienė atlikti streso testai, kuomet kiekviena užklausa buvo laikoma potencialia ataka. Paaiškėjo didelis baltųjų sąrašų privalumas. Naudojant baltuosius sąrašus efektyvumas laiko atžvilgiu vidutiniškai pagerėja 69,28%.

6. IŠVADOS

Taikomųjų žiniatinklio programų analizė parodė, kad Internete šiandien vis dar egzistuoja daug saugumo spragų, o jų išnaudojimo metodai tampa modernesni ir padarantys daugiau žalos.

OWASP atliktų stebėjimų duomenimis sudarytas saugumo grėsmių dešimtukas atskleidė pačias pažeidžiamiausias žiniatinklio programų vietas. Norint išvengti galimų grėsmių reikalinga ne tik teisingai sukurti programą, bet ir naudoti papildomas apsaugos priemones – žiniatinklio programų ugniasienes.

Suprojektuota žiniatinklio ugniasienė pritaikant naujausią metodiką kovai su populiariausiomis grėsmėmis. Realizuoto žiniatinklio ugniasienės SQL injekcijų modulio tyrimas atskleidė tokios ugniasienės privalumus. Remiantis atliktais eksperimentais išaiškėjo SQL injekcijų modulio galimybių ribos, modulis veikia itin preciziškai, geba aptikti SQL injekcijas ir jų požymius.

Ugniasienės modelį galima pritaikyti kuriant naujas ar tobulinant esamas apsaugos priemones, bet kokios platformos žiniatinklio programoms. Sukurtą SQL injekcijų prevencinį modulį galima taikyti kaip atskirą saugumo priemonę realiai veikiančiose žiniatinklio programose. SQL injekcijų modulio eksperimentas atkleidė baltųjų sąrašų naudojimo privalumą. Baltieji sąrašai padeda taupyti tarnybinių stočių resursus.

Operatyvi žiniatinklio programų ugniasienės reakcija į pažeidimus leidžia užkirsti tolimesnių pažeidimų plitimo galimybę informuojant apie incidentus atsakingas institucijas, o konfigūracijos valdymo modulis padeda sekti naudojamos programinės įrangos versijas.

Žiniatinklio programų ugniasienės realizuoto modulio programinį kodą galima optimizuoti, čia eksperimentu tikrinti tik esminiai dalykai.

7. LITERATŪRA

[1] Internet world stats – interneto vartotojų statistika [interaktyvus]. [žiūrėta 2009-01-19]. Prieiga per Internetą: http://www.internetworldstats.com/stats.htm

[2] Žiniatinklio saugumo konsorciumas Webappsec – informacijos klasifikavimo svarba kuriant saugias programas [interaktyvus]. [žiūrėta 2009-01-20]. Prieiga per Internetą: http://www.webappsec.org/projects/articles/041607.shtml

[3] Saugios žiniatinklio technologijos [interaktyvus]. [žiūrėta 2009-01-19]. Prieiga per Internetą: http://www.acunetix.com

[4] MSDN saugių ijų kūrimo gairės [interaktyvus]. [žiūrėta 2009-01-19]. Prieiga per Internetą: http://msdn.microsoft.com/en-us/library/aa302420.aspx#c04618429_004

[5] SANS institutas – The Top Cyber Security Risks 2007m. pažeidžiamumai [interaktyvus]. [žiūrėta 2009-01-20]. Prieiga per Internetą: http://www.sans.org/top20/?ref=1697#s1

[6] Atviras žiniatinklio aplikacijų saugumo projektas OWASP [interaktyvus]. [žiūrėta 2009-01-20]. Prieiga per Internetą:

[7] Standfordo universitetas Data Classification, Access, Transmittal and Storage [interaktyvus]. [žiūrėta 2009-06-13]. Prieiga per Internetą: http://www.stanford.edu/group/security/securecomputing/dataclass_chart.html

[8] OWASP TOP 10 – 2010 [interaktyvus]. [žiūrėta 2010-04-10]. Prieiga per Internetą: http://www.owasp.org/index.php/Top_10_2010

[9] Hope, Paco; Walther, Ben (2008), Web Security Testing Cookbook, O'Reilly Media, Inc., p. 254, ISBN 978-0-596-51483-9

[10] Justin Clarke, SQL Injection Attacks and Defense (2009), ISBN 978-1597494243

[11] Watson, Carli (2006) Beginning C# 2005 databases , psl 201-5 ISBN 978-0-470-04406-3

