KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Aidas Ramoška

Apsaugos nuo SQL injekcijų el.verslo svetainėse metodikos sudarymas ir tyrimas

Magistro darbo

Darbo vadovas

Dr. Audronė Janavičiūtė

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Aidas Ramoška

Apsaugos nuo SQL injekcijų el.verslo svetainėse metodikos sudarymas ir tyrimas

Development and research of method of protection against SQL injections in e-commerce websites

Magistro darbo

Recenzentas
prof. dr. B.Paradauskas
2012-05-

Darbo vadovas
Dr. Audronė Janavičiūtė
2012-05-

Atliko
IFN-0/3 gr. stud.
Aidas Ramoška
2012-05-23

Kaunas, 2012

TURINYS

Santrauka

SQL injekcijos atakos taikinys – interaktyvios interneto programos, kurios naudoja duomenų bazės serverius. Šios programos leidžia vartotojams įvesti informaciją ir ją įvedus formuojamos SQL užklausos, kurios siunčiamos į duomenų bazės serverį. Darydamas SQL injekcijos ataką, atakuotojas per įvesties laukus suformuoja kenksmingą SQL užklausos segmentą, kuris modifikuoja buvusią užklausą. Naudodamas SQL injekcijos ataką, atakuotojas gali prieiti prie konfidencialios informacijos, ją modifikuoti ar, apeidamas autorizacijos scenarijų, prisijungti prie sistemos nežinodamas slaptažodžio.

Šiame darbe pasiūlytas saugos modulis perima visą vartotojo įvedamą informaciją, pritaiko saugumo taisykles ir taip padidina saugumą apsisaugant nuo SQL injekcijų el. verslo žiniatinklio programose bei registruoja potencialius bandymus sutrikdyti normalų sistemos darbą.

Norint įdiegti pasiūlytą saugos modulį, nereikia konfigūruoti serverio ar jo programinės įrangos – modulio diegimo metu keičiasi tik žiniatinklio programos failai.

Darbui atlikti pasirinkta PHP programavimo kalba ir MySQL duomenų bazė. Tyrimo metu atlikti testavimo rezultatai parodo, kokius saugos modulio konfigūravimo parametrus reikia taikyti norint užtikrinti maksimalų saugumo lygį.

Development and research of method of protection against SQL injections in e-commerce websites

Summary

The target of SQL injection attack – interactive web programs, which use database servers. Those programs allow users to input information and as it is imputed, it forms SQL queries, which are sent into database server. With SQL injection help, the attacker_using input fields forms harmful section of SQL query, which modifies previous query. Exploiting attack of SQL injection, the attacker may learn confidential information, modify it or connect to system without knowing the password by authorisation bypass.

In this research-paper the proposed security model takes over all information inputted by user, adjusts the safety rules and that way it improves the safety in order to guard from SQL injections at electronic business web systems as well as it register potential attempts to disrupt normal work of the system.

In order to install the proposed safety model there is no need to configure the server or its software because in the moment of installation it changes only files of website programs.

For purpose of executing this work, we use PHP programming language and MySQL database. During the analysis, the received test results show what configuration parameters of safety model we need to use in order to guarantee the maximum level of safety.

1. Įvadas

Žiniatinklio programos vis labiau populiarėja ir tampa įvairių įmonių kasdienybe. Didėjant paklausai žiniatinklio programų kūrėjai vis didesnį dėmesį telkia į programų veikimą, funkcionalumą, tačiau dėl laiko stokos dažniausiai neužtikrina pačio geriausio saugumo lygio. Daugelis programų kuriamos lengvo stiliaus scenarijais (Scripts) ir naudojant tik vieną vartotoją, kuris yra pagrindinis administratorius ir vienintelis valdo sistemą. Šiuo atveju žiniatinklio programos tampa dar labiau pažeidžiamos, nes, nulaužus šio vartotojo prisijungimo duomenis, galima perimti visą sistemą. Gero kodo trūkumas ir nepatvirtinti priėjimai prie duomenų bazės valdymo padidina riziką SQL injekcijos pažeidžiamumams, nebent programuotojas išlaiko įprotį griežtai apriboti visą įvedamą informaciją. Norint teisingai įvertinti problemą, reikia išanalizuoti galimus pažeidžiamumus ir galimas apsisaugojimo priemones.

Pagal projekto OWASP [1] pateiktą statistiką, kurioje įvardinamos labiausiai pažeidžiamų žiniatinklio taikomųjų programų vietos, injekcijos yra pirmoje vietoje. SQL injekcijos yra dažniausiai pasitaikančios verslo sistemose, nes beveik kiekviena žiniatinklio programa turi savo duomenų bazę, iš kurios ima ir į kurią rašo įvairią informaciją. Vienos iš populiariausių duomenų bazių valdymų sistemos yra iš SQL šeimos: MS SQL Server, MySQL, Oracle, PostgreSQL. Visos šios sistemos nėra atsparios prieš SQL injekciją, todėl tai yra labai opi problema – radus spragą verslo sistemoje galima sutrikdyti visos sistemos darbą ar išgauti konfidencialią informaciją – dėl to verslo įmonė gali patirti didelių nuostolių.

Šio darbo paskirtis – pasiūlyti saugos modulio sprendimo metodiką, kuri apsaugotų nuo SQL injekcijų elektroninio verslo žiniatinklio programas, ir sukurti saugos modulį, kuris realizuos pasiūlytą metodiką. Apsaugos modulio tikslas – pritaikyti apsaugos funkcijas įvedamai ir išvedamai informacijai, taip pat registruoti potencialias SQL injekcijos atakas.

Atlikus saugumo grėsmių ir galimų sprendimo būdų analizę, yra sudaromas taisyklių rinkinys. Remiantis juo sukuriamas saugos modulis, kuris šį taisyklių rinkinį integruoja į pasirinktą elektroninio verslo žiniatinklio programą, taip padidindamas jos saugumą.

Tyrimo metu, naudojant sudarytus SQL injekcijų šablonų ir galimų vartotojo įvedamų duomenų failus, patikrinamas incidentų registravimo funkcionalumas, panaudojant žiniatinklio saugumo skenavimo įrankį patikrinamas atsparumas SQL injekcijoms.

Darbo struktūra:

- Pirmajame skyriuje apžvelgti SQL injekcijų atakų tipai ir apsaugos priemonės, aptarti ir palyginti žiniatinklio programų auditui skirti įrankiai.
- Antrajame skyriuje iškeltas darbo tikslas ir suformuluoti uždaviniai bei sudarytas taisyklių rinkinys. Taip pat suprojektuotas saugos modulis ir apibrėžti funkciniai ir nefunkciniai reikalavimai.
- Trečiajame darbo skyriuje pateikiamas saugos modulio veikimo algoritmas, jo integravimo į el. verslo žiniatinklio programą aspektai ir įrankio nustatymai.
- Ketvirtasis skyrius skirtas eksperimentui. Atliekamas SQL injekcijų atakų atrėmimo ir registravimo testas bei tikrinamas klaidų apdorojimas.

Šio darbo metu sukurtas saugos modulis įdiegtas plėtojant „Mokslininkų ir tyrėjų gebėjimų išnaudoti gigabitinio tinklo technologijas ugdymas“ projektą, vykdomą Kauno technologijos universiteto.

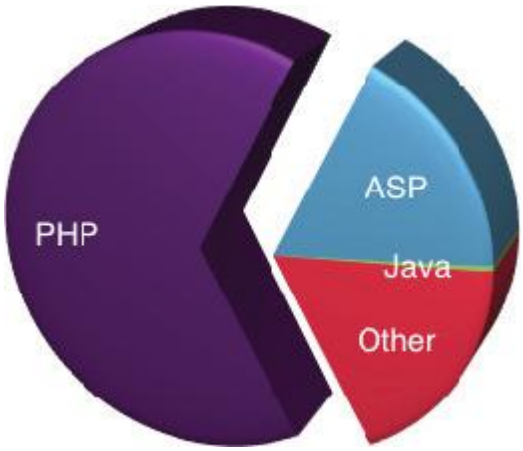
2. SQL injekcijų galimų atakų pavojų ir apsaugos priemonių analizė

2.1. Analizės tikslas

Analizės tikslas – ištirti saugumo grėsmes el.verslo sistemose, išsiaiškinti SQL injekcijų atakų tipus ir galimas apsaugos priemones. Taip pat išanalizuoti esamas apsaugos priemones nuo SQL injekcijų ir įrankius, skirtus aptikti SQL injekcijų pažeidžiamumams bei sudaryti SQL injekcijų atakų šablonų failą.

2.2. Tyrimo sritis, objektas ir problema

Elektroninio verslo žiniatinklio programos saugumas apima vartotojų prisijungimą į sistemą, informacijos rašymą ir skaitymą iš duomenų bazės. Verslo valdymo sistema – programinė įranga, skirta kompiuterizuoti įmonės valdymą, galinti apimti ir integruotis į visus įmonės verslo procesus, naudojama apskaitos vedimo palengvinimui, efektyviam visų resursų išnaudojimui, kontaktų valdymui, efektyviam tiekimo grandinės veikimo užtikrinimui, analitinės įmonės veiklos ataskaitų sudarymui, produkcijos pardavimui ar įmonės teikiamų paslaugų suteikimui. Kadangi PHP (Hypertext Preprocessor) dažniausiai naudojama žiniatinklio taikomųjų programų kūrimui ir ji turi daugiausiai užregistruotų pažeidžiamumų (žr. 1 pav.), tai tirsime programas sukurtas šia kalba.



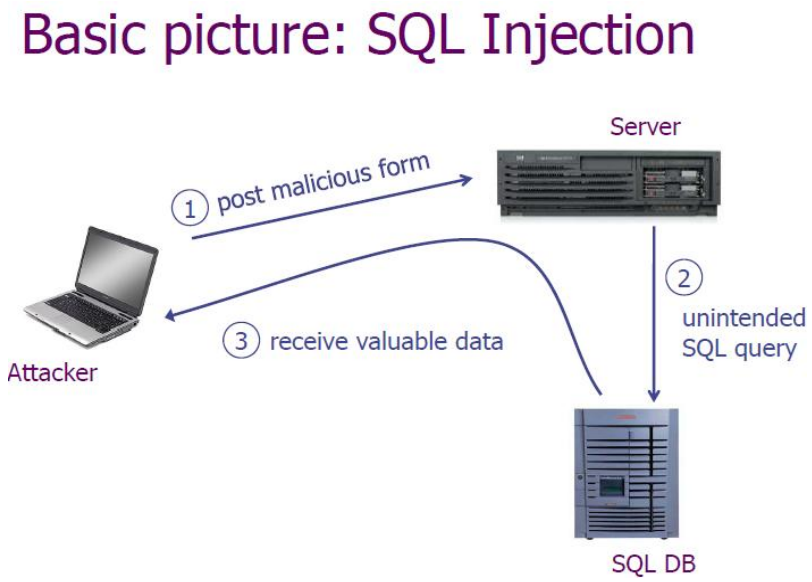
1 pav. Užregistruoti injekcijų pažeidžiamumai „Nacionalinėje pažeidžiamumų duomenų bazėje" [2]

2.3. SQL injekcijos

SQL, kitaip dar žinoma kaip struktūrinė užklausų kalba, yra naudojama siekiant gauti informaciją iš duomenų bazės serverio.

SQL injekcija yra metodas, dažniausiai naudojamas duomenų bazėms eksploatuoti per pažeidžiamas žiniatinklio programas [13]. Šios atakos tipas priklauso bendrai injekcijų atakų kolekcijai. SQL injekcija naudoja SQL užklausas, kurios jungia duomenų bazės serverį per žiniatinklio taikomąsias programas ir pasisavina vartotojo pateiktą įvedamą informaciją užklausos parametrams. Bet kuriuo atveju, programa naudoja iš anksto aprašytas SQL komandas, kuriose gali būti būdų atlikti SQL injekcijas, jeigu nėra įdiegtas teisingas apsaugos mechanizmas.

SQL injekcijos paveikia visas reliacines duomenų bazes visose platformose [20]. Puolimai gali būti naudojami norint išgauti konfidencialią informaciją, apeiti autentiškumo mechanizmus, pakeisti duomenų bazes, taip pat, kai kuriais atvejais, panaudojant saugojimo procedūras galima įvykdyti komandas operacinės sistemos lygmenyje.



2 pav. Paprastas SQL injekcijos atakos pavyzdys[23].

2.3.1. SQL atakų grėsmė

SQL injekcijos ataka leidžia puolėjui ne tik pavogti visą duomenų bazės turinį, bet ir savavališkai atlikti pakeitimus duomenų bazėje. Jis taip pat gali vykdyti elementarias SQL komandas, tokias kaip INSERT (įterpti), UPDATE (atnaujinti), DELETE (trinti) ir t.t. Puolėjas, gavęs administratoriaus slaptažodį, galės vykdyti visas administratoriaus operacijas ar prieiti prie duomenų bazės valdymo sistemos (DBMS) [21]. Dažniausiai pasitaikantys duomenų bazių

serverių produktai neturi jokių mechanizmų, kovojančių su SQL injekcija, kadangi problema yra įsišaknijusi ne pačiose duomenų bazės valdymo sistemose, bet pažeidžiamose programose, kuriose suteikiamos pernelyg didelės teisės vartotojams.

Dažniausiai SQL injekcijos atakų aukos dar ilgai po įsilaužimo net nežino, kad informacija yra sukompromituota ar nutekinta. Daugelyje pavydžių nukentėjusieji net neįtaria, kad jų konfidencialūs duomenys yra pavogti ar pakeisti [4].

2.3.2. SQL injekcijos veikimo principas

Nors SQL injekcijų atakos gali būti įvykdytos prieš bet kurią programą, žiniatinklio programos yra labiausiai pažeidžiamos. Puolėjas gali lengvai žvalgytis po puslapį ir nebijoti būti pagautas. Yra įvairių savanaudiško pasinaudojimo būdų, tačiau visada pagrindinis puolėjo tikslas išlieka toks pat – sugebėti pakeisti SQL užklausą, kad veiktų kitos SQL formuluotės, o ne tos, kurios buvo sukurtos programuotojo. SQL injekcijos gali būti aptiktos visose programos vietose, kur vartotojo įvedama informacija naudojama formuoti SQL užklausas. Parametrais gali būti „Cookie“ reikšmės, POST ar GET metodų duomenys, URL parametrų eilutė ar HTTP antraštė [13].

2.3.3. Pažeidžiamumo aptikimas

Paprasčiausiai aptikti SQL injekciją galima įvedant nenumatytą simbolį ar ženklą į parametrą, per kurį formuojama SQL užklausa, ir žiūrint kaip programa reaguoja į tai. Jei išmetamas pranešimas apie duomenų bazės klaidą, tai tikėtina, kad ši programa yra pažeidžiama. Pavyzdžiui:

```
Http://www.adresas.lt/products.php?id='1
Warning: mysql_fetch_assoc() expects parameter 1 to be resource, boolean given in products.php on line 102
```

2.3.4. Pažeidžiamumo realizavimas

Siekiant išsiaiškinti galimus SQL injekcijų pažeidžiamumus ir jų pasekmes, reikia įsigilinti į konkretų SQL injekcijos atakos realizavimo atvejį.

