LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Tīmekļa lietojumprogrammu drošība

BAKALAURA DARBS

Autore: **Stella Tīda**
Studenta apliecības Nr.: st12048
Darba vadītājs: prof., Dr.dat. Māris Vītiņš

RĪGA 2016

Anotācija

Darbs “Tīmekļa lietojumprogrammi drošība” ir pētījums par tīmekļa lietojumprogrammu drošības apdraudējumiem, uzbrukumu iespējām un drošības risinājumiem.

Darbu veido 54 lappuse. Izmantotās literatūras sarakstu veido 21 avots.

Darba mērķis bija izpētīt, kā interneta vidē ir iespējams izdarīt noziedzīgas lietas, kas ir apdraudējumu cēloņi, un meklēt risinājumus, kā var izvairīties no uzbrukumiem. Darba gaitā tika pievērsta uzmanība tādām tīmekļa lietotņu zināmām vājām vietām, kā klienta puses sniegtie dati, pilnvarošanas mehānisms un citi. Tika parādīts, ka drošības pasākumu neievērošana var krietni apdraudēt informācijas konfidencialitātes nodrošināšanai un sistēmas funkcionēšanai.

Rezultātā tika piemeklēti drošības risinājumi, ko kuriem tika izvelēti efektīvākie, un sastādītas drošas tīmekļa lietotnes izstrādes vadlīnijas. Iegūtas zināšanas tika pielietotas praksē, izstrādājot drošu pret izpētītiem uzlaušanas paņēmieniem tīmekļa lietotni.

Atslēgvārdi: tīmekļa lietotne, urķis, tīmekļa lietotnes drošība, globālais tīmeklis

Abstract

Web application security

"Web application security" is the research of web application security threats, the attack capabilities and security solutions.

The work consists of 54 pages. Bibliography list consists of 21 source.

The aim was to explore how the Internet can be done criminal things that are threats to the causes and find solutions how to avoid attacks. Attention was paid to such web applications vulnerabilities as client-side data provided, authorization mechanism and others. It was shown that the failure to comply with security measures may fail the confidentiality of information and functioning of the system.

In result, there were made secure web application development guidelines from chosen most effective security solutions. These guidelines were applied in development of secure web application.

Keywords: web application, hacker, web application security, world wide web

SATURA RĀDĪTĀJS

APZĪMĒJUMU SARAKSTS

Tīmekļa lietojumprogramma - informācijas un interaktīvu pakalpojumu sakopojums, kas uzbūvēts no tīmekļa lapām un cita veida datiem.

Urķis - tehniski izglītots datoru entuziasts, kam sagādā prieku izpētīt dažādas datoru sistēmas un atrast pieeju tajās uzglabātajai informācijai.

Sensitīvā informācija - informācija, kuru kompetenta persona atzīst par aizsargājamu, jo tās nesankcionēta izmaiņa, bojāšana, atklāšana vai zaudēšana var būt postoša.

HTML - Hiperteksta iezīmēšanas valoda, kas ir izstrādāta tīmekļa lappušu un citas pārlūkprogrammā attēlojamas informācijas glabāšanai.

XSS - tīmekļa vietņu drošības ievainojamību paveids.

SQL - strukturēto vaicājumu valoda datu manipulēšanai datubāžu pārvaldības sistēmās.

SQLI jeb SQL injekcija - uzbrukuma paveids datubāzei, izmantojot SQL.

URL - vienotais resursu vietrādis internetā.

DBMS - datubāzes vadības sistēma

IEVADS

Laiks, kurā dzīvojam, ir saukts par datoru ēru, kuru raksturo iespēja brīvi gūt un sniegt informāciju, un ir iespējama tūlītēja pieeja datiem un zināšanām. Tas ietekmē gan ekonomiku, gan politiku, gan kultūru, un it īpaši strauji attīstās informācijas tehnoloģiju industrija. Mūsdienās vairs nevar iedomāties veiksmīgu biznesu bez tīmekļa vietnes.

Augot pieprasījumam, aug piedāvājums un globālais tīmeklis strauji piepildās ar skaistām biznesa tīmekļa vietnēm, interneta veikaliem, sociālajiem tīkliem un tiešsaistes servisiem, bet izstrādātājus industrija piesaista pat no citām profesijām. Pastāv daudzas iespējas apgūt tīmekļa vietņu izstrādi, tā ir gan augstākā izglītība, gan dažu nedēļu kursi, gan bezmaksas pamācības interneta resursos. Diemžēl ne vienmēr izstrādātājs ir pietiekami izglītots, lai spētu nodrošināt ne tikai sistēmas funkcionalitāti, bet arī ne mazāk nozīmīgu kvalitātes kritēriju - tīmekļa vietnes drošību. Par drošības neievērošanu arvien biežāk liecina ziņas par sensitīvo datu zagšanu un augošs slaveno urķu skaits.

Šī bakalaura darba mērķis ir izpētīt, kā interneta vidē ir iespējams izdarīt noziedzīgas lietas, kas ir apdraudējumu cēloņi, un meklēt risinājumus, kā var izvairīties no uzbrukumiem, kā arī izveidot ieteikumus un vadlīnijas drošas tīmekļa vietnes izstrādei, un pielietot iegūtās zināšanas praksē, izstrādājot praktiskos piemērus, ievērojot iepriekš noteiktās vadlīnijas.

Tā kā visus uzlaušanas paņēmienus šī darba ietvaros izpētīt nebūs iespējams, tika izvēlēti tikai daži, kas ir aktuālāki, kā liecina Open Web Application Security Project publicētie dati, kā arī balstoties uz autores iepriekšējās profesionālās pieredzes.

Darbs sastāv no piecām daļām. Pirmā daļa ir ieskats interneta vēsturē, kibernoziegumos un tiesiskajā regulējumā. Otrā daļa ir potenciālo tīmekļa lietotņu vājo vietu un uzbrukumu virzienu apraksts. Trešā daļa ir piemeklētie drošības risinājumi. Ceturtā daļa ir efektīvāko risinājumu izlase un apkopojums drošas tīmekļa lietojumprogrammas izstrādes vadlīnijās. Piektā daļa ir izstrādātās tīmekļa lietotnes apraksts.

1. GLOBĀLAIS TĪMEKLIS KĀ VIDE

Tīmekļa lietojumprogramma jeb tīmekļa lietotne, ir klienta-servera arhitektūras lietojumprogramma, kur klients ir tīmekļa pārlūkprogramma. Klients atspoguļo lietotāja saskarsmi, veic pieprasījumus serverim un apstrādā servera atbildes. Serveris saņem pieprasījumus no klienta, veic darbības, ģenerē tīmekļa lapu, un sūta to klientam pa HTTP protokolu [1].

Tīmekļa lietojumprogrammatūra kļūst arvien populārāka, jo tai ir daudz priekšrocību - tādas kā zemas resursu prasības, jo lietotne strādā neatkarīgi no lietotāja datorā instalētās operētājsistēmas, bet tās lietošanai nepieciešama tikai tīmekļa pārlūkprogramma un pieslēgums internetam. Tipiski tīmekļa lietotņu piemēri ir elektroniskais pasts, interneta veikali, tīmekļa žurnāli jeb emuāri, ziņu vai informācijas portāli, tiešsaistes biļešu rezervēšanas sistēmas, atskaišu sistēmas un citas sistēmas, kas tiek lietotas, izmantojot tīmekļa pārlūkprogrammas [2].

1.1. Globāla tīmekļa vēsture

Ideja par globālo tīmekli radās jau 1946. gadā. Slavenais amerikāņu zinātniskās fantastikas autors Murrajs Leinsters (Murray Leinster) uzrakstīja stāstu "A Logic Named Joe", kurā tika pausta globāla tīmekļa koncepcija, kur datori, kuri tika dēvēti par “loģikām”, dzīvoja katrā mājā un bija savienoti ar vienu, centrālo ierīci, tā apmainoties ar informāciju [3]. Pēc gandrīz 40 gadiem, sers Tims Berners-Lī, britu inženieris un datorzinātnieks, piedāvāja hiperteksta projektu. Tajā laikā nevienu šī ideja nesaistīja, izņemot vienu cilvēku - Tima Berners-Lī priekšnieku. Viņš akceptēja projektu, un tas tika realizēts ar nosaukumu World Wide Web. 1990. gadā Tims Berners-Lī ar beļģu datorzinātnieka Roberta Keljo palīdzību ieviesa hiperteksta dokumentu, lai varētu ērti un ātri piekļūt dažāda veida informācijai, to apskatīt un lietot. Pirmā pārlūkprogramma tā arī tika nosaukta - “the World Wide Web” un tika izmantota arī kā hiperteksta redaktors.

Interese par jauno tehnoloģiju bija neliela, to izmantoja pārsvarā zinātnieki un tie, kas bija saistīti ar datorzinātni, tātad tie, kuriem vajadzēja ātri un ērti piekļūt kādiem rakstiem un informācijai. Internets sastāvēja no vienkāršām tīmekļa lapām, statiskiem dokumentiem. Informācijas plūsma bija vienvirziena - no servera uz klientu. Navigācija starp lapām bija neērta.

1993. gadā parādījās Mosaic tīmekļa pārlūkprogramma, kas spēja atskaņot vai rādīt multivides datnes tiešsaistē. Šajā gadā parādījās arī pirmā primitīvā meklētājprogramma, kas ievērojami uzlaboja informācijas iegūšanas procesu, un atvēra durvis jaunām, jaudīgākām meklētājprogrammām [4].

1994. - 1995. gadā parādījās PHP programmēšanas valoda, kura bija viegli apgūstama un deva iespēju programmēt, veidot sarežģītu, loģisku sistēmu gandrīz ikvienam, Apache HTTP serveris, kurš šobrīd ir vispopulārakais serveris, MySQL relāciju datubāzu vadības sistēma, kā arī JavaScript (tajā laikā tās nosaukums bija Mocha, vēlāk - LiveScript), kas deva iespēju saziņā iesaistīties arī klientam. 1999. gadā internets sāka strauji attīstīties, informācijas klāsts strauji pieauga, auga arī lietotāju skaits, internetā parādījās biznesa projekti. Šis interneta attīstības posms vēlāk tika nosaukts par Web 2.0. [5].

Šodien internets krietni atšķiras no tā, kāds tas bija iepriekš. Mūsdienīgas tīmekļa vietnes pārsvarā ir tīmekļa lietojumprogrammas jeb lietotnes, informācijas plūsma ir divvirziena, no servera uz klientu, un otrādi [3].

Globalizācija un publiskā pieeja informācijai, no vienas puses, dod lielas iespējas globālas ekonomikas attīstībai, bet no otras puses, draud gan ekonomikai, gan fiziskām personām, jo cilvēki ir iemācījušies izmantot tehnoloģijas sasniegumus arī noziedzīgo darbību veikšanai.

1.2. Urķi un viņu motīvi

Urķi ir tehniski izglītoti datorentuziasti. Tie var būt ļoti dažādi, katrs ar saviem motīviem. Vārdu urķis ne vienmēr var lietot negatīvā nozīmē, tomēr internets ir vide, kur mīt arī nopietni likumpārkāpēji.

1.2.1. Urķu klasifikācija

Urķus var iedalīt trīs galvenajās kategorijās [6]:

- White hat (no angļu - “baltā cepure”);
- Grey hat (no angļu - “pelēkā cepure”);
- Black hat (no angļu - “melnā cepure”);

White hat urķi ir drošības speciālisti, kuri bieži strādā lielos IT uzņēmumos un nepārkāpj likumu. Grey hat urķi ir entuziasti, kuriem ir raksturīgi nelieli likumpārkāpumi vai nav likumpārkāpumu, bet ir pārkāpti kāda interneta resursa lietošanas noteikumi. Par Black hat urķiem sauc nopietnus likuma pārkāpējus.