[12] Hope, Paco; Walther, Ben (2008), Web Security Testing Cookbook, O'Reilly Media, Inc., psl. 254, ISBN 978-0-596-51483-9

[13] Injection Exploits: Cross-Site Scripting, Sql Injection, Code Injection, Shellcode, Xss Worm, Remote File Inclusion Books LLC (2010) ISBN-13: 978-1155209098

[14] O'Reilly, Cross-Site Scripting (2008) ISBN-13: 978-3897215399

[15] Syngress, Seven Deadliest Web Application Attacks (2010). ISBN-13: 978-1597495431

[16] Gunter Ollmann, Professional Services Director, NGS straipsnis Stopping Automated Attack Tools [žiūrėta 2010-02-20]. Prieiga per Internetą: http://www.ngssoftware.com/papers/StoppingAutomatedAttackTools.pdf

[17] O'Reilly Media, PHP: The Good Parts: Delivering the Best of PHP (2010) ISBN-13: 978-0596804374

[18] Stuart McDonald, SQL Injection: Modes of Attack, Defence, and Why It Matters (2002) [interaktyvus]. [žiūrėta 2010-03-12]. Prieiga Internete: http://www.sans.org/reading_room/whitepapers/securecode/sql-injection-modes-attack-defence-matters_23

[19] OWASP, XSS (Cross Site Scripting) Prevention Cheat Sheet [interaktyvus]. [žiūrėta 2010-03-12] Prieiga per Internetą: http://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

[20] Seth Fogie, Jeremiah Grossman, Robert Hansen, Anton Rager, Petko D. Petkov XSS Attacks: Cross Site Scripting Exploits and Defense (2007) ISBN-13: 978-1-59749-154-9

[21] J.D. Meier, Alex Mackman, Blaine Wastell, Prashant Bansode, Andy Wigley. How To: Prevent Cross-Site Scripting in ASP.NET (2005) [interaktyvus]. [žiūrėta 2010-05-19] prieiga per Internetą: http://msdn.microsoft.com/en-us/library/ff649310.aspx

[22] LaQuSo, Broken Authentication and Session Management [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: https://lab.cs.ru.nl/laquso/7.Broken_Authentication_and_Session_Management

[23] Eylül Karsamba, Broken Authentication and Session Management [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://serdarbuyuktemiz.blogspot.com/2008/09/broken-authentication-and-session.html

[24] OWASP, Broken Authentication and Session Management [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.owasp.org/index.php/Broken_Authentication_and_Session_Management

[25] OWASP, Top 10 2007- Failure to Restrict URL Access [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.owasp.org/index.php/Top_10_2007-Failure_to_Restrict_URL_Access

[26] OWASP, Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

[27] Autoriaus Chris Shiflett knyga Essential PHP security (2005) ISBN-13: 978-0596006563

[28] Adam Barth, Collin Jackson, and John C. Mitchell straipsniu Robust Defenses for Cross-Site Request Forgery, 15th ACM konferencija Computer and Communications Security (CCS 2008) ISBN:978-1-59593-810-7

[29] OWASP,Top 10 2007-Information Leakage and Improper Error Handling [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą:

http://www.owasp.org/index.php/Top_10_2007-A6#Protection

[30] The Web Application Security Consortium, Information Leakage [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://projects.webappsec.org/Information-Leakage

[31] Steven McElwee, Secure Coding Practices, Part 11: The Checklist (2008) [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.redlightsecurity.com/2008/01/secure-coding-practices-part-11.html

[32] Informit, Preventing RFI Attacks (2009) [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.informit.com/guides/content.aspx?g=security&seqNum=345

[33] Joel Scambray. Hacking exposed– Web applications Autorius (2010) ISBN 9780071740647

[34] OWASP,Top 10 2010-A7-Insecure Cryptographic Storage [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.owasp.org/index.php/Top_10_2010-A7

[35] Yash Kadakia Automated Attack Prevention [interaktyvus]. [žiūrėta 2009-05-20]. Prieiga per Internetą: http://www.acunetix.com/vulnerability-scanner/yashkadakia.pdf

[36] Gunter Ollmann, Professional Services Director, NGS straipsnis Stopping Automated Attack Tools (2008) Criteria [interaktyvus]. [žiūrėta 2010-05-25]. Prieiga per Internetą: http://www.ngssoftware.com/papers/StoppingAutomatedAttackTools.pdf