Pavyzdžiui, atakuojamas objektas yra elektroninė muzikos albumų parduotuvė, kuri turi paieškos mechanizmą pagal atlikėją ir albumo pavadinimą. Paieškos užklausa patikrins visus

įrašus duomenų bazėje ir grąžins tuos, kuriuose atlikėjas lygus ieškomam įrašui, tarkime *ATB*. Tokia užklausa butų:

```
SELECT * FROM music WHERE artist = 'ATB'
```

Ši vartotojo įvesta informacija tapo užklausos dalimi. Šiuo atveju vartotojas pasirenka atlikėjo reikšmę. Šiek tiek pakoregavus vartotojo įvestį *ATB' OR 1=1--* galima gauti visai kitokį rezultatą:

```
SELECT * FROM music WHERE artist = 'ATB' OR 1=1–
```

Kadangi apostrofo ženklas įvedamas vartotojo, tai apriboja atlikėjo reikšmę ir yra galimybė įvesti kitą sąlygą, kurios pagalba galima gauti visai kitą rezultatą. Sąlyga OR 1=1-- reiškia, kad bus peržiūrėti visi įrašai duomenų bazėje ir atrinkti tie, kuriuose atlikėjas yra ATB arba ten, kur 1 lygu 1. Kadangi paskutinė įvestis visada teisinga, tai užklausa grąžins visus duomenų bazės įrašus. Du brūkšneliai (--) vartotojo įvestos informacijos gale nurodo užklausos interpretatoriui, kad viskas, kas eina po jų, būtų ignoruojama. Tai yra būtina, nes vartotojo įvesta informacija, kuri sudaryta iš brūkšnelio, sunaikina eilutę ir formuluotė pati turi gale brūkšnelį. Jei komentaras yra praleidžiamas, tai programos interpretatorius užklausoje matys iš viso tris brūkšnelius ir padarys sintaksinę klaidą.

Viršuje pateiktas pavyzdys jokios žalos programai nedaro, tačiau jei šis metodas būtų panaudotas autorizacijos formoje, tai žinant vartotojo vardą (tarkime *admin*), bet nežinant slaptažodžio, įmanoma nesunkiai prisijungti į sistemą. Šiuo atveju, į informacijos įvestį įvedus *admin'*-- gaunama užklausa:

```
SELECT * FROM users WHERE username='admin'-- and password='*****'
```

Kadangi po dviejų brūkšnelių viskas yra ignoruojama, tokiu atveju nereikia žinoti vartotojo slaptažodžio ir galima lengvai įsibrauti į sistemą. Dar šiek tiek pakoregavus kenksmingą įvestį į *' OR 1=1*-- gaunama:

```
SELECT * FROM users WHERE username='' OR 1=1-- and password='*****'
```

Šiuo atveju nereikia žinoti jokio vartotojo vardo ir jokio slaptažodžio, nes viskas, kas eina po „--", yra ignoruojama. Ši užklausa grąžina tas reikšmes, kur vartotojo vardas yra tuščias arba 1 lygu 1, o tai yra pirmas įrašas lentelėje, kuris dažniausiai žymi pagrindinį administratorių.

Žinoma, tai nebus lengva įgyvendinti praktiškai, nes dažniausiai autorizacijos formos būna apsaugotos. Puolėjui kartais reikia modifikuoti injekcijos atakų informacijos įvestis tam, kad pavyktų šis metodas. Tokiu atveju jam gali tekti naudoti HEX ar ASCII (American Standard Code for Information Interchange) koduotes vietoj skaičių ar jungiamųjų simbolių tam, kad būtų

suformuota veiksminga užklausa ir išvengta pavienių kabučių [22]. Tam tikri simboliai, tokie kaip: ‘,”,--, #, turi specialią prasmę, kuri paprastai pamirštama, kai ieškoma programų su SQL injekcijos pažeidžiamumais.

Nors programos yra apsaugotos nuo elementariausių SQL injekcijų atakų, kitokio tipo filtrų apėjimas vis dar yra įmanomas. Problema atsiranda tada, kai apostrofo ženklas išsaugomas duomenų bazėje. Tarkim *admin‘* pavirsta į *admin‘‘*, tada duomenys saugomi duomenų bazėje kaip *admin‘‘* ir vėliau tai panaudojama kitai užklausai. SQL injekcijos rizika vėl didėja, kadangi reikšmė paimta iš serverio bus *admin‘* [5].

2.4. SQL injekcijos tipai

SQL injekcijos ataką galima įvykdyti įvairiais būdais. Šioje dalyje apžvelgiami SQL injekcijų atakų skirtingi tipai, tikslai, aprašymai ir jų pavyzdžiai. Reikėtų atkreipti dėmesį, kad skirtingi atakų tipai nebūtinai atliekami pavieniui, dauguma jų naudojami kartu ar nuosekliai, vienas po kito, priklausomai nuo tam tikro puolėjo tikslų.

2.4.1. Paprastos injekcijos

Šio tipo SQL injekcijos atakos tikslas yra apeiti autentifikavimą, identifikuoti pažeidžiamus programos parametrus ir išgauti duomenis. Atakos tikslas yra įterpti kodą į vieną ar daugiau sąlyginių formuluočių tam, kad jie visada būtų įvertinami kaip teisingi. Šių atakų pasekmės priklauso nuo to, kaip užklausos rezultatai yra naudojami programoje. Atakuotojas panaudoja pažeidžiamus įvesties laukelius, kurių įvesta informacija naudojama užklausoje WHERE (kur) sąlyga. Tam, kad ši ataka veiktų, puolėjas turi atsižvelgti ne tik į pažeidžiamus parametrus, bet taip pat ir į kodavimo konstrukciją, kuri įvertina užklausos rezultatus. Dažniausiai ataka laikoma sėkminga, kai programa parodo visus įrašus arba atlieka tam tikrus veiksmus [19].

2.4.2. Neteisėtos/logiškai neteisingos užklausos

Šios SQL injekcijos atakos tikslas – atpažinti pažeidžiamus parametrus, nustatyti duomenų bazės struktūrą ir išgauti duomenis. Ši ataka leidžia puolėjui rinkti svarbią informaciją apie žiniatinklio programų duomenų bazės tipus ir struktūras, bei internetines paslaugas. Ataka laikoma preliminaria – tai informacijos rinkimo žingsnis kitoms atakoms. Šios atakos metu pažeidžiamumas pasireiškia per numatytą klaidos išvedimą į puslapį, kurį atverčia programos serveris ir kuris dažniausiai yra pernelyg aiškiai aprašytas. Iš esmės, pats faktas kad klaidos

(*error*) žinutė yra generuojama, gali dažnai puolėjui atskleisti parametrų pažeidžiamumą ir kitą vertingą informaciją. Papildoma klaidos informacija, kurios tikslas buvo padėti programuotojui surasti ir pašalinti programos klaidas, dar labiau padeda puolėjui gauti informaciją apie duomenų bazės schemą. Atlikdamas šią ataką puolėjas įterpia į duomenų bazės užklausą sintaksės, loginę ar kitokio tipo (*integer/string*) klaidą. Sintaksinės klaida panaudojama pažeidžiamiems parametrams identifikuoti. Loginės klaidos dažnai atskleidžia lentelių ir stulpelių pavadinimus. Tipo klaida gali būti panaudota, kad atakuotojas sužinotų tam tikrų stulpelių ar duomenų tipus arba išgautų duomenis.

2.4.3. Suvienytos užklausos ataka

Pagrindinis suvienytos užklausos atakos (Union attack) [10] tikslas yra apeiti autentifikavimą (atpažinimą) ir išgauti duomenis. Šios atakos metu puolėjas naudoja pažeidžiamus parametrus, kad pakeistų grįžtamus duomenis iš duotos užklausos. Tokiu būdu puolėjas gali apgauti pačią programą išduoti kitokią (skirtingą) informaciją negu buvo numatyta kūrėjo. Atakuotojai tai padaro įterpdami tokios formos formuluotę :

```
UNION SELECT <injekcijos įterpta užklausa>
```

Kadangi puolėjas pilnai kontroliuoja įterptąją užklausą ir jis gali panaudoti ją taip, kad užklausa išgaus informaciją iš tiksliai numatytos lentelės. Šios atakos rezultatas yra tas, kad duomenų bazė išduoda duomenų paketą, kuris yra originalios pirmosios užklausos rezultato ir įterptos antrosios užklausos rezultato sąjunga. Pavyzdžiui, jei prisijungimo užklausa yra:

```
SELECT vartotojo_id FROM vartotojai WHERE login='*****' AND pass=
'******'
```

Įvesdami atakos eilutę į prisijungimo laukelį:

```
' UNION SELECT Card_Nr from Kreditines_korteles WHERE vartotojo_id=10032 - -
```

Atlieka tokią užklausą:

```
SELECT vartotojo_id FROM vartotojai WHERE login='' UNION SELECT Card_Nr from
Kreditines_korteles WHERE vartotojo_id=100 - - AND pass= '******'
```

Tarkime, kad niekur nėra tokio prisijungimo, atitinkančio "", tad originalioji pirmoji užklausa negrąžina nieko (*return null*), tuo tarpu antroji (papildoma) užklausa pateikia duomenis iš "Kreditines_korteles" lentelės. Tokiu atveju duomenų bazės užklausa grąžintų duomenis iš stulpelio "Card_Nr", kur vartotojo ID yra "100." Duomenų bazė paima šių dviejų užklausų

rezultatus, sujungia juos ir grąžina juos programai. Taip piktavalis gali sužinoti sąskaitos numerį ir kitą vertingą informaciją.

2.4.4. Prisišlietos užklausos (Piggy-backed)

Šios atakos pagrindinis uždavinys yra išgauti, pridėti ar modifikuoti duomenis, atlikti paslaugos atmetimą, vykdyti nuotolines komandas. Su šiuo atakos tipu puolėjas mėgina įterpti papildomas užklausas į originaliąją. Šis tipas išsiskiria iš kitų tuo, kad puolėjas nemėgina modifikuoti originalios užklausos, vietoj to jis mėgina įtraukti naują, skirtingą užklausą, kuri "prisišlieja" prie originalios užklausos. Kaip to rezultatas, duomenų bazė gauna daugialypes SQL užklausas. Pirmoji yra numatytoji užklausa, kuri veikia kaip normali. Sekančios yra įterptos užklausos, kurios veikia kaip papildymas pirmajai. Šio tipo atakos yra ypač žalingos. Puolėjas gali įterpti iš esmės bet kokią SQL komandą, įskaitant ir saugojimo procedūras prie papildomų užklausų ir priversti jas veikti kartu su originaliąja užklausa. Šio tipo atakos pažeidžiamumas slypi duomenų bazių nustatymuose, kurie leidžia daugialypes formules. Jei puolėjas įveda į slaptažodžio laukelį: [6]

```
`; drop table users - -
```

programa generuoja tokią užklausą:

```
SELECT vartotojo_id FROM vartotojai WHERE login='*****' AND pass= ``; drop table users - -'
```

Po pirmosios užklausos užbaigimo, duomenų bazė atpažintų užklausos ribos ženklą (";") ir įvykdytų įterptąją antrąją užklausą. Antrosios užklausos įvykdymo rezultatas – pašalinti lentelės vartotojus, kas tikriausiai sunaikintų vertingą informaciją. Kito tipo užklausos galėtų įterpti naujus vartotojus į duomenų bazę arba įvykdyti saugojimo procedūras.

2.4.5. Saugojimo procedūros

Atakos tikslas – panaudoti saugojimo procedūras privilegijų išplėtimui, paslaugų atmetimui ir nuotolinių komandų vykdymui [3]. Šio tipo SQL injekcijos atakos mėgina vykdyti saugojimo procedūras, pateiktas duomenų bazėje. Dauguma duomenų bazių valdymo sistemų gamintojai pristato programas su standartiniais saugojimo procedūrų komplektais, kurie praplečia duomenų bazės funkcionalumą ir leidžia sąveikauti su operacine sistema. Puolėjas, nustatęs, kuri duomenų bazės sistema naudojama, atakos eilutes gali meistriškai sukurti taip, kad vykdytų

saugojimo procedūras, numatytas pagal nutylėjimą toje duomenų bazėje, taip pat ir procedūras, kurios sąveikauja su operacine sistema.

2.4.6. Išvados darymo tipas

Aklosios injekcijos (*Blind injection*) [19] arba išvados darymo tipas yra viena iš SQL injekcijos atakos rūšių, kai informacija nustatoma iš puslapio elgesio užduodant tiesa/melas klausimus serveryje. Jei pateikta formuluotė įvertinama kaip tiesa (teisinga), tai programa tęsia savo funkcijas įprastai. Jei formuluotė įvertinama kaip melas (neteisinga) - net ir nesant klaidos žinutės aprašo – puslapis skiriasi ženkliai nuo normaliai funkcionuojančio puslapio (dažnai puslapio atvaizdavimas užlaikomas nedideliam laiko tarpui). Šios atakos tikslas yra identifikuoti pažeidžiamus parametrus, išgauti duomenis ir nustatyti duomenų bazės schemas. Su šia injekcija puolėjai paprastai puola svetainę, kuri pakankamai apsaugota. Injekcijos galimybė yra, bet nėra jokio naudingo grįžtamojo ryšio per duomenų bazių klaidų žinutes. Tokioje situacijoje puolėjas paleidžia komandas į svetainę ir tada stebi kaip funkcijos joje keisis. Reikia pažymėti, kad svetainių elgesys, kai jis toks pat ir kai jis keičiasi, yra svarbus, nes puolėjas ne tik gali nustatyti ar tam tikri parametrai pažeidžiami, bet gali nustatyti ir papildomą informaciją apie duomenų bazės reikšmes.

2.4.7. Alternatyvus kodavimas

Šios atakos metu, įterptas tekstas modifikuojamas taip, kad išvengtų apsauginio kodavimo ir kitų automatizuotų prevencinių metodų aptikimo. Ši ataka naudojama susiejant ją su kitomis atakomis. Kitaip sakant, alternatyvi koduotė nepateikia jokio unikalaus būdo atakuoti programą, ji yra tik metodas, kuris leidžia išvengti aptikimo ir užkirtimo technikos ir šitaip eksploatuoti pažeidžiamumus, kurie kitu atveju neveiktų. Šie išvengimo manevrai dažnai labai reikalingi, nes gynybinė kodavimo praktika skenuoja ir ieško tam tikrų jai žinomų blogų simbolių, tokių kaip viengubos kabutės ir komentarų operatoriai. Norint išvengti tokios gynybos, puolėjai turi pajungti alternatyvius metodus koduojant savo atakų eilutes. Naudojami hexadecimal, ASCII ir Unicode simbolių kodavimai [11]. Bendri skenavimo ir aptikimo metodai nesistengia įvertinti visų specialiai užkoduotų eilučių, šitaip leisdami tokioms atakoms veikti neaptiktoms.

2.5. Apsisaugojimas nuo SQL injekcijų

Yra žinoma įvairių daugiau ar mažiau patikimų funkcijų, skirtų apsisaugoti nuo SQL injekcijų: įvairūs scenarijai, saugojimo procedūros, įvairios funkcijos. Dauguma kūrėjų apsisaugojimui nuo SQL atakų yra linkę naudoti saugojimo procedūras [14]. Tiesa, kad saugojimo procedūros duoda naudos, tačiau viskas priklauso nuo to, kaip parašyta saugojimo procedūra ir kaip ją panaudoja programuotojas. Jei saugojimo procedūra silpnai parašyta ar netinkamai panaudota – išlieka nemaža tikimybė SQL injekcijos atakai.