Ir vēl daži urķu paveidi - tādi kā elite, kuriem ir labākas iemaņas, skriptu bērni, kuriem biežāk nav iemaņu, un viņi lieto uzlaušanai skriptus, kurus uzrakstījuši citi, biežāk - melnās cepures, vēl ir iesācēji, zilās cepures urķi, kuri konsultē kompānijas drošības jautājumos, un haktivisti, kuru aktivitātes ir saistītas ar politiskām idejām, vārda brīvību, cilvēktiesību aizsardzību un informācijas brīvības nodrošināšanu [3].

1.2.2. Urķu mērķi

Līdz 90. gadu vidum urķu galvenie motīvi bija [6]:

- Interese;
- Slava;
- Nauda;

Tā kā internets tajā laikā masveida lietotājiem bija kaut kas jauns, daudziem entuziastiem bija liela interese izpētīt, kas tas ir. Un tīmekļa vietņu uzlaušana tika realizēta šo eksperimentu ietvaros. Tie, kuriem izdevās kaut ko interesantu uzlauzt, stāstīja par to draugiem un kolēģiem, un tas bija prestiži. Tā kā daudz iespēju iegūt naudu nebija, pēdējo punktu varētu arī uzskatīt par interesi, jo, piemēram, piezvanīt uz ārvalstīm, izmantojot cita līdzekļus ir ieguvums, tomēr ne tik liels, cik liela ir vēlme to izdarīt.

Mūsdienās galvenais urķu dzinējspēks ir nelegāla finansu līdzekļu iegūšana. Internetā šodien atrodas bizness, uzplaukst veikali, notiek maksājumi, parādās daudz vairāk potenciālu ievainojamību nekā agrāk. Protams, interese un slava joprojām saista entuziastus, tomēr jau

krietni mazāk. Urķi pielieto savas zināšanas, arī sniedzot nelegālus pakalpojumus, tādus kā elektroniskā pasta vai sociālo tīklu profilu uzlaušana.

Nevar nepieminēt, ka pelnīt var ne tikai melnās cepures, nodarbojoties ar zādzībām, bet arī baltās cepures, jo lielas kompānijas, tādas kā Google nereti izsludina konkursus, kuru mērķis ir atrast ievainojamību produktā, kura meklējumos var piedalīties jebkurš entuziasts, bet balva ir diezgan liela naudas summa.

1.3. Tiesiskais regulējums

Latvijā internetu un viņa piedāvātas iespējas regulāri lieto aptuveni 1.4 miljonu iedzīvotāju, tāpēc drošības tēma ir aktuāla. Informācijas sistēmu drošības incidenti notiek regulāri, par ko liecina statistikas dati. Latvijā ar drošības incidentiem nodarbojas Informācijas tehnoloģiju drošības incidentu novēršanas institūcija CERT.LV, kas ir Latvijas Universitātes Matemātikas un Informātikas Institūta struktūrvienība [7, 8].

Uz noziegumiem elekrtoniskajā vidē attiecas gan Latvijas administratīvo pārkāpumu kodekss, gan krimināllikums (skat. 1. pielikumā). Tā, par piekļūšanu automatizētai datu apstrādes sistēmai urķim var pienākties sods ar brīvības atņemšanu līdz pieciem gadiem, piespiedu darbi vai naudas sods ar vai bez mantu konfiskāciju, atkarībā no ļaunprātības smaguma un sekām [9].

2. TĪMEKĻA LIETOTŅU POTENCIĀLĀS IEVAINOJAMĪBAS

Lai saprastu, kā izstrādāt produktus, kuri būtu pasargāti no vidēja urķa ļaunprātības, pirmkārt, jāzina, kur šajā sistēmā ir potenciālās vājās vietas.

Kā jau tika minēts, tīmekļa lietotnei ir klienta-servera arhitektūra, tas nozīmē, ka serveris apstrādā pieprasījumus no klienta - tīmekļa pārlūkprogrammas. Te arī slēpjas būtiskākā problēma - lietotājs var ievadīt patvaļīgu pieprasījumu, kas var izraisīt negaidītu (no izstrādātāja puses) servera uzvedību, jo nevar kontrolēt to, kas notiek klienta pusē.

Šajā nodaļā tiek analizētas tipiskākās vājās vietas, kuras tiek uzlauztas visbiežāk [10], tādas kā pilnvarošanas mehānisms, vaicājumi datubāzē, klienta puses loģika un citas.

2.1. Klienta puses apvedceļš

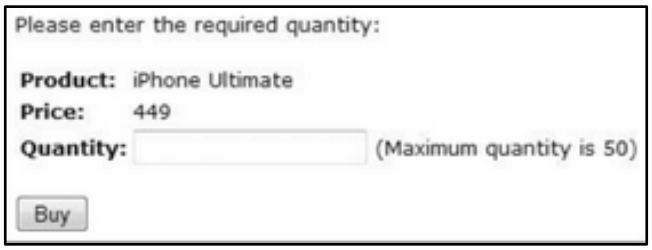
Izstrādātājs, implementējot klienta puses loģiku JavaScript valodā, bieži vien uzskata, ka viss strādās tā, kā viņš to ir kodējis. Taču tā ne vienmēr ir tā.

2.1. att. **Firebug konsolē tika izpildīts JavaScript kods**

Atverot tīmekļa pārlūkprogrammas konsoli (attēls 2.1.), pietiekami izglītots lietotājs var uzrakstīt jebkādu vajadzīgo funkciju un izmainīt jebkāda mainīgā vērtības, kas var izraisīt nepatīkamas sekas. Tīmekļa pārlūkprogrammas konsole ir rīks, ar kura palīdzību var ne tikai izmantot klienta puses funkcionalitāti savos nolūkos, kā arī izsekot tīmekļa pieprasījumus uz serveri, šādā veidā iegūstot priekšstatu par servera piedāvāto funkcionalitāti.

2.1.1. Paslēptie lauki

Klasisks piemērs ir paslēpto lauku rediģēšana. Tie ir tādi lauki, kas neatspoguļojas uz ekrāna, bet tiem ir kāda noklusēta vērtība. Piemēram, interneta veikalā ir forma, kura sniedz serverim informāciju par preci - unikālu identifikatoru un cenu, kas ir paslēpti, un daudzumu, kuru lietotājs var rediģēt noteiktā formā (2.2. attēls). Izmantojot pārlūkprogrammas konsoli, var mainīt paslēpto lauku vērtību, sniedzot serverim savu cenas piedāvājumu. Interneta veikala zaudējumi ir acīmredzami, turklāt bijuši gadījumi, kad, mainot cenas vērtību uz negatīvu skaitli, varēja pat saņemt naudas "atlīdzību", kas tika ieskaitīta kredītkartē [11].

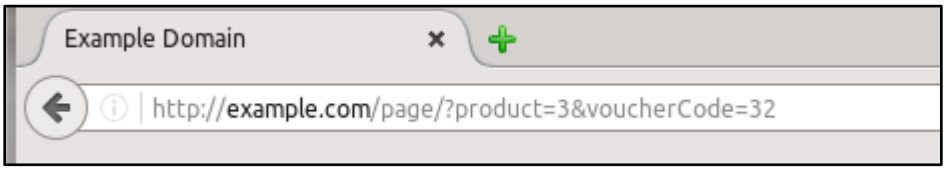


2.2. attēls. **Preces pasūtīšanas forma [11]**

Tas pats notiek arī ar lauku atribūtu "disabled". Ja formā ir kāds nerediģējams lauks, visticamāk tas nozīmē, ka serveris tomēr izmanto tā vērtību, un var mēģināt sniegt šo parametru funkcijas.

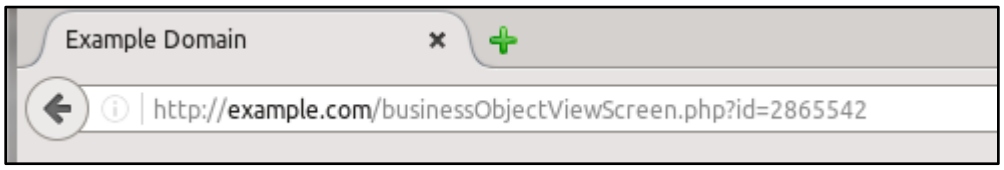
2.1.2. Parametri URL adresē

Cits populārs klienta puses apvedceļš ir URL parametru maiņa. Interneta veikala gadījumā tīmekļa pārlūkprogrammas adreses laukā var atrast un mainīt mainīgo vērtības, kas atbild par atlaižu kuponiem (2.3. attēls).



2.3. attēls. **URL ar atlaižu koda parametru**

Citā gadījumā var nesankcionēti piekļūt datiem, kuri nav paredzēti konkrētajam lietotājam, bet nav pietiekami aizsargāti (2.4. attēls).



2.4 attēls. **URL ar biznesa objekta identifikatora parametru**

2.1.3. JavaScript validācija

Vēl viena vājā vieta klienta pusē ir ienākošo parametru validācija JavaScript kodā. Dažu pārlūkprogrammu konfigurācijā ir iespēja pieslēgt vai atslēgt JavaScript koda realizāciju [11]. Ja pārbaudes ir implementētas tieši JavaScript funkcijās, ir iespēja, ka atslēdzot JS, varēs sniegt serverim nevalīdus datus, ja no tā necietīs funkcionalitāte.

Visticamāk šajā gadījumā funkcionalitāte tomēr cietīs, un elegantākā pieeja, kā apiet klienta puses validāciju, ir vai nu validējošo funkciju izsaukšanas rediģēšana hiperteksta formas kodā, vai nu pašas validējošas JavaScript funkcijas rediģēšana pārlūkprogrammas konsolē tā, lai atbilde vienmēr būtu pozitīva.

2.2. Pilnvarošanās sistēmā

Lietotāja pilnvarošanas mehānisms ir galvenais drošības pasākums pret nesankcionētu piekļuvi, un bieži tas ir galvenais urķu mērķis. Uzlaužot nepietiekami drošu pilnvarošanas funkcionalitāti, var tikt nozagta sensitīva informācija un pat iegūta kontrole pār visu sistēmu. Bez uzticama pilnvarošanas mehānisma ir bezjēdzīgi citi drošības pasākumi, tādi kā sesijas pārvaldība un lietotāju piekļuves kontrole.

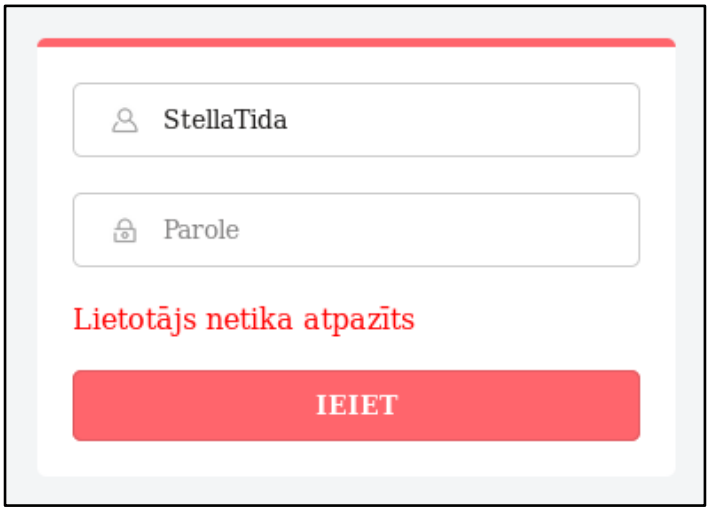
2.2.1. Runīgi kļūdu paziņojumi

Manuālas uzlaušanas procesā urķim var būt noderīgi kļūdu paziņojumi, kuri nepareizas informācijas gadījumā precizē, kāda tieši informācija ir nepareiza.

2.5. attēls. **Lietotāja piekļuves forma ar kļūdas paziņojumu**

Ja kļūda paziņo par to, ka parole ir nepareiza (attēls 2.5.), tas varētu nozīmēt, ka lietotājs ar ievadīto lietotājvārdu sistēmā eksistē.

Citā gadījumā sistēma var paziņot, ka lietotājvārds netika atpazīts (attēls 2.6.), tas nozīmē, ka tāda lietotājvārda sistēmā nav.