[37] Webappsec, Web Application Firewall Evaluation Criteria [interaktyvus]. [žiūrėta 2009-01-20]. Prieiga per Internetą: http://projects.webappsec.org/Web-Application-Firewall-Evaluation-Criteria

[38] Amol Sarwate, Hot or not: Web application firewalls for security and regulatory compliance [interaktyvus]. [žiūrėta 2010-05-20]. Prieiga per Internetą: http://www.scmagazineus.com/Hot-or-not-Web-application-firewalls-for-security-and-regulatory-compliance/article/113146/

[39] Jeffrey H. Rubin ir Ravind Budhiraja, Review: Web Application Firewalls [interaktyvus]. [žiūrėta 2010-05-20]. Prieiga per Internetą: http://www.networkcomputing.com/channels/security/showArticle.jhtml?articleID=185303650&pgno=8

[40] Imperva žiniatinklio aplikacijų ugniasienė [interaktyvus]. [žiūrėta 2010-05-14]. Prieiga per Internetą: http://www.imperva.com/products/waf.html

[41] big-ip žiniatinklio aplikacijų ugniasienė [interaktyvus]. [žiūrėta 2010-05-14]. Prieiga per Internetą: http://www.f5.com/pdf/products/big-ip-application-security-manager-ds.pdf

[42] secunia žiniatinklio aplikacijų ugniasienė [interaktyvus]. [žiūrėta 2010-05-14]. Prieiga per Internetą: http://secunia.com/advisories/product/13434/

[43] breach žiniatinklio aplikacijų ugniasienė [interaktyvus]. [žiūrėta 2010-05-14]. Prieiga per Internetą: http://www.breach.com/products/webdefend.html

[44] Projekto PHP IDS svetainė [interaktyvus]. [žiūrėta 2009-04-20]. Prieiga per Internetą: http://php-ids.org

[45] ModSecority svetainė [interaktyvus]. [žiūrėta 2009-04-20]. Prieiga per Internetą: http://www.modsecurity.org/

[46] CERT-LT – LR nacionalinis elektroninių ryšių tinklų ir informacijos saugumo incidentų tyrimo padalinys. https://www.cert.lt/

[47] SQL injekcijų požymiai [interaktyvus]. [žiūrėta 2009-05-24]. Prieiga per Internetą: http://yehg.net/lab/pr0js/pentest/wordlists/injections/SQL.txt

[48] NeoLoad, Programos plėtotojai Neotys. [interaktyvus]. [žiūrėta 2009-05-23]. Prieiga per Internetą: http://www.neotys.com

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Santrumpa, terminas	**Paaiškinimas**
DB (angl. *Database*)	duomenų bazė.
HTML (angl. *Hyper text Markup Language*)	tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete.
OWASP*(angl. The Open Web Application Security Project)*	atviras žiniatinklio programų saugumo projektas.
XML (angl. Extensible Markup Language)	yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.
UTF-8 (angl. *8-bit Unicode Transformation Format*)	viena kintamo ilgio simbolių koduočių.
PHP (angl. *Hypertext Preprocessor*)	dinaminių puslapių kūrimo kalba.
SQL (angl. *Structured Query Language*)	struktūrizuota užklausų kalba.
HTTP (angl. *HyperText Transfer Protocol*)	pagrindinis protokolas pasiekti informacijai Internete.
HTTPS (angl. *HyperText Transfer Protocol Secure*)	hypertekstinis perdavimo protokolas naudojantis duomenų šifravimą.
GET	dažniausia HTTP užklausa, reikalaujanti tam tikro resurso duotu URL adresu.
POST	panašu į GET, bet siunčiama papildoma informacija nematoma URL.
SSL (angl. *Secure Sockets Layer*)	kriptografinis protokolas, skirtas informacijos, sklindančios internete apsaugojimui šifruojant.
MySQL	viena iš reliacinių duomenų bazių valdymo sistemų.
XSS (angl. *Cross-site scripting*)	papildomų programinių kodų įterpimas
IIS	informacinės interneto paslaugos.
WAF(angl. *Web application firewall*)	Žiniatinklio programos ugniasienė
URL (angl. Uniform Resource Locator)	Unikalus šaltinio adresas naudojamas Internete
REGEXP (angl. Regular expression)	Reguliariosios išraiškos
IP	Interneto protokolas arba IP adresas.
CSRF (angl. *Cross-site request forgery*)	Viena iš pažeidžiamumų rūšių.