Geresnis būdas norint apsisaugoti nuo SQL injekcijos yra parametrizuotos užklausos, kitaip dar žinomos kaip „Prepared statements" – paruoštos formuluotės metodas [5]. Šis metodas užtikrina, kad per vartotojo informacijos įvesties laukus atakuotojas neiškraipytų anksčiau aprašytos SQL užklausos.

2.5.1. Įvestos informacijos tikrinimas

Įvestos informacijos tikrinimas - tai vienas iš patikimiausių būdų išvengti SQL injekcijų. Griežtai tikrinant vartotojų įvedama informaciją nesunkiai galima išvengti ir pavojingiausių, ir gerai apgalvotų atakų. Žinant, kokie tiksliai duomenys turi būti įvesti į tam tikrą lauką, nesunkiai sudaromi šių duomenų šablonai.

Naudojant Regular Expression klasę, sutrumpintai apibrėžiama kaip Regex, sudaro galimybę tikrinti įvedamų duomenų atitikimą apibrėžtiems šablonams. Pavyzdžiui, norint patikrinti ar gerai įvestas elektroninis paštas, reikia pasirašyti funkciją:

```
static function EPastas(&$input)
      {
      Return preg_match('/^[A-z0-9_.]+@[A-z0-9_.]+[.][A-z]{2,4}$/', $input);
      }
//  preg_match  funkcija  tikrina  ar  įvesta  vartotojo  informacija  atitinka
šabloną
// Regex - '/^[A-z0-9_.]+@[A-z0-9_.]+[.][A-z]{2,4}$/'
// Regex taisyklė:
// /^ - pradžia
// [A-z0-9_.] – nurodo, kad turi būti bent vienas simbolis iš apibrėžtų ribų,
tai visi skaičiai, visos lotyniškos raidės bei _ ir . simboliai
// +@ - turi būti būtinai eta simbolis
// [A-z0-9_.] vėl šablonas pakartojamas
// +[.] – turi būti taškas
```

```
// [A-z]{2,4} - po taško turi sekti nuo dviejų iki keturių lotyniškų raidžių
// $/ - pabaiga
```

Tikrinant vartotojo įvestą elektroninį paštą pagal šį šabloną, praktiškai neįmanoma įvykdyti jokios SQL injekcijos. Vartotojui neįmanoma įvesti ‘, “, --, # ar kitų simbolių.

2.5.2. Paruoštos formuluotės metodas

Šio metodo principas labai paprastas. SQL formuluotė šiuo atveju paruošiama dviem etapais. Pirma – programai suteikiama užklausos struktūra, kur, vietoj būsimo vartotojo įvedamos informacijos, yra vietos rezervavimo ženklas (*placeholder*). Antra – visas vietos rezervavimo ženklų turinys tiksliai nustatomas per programą. Pavyzdžiui:

```
$id = $_GET[‘id’];
$statement = $db_connection->prepare(“SELECT name FROM users WHERE id = ?”);
$statement ->bind_param(“i”, $id); // Pirmas argumentas nurodo įvedamos ar gaunamos informacijos tipą, šiuo atveju kintamojo $id tipą.
$statement->execute();
```

Šis metodas konstruodamas ir įvykdydamas SQL užklausą automatiškai tikrina kiekvieno vartotojo įvedamo parametro tipą. Tai yra būtina norint išvengti SQL injekcijos atakos. Šis metodas negali surasti pažeidžiamumų programos kode, tačiau su juo galima išvengti pažeidžiamumų ir įgyti imunitetą SQL injekcijoms.

2.5.3. PHP funkcijos, apsaugančios nuo SQL injekcijų

Kuriant programas taip pat labai svarbu naudoti tam tikras funkcijas, kurios sumažina SQL injekcijos atakų grėsmę. Pirmiausia aptarkime *mysql_query()* [9] – tai funkcija, naudojama kurti užklausoms į duomenų bazę. Ši funkcija nepriima paketinių užklausų su daugialypėmis užklausomis ir taip apsaugo nuo kai kurių elementarių SQL injekcijų atakų. Pavyzdžiui, tampa nebeįmanoma užbaigti užklausos su kabliataškiais ir įvykdyti taip vadinamą DROP TABLE ataką (admin‘ DROP TABLE--).

	ASP	ASP.NET	PHP
MySQL	No	Yes	No
PostgreSQL	Yes	Yes	Yes
Microsoft SQL Server	Yes	Yes	Yes

3 pav. Duomenų bazės valdymo sistemos, kurios palaiko paketines užklausas [13]

Tačiau vengimas daugialyčių formuluočių įvykdymo negarantuoja apsisaugojimo nuo SQL injekcijų. Kai programa neišvalo kintamojo nuo reikiamų specialiųjų simbolių, išlieka didelė grėsmė išgauti slaptą informaciją iš duomenų bazės. Šiuo atveju PHP rekomenduoja naudoti *mysql_real_escape_string()* [19]. Ši funkcija prideda įžambųjį kairinį brūkšnį prieš tam tikrus simbolius, tokius kaip: *\x00, \n, \r, \, ', "*. Pavyzdžiui, jei jūs bandysite prisijungti nežinodami slaptažodžio su įvestimi *admin' OR 1=1--*, tai jums nebepavyks, nes interpretatorius gaus užklausą:

```
SELECT * FROM users WHERE username='admin'\ OR 1=1--' and password=''
```

Taigi, šiuo atveju SQL serveris gražintų užklausą tik tada, jei vartotojo vardas būtų lygus *admin\' OR 1=1--* ir slaptažodis teisingas.

2.6. Prototipų analizė

Sukurta daug sistemų ir apsaugos priemonių skirtų išvengti SQL injekcijų atakų grėsmės: saugojimo procedūros [14], įvairios incidentų aptikimo sistemos (IDS) [12] ir pasiūlyta nemažai metodų saugumui užtikrinti kuriamai naujai programinei įrangai [8,7].

1. lentelė Įrankių apibendrinimo lentelė

Metodo pavadinimas	Pritaikymas serverio pusėje	Pritaikymas kuriamam projektui	Pritaikymas jau sukurtam projektui
Saugojimo procedūros	+		+
Paruoštos formuluotės metodas		+	
IDS sistemos	+		+
Statinės kodo analizės įrankiai		+	
Įvestos informacijos tikrinimas		+	

Norint apsaugoti jau sukurtą el. verslo žiniatinklio programą reikia turėti priėjimą prie serverio, tada būtų galima papildyti duomenų bazės servisą saugojimo procedūromis, kurios apsaugo nuo SQL injekcijų atakų ar įdiegtį į serverį įsilaužimo aptikimo sistemą (IDS).

Kuriant naują el. verslo žiniatinklio programą galima panaudoti paruoštos formuluotės metodą ar griežtai tikrinti visą vartotojo įvedamą informaciją, tačiau norint apsaugoti jau sukurtą el. verslo žiniatinklio programą šie metodai tampa beverčiais, nes reiktų perprogramuoti visą programą.

2.7. SQL injekcijų atakų šablono sudarymas

SQL injekcijų atakų šablonas sudaromas iš dažnai pasitaikančių simbolių, žodžių ar tam tikrų kodo fragmentų, pvz.: ‘, “, SELECT, 1=1, --, #, UNION ir t.t. Šie fragmentai SQL injekcijų atakose atlieka tam tikrą funkciją ir be jų sunku atakuotojui pakenkti sistemai. Sudarant atakų šabloną buvo panaudotas pagal OWASP rekomendacijas kompiuterių saugumo specialistų kompanijos „YEGH“ paruoštas pažeidžiamumų požymių failas [25]. Šis failas papildytas dažnai atakuotojų naudojamomis funkcijomis: waitfor delay, xp_cmdshell, char, ascii ir t.t. [22], kurios pakeičia simbolių reikšmes, grąžinant duomenis uždelsia tam tikrą laiko tarpą ar iškviečia sistemines funkcijas.

Sukurtas atakų šablonų failas bus naudojamas testuojant potencialių atakų registravimą.

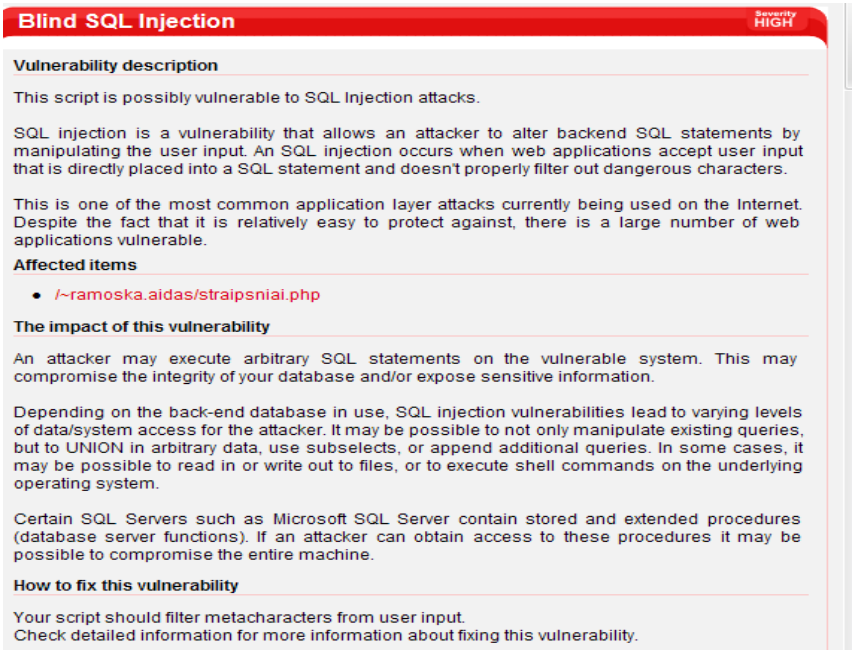
2.8. Testavimo įrankiai

Internete gausu įvairiausių įrankių, skirtų testuoti žiniatinklio taikomąsias programas. Vieni jų yra galingesni, apimantys ne tik pačias sistemas, bet ir interneto serverius, kuriuose tos programos yra patalpintos. Kiti sukonkretinti ir ieško tik tam tikrų pažeidžiamumų, tokių kaip XSS ar SQL injekcijų. Šiame skyriuje apžvelgsime keletą žymesnių produktų, kuriuos naudosime ir savo sukurtos metodikos testavimui.

2.8.1. Acunetix Web Security Scanner

Acunetix – žiniatinklio programų saugumo skeneris [15]. Tai labai galingas įrankis, apimantis daug pažeidžiamumų, taip pat ir SQL injekcijas. Jis skirtas atlikti auditą kuriamoje ar jau esamoje žiniatinklio programoje. Yra daug įvairių nustatymų ir įvairių skenavimo galimybių, taip pat ir specialių įrankių, skirtų SQL injekcijoms. Vaizdžios ir detalios klaidų ataskaitos (pav. 4), kuriose aprašyta ne tik kaip išnaudoti šį pažeidžiamumą, bet ir kaip jį pataisyti. Tačiau, jei

skenuojama sistema yra didelės apimties, šis testas gali ilgokai užtrukti. Acunetix įrankis puikiai susidoroja su aklosiomis SQL injekcijomis (*Blind injection*). Yra prisijungimo į testuojamą sistemą mechanizmas, o tai leidžia patikrinti ir tas tiriamo projekto vietas, kurios neautorizuotiems vartotojams neprieinamos.



4 pav. Acunetix žiniatinklio programų skenavimo įrankio detali ataskaita

2.8.2. Rational AppScan

Rational AppScan – „HP“ kompanijos produktas [16]. Puikus įrankis skirtas patikrinti žiniatinklio programos saugumą. Jame įtrauktas ir serverio saugumo, ir jo programinės įrangos patikrinimas. Realizuotas daugumos pažeidžiamumų patikrinimas: SQL injection, XSS, autorizacijos apėjimo ir pan.

Rational AppScan kiekvienam surastam pažeidžiamumui pasiūlo sprendimo variantą pagal OWASP [1] rekomendacijas.

2.8.3. Havij

Havij – specialiai SQL injekcijoms skirta galinga programa [17]. Šioje programoje galima pasirinkti įvairius nustatymus, galima atlikti aklają SQL injekcijos ataką (*Blind injection*), nurodyti įvairius simbolių kodavimo parametrus (*Unicode, Hex*). Aptikusi pažeidžiamumą, ši sistema išanalizuoja visą duomenų bazės struktūrą, gali pateikti visus įrašus iš duomenų bazės, taip pat ir vartotojo „hash“ slaptažodžius. Ši programa viską atlieka automatiškai.

2.8.4. Įrankių analizės apžvalga

Išanalizavus visus įrankius padaryta jų apibendrinimo lentelė (žr. 2. lentelė). Kaip matome lentelėje „Havij“ įrankis neturi galimybės prisijungti į sistemą ir neformuoja pažeidžiamumų ataskaitų. „Rational AppScan“ skeneris neaptinka aklųjų SQL injekcijų ir neturi pažeidžiamumų išnaudojimo galimybės. Tad apibendrinus galima pastebėti, kad daugiausiai galimybių susijusių su SQL injekcijų pažeidžiamumais turi Acunetix žiniatinklio programų skeneris. Jis turi galimybę prisijungti į sistemą, aptikti akląsias SQL injekcijas, formuoti detalias pažeidžiamumo ataskaitas ir šio įrankio pagalba galima išnaudoti rastus pažeidžiamumus.

2. lentelė Įrankių apibendrinimo lentelė

Testavimo įrankio pavadinimas	Prisijungimo į sistemą galimybė	Detali pažeidžiamumų ataskaita	Aklosios SQL injekcijos palaikymo galimybė	Pažeidžiamumo išnaudojimo galimybė	Yra nemokama versija
Acunetix	Yra	Taip	Yra	Yra	Ribotos galimybės
Havij	Nėra	Nėra	Yra	Yra	Ribotos galimybės
Rational AppScan	Yra	Taip	Nėra	Nėra	Ribotos galimybės

2.9. Analizės išvados

Išnagrinėjus SQL injekcijų atakų tipus nustatyta, kad visi vartotojo įvedami duomenys ir perduodami sistemai per POST, GET, COOKIE parametrus gali būti potencialios SQL injekcijos atakos.

Išsiaiškinus galimas apsaugos priemones PHP žiniatinklio programose, buvo nustatyta, kad norint apsaugoti jau sukurtas programas, dauguma atvejų viskas priklauso nuo serverio konfigūracijos ir jo programinės įrangos.

Išanalizavus SQL injekcijos atakoms būdingus simbolius, kodo fragmentus ir duomenų bazėse naudojamas funkcijas buvo sudarytas atakų šablonų failas.

Susipažinus su žiniatinklio auditui skirtais įrankiais ir apžvelgus jų galimybes Acunetix žiniatinklio programų skeneris atitiko daugiausiai keliamų reikalavimų.