2.6. attēls. **Lietotāja piekļuves forma ar kļūdas paziņojumu**

Izmantojot informatīvus kļūdu paziņojumus, urķis var izveidot eksistējošu lietotājvārdu sarakstu, pēc tam tiem var mēģināt piemeklēt paroles.

2.2.2. Vāja parole

Tas, ko var uzlauzt pat tehniski neizglītots cilvēks, ir vāja parole. Vāju paroli raksturo daži aspekti:

- Pārāk īsa vai tukša;
- Vienkārši "vārdnīcas" vārdi;
- Parole un lietotājvārds ir vienādi;
- Atstāta noklusētā parole;

Lietotnē var vispār nebūt prasību paroles izveidošanai, vai tās var būt nepietiekami stingras, kas tikai atvieglo urķa uzdevumu. Lai noskaidrotu prasības paroles veidošanai, urķis var mēģināt izveidot lietotāja kontu ar dažādiem paroļu variantiem, sākot ar vienkāršāko, un, skatoties uz to, kādas paroles sistēma akceptē, bet kādas apzīmē kā kļūdainas, piemēram "Parolei jābūt vismaz 4 simbolu garai" un tā tālāk. Citā gadījumā visi kritēriji var būt aprakstīti uzreiz, tad

būs skaidri redzams, cik viegli tos var atminēt, piemēram, ja ierobežojums ir tikai simbolu virknes garums, to var uzskatīt par mājienu [12].

Katru gadu tiek publicēts visizplatītāko un šausmīgāko lietotāju paroļu reitingi (2. pielikums), kur dažus gadus pēc kārtas tādi pārstāvji kā "password" un "123456" ir topā, tas liecina par to, ka ar lielu varbūtību šīs paroles varētu derēt dažiem lietotāju kontiem.

2.2.3. Automatizēta rupjā spēka metode

Nopietnāks urķis mēģinās piekļūt kontam nevis manuāli, bet automātiski [11]. Ja sistēma pieļauj neierobežotu daudzumu pilnvarošanas mēģinājumu, tiek pielietoti algoritmi, kas piegādā paroles. Rupjā spēka piemeklēšanas metodi var realizēties divējādi: pirmais izmanto publiski pieejamas datubāzes, kuras iekļauj tūkstošus izplatītāko paroļu variantus, bet otrais ir algoritms, kas ģenerē paroles, izmantojot piemeklēšanas metodi pēc norādītajiem kritērijiem.

2.2.4. Nepilna pārbaude

Var gadīties, ka lietotājvārdi un / vai paroles saglabājas sistēmas datubāzē modificētā veidā, piemēram, saīsinot simbolu virknes garumu, vai pārbaude var būt nepilnīga, piemēram, nereģistrjūtīga, to var noskaidrot, ievadot paroles divus variantus. Tas, ka sistēma tiek pieķerta, ka tai ir šādas nepilnības, atvieglo parametru piemeklēšanu.

2.2.5. Nepareiza paroļu glabāšana

Paroļu glabāšana datubāzē var izraisīt nepatīkamas sekas, kas radušās veiksmīgu SQL injekciju rezultātā (skat. 2.5. nodaļu), kad urķim ir izdevies piekļūt vērtībām, kas bijušas datubāzē. Tāpēc paroļu "paslēpšanai" izmanto jaucējfunkcijas, kas pārveido dažāda garuma ievaddatus noteikta, fiksēta garuma datu virknē. Šo funkciju ideja ir pārveidot simbolu virkni tā, lai to nevarētu atšifrēt. Tomēr tādu izplatītu jaucējalgoritmu kā MD5 un SHA1, ir viegli apiet. Internetā brīvā piekļuvē ir daudz resursu ar tiešsaistes "tulkotājiem", kas pārveido simbolu virkni šifrētā virknē un otrādi. Tās ir tieši vārdnīcas, jo jaucējalgoritms neparedz datu atšifrēšanu, bet ir iespējams sastādīt vārdnīcu, dotā gadījumā ar iespējamām parolēm "tīrā" un sajauktā veidā [11, 13].

2.2.6. Paroles maiņas funkcionalitāte

Paroles maiņas funkcionalitāte ir izdomāta, lai paaugstinātu lietotāju kontu drošību. Tā ir izmantojama gan pēc lietotāja vēlmes, gan tiek aktivizēta automātiski ar noteiktu biežumu - 6 mēneši, gads, vai citu. Tomēr šai funkcionalitātei arī var būt trūkumi, kuri ļauj ļaunprātīgi to izmantot.

Nedroša paroles maiņas funkcionalitāte ir tāda, kura dod saprast, ja padotais lietotājvārds sistēmā eksistē vai neeksistē, ļauj neierobežotu mēģinājumu skaitu vecās paroles atminēšanai, kā arī pārbauda divas reizes ievadītu jauno paroli tikai pēc tam, kad ir veikta ievadīta lietotājvārda un vecās paroles kombinācija, kas dod iespēju piemeklēt pareizos datus piekļuvei sistēmā [11].

2.2.7. Aizmirstās paroles funkcionalitāte

Ir gadījumi, kad aizmirstās paroles atjaunošanas funkcionalitāte ļauj ievadīt e-pasta adresi, uz kuru tiek nosūtīta jauna automātiski ģenerēta parole, vai saite uz lietotni, kur var sastādīt jaunu paroli, kas ir acīmredzami nekorekti, jo sistēma, uzticoties ievadītajai e-pasta adresei, var nodot konta pārvaldību nezināmās rokās. Aizmirstās paroles atjaunošanas saites ģenerēšanas gadījumā, saite var tikt veidota no viegli piemeklējamām sastāvdaļām, kas ļaus pārņemt kontroli pār dažiem lietotāju kontiem.

2.3. Sesijas pārvaldība

Sesijas mehānisms ir vajadzīgs, lai katru reizi, sūtot pieprasījumu uz serveri, nevajadzētu atkārtoti pilnvaroties sistēmā. Tā ir vēl viena potenciāla vāja vieta tīmekļa lietojumprogrammās.

Lietotne glabā tīmekļa pārlūkprogrammas sīkdatnēs sesijas identifikatoru, kas identificē lietotāju, kas pašlaik ir pilnvarots sistēmā. Sesijas identifikators ir unikāla šifrēta simbolu virkne, kuru var veidot vairākas sastāvdaļas. Pašu identifikatoru var atrast tīmekļa pārlūkā, un, ja samainīt savu identifikatoru uz kādu citu, ir iespējams, ka jaunais identifikators sakritīs ar kādu eksistējošu, kas automātiski dos iespēju piekļūt kāda cita lietotāja kontam [3, 11].

2.4. Uzbrukumi lietotājiem: XSS

XSS (no angļu Cross Site Scripting) ir uzbrukums lietotājam. To realizē, ievietojot pašveidotu skriptu lapas hipertekstā tā, lai kods realizētu urķa intereses kā informācijas plūsmas leģitīma daļa [14].

Šādu skriptu injekciju mērķis visbiežāk ir lietotāju sesijas sīkdatnes (angļ. cookies), kas ir, piemēram, lietotāja sesijas identifikators kādā tīmekļa lietotnē, kas ļauj lietotājam palikt pilnvarotam sistēmā, neievadot savu lietotājvārdu un paroli katru reizi, kad notiek pieprasījums serverim.

Skriptu injekcijām ir divi paveidi: saglabājamās un nesaglabājamās.

2.4.1. Saglabājamas skriptu injekcijas

Saglabājamās skriptu injekcijas ir tādas injekcijas, kurās kaitīgais skripts tiek saglabāts sistēmā un atspoguļojas tīmekļa lapas hipertekstā. Šim nolūkam bieži tiek lietoti tīmekļa lapas ievadlauki, piemēram, saglabājot ziņojumu forumā, ievieto teksta ievadlaukā tagu “script”, kuru pēc noklusējuma tīmekļa pārlūkprogramma uztver kā leģitīmu kodu, vai ievietojot tagu “href”, kas izveidos saiti un novirzīs lietotāju tur, kur ir izdomājis urķis. Tādā gadījumā iemānītais skripts realizēsies tik ilgi, cik tas atradīsies sistēmā, sūtot sesijas sīkdatnes un citu sensitīvu informāciju uz nezināmu galamērķi [14, 15].

2.4.2. Nesaglabājamas skriptu injekcijas

Nesaglabājamā skripta injekcija ir injekcija, kas tīmekļa lapas hipertekstā nesaglabā kaitīgu skriptu, tas tiek ievietots URL adresē. Lai šis uzbrukums realizētos, ir nepieciešams, lai potenciālais upuris uzklikšķinātu uz pārveidotās saites.

Ne vienmēr var redzēt, no kā sastāv saite, jo tās parametrus, kā arī skriptu, var "paslēpt" ar base64 kodēšanu, piemēram, HTML birka "<script>" izskatīsies kā "PHNjcmlwdD4NCg==", ko parastais lietotājs nevarēs saprast [14, 15].

2.5. Datubāze: SQLI

Situāciju, kad uzbrucējs ar klienta tiesībām mēģina piekļūt datubāzei, kas ir tīmekļa lietotnes pamatā, dēvē par SQL injekcijām.

2.7. attēls. **Lietotāja piekļuves forma**

Ja servera pusē nav pietiekami daudz pārbaužu, urķis var ievadīt kādā no tīmekļa lapas ievadlaukiem SQL vaicājuma fragmentu, un, ja kods realizējas, lietotājs var pēc paša ieskatiem rediģēt vai kopēt datubāzi. Klasisks piemērs ir SQL koda injekcija lietotāja piekļuves formas ievadlaukos (attēls 2.7.) [14].

Pēc formas iesniegšanas realizējas SQL pieprasījums datubāzē (2.8. attēls).

```
6
7   $query = 'SELECT * FROM Users '.
8            'WHERE Username = \''.$username.'\''.
9            'AND Password = \''.$password.'\';';
10
```

2.8. attēls. **SQL vaicājums PHP kodā**

Ja forma ir nedroša, tad, ievadot tajā viltīgu vērtību (2.9. attēls), kura tiks uztverta kā kods, var iegūt nepieciešamo informāciju.

2.9. attēls. **Lietotāja piekļuves forma**

Pēc šī ievadītā SQL pieprasījuma (2.10. attēls) ir vienmēr pozitīvs nosacījums, un tiek atlasīta visa lietotāju tabula un nesankcionēta piekļuve sistēmai urķim ir pieejama, izmantojot izvēlētā lietotāja datus.

```
10
11  SELECT * FROM Users
12      WHERE Username = '1' OR '1' = '1'
13      AND Password = '1' OR '1' = '1';
14
```

2.10. attēls. **SQL vaicājums pēc formas iesniegšanas**

SQLI tiek dalītas: injekcijas ar iekšējām atbildēm, ar ārējām atbildēm un aklās injekcijas.

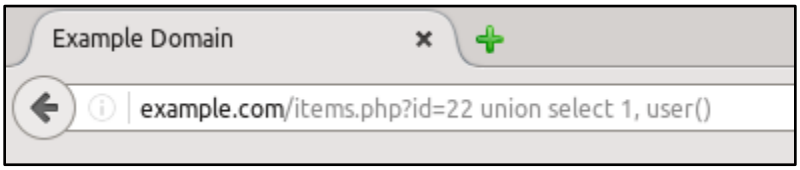
2.5.1. Uz kļūdām bāzētas SQL injekcijas

SQLI ar iekšējām atbildēm jeb uz kļūdām bāzētas injekcijas ir iespējamas, kad kļūdas, kuras izraisa nekorekti vaicājumi datubāzē, ir viegli pieejamas tīmekļa lapā, tāpēc tās tiek dēvētas par SQL injekcijām, kas bāzējas uz kļūdām. Kļūdas ir ērti izmantojamas, lai noskaidrotu datubāzes iekārtu, tabulu, kolonnu nosaukumus, pārējo informāciju.