3. Apsisaugojimo metodika ir apsauginės sistemos projektas

3.1. Darbo tikslas ir keliami reikalavimai

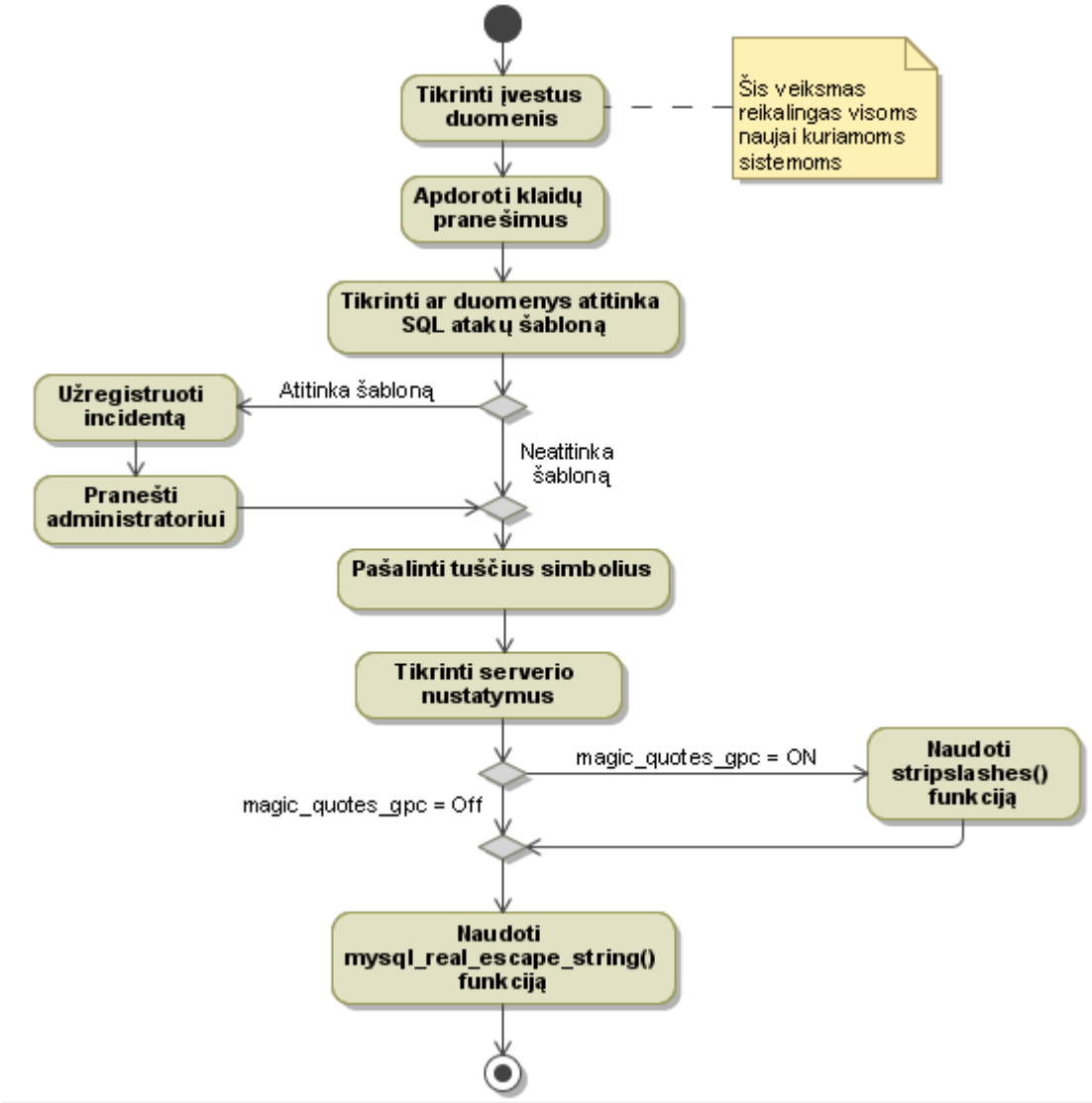
Darbo tikslas – remiantis SQL injekcijų grėsmių ir apsisaugojimo priemonių analizės duomenimis sukurti metodiką, kuri užtikrintų el. verslo sistemos apsaugą nuo SQL injekcijų.

Pagrindinis šioje darbo dalyje sprendžiamas uždavinys – sukurti saugos modulį, kuris realizuotų pasiūlytą metodiką ir būtų naudojamas siekiant apsaugoti jau sukurtą elektroninio verslo žiniatinklio programą. Kuriamo modulio tikslas nėra vien tik sustabdyti galimą ataką. Šis apsaugos mechanizmas turi registruoti bandymus pakenkti sistemai ir apie tai informuoti atsakingus asmenis.

3.2. Apsaugos nuo SQL injekcijų metodikos sudarymas

Norint apsisaugoti nuo SQL injekcijų reikia filtruoti visą vartotojo įvedamą informaciją, tikrinti duomenis ar juose nėra SQL injekcijoms būdingų žodžių ar simbolių. Radus SQL injekcijoms būdingų bruožų – sutvarkyti duomenis ir visus incidentus užregistruoti, ir apie tai informuoti administratorių.

Taip pat reikia tikrinti visus klaidos pranešimus grąžinamus vartotojui.



5 pav. Metodikos veiklos schema

3.2.1. Klaidų apdorojimas

Norint išvengti SQL injekcijų atakų visų pirmą reikia apdoroti visas galimas klaidas, nes atakuotojas gali iš jų išgauti atakai naudingos informacijos ir pastebėti, kad sistema yra pažeidžiama. Ši metodika pritaikyta *php* programavimo kalbai, o ši kalba turi dvi naudingas funkcijas klaidų apdorojimui:

error_reporting(0) – išjungia klaidų atvaizdavimą;

set_error_handler(klaidu_funkcija) – įvykus klaidai iškviečia numatytą funkciją, kuri apdoroja klaidą.

Dėka pastarosios funkcijos, galima informuoti vartotoją apie įvykusią klaidą, nors ir išjungtas klaidų atvaizdavimas. Vietoj informatyvios atakuotojui klaidos, kuri parodo detalią informaciją, pvz.:

```
Warning: mysql_fetch_assoc() expects parameter 1 to be resource, boolean given in E:\Programos\wamp\www\mano darbai\Zurnalas\straipsniai.php on line 108
```

bus išvedamas pranešimas apie blogus duomenis:

```
Klaida:[2]Duomenų grąžinimo klaida
```

3.2.2. Serverio nustatymų patikrinimas

Elektroninio verslo sistemos gali būti eksploatuojamos įvairiuose serveriuose, kurių konfigūracija gali skirtis. Todėl reikia patikrinti *magic_quotes_gpc()* funkciją – ar ji išjungta, ar įjungta ir, jei išjungta - imtis atitinkamų veiksmų. Ši funkcija pagal nutylėjimą yra įjungta, tačiau dažnai pasitaiko serverių, kuriuose ji yra išjungiama. Visi serverio nustatymai susiję su *php* konfigūracija yra saugomi php.ini faile.

Magic_quotes_gpc() funkcija automatiškai prideda \ (*backslash*) simbolį prieš viengubas ir dvigubas kabutes, \ (*backslash*) ir NULL simbolius, beveik visoms HTTP užklausoms (GET, POST, ir COOKIE).

Sužinoti, ar įjungtas šis nustatymas, galima pasinaudojus *get_magic_quotes_gpc()* funkcija.

Jei ši funkcija įjungta, reikia prieš kiekvieną iš vartotojo gaunamą kintamąjį įvykdyti *stripslashes()* funkciją, kurios paskirtis pašalinti \ (*backslash*), kad panaudojus *mysql_real_escape_string()* nebūtų simboliai apdorojami antrą kartą.

3.2.3. Funkcijos *mysql_real_escape_string()* naudojimas

Funkcija *mysql_real_escape_string()* prideda įžambųjį kairinį brūkšnį prieš tam tikrus simbolius, tokius kaip: *\x00, \n, \r, \, ', ".* Ji padeda išvengti elementarių SQL injekcijų. Šią funkciją reiktų iškviesti visai per POST, GET užklausas gaunamai vartotojo įvestai informacijai apdoroti. Taip pat pravartu patikrinti ir COOKIE kintamuosius, nes patyręs atakuotojas gali pakeisti ir šiame kintamajame esančią informaciją. Ši funkcija iškviečiama tik prisijungus prie duomenų bazės, o pasiūlytas saugos modulis bus iškviečiamas dar neprisijungus programai prie duomenų bazės.

Išnagrinėjus simbolius, kuriuos apdoroja *mysql_real_escape_string()* funkcija ir panaudojus *str_replace()* [25] funkciją, galima prieš pavojingus simbolius uždėti \ ir taip nukenksminti potencialiai pavojingus vartotojo įvedamus duomenis. Pvz.:

```
return str_replace(array('\\', "\0", "\n", "\r", "'", '"', "\x1a"),
array('\\\\', '\\0', '\\n', '\\r', "\\'", '\\"', '\\Z'), $inp);
// str_replace funkcija keičia vieno masyvo elementus į kito masyvo elementus.
// $inp kintamasis – vartotojo įvesti duomenys.
```

3.2.4. Tuščių simbolių išvalymas

Svarbu pašalinti tarpo simbolius kintamojo pradžioje ir pabaigoje. Tai nesunku padaryti panaudojus *trim()* funkciją. Ji ištrina iš kintamojo pradžios ir pabaigos tuščias vietas, naujos eilutes pradžią, nulinius baitus.

O funkcija *trim($kintamasis, "\x00..\x1F")* išvalo „binary“ tipo kintamuosius.

3.2.5. Incidentų fiksavimas

Kiekvienas bandymas pakenkti sistemai turi būti užregistruotas ir tą bandymą atlikęs asmuo turi būti identifikuotas. Prireikus priėjimas prie sistemos šiam asmeniui turi būti užblokuotas.

Atlikus pakartotinį incidentą sistema identifikuoja vartotoją kaip potencialų atakuotoją ir apie tai informuojamas atsakingas asmuo, kuris gali šį vartotoją užblokuoti ar tiesiog ištrinti.

Incidentų fiksavimas vyksta tada, kai vartotojo įvesti duomenys atitinka SQL atakų šablonus. Šablonai sudaromi panaudojant žodžius UNION, SELECT, DROP, INSERT, UPDATE, DELETE, kurie dažniausiai naudojami SQL injekcijoms atlikti. Taip pat įvairių simbolių kombinacijos, tokios kaip: 1=1, -- ir t.t.

Prieš tikrinant, ar vartotojų įvedami duomenys atitinka SQL injekcijų atakų šablonus, reiktų pašalinti SQL kalbos komentarus. Komentarus reikia išmesti panaudojant *preg_replace()*. Komentaras prasideda /* ir baigiasi */ šiais simboliais. Sudarius regex (Regular expression) ir panaudojus *preg_replace()* funkciją nesunkiai galima tai padaryti:

```
$regex = '#\/\*.*\*\/#U';
// Regex taisyklė:
// # - pradžia
// \/\* - ieško /*
// .* - nurodo, kad gali būti bet kiek simbolių
```

```
// \*\/ - ieško */
// # - pabaiga
// U – nurodo, kad regex būtų negodus ir neimtų nuo pirmo iki paskutinio komentaro ženklo, o tikrintu juos po vieną.
$Be_komentaru = preg_replace($regex,"",$Su_komentarais);
```

Atsakingas asmuo informuojamas siunčiant jam elektroninį laišką su detaliu aprašymu: kada įvyko incidentas, kas vykdė ir ką bandė padaryti.

3.3. Apsauginės sistemos projektas

Šis saugos modulis integruos pasiūlytą taisyklių rinkinį į jau sukurtą elektroninio verslo žiniatinklio programą, taip padidindamas esamos programos saugumą.

3.3.1. Elektroninio verslo sistemai keliami reikalavimai

Norint integruoti sukurtą modulį, kuris realizuoja pasiūlytą metodiką į elektroninio verslo sistemą, jis turi atlikti šiuos reikalavimus:

Reikalavimai duomenims

Duomenys gaunami iš HTTP užklausų GET, POST, COOKIE. Šie duomenys yra nešifruoti.

Reikalavimai programavimo kalbai

Programa turi būti parašyta PHP programavimo kalba. Visi programos vykdomi failų plėtiniai turi būti **.php* ir **.inc*.

Reikalavimai duomenų bazei

Duomenų bazė turi būti reliacinė.

3.3.2. Funkciniai ir nefunkciniai reikalavimai

Funkciniai reikalavimai keliami moduliui:

- Klaidų apdorojimas;
- Apsauga nuo kenksmingų vartotojo įvedamų duomenų;
- Įvestų duomenų filtravimas ir išvalymas;
- Vartotojų identifikavimas;
- Įvykių registravimas;
- Atsakingo asmens informavimas elektroniniu laišku.

Nefunkciniai reikalavimai keliami moduliui:

- Įvesti duomenys turi neprarasti savo prasmės;
- Modulio integravimui į sistemą naudojama grafinė vartotojo sąsaja.

3.3.3. Siūlomas modulis

Siūlomas modulis, kuris realizuos pasiūlytą metodiką, bus integruojamas į taikomosios programos lygmenį. Suprogramuotas SQL injekcijos apsaugos modulis, kuris realizuos pateiktą metodiką, bus pridėtas panaudojus *include* funkciją į kiekvieną programos failą, kurio plėtinys bus *.php* ar *.inc*. Integruota metodika bus kiekvieno failo viršuje. Tokiu būdu ji suaktyvės vos tik sistema kreipsis į šį failą. Visas vartotojo įvestas srautas bus automatiškai patikrinamas, aptikus ataką bus imtasi atitinkamų veiksmų.

Metodika bus realizuota per funkcijas, kurios bus iškviečiamos, kai sistema perims duomenis iš vartotojo.

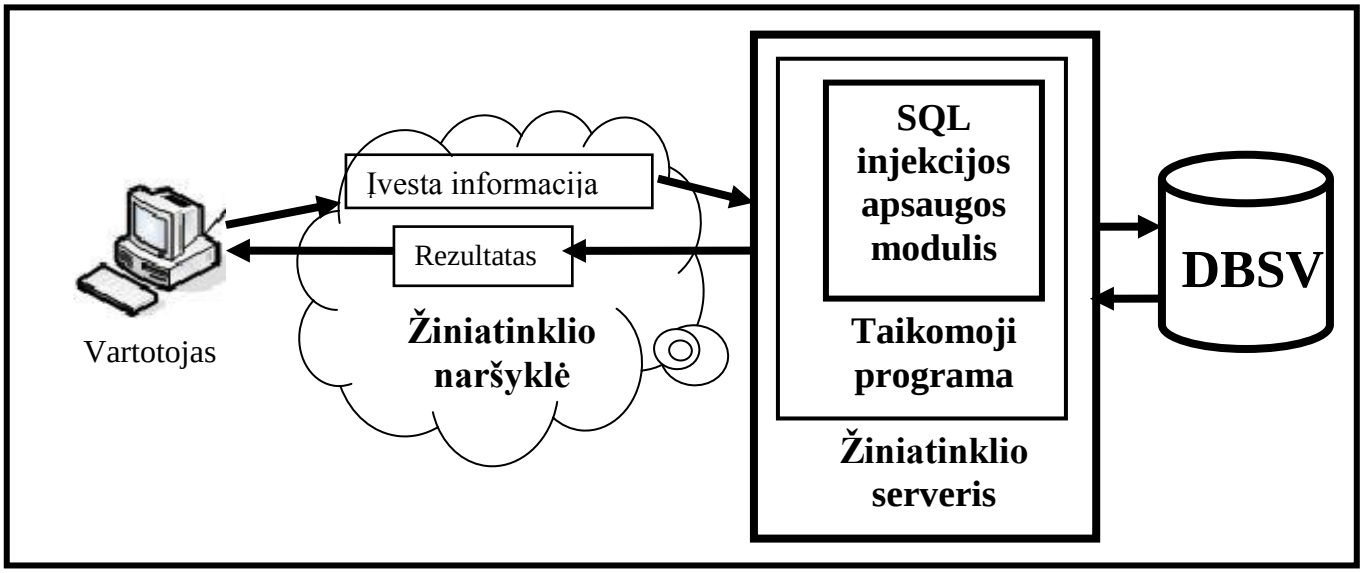
Vartotojo įvestų duomenų srautas bus perimamas iš visų HTTP užklausų GET, POST, ir COOKIE. Pereinant per visus kintamuosius:

```
foreach($_GET as $key => $v)
        $_GET[$key] = Protect($v,$mysql_real,$login_check,$key, "GET");}
```

Šiuo atveju tikrinamos GET užklausos ir, jei juose yra duomenų – įvykdoma saugumo funkcija ir grąžinami jau sutvarkyti duomenys tam pačiam GET kintamajam.

3.3.4. Sistemos architektūra

Šiame modelyje parodyta kur bus projektuojama sistema ir kokie elementai sudaro sistemą (žr. 6 pav.). Vartotojas įveda informaciją į jo kompiuteryje esančią naršyklę. Šie duomenys keliauja į žiniatinklio serverį, kuriame yra įdiegta sistema, o toje sistemoje - įdiegtas SQL injekcijos apsaugos modulis, kuris realizuoja metodiką, apsaugančią nuo SQL injekcijų. Tik tada, kai duomenys patikrinami, jie keliauja į duomenų bazės valdymo sistemą.



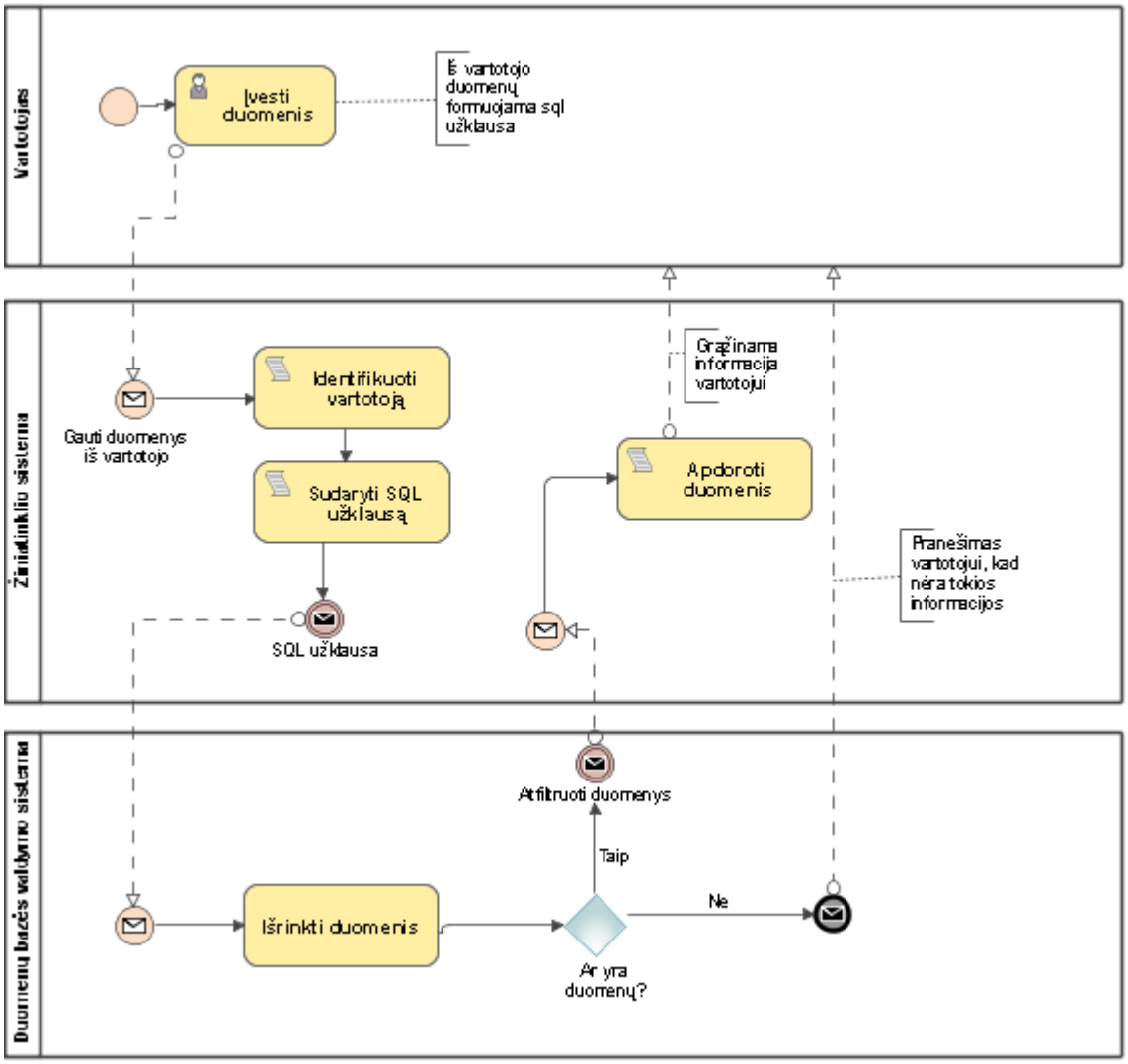
6 pav. Projektuojamo saugos modulio koncepcija

3.3.5. Dabartinis veiklos procesas

Veiklos proceso modelyje (angl., Business Process Diagram - BPD) (žr. 7 pav.) parodoma, kaip formuojamos SQL užklausos, kai sistemoje dar neįdiegtas SQL injekcijos apsaugos modulis. Vartotojas įveda informaciją ir sistema, ją panaudodama sudaro užklausą, kuri keliauja tiesiai į duomenų bazę. Vartotojui grąžinami duomenys, o jei tokių nėra - pranešimas, kad duomenų nerasta.

Vartotojo lygmenyje yra neapibrėžtas pradžios įvykis. Įvedus informaciją į įvedimo lauką, sistema atpažįsta vartotoją ir formuoja SQL užklausą. Suformuota SQL užklausa perduodama SQL serveriui. SQL serveris atrenka duomenis pagal suformuotą užklausą ir grąžina juos vartotojui.

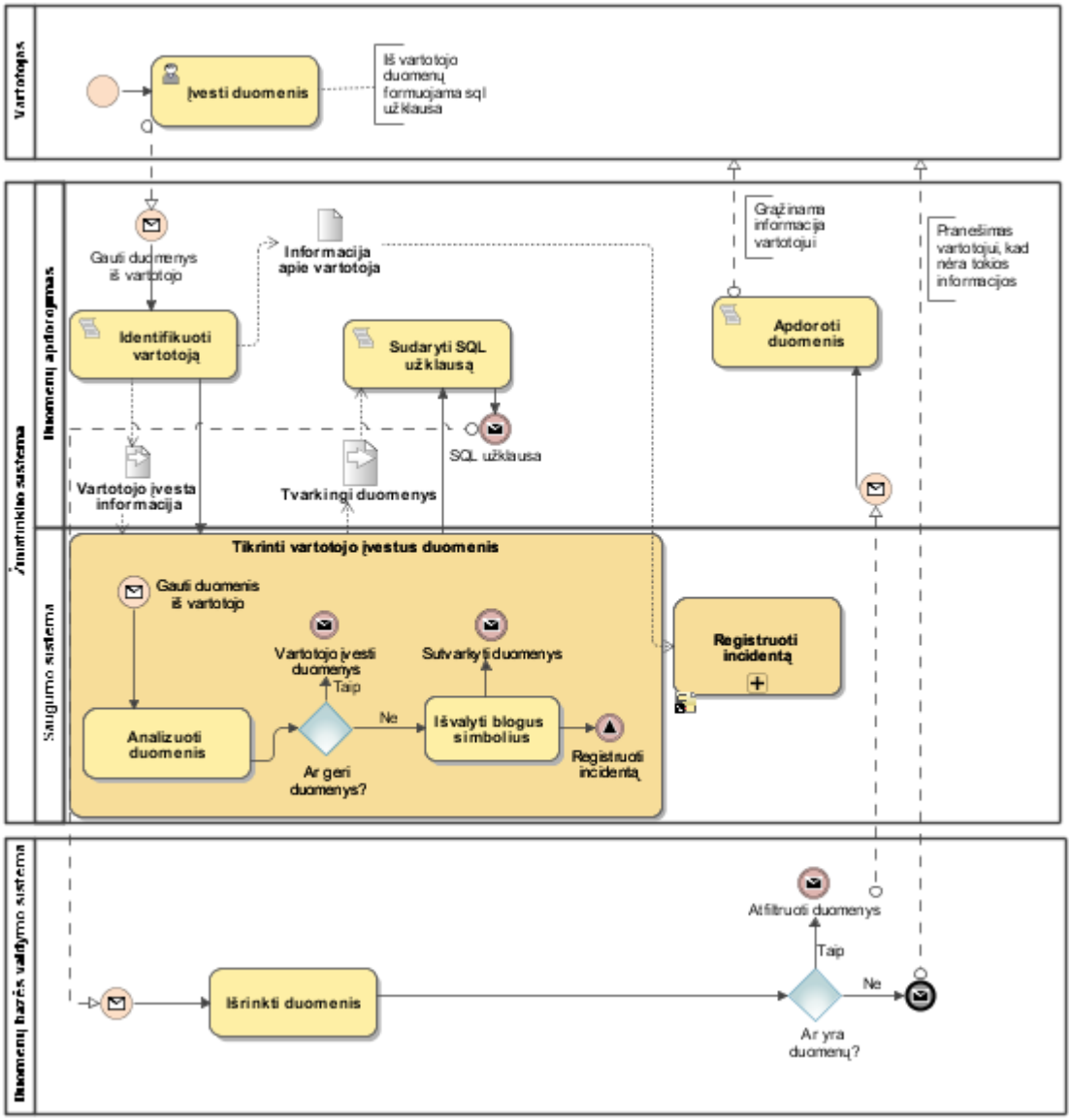
Šiame veiklos procese nėra jokios vartotojo įvestos duomenų kontrolės, tokiu atveju sistema yra pažeidžiama SQL injekcijoms.



7 pav. Veiklos proceso modelis („as-is“) SQL užklausų sudarymas

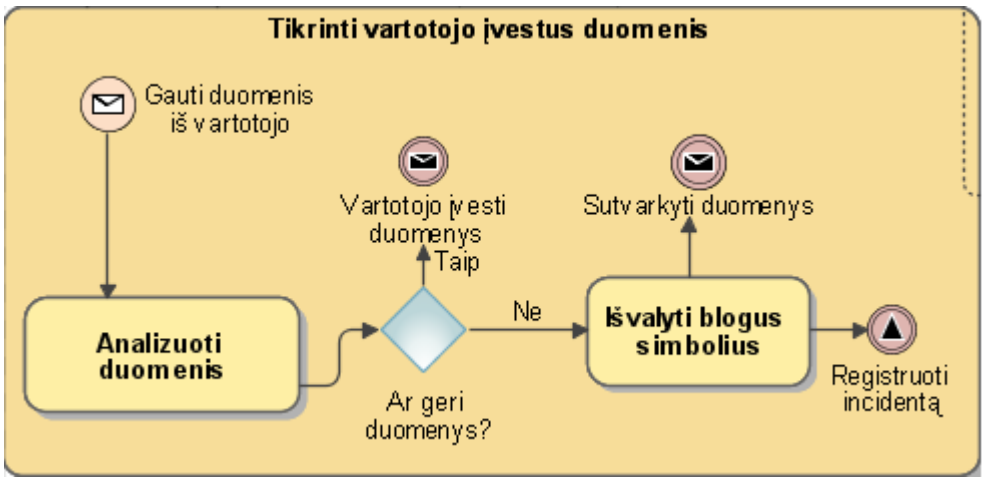
3.3.6. Būsimas veiklos procesas

Integravus apsaugos sistemą į prieš tai buvusią sistemą, visi duomenys yra tikrinami ir sutvarkomi (žr. 8 pav.).



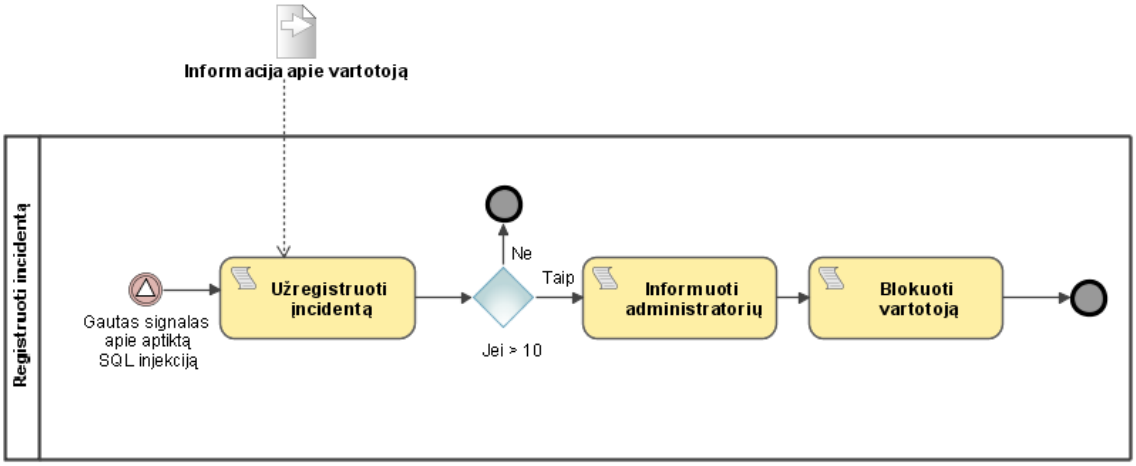
8 pav. Veiklos proceso modelis („to-be“) su įdiegtu apsaugos mechanizmu

SQL injekcijų apsaugos modulyje realizuotas vartotojo įvestų duomenų filtravimas, kuris tikrina, ar vartotojo duomenyse nėra kenksmingų simbolių, ar SQL injekcijos atakoms vartotinų žodžių, tokių kaip: SELECT, UNION, INSERT ir pan. Tai subprocesas (žr. 9 pav.), kurio pradžios įvykis – gauti duomenys iš vartotojo. Šis mechanizmas tikrina, ar vartotojas bando pakenkti sistemai, jei taip, tai sutvarkomi duomenys, kad nepadarytų jai žalos. Toks įvykis užregistruojamas ir perduodamas signalas incidentų registravimo sistemai.