Noskaidrot, vai sistēma ir neaizsargāta pret SQL injekcijām, var pievienojot tīmekļa pārlūkprogrammas adreses ievadlaukā apostrofu. Ja lapā parādas kļūda "Error in your SQL Syntax", tas nozīmē, ka var mēģināt veikt uzbrukumu. Pat ja kļūda neparadās, var mēģināt veikt uzbrukumu. Parasti izmanto vienmēr patieso nosacījumu (2.11. attēls).

2.11. attēls. **SQL injekcija ar vienmēr patieso nosacījumu**

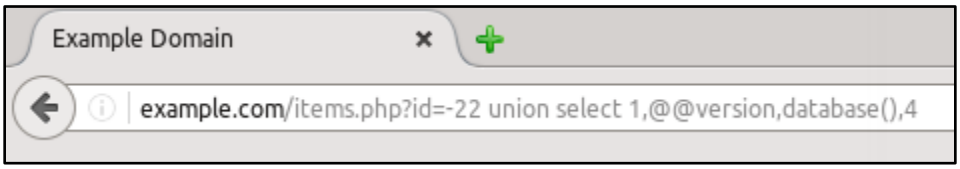
Atradot sistēmā vājo vietu, kas padodas SQL injekcijai, var ķerties pie datu izvilkšanas.



2.12. attēls. **SQL injekcija lietotāja nosaukuma noskaidrošanai**

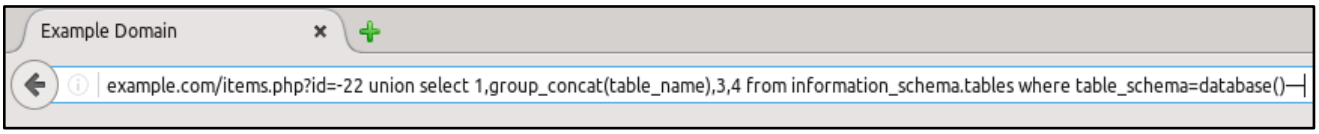
Izmantojot dažādu DBMS specifiskās komandas, var noskaidrot tās tipu, piemēram, MySQL atbilst funkcija "user()", MS-SQL funkcija "user_name()", u.c. Piemēram, ja DBMS ir MySQL, var noskaidrot lietotāja nosaukumu, padodot vaicājumu ar funkcijas "user()" izmantošanu (2.12. attēls).

Izmantojot “UNION” vai “ORDER BY” konstrukcijas, ir viegli noskaidrot tabulas kolonnu skaitu, lai SQL kļūdas gadījumā no tās izvairītos. Kad ir zināms tabulas kolonnu skaits, piemēram, 4, var noskaidrot datubāzes nosaukumu (2.13. attēls).



2.13. attēls. **SQL injekcija datubāzes versijas un nosaukuma noskaidrošanai**

Izmantojot datubāzes nosaukumu, var izvilkt datubāzes metadatus ar informāciju par visām datubāzes tabulām (2.14. attēls).

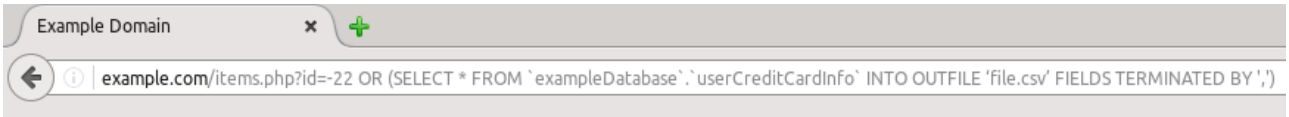


2.14. attēls. **SQL injekcija datubāzes tabulu nosaukumu noskaidrošanai**

Pēc datubāzes tabulu nosaukumu noskaidrošanas var izvilkt tās kolonnu nosaukumus, un visus interesējošus datus.

2.5.2. SQL injekcijas ar ārējām atbildēm

SQL injekcijas ar ārējām atbildēm ir injekcijas, kad urķis sūta datubāzes atbildes ar kļūdām vai datiem uz kādu galamērķi, piemēram, savu e-pastu.

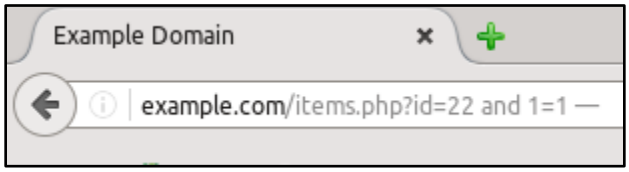


2.15. attēls. **SQL injekcija ar informācijas ierakstīšanu datnē**

Tas ir iespējams, mainot datubāzes pārvaldības sistēmas konfigurācijā e-pastu saņēmējus, pievienojot savu, vai saglabājot vaicājuma rezultātu datnē (2.15. attēls).

2.5.3. Aklās SQL injekcijas

Atšķirībā no SQL injekcijām ar iekšējām un ārējām atbildēm, aklās SQL injekcijas ir injekcijas, kas tiek izmantotas tad, ja nav iespējams saņemt kļūdas no datubāzes, tomēr ir iespēja "komunicēt" ar serveri, saņemot atbildes, kas ir ekvivalentas "jā" vai "nē".



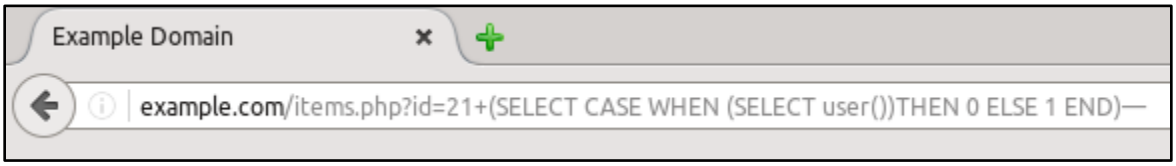
2.16. attēls. **SQL injekcija**

Šajā gadījumā, pievienojot pieprasījumam patiesu nosacījumu (2.16. attēls) lapas saturs nemainīsies, jo abi nosacījumi ir patiesi.

Ja pievienojot pieprasījumam aplamu nosacījumu (attēls 2.17.), lapā parādījās kļūdas paziņojums, vai lapa palika tukša, tas nozīmē, ka SQL pieprasījums izpildījās ar ievadīto nosacījumu, un var turpināt uzbrukumu.

2.17. attēls. **SQL injekcija**

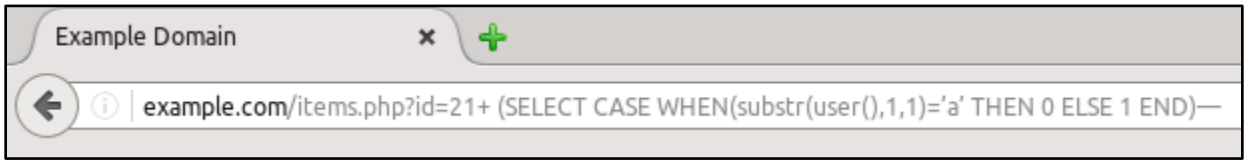
Pieņemot, ka objekta ar identifikatoru 21 datubāzē neeksistē, tad vienkāršais vaicājums, samainot oriģinālajā URL adresē identifikatoru no 22 uz 21, tiks saņemta negatīva atbilde, tāda pati, ka iepriekšējā piemērā, kad tika pievienots aplams nosacījums. Šo vaicajumu var izmantot informācijas iegūšanai, piemēram, DBMS versijas noskaidrošanai (2.18. attels).



2.18. attēls. **SQL injekcija**

Vaicājums ir sastādīts tā, lai palielinātu pieprasīta objekta identifikatoru par vieninieku, kas nodrošinās tīmekļa lapas korektu atspoguļošanu, gadījumā, ja funkcija user() nostrādās, kas nozīmēs, ka DBMS ir MySQL.

Tālāk var mēģināt noskaidrot lietotāja nosaukumu pa vienam burtam, pārbaudot sakritību ar visiem alfabēta burtiem pēc kārtas (2.19. attēls).



2.19. attēls. **SQL injekcija**

Aklo SQL injekciju uzbrukums ir grūtāks, jo prasa lielāku pacietību un daudz laika, tomēr informācijas izvilkšana no datubāzes un datu bojāšana ir iespējami.

3. TĪMEKĻA LIETOTŅU IEVAINOJAMĪBU NOVĒRŠANAS IESPĒJAS

Iepriekšējā nodaļā tika aplūkotas potenciālā tīmekļa lietotņu ievainojamība un izplatītākie uzbrukumu veidi, kas ir iespējami sistēmā esošo jau zināmo vājo vietu dēļ. Analizējot ievainojamības situācijas, var secināt, ka būtiskāka problēma tīmekļa lietojumprogrammu drošībā ir dati, ko sniedz klients, tomēr problēmas var radīt arī servera piedāvātā loģika.

Visas internetā esošas tīmekļa lietotnes var dalīt trīs daļās [16]:

- Lietotnes, kuras jau ir uzlauztas;
- Lietotnes, kuras vēl nav uzlauztas;
- Lietotnes, kuru izstrādātāji zina par potenciāliem uzbrukumu vektoriem un pasargā sistēmu no tiem;

Vēlamā kategorija, kurā jācenšas iekļauties, ir pēdējā, tātad, lai izstrādātu no vidēja līmeņa ļaunprātības drošas tīmekļa lietotnes, jāzina, kādiem sistēmas elementiem jāpievērš uzmanība, un jāpiemeklē atbilstoši drošības pasākumi pret izplatītākajiem uzbrukumiem.

3.1. Ieejas datu pārbaude

Lielākais uzbrukumu vektoru skaits ir virzīts uz ieejas datu viltošanu[11]. Tas ietver sevī tādus uzbrukumus kā skriptu un SQL vaicājumu injekcijas, klienta puses skriptu rediģēšanu vai atslēgšanu. Jāatceras, ka visi ienākošie dati ir neuzticami.

Lai izvairītos no potenciālajām injekcijām, jānodrošina datu pārbaude un filtrācija. Pie tam līdz filtrācijai būtu labāk atteikties no datiem, kuri acīmredzami neatbilst noteiktam formātam. Piemēram, ja sistēma sagaida objekta identifikatoru, tam jāatbilst nosacījumiem, tādiem kā simbolu virknes garums, simbolu kopa (burti / cipari), ja tas ir skaitlis, tad skaitļa pieļaujamais apgabals u.c. Ja ieejas parametri neatbilst nosacījumiem, koda realizācijai jābeidz darbs un jāsniedz lietotājam informācija par kļūdu.

Pretējā gadījumā, ja sagaidāmā ievade satur lielāku simbolu klāstu, jāpārbauda, vai ienākošie parametri nesatur HTML birkas vai SQL vaicājumu elementus. Tomēr tādām pārbaudēm ir pielietošanas ierobežojumi, kā arī savi mīnusi. Bieži sastopamā speciālu simbolu kombināciju pārbaude ir tā saucamais “melnais saraksts”, kas iekļauj sevī visas (kā domā

izstrādātājs) iespējamās bīstamās ievades, un meklē ienākošajā tekstā atbilstības tām. Tomēr tāda implementācija nav eleganta, un cilvēciskā faktora dēļ var tikt pieļautas kļūdas vai nav ņemti vērā visi gadījumi. Klasisks melnā saraksta nepilnību piemērs var būt visu simbolu reģistra neievērošana, piemēram, ievade ar SQL komandu "SELECT" nederēs, bet "SeLeCt" derēs, un gadījumā ar skripta injekciju "alert ('Hello webpage!')" nerealizēsies, bet "prompt('Hello webpage!')" realizēsies.

Gadījumi, kad kods tomēr jāpieņem, ir saistīti ar elektronisko žurnālu un citu rakstu portālu ievadlaukiem, kur ir pieņemama koda izmantošana, piemēram, ja tas ir raksts par hiperteksta vai programmēšanas valodas izmantošanu. Šajā gadījumā ir svarīgi padarīt drošas speciālas simbolu kombinācijas, ko nevarēs pārvērst realizējamajā kodā, saglabājot tās datubāzē un atspoguļojot tīmekļa lapā lietotājam drošajā veidā.

Lai to realizētu, izmanto speciālus modifikācijas simbolus (angļ. escape characters) (3.1. attēls) vai modificēšanas funkcijas, kuras nodrošina teksta kā parasta teksta uztveršanu.