9 pav. Vartotojo duomenų tikrinimas

Įvykių registravimas realizuotas neišskleistu subprocesu (žr. 10 pav.), kurį pradeda gautas signalas. Šis subprocesas pasiima vartotojo duomenis iš vartotojo identifikavimo scenarijaus ir užregistruoja incidentą. Taip pat tikrina ar šis vartotojas nėra potencialus įsilaužėlis, kuris bando surasti sistemoje skylę. Jei toks šio vartotojo veiksmas pasikartoja nebe pirmą kartą, vartotojas įtraukiamas į juoduosius sąrašus.



10 pav. Incidentų registravimo išskleistas subprocesas

Į elektroninio verslo sistemą integravus SQL injekcijos apsaugos modelį, visa vartotojo įvedama informacija būtų tikrinama ir SQL injekcijos atakos rizika būtų minimali. Taip pat kiekvienas pažeidimas būtų užregistruotas ir apie tai informuotas administratorius. Paprastam vartotojui šis saugumo mechanizmas nejaučiamas: jis kaip gaudavo duomenis iš sistemos, taip ir gauna.

3.3.7. Rezultato kokybės kriterijai

SQL injekcijos saugos modulio pagrindiniai teigiami kokybės kriterijai:

- Apdoroja visus klaidų pranešimus;
- Perima visą vartotojo įvedamą informaciją (POST, GET, COOKIES);
- Apsaugo nuo visų žinomų SQL injekcijų atakų tipų.

3.4. Išvados

Šiame skyriuje apibrėžtas darbo tikslas ir iškelti pagrindiniai uždaviniai, taip pat sudarytas taisyklių rinkinys, kuris bus naudojamas didinant saugumą el. verslo žiniatinklio programoms, parašytoms PHP programavimo kalba.

Suprojektuotas saugos modulio projektas, aptarti jam keliami funkciniai ir nefunkciniai reikalavimai.

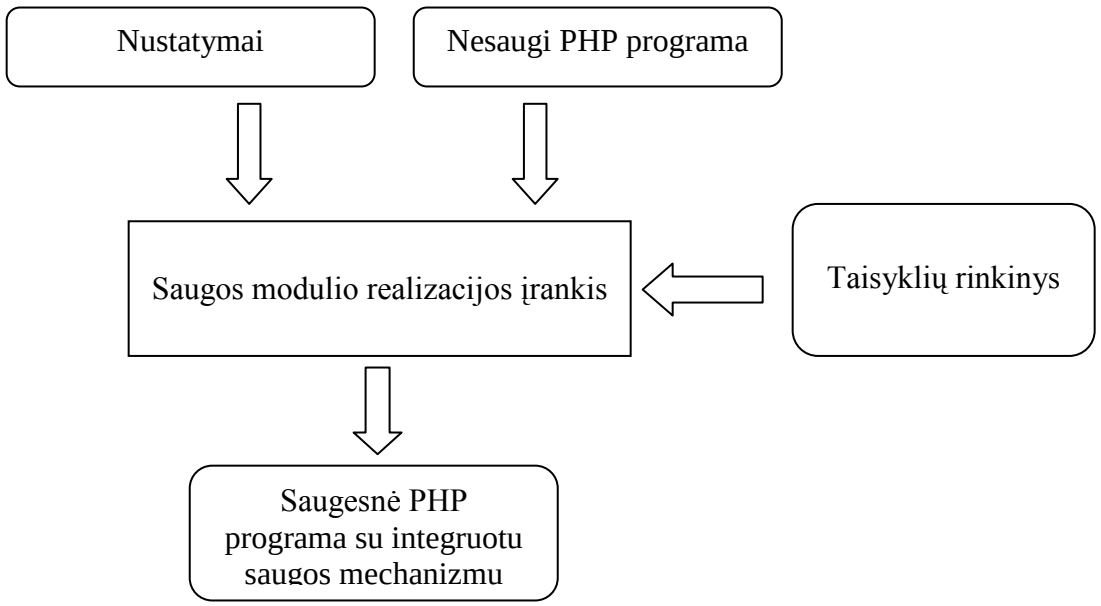
Pateiktas siūlomo modulio veikimo principas ir jo architektūra. Taip pat iškelti saugos modulio kokybės kriterijai.

4. Saugos modulio realizacija

Šioje darbo dalyje pateikiamos realizuojamo saugos modulio detalės. Pateikiama saugos modulio integravimo schema, modulio algoritmas, pagrindiniai nustatymai.

4.1. Saugos modulio integravimas į nesaugią sistemą

Saugumo modulio realizacija (žr. 11 pav.) skirta apsaugoti nesaugias PHP žiniatinklio programas, pritaikius tam tikras saugos taisykles ir funkcijas. Įrankis savyje turi taisyklių ir funkcijų rinkinį, kuris integruojamas į programą pagal vartotojo įvestus nustatymus ir esamą programos saugumo lygį. Paleidus įrankį reikia nustatyti kokių saugumo funkcijų pageidaujama ir nurodyti PHP programos direktoriją. Programa nustato esamą nurodytos programos saugumo lygį ir pritaiko reikiamas taisykles.



11 pav. Integracijos modelis

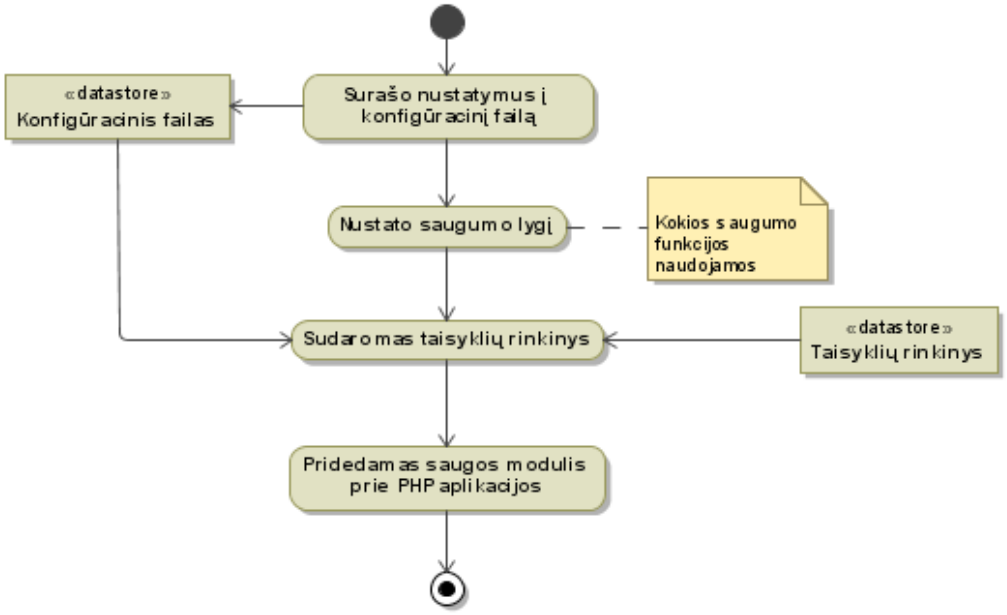
4.2. Saugos modulio realizacijos įrankio algoritmas

Svetainės administratoriui nurodžius nustatymų parametrus ir programos katalogą, saugos modulio realizacijos įrankis atlieka šiuos veiksmus:

- Surašo nustatymus į konfigūracinį failą (config.cfg);
- Nustato esamos programos saugumo lygį, nuskaito visus projekto failus ir tikrina, kokios saugumo funkcijos ir komandos yra naudojamos;

- Pagal nustatytą saugumo lygį ir konfigūracinį failą sudaro taisyklių rinkinį;
- Prideda šį taisyklių failą į PHP programą.

Integravus saugos modulį, kuris perima visus vartotojo įvestus duomenis ir pritaiko parinktas taisykles, PHP programa tampa žymiai saugesnė.



12 pav. Realizacijos įrankio algoritmas

4.3. Saugumo lygio nustatymas

Vienas iš pagrindinių kokybės reikalavimų yra vartotojų įvedamų duomenų nesugadinimas, todėl prieš apdorojant vartotojo įvestą informaciją reikia nustatyti kokias apsaugos funkcijas naudoja sistema, kad jos nesidubliuotų.

Saugos modulio realizacijos įrankis nuskaito visus PHP programos failus ir ieško juose saugumą realizuojančių funkcijų, pvz.: *trim() mysql_real_escape_string().*

Gautus rezultatus išsaugo konfigūraciniame faile.

4.4. Taisyklių rinkinio sudarymas

Šis įrankis turi savyje bendrą taisyklių rinkinį, tačiau konkrečiai verslo sistemai taisyklių rinkinys vykdomas individualus, pagal programos saugumo lygį ir įvestus nustatymus.

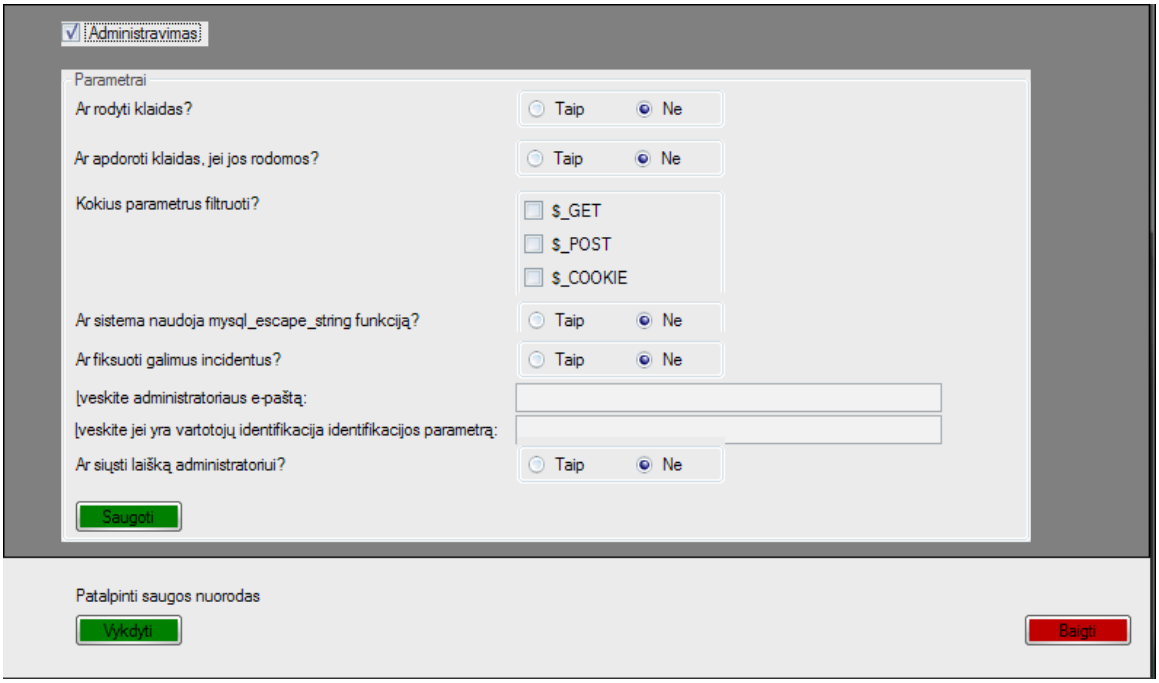
Taisyklių rinkinį sudaro įvairios funkcijos, kurios realizuoja 3.2. skyrelyje aptartą saugumo metodiką. Šios funkcijos iškviečiamos prieš tai nuskaičius konfigūracinį failą.

Konfigūraciniame faile esantys nustatymai perduodami pagrindinei procedūrai, kuri iškviečiama vykdant operaciją sistemoje, reikalaujančioje vartotojo įvedamų duomenų. Pagrindinė procedūra nuskaito parametrus ir iškviečia reikiamas funkcijas.

13 pav. Taisyklių sudarymo algoritmas

4.5. Įrankio nustatymai

Kiekvienas projektas yra unikalus, todėl norint, kad saugos mechanizmas pilnai funkcionuotų, reikia nustatyti tam tikrus parametrus. Nustatymai yra unikalūs kiekvienai sistemai pagal saugumo poreikį ir kintamųjų reikšmes.



14 pav. Įrankio konfigūravimo grafinė sąsaja

4.5.1. Globalių kintamųjų nustatymas

Norint apsaugoti pasirinktą el. verslo žiniatinklio programą, reikia išsiaiškinti, kokius globalius kintamuosius ji naudoja. Kintamieji, kuriuos gali keisti vartotojas, yra šie: $_POST, $_GET, $_COOKIES.

Globalių kintamųjų nustatymas reikalingas tam, kad būtų galima nustatyti, kokius kintamuosius ($_POST, $_GET, $_COOKIES) tikrins ši sistema. Saugos modulis perims pasirinktų globalių kintamųjų duomenis ir jiems pritaikys saugos funkcijas.

4.5.2. Klaidų apdorojimo parametrai

Išvedamos į ekraną klaidos labai padeda piktadariui įsilaužti į sistemą. Dėka į ekraną išvedamų klaidų, įsilaužėlis gali sužinoti MySQL versiją, skriptų pavadinimus ar net kintamųjų vardus. Todėl rekomenduojama tvarkingai apdoroti klaidas visoms sistemoms.

Yra įvairių sistemų, kurioms reikia nustatyti skirtingus parametrus ar išvesti skirtingus pranešimus apie klaidas. Vienoms nereikia rodyti klaidų pranešimų apie įvairius perspėjimus ar kritinių klaidų įvykius, kitoms - reikia rodyti viską, bet tvarkingai apdorojus išvedamus klaidų pranešimus.

Šio saugos modulio pagalba galima tvarkingai apdoroti klaidas, reikia tik nustatyti tam tikrus parametrus:

- Apdoroti ar neapdoroti klaidas;
- Rodyti ar nerodyti klaidų pranešimus.

4.5.3. Incidentų fiksavimo parametrai

Incidentų registravimas padeda administratoriui fiksuoti galimas pažeidžiamas vietas ir potencialius įsilaužėlius į sistemą.

Jei sistemoje saugoma informacija yra ypač konfidenciali ir visi vartotojai yra autentifikuojami, tai reikalingas labai didelis saugumo lygis su vartotojų kontrole.

Potencialiai blogų įvykių fiksavimas vyksta tikrinant vartotojo įvestus duomenis pagal saugumo modelyje sudarytus SQL injekcijų atakų šablonus.

Incidentų fiksavimo pagrindiniai parametrai:

- Tikrinti ar netikrinti vartotojų įvedamus duomenis pagal šablonus;
- Siųsti ar nesiųsti incidentų žurnalus administratoriui (jei siųsti, tai reikia nurodyti „Administratoriaus elektroninis paštas“);
- Jei programoje identifikuojami vartotojai – įvesti identifikavimo kintamąjį pvz.: *$_SESSION[“username”]*.

4.6. Realizacijos apibendrinimas

Saugos modulio integravimui į el. verslo žiniatinklio programą naudojama grafinė sąsaja, kurioje parenkami tinkami nustatymai ir nurodomas programos šakninis aplankas. Nurodžius programos aplanką šis saugos modulis tikrina programos failus ir nustato esamą saugumo lygį.

Pagal nustatytus nustatymus ir gautą saugumo lygį sudaromas taisyklių rinkinys, kuris integruojamas į el. verslo žiniatinklio programą.