Escape	Description	Example	Output String
\'	Single quote mark	"couldn\'t be"	couldn't be
\"	Double quote mark	"I \"think\" I \"am\""	I "think" I "am"
\\	Backslash	"one\\two\\three"	one\two\three
\n	New line	"I am\nI said"	I am I said
\r	Carriage return	"to be\ror not"	to be or not
\t	Tab	"one\ttwo\tthree"	one two three
\b	Backspace	"correctoin\b\b\bion"	correction
\f	Form feed	"Title A\fTitle B"	Title A then Title B

3.1. attēls. **Modifikācijas simboli JavaScript valodā [17]**

Ērtākas modificēšanas lietošanai daudzās programmēšanas valodās ir implementētas iebūvētas funkcijas [13]. Piemēram, JavaScript valodā tā ir funkcija escape(), PHP valodā - mysql_escape_string(), mysqli_escape_string(), sqlite_escape_string(), addslashes() un citas funkcijas, kas pārveido tekstus dažādi reprezentējot, ar dažādām niansēm.

Cits variants ir sagatavotu vaicājumu izmantošana (angļu - prepared statements) [17], kas pastāv, piemēram, PHP valodā (3.2. attēls).

Sagatavota vaicājuma priekšrocība ir tas, ka vaicājums un parametri tiak padotas atsevišķi, un tiek stingri definēti. Vaicājums tiek sintaktiski validēts jau pirms tam, kad tajā tiek padoti parametri, kuri tiek automātiski modificēti pēc augstākminēta principa.

```php
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();
```

3.2. attēls. **PHP sagatavota vaicājuma piemērs [13]**

Secinājums, ko var izdarīt, analizējot izpētīto tīmekļa lietotņu ievainojamību, ir - nedrīkst paļauties uz kodu, kas ir implementēts klienta pusē, jo tas ir modificējams un pat atslēdzams. Līdz ar to obligāts drošības pasākums ir visu ienākošo datu daudzpakāpju pārbaude, kā klienta pusē (kā ātrākais ceļš atmest neleģitīmu informāciju), tā arī servera pusē, tas nodrošinās pārbaudi pēc būtības un drošību. Ideāls variants ir katras funkcijas pārbaudes nodrošināšana, kas varētu atvieglot arī kļūdu atrašanu sistēmas loģikā.

3.2. Lietotāju piekļuves pārvaldība

Svarīga tīmekļa lietotnes drošības sastāvdaļa ir lietotāju piekļuves kontrole sistēmas datiem un funkcionalitātei. To veido [11]:

- Piekļuves mehānisms (angļ. authentication);
- Sesijas pārvaldība (angļ. session management);
- Piekļuves kontroles sistēma (angļ. access control);

Katra no šīm sastāvdaļām ir vienotas ķēdes posms, un drošības mehānisms ir tik drošs, cik ir drošs ir vājākais no posmiem.

3.2.1. Drošs pieteikšanās mehānisms

Lai izveidotu drošu piekļuves mehānismu, ir jāakceptē vairāki aspekti. Pirmkārt, jāpārliecina lietotāji, ka ir vērts veidot sarežģītas paroles, kuras sastāv no dažādu reģistru burtiem un cipariem, un kuras nav pārāk īsas. Jo garāka parole, jo drošāka (no tehniskās puses) tā ir, jo ar rupjā spēka metodi tādu paroli ir grūtāk piemeklēt.

Otrkārt, ja lietotājs pirmajā reizē neievada pareizus datus, ir iespēja, ka ir sācies nesankcionētas piekļuves mēģinājums, un tas, kas būtu jānodrošina, ir iespējamā urķa mērķa sasniegšanas apgrūtināšana. Šim nolūkam var izmantot speciālus palīglīdzekļus, tādus kā simbolu ievadīšana no automātiski ģenerējama attēla (CAPTCHA), kas nepieļaus automātisku paroles piemeklēšanu, un krietni kavēs manuālu piemeklēšanu, lai administrators nepieciešamības gadījumā varētu rīkoties, kamēr nav par vēlu.

Treškārt, lai nodrošinātu paroļu uzticamu glabāšanu, jāizmanto drošs jaucējalgoritms. Kā jau tika minēts, tāds populārs un plaši izmantojams algoritms kā MD5 nav uzticams. PHP valodā ir speciālas drošas jaucējfunkcijas [13], kas palīdz ērti un droši šifrēt paroles, kuras tiek atjauninātas, lai paaugstinātu drošības līmeni.

3.2.2. Sesijas pārvaldība

Sesijas drošība ir atkarīga no tās identifikatora sarežģītības. Tāpat kā ar paroli sesijas identifikatoru var mēģināt piemeklēt, tāpēc tā veidošana nedrīkst būt prognozējama, piemēram - sastādīta no datuma un lietotājvārda [11].

Drošības nolūkā, ja lietotājs kādu laika periodu (piemēram, 15 minūtes) nav aktīvs lietotnē, sesiju var beigt. PHP valodā pēc noklusējuma sesija ilgst 1440 sekundes, kas ir 24 minūtes [13].

3.2.3. Lietotāju piekļuves vadība

Ne mazāk svarīgs lietotāju pārvaldības aspekts ir lietotāju piekļuves vadības sistēma, kas nodrošina lietotāju pieejas ierobežojumu sistēmas datiem un funkcionalitātei, tajā var nodrošināt vairāku pieejas līmeņu hierarhiju. Šim nolūkam sistēmā tiek definētas lietotāju lomas, un katram lietotājam tiek piešķirta viena vai vairākas lomas.

Servera līmenī jāpārbauda, vai lietotājam ir pietiekams piekļuves līmenis, lai veiktu darbību vai piekļūtu informācijai. Tas attiecas uz 2.1.2. nodaļā izskatītu gadījumu, kad URL adresē var tikt mainīti parametri, veicot pieprasījumu nesankcionetai piekļūšanai datiem, tāpēc ir svarīgi pārbaudīt lietotāja tiesības pirms katra pieprasījuma izpildes.

3.3. Uzbrucēju pārvaldība

Būvējot drošu tīmekļa lietojumprogrammu, jādomā par to, ka sistēmas reakcijām uz uzbrukumiem ir jābūt kontrolētiem. Ir korekti jāapstrādā kļūdas un jāziņo par tām administratoram.

3.3.1. Kļūdu kontrolēšana

Svarīga kļūdu atspoguļošanā ir tāda atspoguļošana, no kuras nevar iegūt tehnisko un sensitīvo informāciju, lai pēc tam to ļaunprātīgi izmantotu [16]. Lietotnei ir jāpieņem kļūdu ziņojumi, kurus ģenerē sistēma, jo tie bieži vien var iekļaut sevī pārāk daudz informācijas, kas var pastāstīt tehniski izglītotam lietotājam par sistēmas iekšējo iekārtojumu, datubāzes struktūru un glabājamo informāciju - kā, piemēram, SQL injekciju gadījumā.

Daudzās programmēšanas valodās, arī PHP, pastāv try-catch bloku konstrukcijas, kas nodrošina kļūdu fiksēšanu un drošu apstrādi. Labs variants ir šo konstrukciju izmantošana, un kļūdu gadījumā lietotājam tiek piedāvāts neinformatīvs kļūdas paziņojums.

3.3.2. Notikumu žurnāls

Lai zinātu, kas notiek ar sistēmu, kur parādās kļūdas un kur notiek uzlaušanas mēģinājumi, ir lietderīgi ieviest notikumu žurnālu, lai problēmsituācijas gadījumā saprastu, kas ir noticis ar sistēmu, un reproducētu darbības nepieciešamības gadījumā, lai izpētītu loģisku kļūdu vai ievainojamību. Noderīga informācija var būt [11]:

- Notikumi, kas saistīti ar pilnvarošanos sistēmā, kā veiksmīgi, tā arī neizdevušies mēģinājumi, paroles maiņa;
- Svarīgas transakcijas, ja tādas ir, piemēram, apmaksa;

- Ievades un pieprasījumi, kurus sistēma ir fiksējusi kā kaitīgus;
- Mēģinājumi piekļūt neatļautai informācijai;

Tipiski, efektīvi notikumu žurnāli vāc informāciju par notikumiem, šo informāciju veido laika zīmogs, pieprasījuma IP adrese un lietotāja konts, ja tāds eksistē un lietotājs ir piekļuvis sistēmai. Svarīgi ir atcerēties, ka notikumu žurnālam arī jābūt drošam pret uzbrukumiem, jo tajā var glabāties sensitīva un urķim ļoti noderīga informācija, tādēļ notikumu žurnālus bieži implementē atsevišķā autonomā sistēmā, kura ļauj tikai saglabāt objektus.

3.3.3. Paziņojumi administratoriem

Notikumu žurnāls palīdz aplūkot un analizēt notikumus retrospektīvi, tomēr dažos gadījumos ir jārīkojas tūlīt [19]. Tad jānodrošina tūlītēja paziņojuma nogāde sistēmas administratoriem, kuri nekavējoties var reaģēt uz situāciju un, ja nepieciešams, bloķēt IP adresi, no kuras notiek uzbrukums, vai pat izslēgt sistēmu, kamēr nebūs atrasts risinājums drošības nepilnību novēršanai.

Paziņošanas mehānisms var būt integrēts notikumu žurnālā, tas ir diezgan ērti. Administrators var saņemt īsziņu uz mobilo telefonu, tajā būs iekļauts īss apraksts par notikušo, piemēram, laiks, notikuma kategorija (kļūda, kritiskā kļūda u.c.), sistēmas daļa, kur tas ir noticis, un notikumu žurnāla ieraksta numurs. Vienlaicīgi ar īsziņu var pienākt e-pasta vēstule ar saiti, kas norāda uz notikumu žurnālu, kur būs iespējams aplūkot visu informāciju par notikumu. Lietderīgi var būt paziņojumi par šādiem notikumiem [11]:

- Netipiski liels pieprasījumu skaits no vienas IP adreses;
- Biznesa anomālijas, tādas kā netipiski liels transakciju skaits no viena bankas konta;
- Pieprasījumi, kuri satur zināmus uzbrukumu paņēmienus;
- Pieprasījumi, kuri rediģē informāciju, kurai parastiem lietotājiem nav piekļuves;

3.3.4. Automātiska reaģēšana uz uzbrukumu

Kā papildus drošības pasākums, sistēmā var tikt realizēta automātiska reaģēšana uz uzbrukumu, negaidot, kamēr administratori sāks rīkoties. Tas var apgrūtināt urķa uzdevumu. Tāda funkcionalitāte, pamanot, ka notiek aizdomīgi pieprasījumi, var pakāpeniski samazināt

servera atbilžu ātrdarbību konkrētam lietotājam, vai pirms katras darbības beigt lietotāja sesiju un pieprasīt iekļūt sistēmā ar savu lietotājvārdu un paroli. Protams, tas negarantēs sistēmas drošību, bet var palīdzēt kavēt laiku, kamēr administratori sāks rīkoties.

3.4. Lietotnes administrēšana

Jebkurai nopietnai tīmekļa lietojumprogrammai nepieciešama pārvaldība un administrēšana. Administratori var pārskatīt notikumu žurnālus, reaģēt uz kļūdām, meklēt ievainojamības, pārvaldīt lietotāju profilus, arī lietotāju piekļuves tiesības, rīkoties uzbrukumu gadījumā, atslēdzot sistēmas funkcionalitātes daļu, vai nepieciešamības gadījumā pat visu sistēmu.

4. DROŠAS TĪMEKĻA LIETOTNES IZSTRĀDES VADLĪNIJAS

Apkopojot iepriekšējā nodaļā drošības risinājumus, var izcelt sekojošas drošas tīmekļa lietojumprogrammatūras izstrādes vadlīnijas:

- Ienākošo datu pārbaude
    - Obligāta pārbaude servera puses kodā
    - Visu funkciju ienākošo parametru pārbaude
- Ienākošo parametru modificēšana, sagatavotu vaicājumu izmantošana
- Droša lietotāju pieteikšanās sistēma
    - Sarežģītu paroļu pieprasīšana
    - Uzticamu jaucējalgoritmu izmantošana
    - Sarežģītu sesijas identifikatoru ģenerēšana
    - Sesijas automātiska beigšana
    - Drošības kodu pieprasīšana neveiksmīga pieteikšanās mēģinājuma gadījumā
- Lietotāju piekļuves kontrole
    - Lietotāju kontu sadalīšana pa kategorijām
    - Nosacījumu un tiesību izstrāde lietotāju kategoriju darbam ar sistēmu
    - Visu darbību pārbaude
- Kļūdu kontrolēšana
    - Try-catch bloku konstrukciju izmantošana
    - Neinformatīvi kļūdu paziņojumi lietotāja saskarnē
- Notikumu žurnāls
    - Transakciju vēstures reģistrēšana
    - Piekļuves vēstures reģistrēšana
    - Paslēptās informācijas pieprasījumu reģistrēšana
    - Potenciāli kaitīgu pieprasījumu reģistrēšana
    - Nesekmīgu sistēmas lietošanas meģinajumu reģistrēšana
    - Īsziņas un epasta paziņojumi administratoriem problēmu gadījumā
- Automātisks drošības mehānisms
    - Servera atbilžu ātruma samazināšana

- Sesijas pārtraukšana

Dotās vadlīnijas var nodrošināt aizsardzību pret vidēja datoru entuziasta uzbrukumu mēģinājumiem, taču jāņem vērā, ka minimālais vēlamais drošības līmenis var atšķirties dažādām tīmekļa lietotnēm, atkarībā no riskiem informācijas sensitivitāti, sniegto pakalpojumu sarežģītību un nozīmību, un potenciālo ieguvumu no sistemas uzlaušanas, tā, piemēram, interneta veikalam drošības prasības būs zemākas, nekā interneta bankai.

5. VADLĪNIJU PRAKTISKAIS PIELIETOJUMS

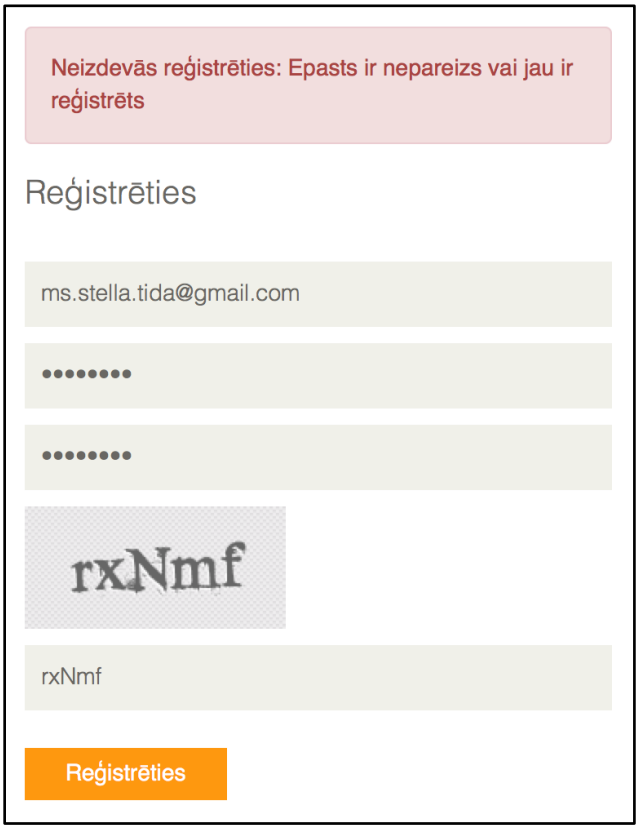
Tīmekļa lietotnes izstrādei tika izvēlēta PHP programmēšanas valoda un MySQL datubāze, jo autorei tie ir vairāk ierasti. Tīmekļa lietotne ir interneta veikals, kurā lietotāji var reģistrēties, pilnvaroties, mainīt vai atjaunot paroli, izvēlēties preces, ielikt vai izlikt tās no sava iepirkumu groza. Ir divas lietotāju kategorijas - klienti un administratori.

Reģistrēšanai sistēmā (funkcijas pirmkods pieejams 3. pielikumā) tiek prasīta e-pasta adrese un pietiekami sarežģīta divas reizes ievadīta parole. Jauniem lietotājiem automātiski tiek piešķirta loma "klients". Rediģēt lomas var tikai esošais administrators.

Administratoram pilnvarojoties, automātiski atveras administratora panelis, kur viņam ir tiesības rediģēt datus, savukārt klientam atveras parastā lietotāja skats. Parastais lietotājs nevar tikt administratora panelī, jo administratora paneļa saturs netiek radīts lietotājiem bez administratora lomas.

Autore atteicās no lietotājvārdu izmantošanas lietotnē, jo lietotāju reģistrācijas formā tik un tā būtu jāinformē, ka lietotājvārds ir aizņemts (attēls 5.1.), kas dotu iespēju urķim piemeklēt kādu eksistējošu izplatītu lietotājvārdu (piem. janis, user, 12345). E-pasta adresi ir sarežģītāk piemeklēt, tāpēc tā ir drošāka.

Papildus, lai izvairītos no automatizētas reģistrēto e-pastu piemeklēšanas, tika izmantoti automātiski ģenerējami attēli (CAPTCHA) [20], kuros attēlotus simbolus lietotājam ir jāievada formā.



5.1. attēls. **Lietotāja reģistrācijas forma**

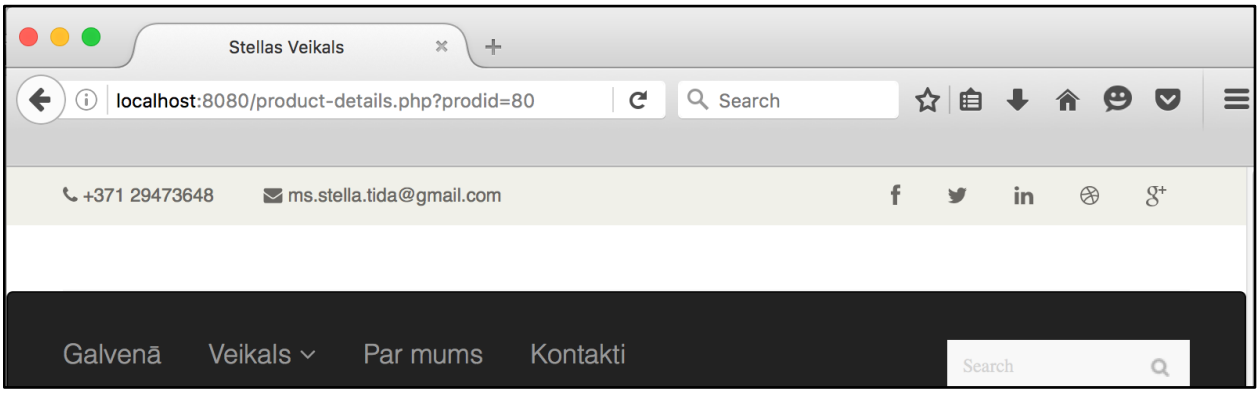
Veiksmīgai reģistrēšanai parolei arī jāatbilst prasībām. Tai jābūt vismaz 8 simbolu garai, tajā jābūt iekļauti vismaz 2 lielie burti, vismaz 2 mazie burti, un vismaz 2 cipari. Parole tiek šifrēta ar jaucējalgoritmu, izmantojot iebūvētu PHP funkciju. Jaucējalgoritma izejas virknes garums ir sešdesmit simboli, un tā ģenerēšanai tiek izmantota atslēga, kas padara to drošu.

Pilnvarošanās mehānisms skaita neveiksmīgu piekļuves mēģinājumu skaitu, un sākot ar otro, pieprasa kodu no bildes (attēls 5.2.).

5.2. attēls. **Neveiksmīgs pilnvarošanas mēģinājums**

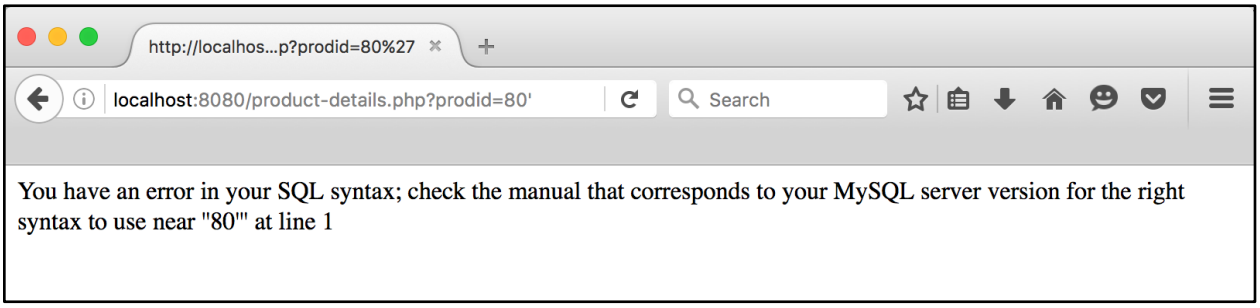
Ja lietotājs ir aizmirsis paroli, viņam tiek dota iespēja atjaunot paroli. Šim nolūkam ir implementēta funkcionalitāte (pirmkods ir atrodams 4. pielikumā), kura uz īsu laiku periodu - uz piecām minūtēm ģenerē saiti, kuru lietotājs saņem uz savu e-pasta adresi. Saitē tiek izmantots unikāls identifikators (token), kas sastāv no šifrētas e-pasta adreses un laika zīmoga ar precizitāti līdz mikrosekundēm, kas padara to drošu. Saites parametri tiek kodēti ar base64 papildus drošības nolūkos.

Lai izvairītos no injekcijām, lietotnē tika izmantoti sagatavoti SQL vaicājumi. Kā piemēru, var paņemt preces detalizētu skatu (5.3. attēls), kur ir redzami padodamie parametri URL adresē.



5.3. attēls. **Preces detalizēts skats**

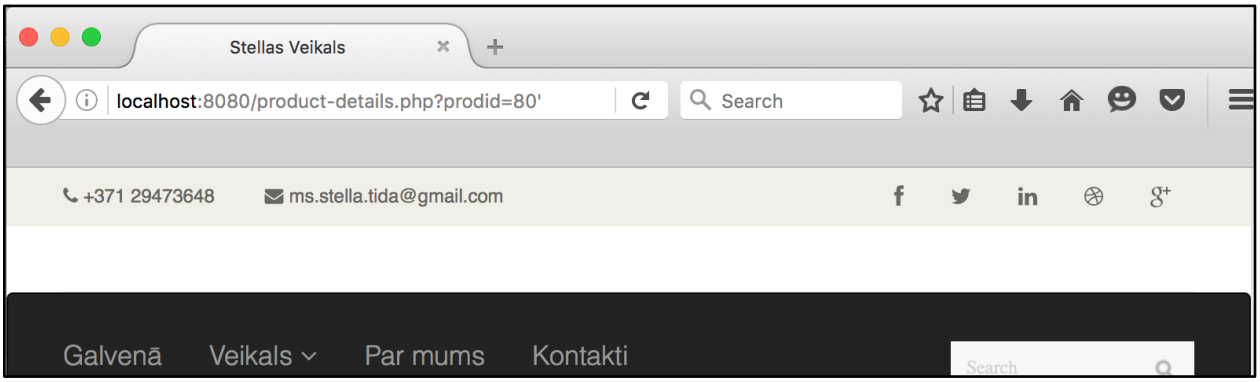
Ja kods ir nedrošs pret injekcijām un SQL vaicājumā tiek tīrā veidā padoti parametri, tas ir redzams, ja pieprasījuma beigās pievienot apostrofu (attēls 5.4.).



5.4. attēls. **Kļūda SQL pieprasījumā apostrofa dēļ**

Tādas kļūdas nozīmē potenciālas problēmas ar informācijas zaudēšanu vai bojaeju.