Saugos modulis perima visą vartotojų įvedamą informaciją ir pritaiko jai sudarytą taisyklių rinkinį.

5. Saugos modulio realizacijos tyrimas

Šioje darbo dalyje atliekamas realizuoto saugos modulio apsaugai nuo SQL injekcijų tyrimas. Patikrinta, ar modulis atitinka ankstesniuose skyriuose iškeltus teigiamus kokybės kriterijus. Testavimas atliekamas elektroninėje verslo programoje sukurtoje „PHP“ programavimo kalba. Ši programa naudoja „MySQL“ duomenų bazę.

5.1. Klaidų apdorojimo testas

Šis saugos modulis turi du pasirinkimus apdorojant klaidų pranešimus: blokuoti visus išvedamus perspėjimų ir klaidų pranešimus arba apdoroti visus išvedamus klaidų pranešimus.

Norint patikrinti klaidų apdorojimą reikia testuojamoje programoje sudaryti sąlygas įvykti klaidai. Pvz.: nurodžius netinkamą parametrą scenarijuje (straipsniai.php), kuris rodo straipsnį, įvyksta duomenų bazės klaida ar programuotojo netyčia palikta klaida.

3. lentelė Klaidų pranešimai vartotojui

Klaidos tipas	Neapdorotas klaidos pranešimas	Apdorotas klaidos pranešimas	Klaidos išjungtos
Duomenų bazės klaida	`Warning: mysql_fetch_assoc() expects parameter 1 to be resource, boolean given in ..\straipsniai.php on line 119`	`Klaida [2] mysql_fetch_assoc() expects parameter 1 to be resource, boolean given`	Į ekraną neišveda jokios informacijos
Programuotojo palikta klaida	`Notice: Undefined index: id in ..\straipsniai.php on line 107`	`Klaida [8] Undefined index: id`	Į ekraną neišveda jokios informacijos
Nežinoma klaida	`...`	`Nežinoma klaida [klaidos nr] Klaidos pranešimas`	Į ekraną neišveda jokios informacijos

Neapdorotos klaidos pranešimai (žr. 3. lentelė) atskleidžia daug naudingos informacijos atakuotojui, o apdorojus klaidos pranešimus atakuotojas gauna minimalų žinių apie sistemą kiekį. Tačiau saugiausia išvis uždrausti klaidų išvestį vartotojams.

5.2. Incidentų registravimo testas

Incidentų registravimo testavimui naudojama:

- Iš anksto paruoštas failas su SQL injekcijoms būdingais atakų šablonais;
- Failas su galimais vartotojo įvedamais duomenimis, kurie panašūs į SQL injekcijų atakų šablonus, tačiau visiškai nepavojingi sistemai.

Testavimo eiga:

- Parengiamas SQL injekcijų šablonų failas;
- Parengiamas vartotojo įvedamų duomenų failas;
- Šių failų duomenys patikrinami su incidentų fiksavimo įrankiu. Jei duomenys atitinka nustatyta taisyklių kiekį, incidentas užregistruojamas;

Atakų šablono failas paruoštas pagal atliktos analizės duomenis, o vartotojo duomenų failas parengiamas panaudojant SQL atakoms būdingus simbolius ar žodžių frazes. Pvz.: O'Neal, “Nepriklausomybės diena”, Mike and Jane, select this item, 1+1=2 ir pan.

5.2.1. Incidentų registravimo testavimo vykdymas

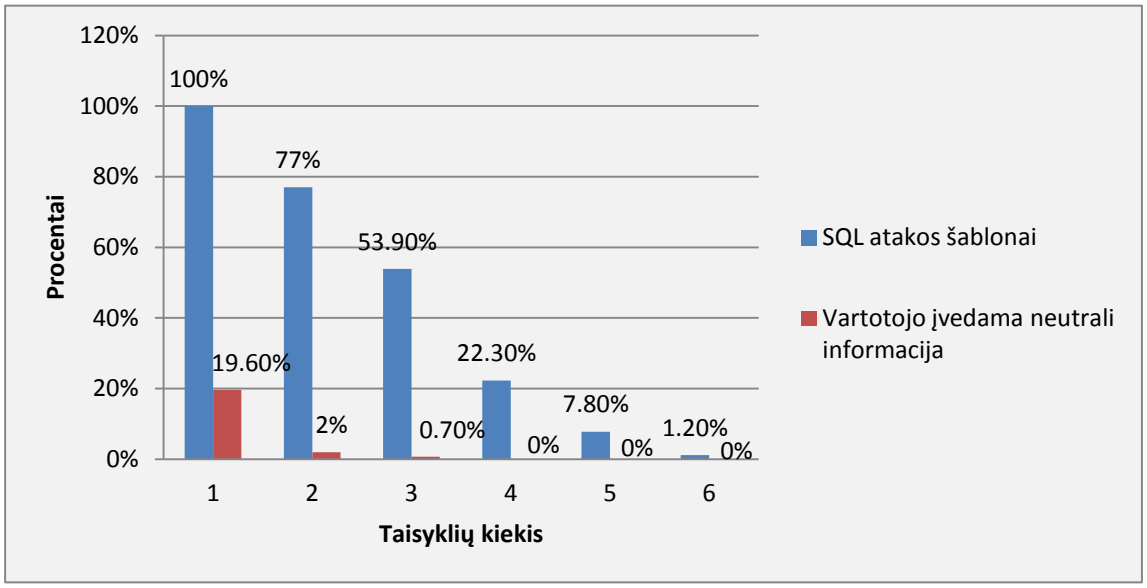
Vykdomas ciklas, kurio metu imama eilutė iš SQL injekcijų atakų šablonų failo ir lyginama su aprašytomis taisyklėmis. Radus tam tikrą atitikimų kiekį, incidentas užregistruojamas. Šiam failui pasibaigus ciklas kartojamas su vartotojų įvedamų duomenų failu nepakeitus taisyklių atitikimų kiekio.

Patikrinus visus duomenis didinamas taisyklių atitikimų kiekis, bandant surasti optimalų atitikimų kiekį.

5.2.2. Incidentų registravimo testo rezultatai

4. lentelė Įrašų registravimo rezultatai

Kiek taisyklių atitinka	Atakų šablonų failas	Vartotojo įvedamų duomenų failas
1	Iš 243 įrašų atitiko 243 įrašai, t.y. 100%	Iš 143 įrašų atitiko 28 įrašai, t.y. 19.6%
2	Iš 243 įrašų atitiko 187 įrašai, t.y. 77%	Iš 143 įrašų atitiko 3 įrašai, t.y. 2%
3	Iš 243 įrašų atitiko 131 įrašai, t.y. 53.9%	Iš 143 įrašų atitiko 1 įrašai, t.y. 0.7%
4	Iš 243 įrašų atitiko 59 įrašai, t.y. 22.3%	Iš 143 įrašų atitiko 1 įrašai, t.y. 0%
5	Iš 243 įrašų atitiko 19 įrašai, t.y. 7.8%	Iš 143 įrašų atitiko 1 įrašai, t.y. 0%
6	Iš 243 įrašų atitiko 3 įrašai, t.y. 1.2%	Iš 143 įrašų atitiko 1 įrašai, t.y. 0%



15 pav. Įrašų registravimo rezultatų diagrama procentais

Kaip matyti iš pateiktų rezultatų, norint užfiksuoti visus galimus pažeidimus reikia nustatyti, kad saugumo modulis registruotų incidentus, kurie atitinka bent vieną aprašytą taisyklę, tačiau tokiu atveju dalis vartotojų įvedamos informacijos, tokios kaip: O`Neal, “Pavadinimas” ir t.t., taip pat patektų į užfiksuotų incidentų žurnalą.

Norint to išvengti reiktų nustatyti didesnį taisyklių atitikimo skaičių. Tačiau dalis potencialių atakų gali būti, kad nebus užfiksuotos.

Sistemos administratorius turėtų įvertinti kokio apsaugos lygio reikia administruojamai elektroninio verslo programai.

5.3. Apsaugos nuo SQL injekcijų testas

Apsaugos nuo SQL injekcijų testavimui naudojama:

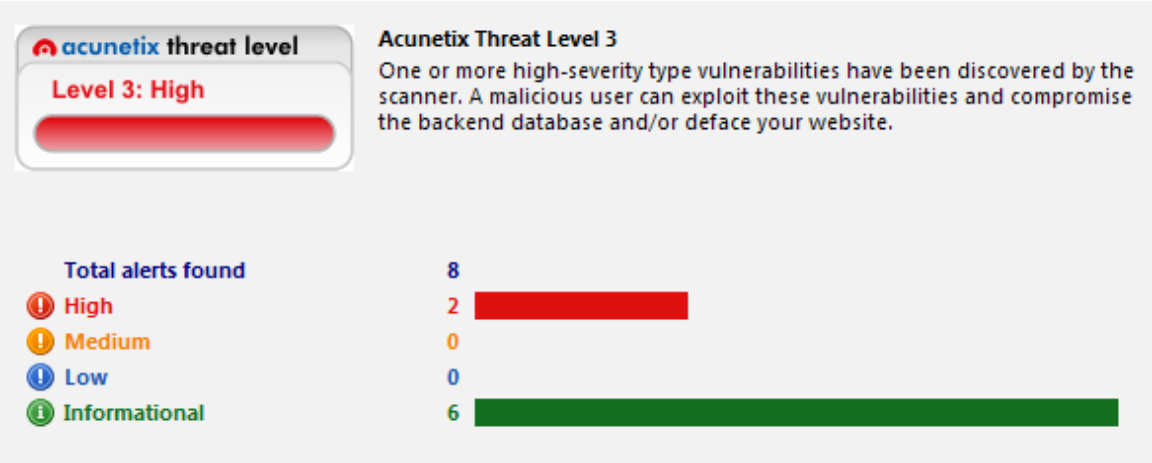
- „Acunetix Web Vulnerability Scanner“ pažeidžiamumų skenavimo programa;
- Testavimui skirta žiniatinklio programa.

Testavimo eiga:

- Su pažeidžiamumų skenavimo įrankiu patikrinima nesaugi žiniatinklio programa;
- Sukonfigūruojamas saugos įrankis:
  - Tikrinti visus globalius kintamuosius;
  - Pritaikyti apsauga nuo SQL injekcijų;
- Saugos modulis integruojamas į nesaugią žiniatinklio programą;
- Pakartojamas skenavimas.

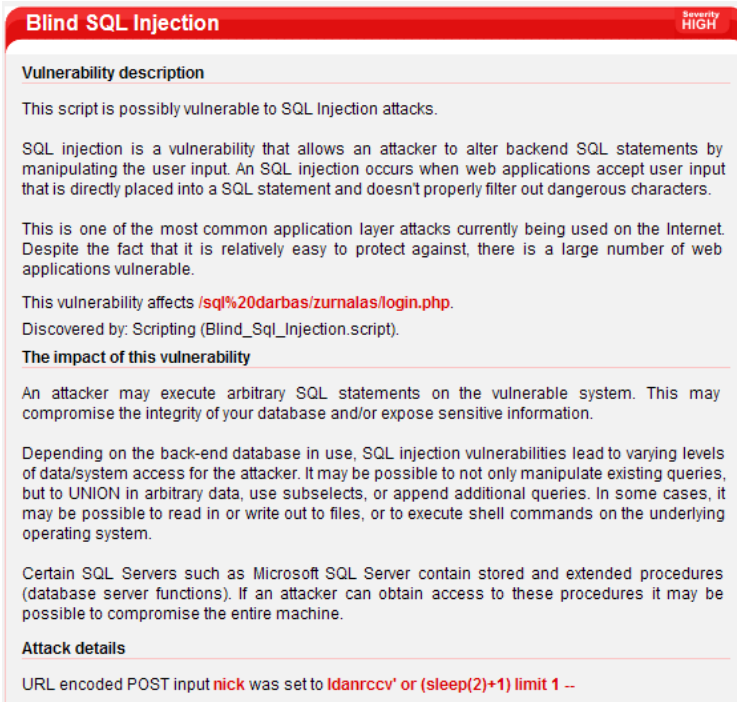
5.3.1. Nesaugios sistemos skenavimo rezultatai

Skenavimo įrankis nustatomas taip, kad skenuotų tik SQL injekcijų pažeidžiamumus.

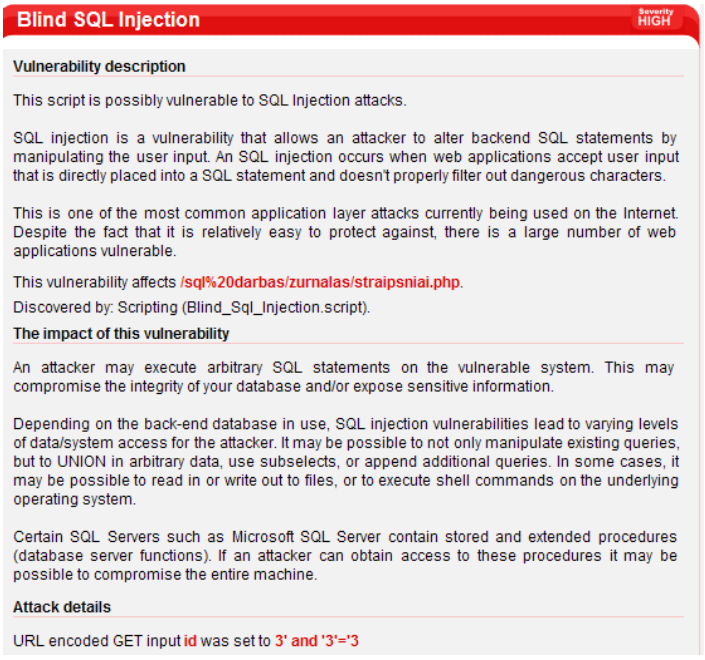


16 pav. Pažeidžiamumų informacija nenaudojant jokios apsaugos

Patikrinus nesaugią programą (žr. 16 pav.) randami du pažeidžiamumai. Detaliau nagrinėjant rastus pažeidžiamumus matoma, kad pirmas pažeidžiamumas (žr. 17 pav.) yra scenarijuje login.php, pažeidžiamas „nick“ parametras, kuris perduodamas POST metodu. Antrasis pažeidžiamumas (žr. 18 pav.) rastas scenarijuje straipsniai.php, pažeidžiamas „id“ parametras, kuris perduodamas GET metodu.



17 pav. Pirmas pažeidžiamumas login.php scenrijuje

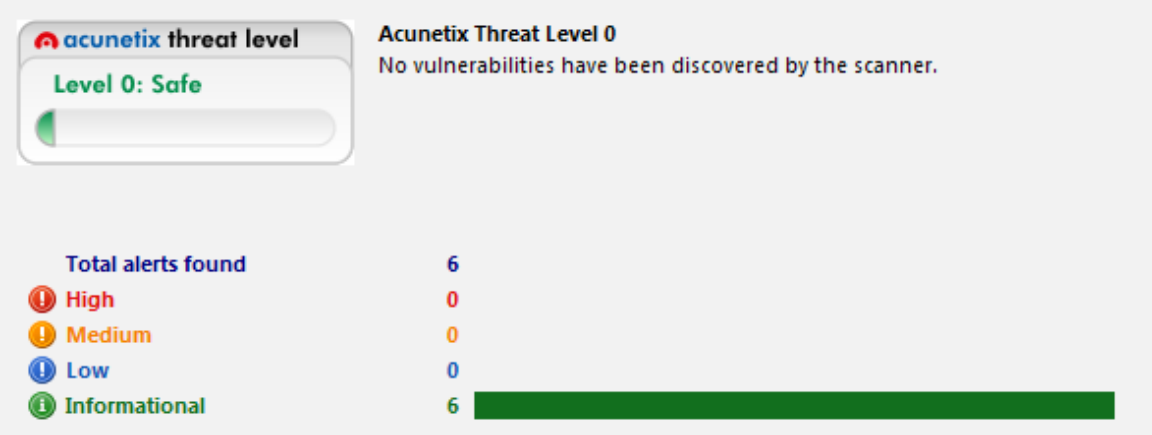


18 pav. Antras pažeidžiamumas straipsniai.php scenarijuje

5.3.1. Saugios sistemos skenavimo rezultatai

Į pažeidžiamą žiniatinklio programą buvo integruotas saugos modulis. Kuris apsaugo nuo SQL injekcijų ir užregistruoja visus bandymus pažeisti sistemą.

Skenavimo įrankis nustatomas taip, kad skenuotų tik SQL injekcijų pažeidžiamumus.



19 pav. Pažeidžiamumų informacija naudojant saugos modulį

Atlikus apsaugotos programos skenavimą (žr. 19 pav.) ir išanalizavus prieduose (žr. Priedai) pateiktą užregistruotų incidentų failą, galima teigti, kad saugos modulis puikiai apsaugo pažeidžiamą programą ir užregistruoja bandymus įsilaužti į sistemą.

5.4. Tyrimo išvados

Norint užtikrinti patį didžiausią sistemos saugumą rekomenduojama išjungti visus vartotojams išvedamus klaidų pranešimus.

Incidentų registravimas užfiksavo visą potencialiai pavojingą vartotojo įvestą informaciją, kuri atitiko bent vieną aprašytą taisyklę, tačiau taip pat užregistravo 19.60% klaidingos informacijos.

Norint pasiekti optimalų incidentų registravimą, rekomenduojama nustatyti, kad įrankis aptiktų incidentus, kurie atitinka bent dvi aprašytas taisykles. Tokiu atveju įrankis aptiks tik 2% klaidingos informacijos ir 77% bandymus pakenkti sistemai.

Atsparumo SQL injekcijoms testas parodė, kad saugos modulis apsaugo visas rastas pažeidžiamas vietas.

6. Išvados

Analizės metu paaiškėjo SQL injekcijoms naudojami simboliai ir kodo fragmentai. Taip pat buvo nustatyti parametrai, kuriuos reikia tikrinti nuo galimai pavojingų atakuotojo įvedamų duomenų.

Esamų apsaugos priemonių analizė parodė, kad didžioji dalis apsaugos priemonių apsaugančių nuo SQL injekcijų yra diegiamos į serverį arba panaudojamos kuriant naują programinę įrangą.

Sudaryta metodika, skirta apsaugoti el. verslo žiniatinklio programas nuo SQL injekcijų atakų. Ji naudojama realizuojant saugos modulį.

Pasiūlytas saugos modulis, kuris diegiamas į jau esamą elektroninio verslo žiniatinklio programą. Ši programa turi būti parašyta PHP programavimo kalba, duomenys gaunami iš HTTP užklausų GET, POST, COOKIE turi būti nešifruoti, programa naudoja reliacinę duomenų bazę. Įdiegtas saugos modulis perima visus vartotojų įvestus duomenis ir pritaiko atitinkamas saugos funkcijas.

Realizuota programinė priemonė apsaugo nuo visų žinomų SQL injekcijų atakų tipų ir registruoja potencialius bandymus sutrikdyti normalų elektroninio verslo svetainės darbą ar išgauti konfidencialią informaciją.

Atliktas eksperimentas parodė, kad šis saugos modulis apsaugo nuo visų bandytų SQL injekcijų ir nustačius, kad registruotų incidentus, kurie atitinka bent vieną aprašytą taisyklę, užregistravo 100% bandymų įsilaužti į sistemą.

Naudota literatūra

1. OWASP – Open Web Application Security Project [interaktyvus]. [Žiūrėta 2012-05-18], Prieiga per internetą: http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project;
2. National Vulnerability Database [interaktyvus]. [Žiūrėta 2012-05-18], Prieiga per internetą: http://nvd.nist.gov
3. Andrews C., Litchfield D., B. Grindlay. SQL Server security. The McGraw-Hill Companies, 2003
4. Boyd S. W., Keromytis A. D. : Preventing SQL Injection Attacks. Lecture Notes in Computer Science, 2004, Volume 3089/2004, 292-302, DOI:10.1007/978-3-540-24852-1_21
5. Clarke J.. SQL Injection attacks and defense. Elsvier, Inc. 2009.
6. Cumming A.,. Russell G. SQL HACKS Tips & Tools for Digging into Your Data. O'Reilly Media, 2007
7. Dysart F., Sherriff M.. Automated Fix Generator for SQL Injection Attacks. Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4700351 [2012-05-16]
8. Merlo E., Letarte D., Antoniol G.. Automated Protection of PHP Applications. Against SQL-injection Attacks. Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4145037 [2012-05-16]
9. Merlo E., Letarte D., Antoniol G.. SQL-Injection Security Evolution Analysis in PHP. Web Site Evolution, 2007. WSE 2007. 9th IEEE International Workshop on. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4380243 [2012-05-17]
10. Nystrom M.G.. SQL Injection Defenses. O'Reilly Media, 2007.
11. Poel N. L. Automated Security Review of PHP Web Applications with Static Code Analysis. May 28, 2010. http://scripties.fwn.eldoc.ub.rug.nl/FILES/scripties/Informatica/Master/2010/Poel.N.L.de./INF-MA-2010-N.L._de_Poel.pdf [2012-05-19]
12. Ruse M., Sarkar T., Basu S.. Analysis & Detection of SQL Injection

Vulnerabilities via Automatic Test Case Generation of Programs. 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet. http://www.computer.org/portal/web/csdl/doi/10.1109/SAINT.2010.60 [2012-05-16]

13. Vernesson S.. Penetration Testing in a Web Application Environment. 2010-10-12 Subject: Computer Science. http://lnu.diva-portal.org/smash/get/diva2:356502/FULLTEXT01 [2012-05-17]

14. Wei K., . Muthuprasanna M, Suraj K.. Preventing SQL injection attacks in stored procedures. Software Engineering Conference, 2006. Australian http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1615052&tag=1 [2012-05-18]

15. Website Security - Acunetix Web Security Scanner [interaktyvus]. [Žiūrėta 2012-05-16], Prieiga per internetą: http://www.acunetix.com/;

16. Rational AppScan [interaktyvus]. [Žiūrėta 2012-05-16], Prieiga per internetą: http://www-01.ibm.com/software/awdtools/appscan/

17. Havij v1.14 Advanced SQL Injection [interaktyvus]. [Žiūrėta 2012-05-16], Prieiga per internetą: http://www.itsecteam.com/en/projects/project1.htm

18. Scambray J., Shema M., Sima C. Hacking Exposed Web Applications (2nd Edition). 06/2006

19. Stuttard D., Pinto M. Web Application Hackers Handbook : Discovering and Exploiting Security Flaws. 01/2008

20. Huseby Sverre H. Innocent Code : A Security Wake-up Call for Web Programmers. 12/2004

21. Cannings R., Dwivedi H., Lackey Z. Hacking Exposed Web 2. 0 : Web 2. 0 Security Secrets and Solutions. 12/2007

22. Boneh D. Hacking SQL injection: attacks and. 2009 [interaktyvus] [Žiūrėta 2012-05-18], Prieiga per internetą: http://crypto.stanford.edu/cs142/lectures/16-sql-inj.pdf

23. Dagienė V., Grigas G., Jevsikova T. Enciklopedinis kompiuterijos žodynas (II papildytas leidimas). 2009-01-01 [interaktyvus] [Žiūrėta 2012-05-16], Prieiga per internetą: http://aldona.mii.lt/pms/terminai/term/enciklo.html

24. SQL injekcijų požymiai. [interaktyvus] [Žiūrėta 2012-05-10]. Prieiga per Internetą: http://yehg.net/lab/pr0js/pentest/wordlists/injections/SQL.txt

25. PHP programavimo kalbos funkcijos. [interaktyvus] [Žiūrėta 2012-05-10]. Prieiga per Internetą: http://lt.php.net/manual/en/function.str-replace.php

Priedai

1 priedas. Įvykių registravimo failas skenuojant su Acunetix įrankiu

[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp' and sleep(4)='
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp" and sleep(4)="
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: '"'");|]*{%0d%0a<%00>
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: ;; print; Klientas: 127.0.0.1 Įvesta reikšmė: print(md5(acunetix_wvs_security_test));die();/*
[2012-05-04 15:20:33] Metodas: POST; Parametras: password; Pavartoti žodžiai: 0X; Klientas: 127.0.0.1 Įvesta reikšmė: 0XhRi10w
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: #; Klientas: 127.0.0.1 Įvesta reikšmė: ^(#$!@#$)(()))******
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; --; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp' or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:33] Metodas: POST; Parametras: nick; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: --; "; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp" or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}\
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: ;; print; Klientas: 127.0.0.1 Įvesta reikšmė: print(md5(acunetix_wvs_security_test));die();/*
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp'=sleep(4)='
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; Klientas: 127.0.0.1 Įvesta reikšmė: 1'
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: #; Klientas: 127.0.0.1 Įvesta reikšmė: ^(#$!@#$)(()))******
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp"=sleep(4)="
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: '"'");|]*{%0d%0a<%00>
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}\
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; --; ;; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp'; waitfor delay '0:0:4' --
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; --; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: flsjbmwp"; waitfor delay '0:0:4' --
[2012-05-04 15:20:34] Metodas: POST; Parametras: submit; Pavartoti žodžiai: ;; print; Klientas: 127.0.0.1 Įvesta reikšmė: print(md5(acunetix_wvs_security_test));die();/*
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: submit; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: "; Klientas: 127.0.0.1 Įvesta reikšmė: 1"
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: '"
[2012-05-04 15:20:34] Metodas: POST; Parametras: submit; Pavartoti žodžiai: #; Klientas: 127.0.0.1 Įvesta reikšmė: ^(#$!@#$)((()))******
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: submit; Pavartoti žodžiai: print; Klientas: 127.0.0.1 Įvesta reikšmė: ${@print(md5(acunetix_wvs_security_test))}\
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: --; Klientas: 127.0.0.1 Įvesta reikšmė: <!--
[2012-05-04 15:20:34] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: '"'");|]*{%0d%0a<%00>
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD' and sleep(4)='
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: '"
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:34] Metodas: POST; Parametras: nick; Pavartoti žodžiai: =; Klientas: 127.0.0.1 Įvesta reikšmė: JyI=
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD" and sleep(4)="
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: --; Klientas: 127.0.0.1 Įvesta reikšmė: <!--
[2012-05-04 15:20:35] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; --; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD' or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:35] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: □□
[2012-05-04 15:20:35] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:35] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: '"

[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: --; "; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD" or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:35] Metodas: POST; Parametras: nick; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: ▪""
[2012-05-04 15:20:35] Metodas: POST; Parametras: submit; Pavartoti žodžiai: --; Klientas: 127.0.0.1 Įvesta reikšmė: <!--
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD'=sleep(4)='
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD"=sleep(4)="
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; Klientas: 127.0.0.1 Įvesta reikšmė: 1'
[2012-05-04 15:20:35] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; --; ;; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD'; waitfor delay '0:0:4' --
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: "; Klientas: 127.0.0.1 Įvesta reikšmė: 1"
[2012-05-04 15:20:35] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; --; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: g00dPa$$w0rD"; waitfor delay '0:0:4' --
[2012-05-04 15:20:36] Metodas: POST; Parametras: password; Pavartoti žodžiai: user; Klientas: 127.0.0.1 Įvesta reikšmė: user
[2012-05-04 15:20:36] Metodas: POST; Parametras: password; Pavartoti žodžiai: =; Klientas: 127.0.0.1 Įvesta reikšmė: JyI=
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: 0x; Klientas: 127.0.0.1 Įvesta reikšmė: lKiYk0xJ
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti' and sleep(4)='
[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:36] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: □□
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti" and sleep(4)="
[2012-05-04 15:20:36] Metodas: POST; Parametras: password; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: ▪""
[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; --; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti' or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: --; "; or; limit; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti" or (sleep(4)+1) limit 1 --
[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; Klientas: 127.0.0.1 Įvesta reikšmė: 1'
[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: =; '; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti'=sleep(4)='

[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:36] Metodas: COOKIE; Parametras: memberpw; Pavartoti žodžiai: =; '; or; '); Klientas: 127.0.0.1 Įvesta reikšmė: ')or isnull(1/0) or(1='

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: =; "; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti"=sleep(4)="

[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: "; Klientas: 127.0.0.1 Įvesta reikšmė: 1"

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; --; ;; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti'; waitfor delay '0:0:4' --

[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: =; Klientas: 127.0.0.1 Įvesta reikšmė: JyI=

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; --; "; ;; Klientas: 127.0.0.1 Įvesta reikšmė: Prisijungti"; waitfor delay '0:0:4' --

[2012-05-04 15:20:36] Metodas: POST; Parametras: nick; Pavartoti žodžiai: admin; Klientas: 127.0.0.1 Įvesta reikšmė: admin

[2012-05-04 15:20:36] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: □□

[2012-05-04 15:20:37] Metodas: POST; Parametras: submit; Pavartoti žodžiai: '; "; Klientas: 127.0.0.1 Įvesta reikšmė: ▪""

2 priedas. Darbo rezultatų panaudojimo aktas

Akte nurodoma, kad sukurtas saugos modulis įdiegtas plėtojant „Mokslininkų ir tyrėjų gebėjimų išnaudoti gigabitinio tinklo technologijas ugdymas“ projektą, vykdomą Kauno technologijos universiteto.

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMACINIŲ TECHNOLOGIJŲ PLĖTROS INSTITUTAS
ITPI
Kodas 1119 50581 • Studentų g. 48A, LT-51367 Kaunas
Rašt. tel. (8-37) 30 06 00, faksas (8-37) 33 19 25 • http://www.itpi.ktu.lt • El. paštas itpi@ktu.lt

KTU Informatikos fakulteto
Kompiuterių katedrai

2012-05-21 Nr.DV17-F-20-150

DĖL SAUGOS MODULIO NAUDOJIMO

Aido Ramoškos magistrinio darbo „Apsaugos nuo SQL injekcijų el.verslo svetainėse metodikos sudarymas ir tyrimas“ rezultatai įdiegti plėtojant „Mokslininkų ir tyrėjų gebėjimų išnaudoti gigabitinio tinklo technologijas ugdymas“ projektą, vykdomą Kauno technologijos universiteto, finansuojamą ES struktūrinių fondų lėšomis pagal priemonę VP1-3.1-ŠMM-02-V.

Direktorius Rimantas Šeinauskas

D. Tribandis, (8 37) 32 76 32