Pateicoties sagatavotiem vaicājumiem PHP kodā, apostrofs nerada problēmas, jo neietekmē vaicājumu (attēls 5.5.).



5.5. attēls. **Drošs pret injekcijām pieprasījums**

Sagatavotu SQL vaicājumu izmantošana nav sarežģīta (attēls 5.6), bet apjomīgu pieprasījumu gadījumā tikai atvieglotu pieprasījuma lasāmību un uztveri.

```
10
11      $statement = $DBH->prepare(
12              "SELECT * FROM products ".
13              "WHERE id = :prod_id LIMIT 1"
14              );
15
16      $statement->execute(array(
17          'prod_id' => $prodID
18          ));
19
20      $getProdInfo = $statement->fetch();
21
```

5.6. attēls. **Sagatavota SQL pieprasījuma kods**

Tā kā izstrādāta tīmekļa lietotne ir interneta veikals, ir aktuāla paslēpto lauku rediģēšana preces cenas vai citu datu rediģēšanai. Tātad, drošības nolūkos, lai caur tīmekļa pārlūkprogrammu nebūtu iespējams rediģēt cenas vērtību, no servera puses tiek pieņemts tikai preces identifikators, bet pārēja informācija tiek izvilkta no datubāzes.

Notikumu žurnālu autore neimplementēja pati, jo ir iespējams paņemt jau gatavu. Tika atrasts atvērta koda notikumu žurnāls "PHP Server Monitor" [21], kas atspoguļo notikumu vēsturi, un par noteiktām notikumu kategorijām var sūtīt paziņojumus gan pa e-pastu, gan īsziņu veidā.

REZULTĀTI

Darba “Tīmekļa lietojumprogrammu drošība” ietvaros tika izpētīti izplatīti pēc OWASP statistikas tīmekļa lietotņu uzlaušanas paņēmieni un apdraudējumu cēloņi, tika atrasti risinājumi, kas nepieļautu realizēt uzbrukumus, sastādītas drošas tīmekļa lietotnes izstrādes vadlīnijas, kuras tika pielietotas praksē, izstrādājot tīmekļa lietotni.

Tika izpētītas tādas tīmekļa lietotņu potenciālas ievainojamības, kā paslēptie lauki JavaScript formās, parametri URL adresē, kurus var mainīt gan SQL injekcijas realizēšanai, gan nesankcionētai piekļuvei paslēptiem objektiem, datu pārbaudes klienta pusē, pilnvarošanas mehānisms, skriptu injekcijas un SQL injekcijas. Tika iegūta pieredze uzlaušanas paņēmienu pielietošanā, lai labāk saprastu uzlaušanas procesu.

Uzlaušanas metodēm tika atrasti risinājumi, kas neļauj tos pielietot, vai krietni samazina efektivitāti, kas arī dažos gadījumos var palīdzēt glābt situāciju. Tika piemeklēti risinājumi gan iespējamiem tehniskiem trūkumiem, tādiem, kā nedrošs jaucējalgoritms, gan loģiskām lietotņu nepilnībām, kā, piemēram, runīgi kļūdu paziņojumi.

Apkopojot iegūtas zināšanas un idejas, tika sastādītas drošas tīmekļa lietotnes izstrādes vadlīnijas, kas tika pielietotas, izstrādājot interneta veikalu. Lietotnes izstrāde palīdzēja saprast, kāpēc jālieto iepriekš sastādītus ieteikumus, jo paralēli izstrādes procesam tīmekļa lietotne tika lauzta ar izpētītiem paņēmieniem. Šādā veidā bija acīmredzami, ka vadlīnijas strādā, paaugstinot tīmekļa lietotnes drošības līmeni.

SECINĀJUMI

Darba mērķis bija izpētīt, kā interneta vidē ir iespējams izdarīt noziedzīgas lietas, kas ir apdraudējumu cēloņi, un piemeklēt risinājumus, kā var izvairīties no uzbrukumiem, kā arī izveidot ieteikumus un vadlīnijas drošas tīmekļa vietnes izstrādei, un pielietojot iegūtās zināšanas praksē, izstrādāt praktiskus piemērus, ievērojot iepriekš noteiktās vadlīnijas. Darba gaitā tika izpētītas tādas potenciāli vājas vietas tīmekļa lietojumprogrammtūrā kā klienta pusē implementēta loģika, ienākošie parametri un lietotāju pilnvarošana. Tika piemeklēti un analizēti risinājumi, kā varētu paaugstināt tīmekļa lietotnes drošību, daži no tiem tika iekļauti izstrādātās drošas tīmekļa lietojumprogrammatūras vadlīnijās.

Veicot darba uzdevumus, tika padziļinātas zināšanas par tīmekļa lietotņu drošību, iegūta praktiska pieredze kā laušanā, bez kuras nevar pilnvērtīgi izprast tēmu, tā arī sistēmas pasargāšanā, kas arī bija galvenais mērķis.

Iegūtā pieredze palīdzēs turpmākā tīmekļa lietojumprogrammu izstrādē, to drošības līmeņa pārbaudē, izmantojot apgūtos uzlaušanas paņēmienus, un vājo vietu novēršanā, paaugstinot izstrādājamo sistēmu kvalitāti. Tika secināts, ka tīmekļa lietotnes drošības līmenis ir atkarīgs gan no tehniskiem aspektiem, gan no loģiskās implementācijas.

Tomēr, pārskatot dažus avotus, tika secināts, ka tika izpētīti tikai daži tīmekļa lietojumprogrammu drošības risinājumi, kas var glābt no vidēja urķa uzbrukumiem, taču augstākās klases datorlauži, tā saucama elite, joprojām var piemeklēt veidu, kā uzlauzt tīmekļa lietotni un izmantot to savām vajadzībām. Ne velti pasaulē tik bieži notiek skandāli, kas saitīti ar slepenās informācijas nozagšanu un publisku atklāšanu, pat no lielām valsts informācijas sistēmām. Kamēr baltās cepures speciālisti strādā pie drošības risinājumu meklēšanas un realizēšanas, melnās cepures urķi izgudro veidus, kā tos var apiet un uzlauzt.

Darba gaitā tika secināts, ka drošība ir būtiska un aktuāla tēma, kuras izpēte ir turpināma.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. Web Applications: What are They? (valoda - angļu) [tiešsaiste]. [atsauce 05.05.2016.] Pieejams internetā: http://www.acunetix.com/websitesecurity/web-applications/

2. Wikipedia: Web application (valoda - angļu) [tiešsaiste]. [atsauce 10.05.2016]. Pieejams internetā: https://en.wikipedia.org/wiki/Web_application

3. History of the WWW (valoda - angļu) [tiešsaiste]. [atsauce 10.05.2016.]. Pieejams internetā: http://www.whoishostingthis.com/resources/history-of-web/

4. A Brief History of the World Wide Web (valoda - angļu) [tiešsaiste]. [atsauce 10.05.2016.]. Pieejams internetā: http://www.whoishostingthis.com/resources/history-of-web/

5. Paul Graham. Web 2.0 (valoda - angļu) [tiešsaiste]. [atsauce 11.05.2016.]. Pieejams internetā: http://www.paulgraham.com/web20.html

6. Mail.Ru Group. Безопасность веб-приложенийю Риски и угрозы (valoda - krievu) [tiešsaiste]. [atsauce 12.05.2016.]. Pieejams internetā: https://www.youtube.com/watch?v=xAt--Bj9Vak&list=PLp2kHCN7pLe1azEbJa5ucnLpD0nGex0tg

7. CERT.LV. Biežāk sastopamie IT drošības incidenti (valoda - latviešu) [tiešsaiste]. [atsauce 15.05.2016.]. Pieejams internetā: https://www.cert.lv/section/show/46

8. Informācijas tehnoloģiju drošības likums (valoda - latviešu) [tiešsaiste]. [atsauce 15.05.2016.]. Pieejams internetā: http://likumi.lv/doc.php?id=220962

9. Krimināllikums. Vispārīgā daļa (valoda - latviešu) [tiešsaiste]. [atsauce 20.05.2016.]. Pieejams internetā: http://m.likumi.lv/doc.php?id=88966

10. OWASP Top 10 Project (valoda - angļu) [tiešsaiste]. [atsauce 18.05.2016.]. Pieejams internetā: https://www.owasp.org/index.php/Top_10_2013-Top_10

11. Dafydd Stuttard, Marcus Pinto. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, Second Edition. Wiley Publishing, Inc. 2011 (valoda - angļu).

12. Information Security: What Is A Weak Password? (valoda - angļu) [tiešsaiste]. [atsauce 18.05.2016.] Pieejams internetā: http://www.alertboot.com/blog/blogs/endpoint_security/archive/2010/09/16/information-security-what-is-a-weak-password.aspx

13. PHP programmešanas valodas dokumentācija (valoda - angļu) [tiešsaiste]. [atsauce 15.05.2016.] Pieejams internetā: http://php.net/docs.php

14. Stella Tīda. Tīmekļa vietņu drošība. 2015 (valoda - latviešu) [rokraksts].

15. OWASP: XSS (valoda - angļu) [tiešsaiste]. [atsauce 20.05.2016.] Pieejams internetā: https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

16. OWASP TOP-10: практический взгляд на безопасность веб-приложений (valoda - krievu) [tiešsaiste]. [atsauce 20.05.2016.] Pieejams internetā: https://habrahabr.ru/company/simplepay/blog/258499/

17. Modifikācijas simbolu tabula JavaScript valodā [tiešsaiste]. [atsauce 20.05.2016.] Pieejams internetā: http://ptgmedia.pearsoncmg.com/imprint_downloads/informit/learninglabs/9780133902990/graphics/03tab05.jpg

18. Prepared statements and bound parameters (valoda - angļu) [tiešsaiste]. [atsauce 19.05.2016.] Pieejams internetā: http://www.w3schools.com/php/php_mysql_prepared_statements.asp

19. Vides aizsardzības un reģionālās attīstības ministrija: Informācijas sistēmu drošības pārbaudes vadlīnijas (valoda - latviešu) [tiešsaiste]. [atsauce 25.05.2016.] Pieejams internetā: http://www.varam.gov.lv/lat/publ/met/?doc=12954

20. A simple PHP CAPTCHA script [tiešsaiste]. [atsauce 26.05.2016.] Pieejams internetā: https://github.com/claviska/simple-php-captcha

21. PHP Server Monitor. Open source tool to monitor your servers and websites [tiešsaiste]. [atsauce 26.05.2016.] Pieejams internetā: http://www.phpservermonitor.org

PIELIKUMI

1. Pielikums. Krimināllikums attiecībā uz kibernoziegumiem

241.pants. Patvaļīga piekļūšana automatizētai datu apstrādes sistēmai

(1) Par patvaļīgu piekļūšanu automatizētas datu apstrādes sistēmas resursiem, ja tas saistīts ar sistēmas aizsardzības līdzekļu pārvarēšanu vai ja tas izdarīts bez attiecīgas atļaujas vai izmantojot citai personai piešķirtas tiesības un ja ar to radīts būtisks kaitējums, —

soda ar brīvības atņemšanu uz laiku līdz diviem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu.

(2) Par šā panta pirmajā daļā paredzēto noziedzīgo nodarījumu, ja tas izdarīts mantkārīgā nolūkā, —

soda ar brīvības atņemšanu uz laiku līdz četriem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu, konfiscējot mantu vai bez mantas konfiskācijas.

(3) Par šā panta pirmajā daļā paredzētajām darbībām, ja tās izraisījušas smagas sekas vai ja tās vērstas pret automatizētu datu apstrādes sistēmu, kas apstrādā informāciju, kura saistīta ar valsts politisko, ekonomisko, militāro, sociālo vai citu drošību, —

soda ar brīvības atņemšanu uz laiku līdz pieciem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu, konfiscējot mantu vai bez mantas konfiskācijas.

243.pants. Automatizētas datu apstrādes sistēmas darbības traucēšana un nelikumīga rīcība ar šajā sistēmā iekļauto informāciju

(1) Par automatizētā datu apstrādes sistēmā esošās informācijas neatļautu grozīšanu, bojāšanu, iznīcināšanu, pasliktināšanu vai aizklāšanu vai apzināti nepatiesas informācijas ievadīšanu automatizētā datu apstrādes sistēmā, ja ar to radīts būtisks kaitējums, —

soda ar brīvības atņemšanu uz laiku līdz diviem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu.

(2) Par automatizētas datu apstrādes sistēmas darbības apzinātu traucēšanu, ievadot, pārnesot, bojājot, izdzēšot, pasliktinot, izmainot vai aizklājot informāciju, ja ar to tiek bojāta vai iznīcināta aizsardzības sistēma un radīts būtisks kaitējums, —

soda ar brīvības atņemšanu uz laiku līdz trim gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu.

(3) Par šā panta pirmajā vai otrajā daļā paredzēto noziedzīgo nodarījumu, ja tas izdarīts mantkārīgā nolūkā, —

soda ar brīvības atņemšanu uz laiku līdz pieciem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu, konfiscējot mantu vai bez mantas konfiskācijas, un ar probācijas uzraudzību uz laiku līdz trim gadiem vai bez tās.

(4) *(Izslēgta ar 13.12.2012. likumu)*

(5) Par šā panta pirmajā vai otrajā daļā paredzētajām darbībām, ja tās izdarījusi organizēta grupa vai ja tās izraisījušas smagas sekas, vai ja tās vērstas pret automatizētu datu apstrādes sistēmu, kas apstrādā informāciju, kura saistīta ar valsts politisko, ekonomisko, militāro, sociālo vai citu drošību, —

soda ar brīvības atņemšanu uz laiku līdz septiņiem gadiem, konfiscējot mantu vai bez mantas konfiskācijas, un ar probācijas uzraudzību uz laiku līdz trim gadiem vai bez tās.

244.pants. Nelikumīgas darbības ar automatizētas datu apstrādes sistēmas resursu ietekmēšanas ierīcēm

(1) Par tāda rīka (ierīces, datorprogrammas, datorparoles, pieejas koda vai līdzīgu datu) neatļautu izgatavošanu, pielāgošanu izmantošanai, realizēšanu, izplatīšanu vai glabāšanu, kurš paredzēts automatizētas datu apstrādes sistēmas resursu ietekmēšanai vai ar kura palīdzību var piekļūt automatizētas datu apstrādes sistēmai vai tās daļai nolūkā izdarīt noziedzīgu nodarījumu, —

soda ar brīvības atņemšanu uz laiku līdz diviem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu.

(2) Par tādām pašām darbībām, ja tās izraisījušas smagas sekas, —

soda ar brīvības atņemšanu uz laiku līdz pieciem gadiem vai ar īslaicīgu brīvības atņemšanu, vai ar piespiedu darbu, vai ar naudas sodu.

245.pants. Informācijas sistēmas drošības noteikumu pārkāpšana

Par saskaņā ar informācijas režīmu vai tās aizsardzību izstrādātu informācijas glabāšanas un apstrādes noteikumu vai citu informācijas datorsistēmas drošības noteikumu pārkāpšanu, ko

izdarījusi persona, kura ir atbildīga par šo noteikumu ievērošanu, ja tas bijis par iemeslu informācijas nolaupīšanai, iznīcināšanai vai bojāšanai vai ja ar to radīts cits būtisks kaitējums, —

soda ar īslaicīgu brīvības atņemšanu vai ar piespiedu darbu, vai ar naudas sodu [9].

2. Pielikums. 25 populārākas paroles pasaulē 2011. - 2015. gados

	2015	2014	2013	2012	2011
#1	123456	123456	123456	password	password
#2	password	password	password	123456	123456
#3	12345678	12345	12345678	12345678	12345678
#4	qwerty	12345678	qwerty	abc123	qwerty
#5	12345	qwerty	abc123	qwerty	abc123
#6	123456789	1234567890	123456789	monkey	monkey
#7	football	1234	111111	letmein	1234567
#8	1234	baseball	1234567	dragon	letmein
#9	1234567	dragon	iloveyou	111111	trustno1
#10	baseball	football	adobe123	baseball	dragon
#11	welcome	1234567	123123	iloveyou	baseball
#12	1234567890	monkey	admin	trustno1	111111
#13	abc123	letmein	1234567890	1234567	iloveyou
#14	111111	abc123	letmein	sunshine	master
#15	1qaz2wsx	111111	photoshop	master	sunshine
#16	dragon	mustang	1234	123123	ashley
#17	master	access	monkey	welcome	bailey
#18	monkey	shadow	shadow	shadow	passw0rd
#19	letmein	master	sunshine	ashley	shadow
#20	login	michael	12345	football	123123
#21	princess	superman	password1	jesus	654321
#22	qwertyuiop	696969	princess	michael	superman
#23	solo	123123	azerty	ninja	qazwsx
#24	passw0rd	batman	trustno1	mustang	michael
#25	starwars	trustno1	000000	password1	football

3. Pielikums. Izstrādātās tīmekļa lietotnes pirmkoda fragments. Reģistrācijas funkcija.

```
221     // user registration
        function register()
223     {
224         // checking input parameters presence
            if (!$_POST['captcha'] || !$_POST['password'] || !$_POST['password2'] ||
                !$_POST['email'] || !$_SESSION['captcha']['code'])
227         {
228             echo 'Ne visi lauki aizpildīti';
229             exit;
230         }
231
            $captcha = $_POST['captcha'];
            $password = $_POST['password'];
            $password2 = $_POST['password2'];
            $email = $_POST['email'];
236         $realCaptcha = $_SESSION['captcha']['code'];
237
238         global $DBH;
239
240         // functionality wrapped in try-catch construction for safety
241         try
242         {
243             if ($captcha !== $realCaptcha)
244             {
245                 $errorMsg = 'Nepareizs kods no bildes';
246                 echo $errorMsg;
247                 exit;
248             }
249
250             // checking email uniqueness using prepared statement for safety
251             $sql =
252                 'SELECT * FROM shop.user '.
253                 'WHERE email = :email '.
254                 'LIMIT 1;';
255
                $statement = $DBH->prepare($sql);
257             $statement->execute(array(
258                 'email' => $email
259             ));
260             $record = $statement->fetch();
261
262             // checking if email is correct, as well
263             if ((!filter_var($email, FILTER_VALIDATE_EMAIL)) || $record)
264             {
265                 $errorMsg = 'Epasts ir nepareizs vai jau ir reģistrēts';
266                 echo $errorMsg;
267                 exit;
268             }
269
```

```php
            // checking if password matches criteria
            $passwordArr = str_split($password);
            $bigLetters = 0;
            $smallLetters = 0;
            $numbers = 0;
            foreach($passwordArr as  $letter) {
                if (preg_match("/[a-z]/", $letter)) {
                    $smallLetters++;
                }
                if (preg_match("/[A-Z]/", $letter)) {
                    $bigLetters++;
                }
                if (preg_match("/[0-9]/", $letter)) {
                    $numbers++;
                }
            }

            $errorMsgArr = array();
            if (strlen($password) < 8) {
                $errorMsgArr[] = 'vismaz 8 simbolus';
            }
            if ($smallLetters < 2) {
                $errorMsgArr[] = 'vismaz 2 mazuz burtus';
            }
            if ($bigLetters < 2) {
                $errorMsgArr[] =  'vismaz 2 lielus burtus';
            }
            if ($numbers < 2) {
                $errorMsgArr[] = 'vismaz 2 ciparus';
            }

            if ($errorMsgArr) {
                echo 'Parolei jāsatur: ' . implode(', ', $errorMsgArr);
                exit;
            }

            // checking if passwords match
            if ($password !== $password2) {
            echo 'Paroles nesakrīt';
            exit;
            }

            // if no errors, recording data to database
            $sql =
                'INSERT INTO shop.user (email, password) '.
                'VALUES (:email, :password); ';

            $statement = $DBH->prepare($sql);

            // hashing password using a strong one-way hashing algorithm
            $statement->execute(array(
                'email' => $email,
                'password' => password_hash($password, PASSWORD_DEFAULT)
            ));
```

4. Pielikums. Izstrādātās tīmekļa lietotnes pirmkods. Paroles atjaunošanas funkcija.

```php
    // password renewal function
    function password_renewal_token()
    {
        global $DBH;

        $msg = 'Saites darbības laiks ir beidzies';
        if (!array_key_exists('token', $_SESSION) ||
                !array_key_exists('token_timeout', $_SESSION) ||
                !array_key_exists('token_email', $_SESSION))
        {
            echo $msg;
            exit;
        }

        if ($_SESSION['token'] !== $_POST['token']) {
            echo $msg;
            exit;
        }
        if ($_SESSION['token_timeout'] < time()) {
            echo $msg;
            exit;
        }

        $password = $_POST['password'];

        // checking if password matches criteria
        $passwordArr = str_split($password);
        $bigLetters = 0;
        $smallLetters = 0;
        $numbers = 0;
        foreach($passwordArr as  $letter) {
            if (preg_match("/[a-z]/", $letter)) {
                $smallLetters++;
            }
            if (preg_match("/[A-Z]/", $letter)) {
                $bigLetters++;
            }
            if (preg_match("/[0-9]/", $letter)) {
                $numbers++;
            }
        }

        $errorMsgArr = array();
        if (strlen($password) < 8) {
            $errorMsgArr[] = 'vismaz 8 simbolus';
        }
        if ($smallLetters < 2) {
            $errorMsgArr[] = 'vismaz 2 mazuz burtus';
        }
```

```
399             if ($bigLetters < 2) {
400                 $errorMsgArr[] =  'vismaz 2 lielus burtus';
401             }
402             if ($numbers < 2) {
403                 $errorMsgArr[] = 'vismaz 2 ciparus';
404             }
405
406             if ($errorMsgArr) {
407                 echo 'Parolei jāsatur: ' . implode(', ', $errorMsgArr);
408                 exit;
409             }
410
411
412             // if no errors, recording data to database
413             $sql =
414                 'UPDATE shop.user SET password = :password WHERE email = :email; ';
415
416             $statement = $DBH->prepare($sql);
417
418             // hashing password using a strong one-way hashing algorithm
419             $statement->execute(array(
420                 'email' => $_SESSION['token_email'],
421                 'password' => password_hash($password, PASSWORD_DEFAULT)
422             ));
423
424             echo 1;
425             exit;
426         }
427
```

```php
    // password renewal email generation
    function password_renewal_email()
    {
        global $DBH;

        $email = $_POST['email'];

        $sql =
            'SELECT * FROM shop.user '.
            'WHERE email = :email '.
            'LIMIT 1;';

        $statement = $DBH->prepare($sql);
        $statement->execute(array(
            'email' => $email
        ));

        // if no user record found, exit
        $record = $statement->fetch();
        if (!$record) {
            echo 1;
            exit;
        }

        $token = base64_encode(password_hash(
                $email.microtime(),
                PASSWORD_DEFAULT
            ));

        $_SESSION['token'] = $token;
        $_SESSION['token_timeout'] = time() + 5*60;
        $_SESSION['token_email'] = $email;

        $subject = 'Paroles atjaunināšana';
        $msg = 'Lai atjaunotu savu paroli, ' .
                'lūdzam pārvietoties pa saiti: <a href="' .
                $this->url_origin($_SERVER) .
                '/password-renewal-token.php?token=' .
                $token .
                '">the link</a>';

        // send email
        mail($email,$subject,$msg);
        echo 1;
        exit;
    }
```

DOKUMENTĀRĀ LAPA

Bakalaura darbs “Tīmekļa lietojumprogrammu drošība” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti.

Autore: _____ Stella Tīda

Rekomendēju/nerekomendēju darbu aizstāvēšanai (nevajadzīgo izsvītrot)

Vadītājs: prof., Dr.dat. Māris Vītiņš ____

Recenzents: Dr.dat. Aivars Niedrītis

Darbs iesniegts Datorikas fakultātē 30.05.2016.

Dekāna pilnvarotā persona:

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

__.06.2016. prot. Nr. ____

Komisijas sekretār_: