KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Martynas Bendorius

PAŽEIDŽIAMUMŲ TYRIMAS PRIEGLOBOS PASLAUGAS TEIKIANČIŲ SERVERIŲ APLINKOSE

Baigiamasis magistro darbas

Vadovė
Doc. dr. Ingrida Lagzdinytė-Budnikė

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

PAŽEIDŽIAMUMŲ TYRIMAS PRIEGLOBOS PASLAUGAS TEIKIANČIŲ SERVERIŲ APLINKOSE

Baigiamasis magistro darbas
Informacijos ir informacinių technologijų sauga (kodas 621E10003)

Vadovė
(parašas) Doc. dr. Ingrida Lagzdinytė-Budnikė
(data)

Recenzentas
(parašas) Dr. Dangis Rimkus
(data)

Projektą atliko
(parašas) Martynas Bendorius
(data)

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Martynas Bendorius

(Studento vardas, pavardė)

Informacijos ir informacinių technologijų sauga (kodas 621E10003)

(Studijų programos pavadinimas, kodas)

„Baigiamojo projekto pavadinimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. gegužės 13 d.

Kaunas

Patvirtinu, kad mano, **Martyno Bendoriaus**, baigiamasis projektas tema „Pažeidžiamumų tyrimas prieglobos paslaugas teikiančių serverių aplinkose“ yra parašytas visiškai savarankiškai, o 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)*

Bendorius, M. „Baigiamojo projekto pavadinimas“. Magistro baigiamasis projektas / vadovas doc. dr. Ingrida Lagzdinytė-Budnikė; Kauno technologijos universitetas, Informatikos fakultetas, Kompiuterių katedra.

Kaunas, 2016. 64 p.

SANTRAUKA

Magistro darbo tikslas – ištirti prieglobos paslaugoms (angl. *hosting*) teikti naudojamos programinės įrangos pažeidžiamumus, bei pažeidžiamumus atsiradusius dėl netinkamo sistemų konfigūravimo, ir pritaikyti saugumo sprendimus *CustomBuild* automatizuoto PĮ paketų diegimo, atnaujinimo ir konfigūravimo sistemai. Sistema pagal nutylėjimą yra naudojama kartu su *DirectAdmin* valdymo skydu, skirtu prieglobos paslaugų teikimui.

Išanalizavus konkurentų sistemas buvo pastebėta, kad nėra dedama pakankamai pastangų saugumo padidinimui, o daugiau dėmesio yra skiriama funkcionalumui. Nors konkurentų sistemose saugumas yra pakankamas vartotojo duomenų saugumui užtikrinti, tačiau nėra prevencijos priemonių, skirtų apsaugoti vartotojo aplinką, pavyzdžiui, kai vartotojas svetainei naudoja pažeidžiamą turinio valdymo sistemą. Nesiimant prevencijos priemonių nukreiptų prieš konkrečius vartotojus sistemoje, gali būti padaroma žala visai sistemai. Pavyzdžiui, siunčiant didelius kiekius brukalo iš serverio, duomenų centras be įspėjimo gali blokuoti *SMTP* prievadą ar išvis nutraukti interneto ryšio tiekimą serveriui. Pakenkti kitiems vartotojams galima ir paleidus daug resursų naudojančius kenkėjiškus procesus, kai serverio resursai yra išeikvojami. Egzistuoja ir kiti būdai, kurie yra šiame darbe yra plačiai aptariami.

Projektuojant sistemos patobulinimus nesklandumų nekilo, tačiau kai kurių tikslų įgyvendinimui buvo rašomos ir teikiamos pataisos tokiems populiariems projektams kaip *Apache*, *phpMyAdmin*, *PHP*. *CustomBuild* įskiepio, teikiančio grafinę aplinką valdymui, kūrimo metu įtrauktas papildomas funkcionalumas į patį *DirectAdmin* valdymo skydo kodą, kad būtų užtikrintas didelis interaktyvumo lygis. Kitų, darbe pasiūlytų ir aptartų algoritmų realizacijai taip pat atlikti *DirectAdmin* kodo pakeitimai. Sukurta sistema, kuri pilnai atitinka visus darbo analizės etape išsikeltus reikalavimus. Šiuo metu sistema yra naudojama daugiau nei 150 000 serverių.

Bendorius, Martynas. *Research of Vulnerabilities in Shared Web Hosting Server Environments*: *Master's* thesis in Informatics / supervisor doc. dr. Ingrida Lagzdinyte-Budnike. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: information technology security.

Key words: web hosting security, control panels, electronic mail, message systems, unsolicited electronic mail, malware.

Kaunas, 2016. 64 p.

SUMMARY

The purpose of this work is to make a research in shared web hosting sphere, analyze the vulnerabilities found in the software or the environment (configuration, permission issues) and apply solutions to the problems in *CustomBuild* software – modern and automated system for software packages installation, upgrade and configuration. System is used by default together with *DirectAdmin* control panel, used for shared web hosting automation.

Comparing the competition, it was discovered that little efforts are made for security hardening, and most of the work hours are invested into functionality of the products. Even though the security in competition is sufficient for ensuring safety of data in user environments, there is not enough of prevention done to protect user environment in cases where it could be affected, for example, when a vulnerable content management system is used for the sites. Without doing any prevention to protect user environments they could affect the whole system. For example, if huge amount of unsolicited bulk email is sent, datacenters could block *SMTP* port or disconnect the server from the Internet at all, without any compromises and notifications. It is also possible to do harm for other shared web hosting customers by starting resource hungry processes that could use all of the resources available. Other methods to affect user environments also exist, they are analyzed in this work.

There were no notable issues encountered in designing the system, however some of the objectives required to patch a few popular projects like *Apache*, *phpMyAdmin*, *PHP*. *CustomBuild* plugin, which provider graphical user interface for administration, required additional functionality in *DirectAdmin* control panel itself, to ensure a high level of interactivity. Other algorithms offered in the work also required changes in *DirectAdmin* code. The system created fulfilled all the requirements. At the moment system is used in more than 150 000 servers.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

MPM (angl. *Multi Processing Module*) – *Apache* serverio modulis, skirtas užklausų apdorojimui.

OS – operacinė sistema.

RHEL – *RedHat Enterprise* OS.

DDoS – paskirstytas atsisakymas aptarnauti (angl. *Distributed Denial of Service*).

Bootstrap – CSS karkasas.

jQuery – JavaScript biblioteka.

EOL – pasibaigęs gyvavimo ciklas (angl. *end of life*).

Bash – Unix *shell*, komandų interpretatorius ir komandinės eilutės vartotojo sąsaja.

DirectAdmin – valdymo skydas skirtas prieglobos paslaugų teikėjams.

PHP – dinaminė interpretuojama programavimo kalba.

MySQL – *SQL* duomenų bazių serveris.

CloudLinux – operacinė sistema, sukurta *CentOS* sistemos pagrindu.

CageFS – *CloudLinux* OS produktas, skirtas virtualių failų sistemų palaikymui.

LiteSpeed – žiniatinklio serveris.

Nginx – žiniatinklio serveris.

Apache – žiniatinklio serveris.

Pure-FTPd – *FTP* serveris.

ProFTPd – *FTP* serveris.

ClamAV – antivirusinė programinė įranga.

Exim – el. pašto serveris.

SMTP – paprastas pašto perdavimo protokolas.

RFI – nutolusio failo įkėlimas (angl. *Remote File Inclusion*).

FastCGI – protokolas, skirtas interaktyvių programų bendravimui su žiniatinklio serveriu.

RoundCube – el. pašto klientas.

phpMyAdmin – duomenų bazių valdymo įrankis.

SSL – kriptografinis protokolas, skirtas internete perduodamų duomenų šifruojant.

SSH – tinklo protokolas, skirtas kliento prisijungimui prie serverio *shell* aplinkos.

SpamAssassin – el. pašto filtravimo programa.

MTA – pašto serveris.

DNS – sričių vardų serveris.

TVS – turinio valdymo sistema.

DKIM – el. pašto autentifikavimo metodas, paremtas *DNS* įrašais.

SPF – *DNS* įrašas, nustatantis serverius, turinčius teisę siųsti laiškus (angl. *Sender Policy Framework*).

RBL – realaus laiko juodieji sąrašai.

ĮVADAS

Darbas priklauso Informacijos ir informacinių technologijų saugos studijų programai.

Prieglobos (angl. *hosting*) paslaugų sferoje, kaip ir kitose, vyksta nuolatinis vystymasis ir tobulėjimas. Remiantis kompanijos „NetCraft Ltd." paskelbtais duomenimis, internetinių svetainių kiekis internete jau yra perkopęs vieną milijardą, ir jų nuolat daugėja [1]. Šiais laikais yra populiaru įmonėms ar asmenims turėti savo svetainę internete, o šį siekį įgyvendinti padeda prieglobos paslaugas teikiančios įmonės. Prieglobos paslauga yra internetinių paslaugų grupė, kuri leidžia įmonėms ar asmenims skelbti turinį internete. Teikiant prieglobos paslaugas, dažniausiai suteikiama:

- el. pašto saugykla (*IMAP*/*POP3* ir *SMTP* prieiga);
- *DNS* zonų aptarnavimas ir valdymas;
- svetainių saugojimas ir valdymas;
- *FTP* prieiga;
- *MySQL* duomenų bazės saugykla ir duomenų bazės valdymas;
- prieiga naudojant žiniatinklio serverį;
- *PHP* interpretatorius failų apdorojimui;
- prieiga prie statistiką renkančių įrankių.

Kad būtų galima teikti prieglobos paslaugas, turi būti įdiegta aibė programinės įrangos, ji turi būti tinkamai sukonfigūruota, ją reikia nuolat atnaujinti ir prižiūrėti, o atsiradus naujam klientui, reikia keisti konfigūraciją. Tai yra ganėtinai sudėtingas, nestandartizuotas ir neautomatizuotas procesas.

Galimybę lengvai pradėti ir vykdyti prieglobos paslaugų verslą siūlo sukurti valdymo skydai, kurie automatizuoja daugelį procesų. Populiariausi pasaulyje valdymo skydai yra *Plesk* [2], *DirectAdmin* ir *cPanel* [3]. Jie užima didžiausią dalį rinkoje ir vyrauja tam tikruose regionuose. Pvz. *cPanel* pagrindinė rinkos dalis yra JAV, *DirectAdmin* – Europoje. Valdymo skydai leidžia naudojant interneto naršyklę kelių mygtukų paspaudimu sukurti naujo kliento paskyrą, kuris iškart turės priėjimą prie reikiamų paslaugų, tokių kaip el. pašto, duomenų bazių, *DNS* įrašų, domenų, *SSL* sertifikatų, periodinių veiksmų (angl. *cronjob*), *FTP* vartotojų valdymo ar kt.

Valdymo skydų esmė – lengvas valdymo skydo ir jo naudojamų komponentų (reikalingų paslaugoms teikti) diegimas, automatizuotas sistemos konfigūravimas, paprastas naudojimas tiek iš administravimo, tiek iš vartotojo pusės. Prieglobos paslaugų sferoje valdymo sistema yra suprantama kaip produktas leidžiantis vartotojams (klientams) valdyti jų gaunamas paslaugas vienoje vietoje.

Iš kliento pusės, valdymo skydas naudojant žiniatinklį teikia grafinėje sąsajoje pasiekiamas tokias paslaugas kaip: paskyros valdymas, *FTP* vartotojų valdymas, el. pašto vartotojų valdymas, DNS administravimas, statistika apie resursų naudojimą, *MySQL* duomenų bazių ir jų vartotojų valdymas, SSL sertifikatų valdymas, domenų administravimas ir kt.

Iš administratoriaus pusės turi būti užtikrinama aibė funkcijų, pasiekiamų per grafinę sąsają: vartotojų valdymas, *DNS* zonų administravimas, *IP* adresų valdymas it kt. Valdymo skydas turi turėti pilną ir paprastą serverio paruošimą prieglobos (angl. *hosting*) paslaugoms teikti, todėl įvairios programinės įrangos, tokios kaip žiniatinklio serverio (pvz. *Nginx*, *Apache*), el. pašto serverio (pvz. *Exim*, *Postfix*), duomenų bazių serverio (pvz. *MySQL*), FTP serverio (pvz. *ProFTPd*, *Pure-FTPd*) ir kitų diegimu, konfigūravimu ir valdymu (pvz. atnaujinimu, pakeitimu kita programine įranga), irgi rūpinasi valdymo skydai, skirti prieglobos paslaugoms teikti. Tam jie naudoja atskiras posistemes. Pavyzdžiui, valdymo skydas *DirectAdmin* naudoja posistemę *CustomBuild*.

Svetainių prieglobos paslaugas teikiančios įmonės susiduria su daugeliu grėsmių, tokių kaip:

- serveriai būna nepasiekiami dėl *DDoS* atakų;
- serveriai negali išsiųsti el. pašto laiškų, nes dėl kenkėjiškų atakų būna pakliuvę į juoduosius sąrašus (angl. *blacklist*);
- serveryje esantys vartotojai gali paleisti kenkėjiškas programas;
- vartotojai gali nuskaitinėti vienas kito failus;
- dėl programinės įrangos spragų atsiranda galimybė eskaluoti privilegijas;

- nesaugi programinės įrangos konfigūracija leidžia atlikti įsibrovimus nuotoliniu būdu;
- netinkama programinės įrangos konfigūracija leidžia nuotoliniu būdu kenkėjiškai apkrauti serverį tiek, kad jis tampa nebepasiekiamas.

Remiantis „Symantec" kompanijos metine ataskaita „Internet Security Threat Reports" [4], išleista 2016 metais:

- lyginant 2014 ir 2015 metus, asmeninių duomenų atskleidimo atvejų internete padaugėjo 85 proc.;
- bendras brukalo (angl. *spam*) kiekis tarp visų išsiunčiamų el. pašto laiškų 2015 metais siekė 53 proc., o 87 proc. iš jų turėjo bent vieną laiške paminėtą internetinę nuorodą;
- daugiau nei 75 proc. visų tinklalapių turi žinomus pažeidžiamumus, o 15 proc. jų turi kritinių pažeidžiamumų;
- žaibiškų (angl. *zero-day*) programinės įrangos pažeidžiamumų kiekis, naudojamos internetinėms paslaugoms, kuris buvo atskleistas 2015 metais, yra 54, nors 2014 metais jų buvo tik 24.

Darbo problematika ir aktualumas

Remiantis tendencijomis, kurios parodo vis didesnę saugumo svarbą internetinėms paslaugoms, darbe siekiama apibendrinti galimas saugos problemas, identifikuoti tokių problemų atsiradimą dėl naudojamos PĮ ar jos netikslios konfigūracijos pažeidžiamumų bei pateikti pasiūlymų, aprašančių kaip naudoti, konfigūruoti, diegti ir atnaujinti PĮ, kad prieglobos paslaugos būtų teikiamos saugiai.

Darbo tikslas ir uždaviniai

Darbo pagrindinis tikslas – ištirti prieglobos paslaugoms teikti naudojamos programinės įrangos pažeidžiamumus, bei pažeidžiamumus atsiradusius dėl netinkamo sistemų konfigūravimo, ir pritaikyti saugumo sprendimus *CustomBuild* automatizuoto PĮ paketų diegimo, atnaujinimo ir konfigūravimo sistemai.

Darbo uždaviniai:

- Išanalizuoti galimus pažeidžiamumus esamuose automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo įrankiuose;
- Išanalizuoti prieglobos paslaugoms teikti naudojamos programinės įrangos saugos metodus ir technologijas;
- Remiantis analizės etapo rezultatais pasiūlyti priemones ir algoritmus, reikalingus geresnei teikiamų paslaugų apsaugai;
- Realizuoti pasiūlytus patobulinimus automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo įrankyje *CustomBuild*;
- Įvertinti gautus rezultatus.

Darbo rezultatai ir jų svarba

Galutinis darbo rezultatas yra papildyta *DirectAdmin* valdymo skydo posistemė *CustomBuild,* kurioje realizuotos pasiūlytos priemonės ir algoritmai, užtikrinantys geresnę apsaugą prieglobos paslaugas teikiančių serverių aplinkose. Kadangi *DirectAdmin* yra vienas iš trijų populiariausių valdymo skydų pasaulyje, įgyvendinti sprendimai turės įtakos daugiau nei 150 000 serverių, teikiančių prieglobos paslaugas.

Dėl realizuotų sprendimų bus ne tik padidintas prieglobos paslaugas teikiančių serverių saugumas, tačiau taip pat sumažės darbo jėgos poreikis serverių administravimui. To priežastis – automatinis atakų aptikimas, jų izoliavimas, o kai kuriais atvejais (pvz. brukalo siuntimo iš serverio metu), net galutinių vartotojų automatinis informavimas apie incidentus. Šios priemonės neleis serveriams patekti į juoduosius sąrašus, o taip pat bus išvengiama duomenų centrų atliekamų veiksmų prieš prieglobos paslaugas teikiančius serverius, kuomet dėl išeinančio iš serverio brukalo, arba aptiktų kenkėjiškų failų, blokuoja *SMTP* prievadą arba nutraukia interneto ryšio teikimą.

Darbo struktūra

Darbo dokumentą sudaro:

- Įvadas, kuriame analizuojama darbo problematika, tikslai ir uždaviniai;
- Analizės dalis, kurioje nagrinėjama kuriama, bei aptariamos panašios sistemos, jų privalumai ir trūkumai, svarstomas galimas sistemos realizacijos sprendimas;
- Projektavimo dalis, kurioje detaliai aprašoma, kaip analizuoti komponentai buvo bus integruoti į *CustomBuild* programą;
- Realizacijos dalis, kurioje aprašoma kaip buvo realizuotas metodų įgyvendinimas ir kaip jie veikia;
- Eksperimento dalis, kurioje išbandyti realizuoti metodai, atliekant jų analizę;
- Rezultatų apibendrinimas ir išvados;
- Literatūros sąrašas, kuriame pateikiami naudoti literatūros šaltiniai.

1. PRIEGLOBOS PASLAUGAS TEIKIANČIŲ SERVERIŲ APLINKŲ ANALIZĖ

Analizės skyriuje analizuojami valdymo skydų posistemių, skirtų automatiniam PĮ diegimui, atnaujinimui ir konfigūravimui, pažeidžiamumai ir metodai, kurie sumažintų pažeidžiamumų riziką. Darbas orientuotas į galimus *CustomBuild* įrankio, kuris yra naudojamas *DirectAdmin* valdymo skyde, patobulinimus.

1.1. Analizės tikslas

Analizės tikslas yra ištirti populiariausių valdymo skydų, skirtų prieglobos paslaugoms teikti, sukuriamas saugumo problemas ir surasti būdus šioms problemoms spręsti.

1.2. Tyrimo objektas, sritis ir problema

Šio darbo *tyrimo objektas* – valdymo skydų posistemės, skirtos automatiniam PĮ diegimui, atnaujinimui ir konfigūravimui, *sritis* ir *problema* – galimi pažeidžiamumai.

1.3. Pažeidžiamumai ir galimi jų sprendimų būdai prieglobos paslaugas teikiančių serverių aplinkose

Kaip įvade buvo aptarta, svetainių prieglobos paslaugas teikiančios įmonės susiduria su daugeliu grėsmių, kurios didele dalimi yra susijusios su naudojamos PĮ ar jos netikslios konfigūracijos pažeidžiamumais. Tolesniuose analizės skyriuose aptariamos galimos tokio pobūdžio saugos problemos, įvardinami pasiūlymai, aprašantys kaip naudoti, konfigūruoti, diegti ir atnaujinti PĮ, kad prieglobos paslaugos būtų teikiamos saugiai. Analizė atlikta didesnį dėmesį skiriant:

a) Prevencijai prieš *DDoS* atakas (analizuojant kokia PĮ serverių aplinkose yra su šia sritimi susijusi, taip pat paliečiant tokios PĮ našumo, lygiagretumo, operatyvios atminties naudojimo aspektus, 1.3.1 – 1.3.5 skyriai)
b) Problemoms kylančioms el. pašto srityje dėl brukalo aptarti (analizuojant tiek įeinančio, tiek išeinančio į/iš serverio brukalo problemas ir prevencijos būdus, 1.3.6 – 1.3.9 skyriai)
c) Prevencijai prieš įkeliamus kenkėjiškus failus ir rizikos sumažinimui aplinkai atliekant kenkėjiškus veiksmus serveryje (analizuojant kokie būdai sukelia grėsmę priėjimui prie kitų vartotojų ar sistemos failų, kaip įkeliami kenkėjiški failai, 1.3.8, 1.3.10 – 1.3.12 skyriai)

Lygiagrečiai analizės skyriuose minėtais aspektais aptariamas trijų populiariausių valdymo skydų (*Plesk*, *cPanel* ir *DirectAdmin*) teikiamas saugumo lygis.

1.3.1. *PHP-FPM* ir *PHP-FastCGI* palaikymas

PHP-FastCGI SAPI (angl. *Server API*) teikia greitesnę sąsają nei *PHP-CGI*, tačiau jį valdyti yra sudėtinga, nes Apache turi reguliuoti tiek savo procesus, tiek *PHP*. *PHP* nuo 5.3.3 versijos pradėjo palaikyti *PHP-FPM* SAPI (angl. *Server Application Programming Interface*) [5], kurio palaikymą į *CustomBuild 2.0* nuspręsta įtraukti dėl:

- Galimybės *CloudLinux* [6] aplinkoje apriboti operatyvią atmintį ir palaikyti *CageFS* virtualią failų sistemą. Tai suteikia saugumo ir sutaupo sistemos resursų. *CageFS* suteikia virtualias izoliuotas failų sistemas, tad *PHP* procesai negali matyti kitų vartotojų failų sistemų ar jų paleistų procesų. Operatyviosios atminties apribojimas apsaugo nuo žiniatinklio serverio užtvindymo užklausomis;
- *Nginx* žiniatinklio serveris pilnai palaiko *PHP-FPM*, tad bus paruošta tinkama aplinka *PHP* palaikymui *Nginx* žiniatinklio serveryje;
- Mažesnis operatyviosios atminties naudojimas nei *mod_php*, todėl serveris gali atlaikyti daugiau užklausų atakos metu;
- Vartotojo failų sistemos izoliavimo (angl. c*hroot*) galimybės.

PHP-FPM privalumų lentelėje 1.3.1.1 pateikiamas palyginimas su *PHP-FastCGI*.

1.3.1.1 lentelė. *PHP-FPM* palyginimas su *PHP-FastCGI*

Aprašymas	***PHP-FastCGI***	***PHP-FPM***
PHP tarnybos sukūrimas: pid failas, žurnalo failas, *setsid(), setuid(), setgid(), chroot()*	✔	✔
Procesų valdymas. Galimybė grakščiu būdu (angl. *gracefully*) sustabdyti ir paleisti *PHP* procesus neprarandant sujungimų. Tai leidžia nuosekliai atnaujinti konfigūraciją ir vykdomuosius failus neprarandant sujungimų.	-	✔
IP adresų apribojimas *TCP* sujungimams	-	✔
Dinaminis procesų skaičius, atsižvelgiant į sistemos apkrovą (adaptyvus procesų išskyrimas)	-	✔
Galimybė paleisti procesus su skirtingais vartotojo ir grupės identifikatoriais, izoliuota failų sistema, skirtingais aplinkos kintamaisiais, skirtingais prievadais ir atskirais php.ini konfigūracijos failais	-	✔
Standartinės išvesties (angl. *stdout*) ir standartinės įvesties (angl. *stderr*) įrašymas į žurnalo failus.	-	✔
Galimybė greitai ir automatiškai perkrauti visus procesus nutikus atsitiktiniam *opcode* podėlio (angl. *cache*) susigadinimui operatyviojoje atmintyje	-	✔
PHP-FPM siūlomos funkcijos		
Klaidos antraštė (angl. e*rror header*)		✔
fastcgi_finish_request() (speciali funkcija užbaigti užklausai ir išvalyti visus duomenis, kai norima atlikti ilgai trunkančią operaciją (pvz. vaizdo medžiagos konvertavimas, statistikos rinkimas ir pan.))		✔
Lėtai įsivykdančių komandų žurnalo failas su dėklo pėdsakais (angl. *slowlog with backtrace*)		✔
Būsenos puslapis (panašus į *Apache mod_status*)		✔

PHP-FPM palaikymas *Apache* žiniatinklio serveryje realizuojamas naudojant mod_proxy_fcgi [7], kuris naudojant *FastCGI* protokolą ir *UDS* jungtį (angl. *Unix Domain Socket*) atliks sujungimus su *PHP-FPM* procesais. *Apache* konfigūracijoje dėl suderinamumo bus naudojamas *AddHandler* metodas. Bendras *PHP* tipų palyginimas pateikiamas 1.3.1.2 lentelėje:

1.3.1.2 lentelė. *PHP* tipų palyginimas

	Mod_php	*SuPHP*	*FastCGI*	*PHP-FPM*
Neintensyvus procesoriaus resursų naudojimas	✔		✔	✔
Neintensyvus operatyviosios atminties naudojimas	✔	✔		
Veikia vartotojo privilegijomis/vardu (ne apache/nobody)		✔	✔	✔
Didelis saugumas, virtualizuotos failų sistemos galimybė		✔	✔	✔
Didelis funkcionalumas				✔

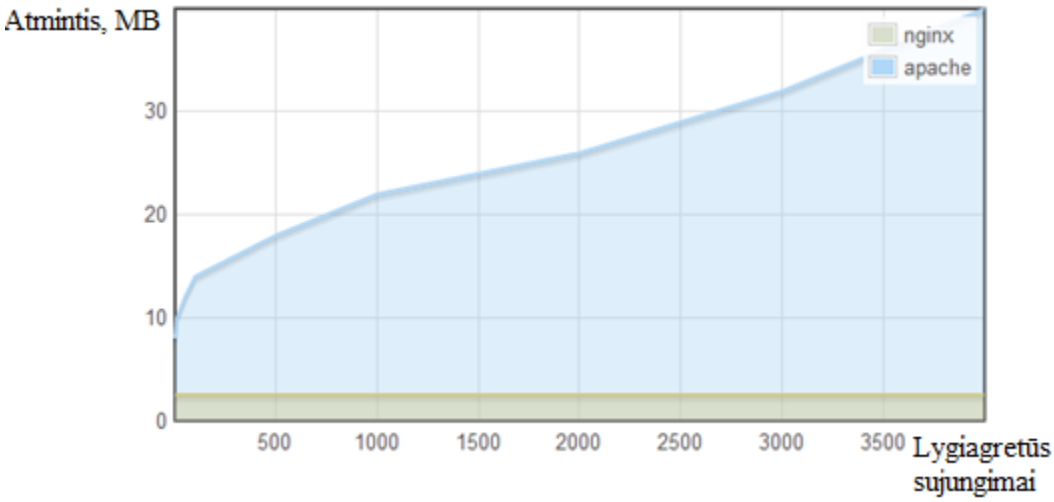
PHP-FPM SAPI (angl. *Server API*) diegimas nėra palaikomas *Plesk* ir *cPanel* valdymo skyduose, naudojant *Apache* žiniatinklio serverį;

1.3.2. *Nginx* palaikymas

Nginx yra atviro kodo žiniatinklio serveris, išleistas 2002 rugpjūčio 6 dieną [12]. *Nginx* buvo kuriamas, kad užtikrintų:

- mažą operatyviosios atminties sunaudojimą
- didelį našumą (angl. *performance*)
- didelį lygiagretumą (angl. *concurrency*).

Šie trys aspektai padeda apsisaugoti nuo atakų, kai bandoma paversti serverį nepasiekiamu dėl resursų išnaudojimo [13]. *Nginx* naudoja asinchroninį įvykiais paremtą metodą užklausoms apdoroti, priešingai nei *Apache* naudojamą į gijas ar procesus orientuotus metodus. Operatyviosios atminties naudojimo palyginimas tarp *Apache* ir *Nginx* žiniatinklio serverių pavaizduotas 1.3.2.1 paveiksle, parengtame remiantis „WebFaction.com iliustracija" [9]. *Nginx* sunaudojamos operatyviosios atminties kiekis nekinta didėjant lygiagrečių sujungimų kiekiui.



1.3.2.1 pav. Atminties kiekio sunaudojimo priklausomybės nuo lygiagrečių sujungimų palyginimas *Apache* ir *Nginx* serveriuose [9]

2016 metų balandžio mėnesio „Netcraft" ataskaitoje teigiama, kad *Nginx* yra naudojamas 13 proc. visų pasaulio tinklalapių pateikimui, o pagal „W3Techs" ataskaitą, 50,6 proc. iš 10000 pasaulio lankomiausių puslapių užklausoms apdoroti taip pat buvo naudojamas *Nginx* [8]. *Nginx* naudojamas tokiuose projektuose kaip „Facebook", „Wikipedia", „Wordpress.org", „Dropbox" ir yra pajėgus atlaikyti daugiau nei 10000 vienalaikių sujungimų (angl. *simultaneous connections*) [9] su mažu operatyviosios atminties naudojimu (~2.5MB 10000 neaktyvių *HTTP keep-alive* sujungimų [10]). Kadangi *mod_php* yra tinkamas *PHP* tipas tik *Apache* žiniatinklio serveriui, anksčiau minėtas *PHP-FPM* tipas bus naudojamas kartu su *Nginx*.

Nginx, kaip *Apache* pakeičiantis žiniatinklio serveris, nėra palaikomas *Plesk* ir *cPanel* valdymo skyduose.

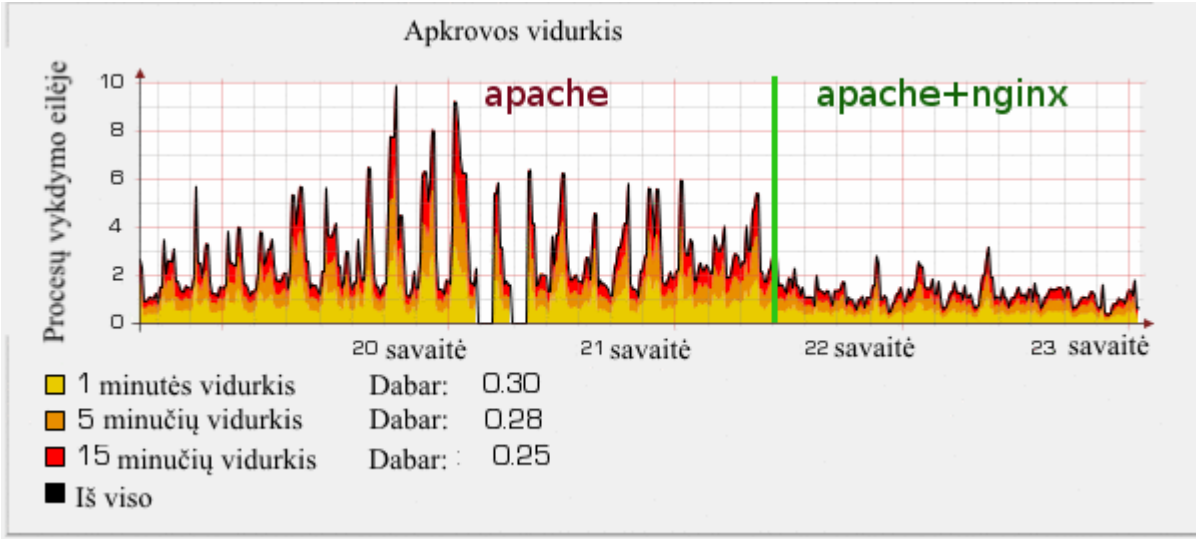
1.3.3. *Nginx* ir *Apache* žiniatinklio serverių kombinacijos palaikymas

Nginx kartu su *Apache* gali pasirodyti bereikalingu resursų švaistymu, tačiau taip nėra. Perėjimas tiesiai prie *Nginx* dėl suderinamumo (*RewriteRule* taisyklės, *.htaccess* failai) gali atnešti daugiau žalos negu naudos. Apache puikiai palaiko dinaminio turinio apdorojimą (pvz. *PHP*, naudojant *mod_php*), o *Nginx* tuo metu gali apdoroti serverį pasiekiančias užklausas ir statinį turinį. Sukombinavus juos abu, sistemos apkrovos sumažėja [14].

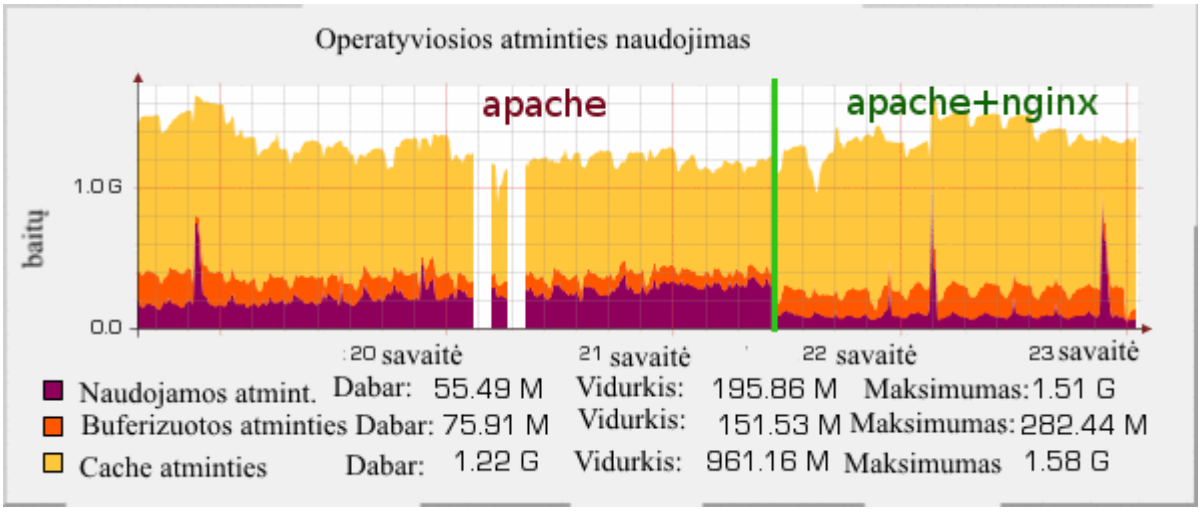
Nors erdvės keisti patį statinio turinio apdorojimą nėra, nes tenka bet kokiu atveju naudoti *fopen* sistemos kvietimus, *Nginx* privalumas yra tas, jog jis nenaudoja daug operatyvios atminties (angl. *RAM*) apdorojant daug užklausų. Taip yra dėl *Nginx* įvykiais paremtos asinchroninės architektūros. *Apache* naudoja nemažai operatyviosios atminties užklausoms apdoroti, ir atmintis nėra

atlaisvinama kol sujungimas nėra baigtas. Todėl turime didelį operatyviosios atminties sunaudojimą kol yra laukiama sujungimų užsibaigimo, o kad sujungimai užsibaigtų, reikia, kad kliento pusėje būtų gautas visas siunčiamas turinys. Šioje vietoje *Nginx* ir leidžia sutaupyti resursus, bei pakelti smarkiai didesnį sujungimų kiekį. *Nginx* gali gauti turinį iš *Apache* (įgaliotojo serverio metodu) labai greitai, nes tai vyksta vietiniame serveryje, ir nepriklauso nuo tinklo apkrovos, todėl *Apache* procesas pasibaigia greičiau, *Apache* lygiagrečiai veikiančių procesų kiekis būna mažesnis, tad mažiau sunaudojama ir operatyviosios atminties. Kadangi operatyviosios sunaudojama mažiau – lieka daugiau erdvės atlaikyti daugiau lygiagrečių sujungimų.

Bendravimui tarp *Nginx* ir *Apache* naudojamas *Apache* modulis *mod_aclr2* [11]. *X-Accel-Internal* užklausos antraštės persiuntimui ir tikrojo *IP* adreso nustatymui *Apache* naudojamas *mod_remoteip* modulis. Procesoriaus apkrovos ir operatyviosios atminties išnaudojimo palyginimas tarp *Apache* žiniatinklio serverio ir dviejų žiniatinklio serverių kombinacijos pateikiamas 1.3.3.1 ir 1.3.3.2 paveiksluose, kurie yra paremti remiantis „opennet.ru" iliustracijomis [36].



1.3.3.1 pav. Sistemos vidutinės apkrovos skirtumas naudojant tik A*pache* ir *Apache+Nginx* kombinaciją [36]



1.3.3.2 pav. Operatyviosios atminties kiekio sunaudojimo skirtumas naudojant tik A*pache* ir *Apache+Nginx* kombinaciją [36]

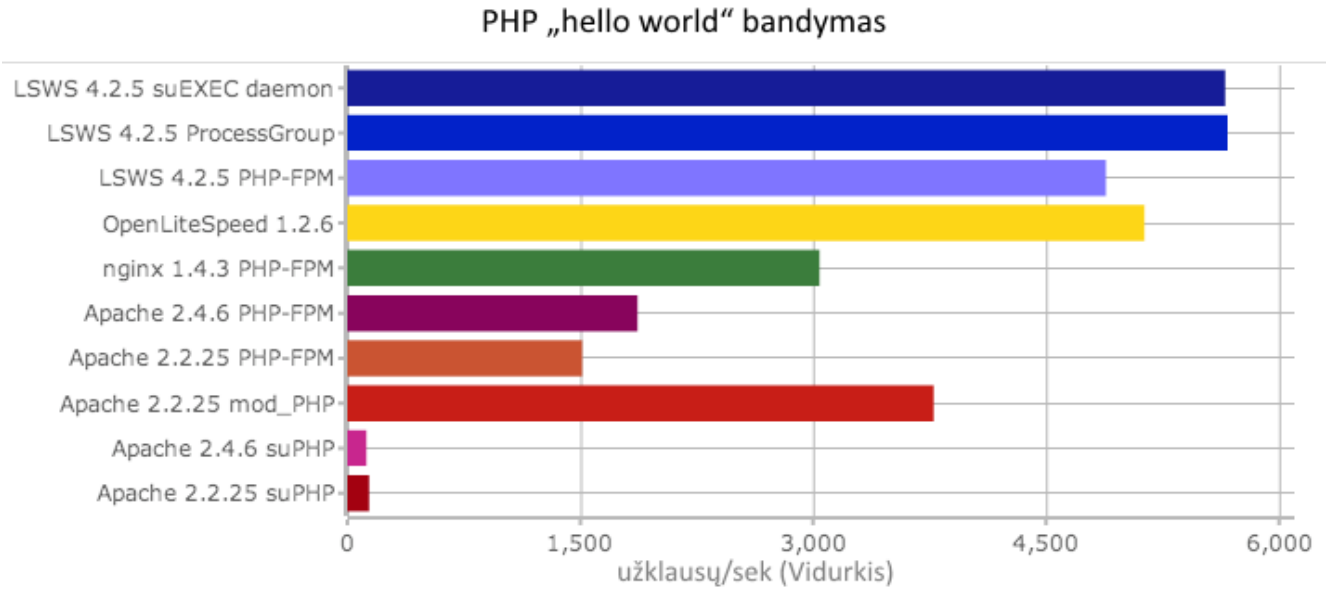
Nginx ir *Apache* žiniatinklio serverių kombinacija, kai *Nginx* apdoroja statinį turinį, o *Apache* – dinaminį, yra palaikomas tik *Plesk* valdymo skyde, bet ne *cPanel*.

1.3.4. *LiteSpeed* palaikymas

LiteSpeed yra komercinis žiniatinklio serveris, kuris pilnai pakeičia *Apache* žiniatinklio serverį [15], t.y. veikia su *Apache* konfigūraciniais failais. Pagrindiniai *LiteSpeed* pranašumai yra:

- Reikalingiausių *Apache* funkcijų palaikymas: *mod_rewrite*, *.htaccess* failai, *mod_security* ar *HTTP* autentifikavimas;
- Grafinė sąsaja konfigūravimui ir valdymui;
- Didesnis našumas, palyginus su *Apache* žiniatinklio serveriu. Taip sutaupomos lėšos, kurios kitu atveju būna išleidžiamos techninės įrangos įsigijimui, tuo pačiu tai reiškia didesnį galimą lygiagrečių sujungimų kiekį serveryje, o tai leidžia efektyviau atlaikyti *DDoS* atakas. Didesniam našumui įtaką daro šie aspektai:
    - *LiteSpeed* naudoja įvykiais paremtą (angl. *event-driven*) architektūrą ir nekuria atskiro proceso ar gijų kiekvienam *HTTP* sujungimui, kaip tai daro *Apache*. Tokiu būdu galima aptarnauti tūkstančius lygiagrečių sujungimų su minimaliu operatyviosios atminties ir procesoriaus išnaudojimu;
    - *LiteSpeed* naudoja efektyvią *PHP* sąsają *LSAPI* bendravimui tarp *LiteSpeed* žiniatinklio serverio ir *PHP* interpretatoriaus;
    - *LiteSpeed* perteikia statinį turinį greičiau, dėl didelio našumo OS branduolio (angl. *kernel*) sisteminių iškvietimų naudojimo.
- Didesnis stabilumas, palyginus su *Apache* žiniatinklio serveriu, nes yra efektyviau neutralizuojamos *DDoS* (angl. *Distributed Denial of Service*) tipo atakos ir atlaikomos didesnės apkrovos.

Sekančioje iliustracijoje (1.3.4.1 pav.) pateikiamas įvairių žiniatinklio serverių palyginimas, pagal užklausų apdorojimo kiekį per sekundę, naudojant skirtingus *PHP* tipus (angl. *PHP SAPI*). Bandymas atliktas naudojant paprastą *PHP* „hello world“ vykdomąjį failą, užimantį 13 baitų. Buvo simuliuojama 50000 užklausų skirtų 100 vartotojų, o maksimalus pastovių (angl. *keep-alive*) užklausų kiekis buvo 100. Paveikslas parengtas remiantis „litespeedtech.com” iliustracija [15].



1.3.4.1 pav. Skirtingų žiniatinklio serverių ir *PHP* tipų palyginimas pagal apdorojamų užklausų kiekį per sekundę [15]

Integruotas *LiteSpeed* žiniatinklio serverio palaikymo neseniai atsirado *cPanel* valdymo skyde. *Plesk* valdymo skydas jo nepalaiko, tačiau yra trečiųjų šalių įskiepiai *LiteSpeed* palaikymui.

1.3.5. *Apache Event MPM* palaikymas

MPM (angl. *Multi Processing Module*), galima sakyti, yra atsakingas už visą *HTTP* sesiją [16]. Nuo „klausymosi“ tinkle, užklausų priėmimo ir apdorojimo iki vaikinių procesų ir jų gijų valdymo. *Unix*/*Linux* serveriams *Apache* siūlo šiuos *MPM*: *Prefork*, *Worker* ir *Event*.

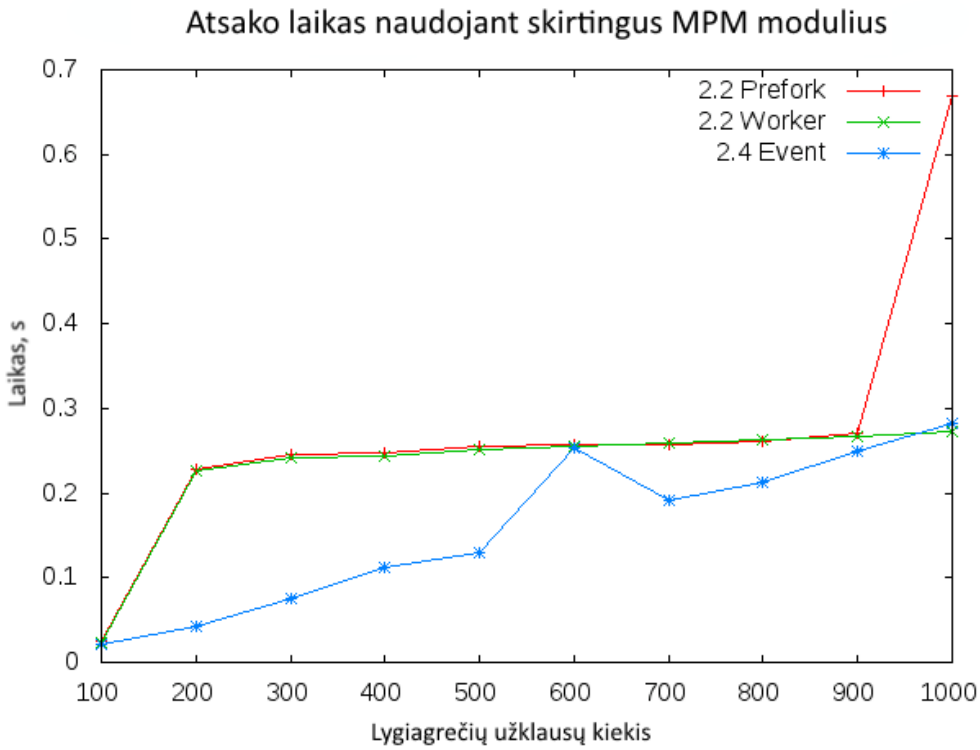
Apache startavus pradeda veikti vienas procesas, jis gali būti vadinamas „koordinatoriumi“. Šis procesas yra atsakingas už kitus procesus, kurie laukia užklausų ir jas apdoroja.

Labiausiai paplitęs *Apache MPM* yra *Prefork MPM*, kuris visiškai nenaudoja gijų, be to visas procesas yra dedikuotas kiekvienai *HTTP* užklausai. Serverį pasiekus vienai užklausai visas procesas yra atsakingas už užklausos apdorojimą tik tam vienam žmogui. Jei yra lygiagrečių sujungimų, pvz. kitas žmogus naršo tą patį failą, turi būti naudojamas visiškai atskiras procesas. Tai suteikia šiek tiek stabilumo dėl to, jog jei kažkas nutinka su tam tikra užklausa, kitos užklausos nėra paveikiamos, nes jos yra aptarnaujamos kitų procesų.

Prefork naudojimas yra privalomas, jei naudojami veikimo gijose nepalaikantys *Apache* moduliai (pvz. *mod_php*). *Mod_php* problemą galima apeiti naudojant *PHP* kaip atskirą procesą, pvz. naudoti *PHP-FPM*.

Jeigu susiduriama su dideliais kiekiais lygiagrečių (angl. *concurrent*) užklausų, tai gali sunaudoti milžiniškus kiekius sistemos resursų [17], nes kiekvienas procesas turi veikti kaip atskiras *Apache* vienetas (angl. *instance*). Tai reiškia, jog procese veikia ir visi moduliai, be to tokie procesai yra kuriami kiekvienai užklausai, o tai sunaudoja daug operatyviosios atminties. Šiai problemai spręsti buvo sukurtas *Worker MPM*, kuris naudoja gijas, tad ir sutaupo operatyviosios atminties vykstant lygiagretiems sujungimams. Naudojant *Worker MPM* yra paleidžiama mažiau procesų, nes nereikia kiekvienai užklausai apdoroti kurti vis naujo proceso. *Worker MPM* sukuria gijas jau esančiuose procesuose, tokiu būdu vienas procesas apdoroja skirtingus sujungimus. Nauji sujungimai *Worker MPM* paprasčiausiai turi palaukti kol bus pasiekiama gija, o ne procesas, kaip naudojant *Prefork MPM*.

Event MPM yra struktūriškai labai panašus į Worker. Stabili versija pasirodė kartu su *Apache 2.4*. Esminis skirtumas tarp *Worker* ir *Event MPM* yra tas, jog *Event MPM* naudoja dedikuotą giją *keep-alive* sujungimų apdorojimui, ir perduoda užklausas vaikiniams procesams tik tada, kada užklausa buvo gauta, tokiu būdu leidžiant gijoms atlaisvinti atmintį nedelsiant kai tik užklausa yra apdorota, nepriklausomai nuo faktinio *HTTP* sujungimo, kuris yra valdomas tėvinio proceso. Kitaip tariant, *Event MPM* gija yra priskiriama tik užklausos apdorojimui, o ne visam *HTTP* sujungimui. Naudojant *Worker MPM* gija būtų priskirta sujungimui ir būtų veikianti nepriklausomai nuo to ar užklausa yra apdorojama, ar ne. Kadangi naudojant *Event MPM* gija yra atlaisvinama iškart po užklausos apdorojimo, ji gali būti iškart naudojama kitoms užklausoms, o tai reiškia, jog bus naudojama mažiau gijų nei *Worker MPM*. Skirtingų *MPM* modulių atsako laikas pagal lygiagrečių užklausų kiekį pateikiamas 1.3.5.1 paveiksle, parengtame remiantis „noqqe.de” duomenimis [37].



1.3.5.1 pav. Atsako laikas naudojant skirtingus *Apache MPM* modulius [37]

Apache 2.4 versijoje atsirado galimybė dinamiškai parinkti *MPM*. T.y. nereikia perkompiliuoti Apache tam, kad pasikeistų *MPM*. Nuo šiol jį galima pakeisti ir užkrauti kaip dinaminį *DSO* modulį, naudojant *LoadModule* direktyvą. Tai leidžia automatiškai parinkti *MPM* pagal naudojamą *PHP* tipą.

Event Apache MPM yra siūlomas su *cPanel* valdymo skydu, bet ne *Plesk*.

1.3.6. Išeinančio iš serverio brukalo prevencija

Nepageidaujami el. laiškai (angl. *spam*) ne tik kenkia verslui ir vartotojams, tačiau kartu sudaro ir reikšmingus srautus bendrame interneto tinkle. Remiantis tarptautinės programinės įrangos saugos grupe „Kaspersky Lab" [18], antrąjį 2015 metų ketvirtį net 53.4 proc. bendro el. laiškų srauto sudarė nepageidaujami el. laiškai.

Egzistuoja daugelis skirtingų būdų siųsti el. laiškams, tačiau vienas populiariausių – brukalo siuntimas per pažeidžiamas svetaines tinklalapių prieglobos paslaugas (angl. *hosting*) teikiančių įmonių serveriuose. To priežastis yra ta, kad net 86 proc. visų tinklalapių turi bent vieną rimtą saugumo spragą, remiantis „2015 Website Security Statistics Report" ataskaita, kurią parengė „WhiteHat Security" [19]. Ataskaitoje taip pat teigiama, kad dažniausiai svetainėse egzistuoja daugiau negu vienas pažeidžiamumas. Kibernetinio ir duomenų saugumo produktų kūrėjas „Imperva" patvirtina „WhiteHat" kompanijos teiginius savo ataskaitoje "Web Apllication Attack Report #5" [20], papildomai pabrėždamas 24 proc. padidėjimą nuotolinių failų įterpimo atakoms (angl. *Remote File Inclusion – RFI*), ir „WordPress" įvardinimą kaip dažniausiai atakuojamą turinio valdymo sistemą (angl. *Content Management System – CMS*). Būtent *RFI* atakos ir yra dažniausiai naudojamas metodas įkelti kenkėjiškiems failams į serverio, kuriame saugomas tinklalapis, failų sistemą.

Pasaulyje pagrinde yra kovojama su įeinančiu brukalu. Tam naudojamos įvairios metodikos, įskaitant el. laiškų filtravimą naudojant populiarius atviro kodo sprendimus kaip *SpamAssassin* [21], pilkuosius sąrašus (angl. *greylisting*), juoduosius sąrašus (angl. *blacklists*), filtravimą *SMTP* metu. Taip pat sutelktas dėmesys ir į tyrimus įeinančio brukalo srityje. Nors minėti metodai padeda sumažinti gaunamų nepageidaujamų el. laiškų kiekį, ne visi pranešimai yra sėkmingai atpažįstami kaip brukalas, ypač kai siunčiančio serverio *IP* yra geros reputacijos ir tinkamai sukonfigūruotas (egzistuoja *DKIM* [22], *SPF* [23], *DMARC* [24] įrašai), o dažniausiai tokie ir yra tinklalapių prieglobos (angl. *hosting*)

paslaugas teikiantys serveriai. Todėl pažeidžiami tinklalapiai ir yra patrauklūs brukalo siuntimui. Be to, įeinančio brukalo filtravimas nesumažinta bendro brukalo kiekio globaliuose tinkluose.

Kad būtų sumažintas nepageidaujamų el. laiškų kiekis globaliame tinkle, turi būti naudojamos išeinančio iš serverio brukalo prevencijos priemonės. Tinklalapių prieglobos paslaugas teikiančioms įmonėms ir galutiniams vartotojams tai suteikia ir papildomos naudos: serverių *IP* adresai nepatenka į juoduosius serverių sąrašus, išlaikoma gera serverių reputacija, mažiau el. laiškų serveryje patenka į el. pašto laiškų eilę. Be to, duomenų centrai dažniausiai turi griežtas paslaugų tiekimo sutartis dėl brukalo, o pažeidus taisykles (aptikus išeinantį brukalą), būna imamasi veiksmų, kad būtų apsaugota viso duomenų centro tinklo gera reputacija. Veiksmai, kuriuos taiko duomenų centrai, gali būti:

- *IP* adreso blokavimas (angl. *null route*) [25], kas turėtų pasekmes visų servisų pasiekiamumui, įskaitant HTTP, FTP, DNS ir kt.;
- *SMTP* prievado (angl. *port*) 25 blokavimas [26], kuomet nebegalima naudotis *SMTP* ir sutrinka įprastų laiškų pristatymas.

Vienas populiariausių metodų išeinančio brukalo prevencijai tinklalapių prieglobos paslaugas teikiančiuose serveriuose yra išsiunčiamų laiškų kiekio ribojimas per tam tikrą laiko tarpą. Šį metodą naudoja ir du populiariausi tinklalapių prieglobos paslaugoms teikti skirti valdymo skydai *cPanel* [27] ir *Plesk* [28]. Šis metodas yra iš dalies efektyvus, nes turi griežtus limitus, tačiau susiduriama su šiais trūkumais:

- jei vartotojas pasiekia išsiųstų el. laiškų limitą neišsiuntus nei vieno nepageidaujamo el. laiško, jis nebegali išsiųsti daugiau laiškų nepasibaigus laiko periodui, kuriam ribojamas laiškų kiekis;
- jei vartotojas siunčia nepageidaujamus el. laiškus, jis jų gali per nustatytą laiko tarpą sėkmingai išsiųsti tiek, kiek numatyta limitų nustatymuose;
- net jei administratoriams yra pranešama, kad kažkuris vartotojas pasiekė jam numatytą el. laiškų limitą per tam tikrą laiko tarpą, tai reikalauja sistemų administratorių peržiūrėti incidentą, nustatyti ar buvo siunčiami nepageidaujami laiškai ir atsižvelgus į išvadas blokuoti vartotoją arba ne, šis procesas nėra automatizuotas ir reikalauja papildomo darbo laiko.

Kitas metodas siūlomas *cPanel* valdymo skydo yra išeinančių laiškų filtravimas naudojant *SpamAssassin* [29], tačiau dėl didelio resursų naudojimo, daug klaidingai aptikto (angl. *false positive*) brukalo ir nepakankamos apsaugos, pats gamintojas nerekomenduoja naudoti šio apsaugos būdo tinklalapių prieglobos paslaugas teikiančiuose serveriuose.

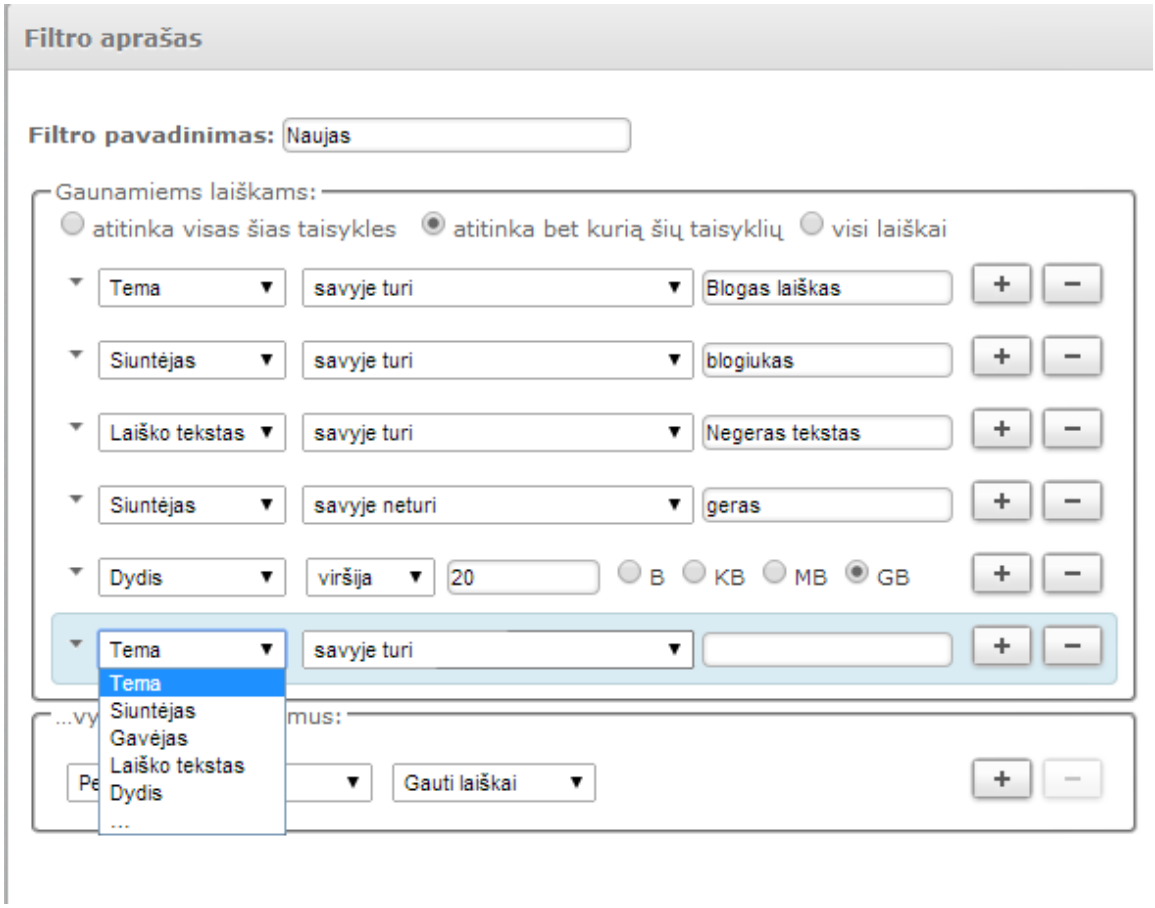
Sprendimas išeinančio iš serverių brukalo prevencijai galėtų būti išsiųstų el. laiškų į neegzistuojančias el. pašto dėžutės kiekio ribojimas per tam tikrą laiko tarpą ir automatinis *PHP* scenarijų analizavimas, iš kurių buvo išsiųsti laiškai. Šie du sprendimai turėtų užtikrinti nepageidaujamus el. laiškus siunčiančių paskyrų blokavimą serveriuose. Kad sprendimas būtų automatizuotas, reikia atlikti skaičiavimus, kiek yra išsiunčiama laiškų iš atitinkamų paskyrų į neteisingus ar neegzistuojančius el. pašto adresus. Nustačius kiek neteisingų el. pašto adresatų gali būti panaudojama per valandą, reikėtų užblokuoti paskyrą tol, kol nebus pakeistas jos slaptažodis. Jei siunčiama naudojant *PHP* scenarijus, tuomet būtų galima blokuoti siuntimą iš atitinkamo pilno kelio iki failo tol, kol nebus pašalintas brukalo šaltinis

1.3.7. *Sieve* kalbos palaikymas *Dovecot IMAP* serveryje (*Pigeonhole*)

Pigeonhole yra projekto pavadinimas, kuris prideda *Sieve* kalbos (RFC 5228) ir *ManageSieve* protokolo (RFC 5804) palaikymą *Dovecot IMAP* serveriui [30]. *Sieve* kalba yra naudojama nurodyti kaip el. paštas turi būti apdorojamas, tai atliekama naudojant *Sieve* scenarijus, kuriuos vartotojai gali tinkinti (angl. *customize*). Juose nurodoma kaip laiškai turi būti pristatomi, t.y. ar jie turi būti peradresuojami, ar saugomi specialiuose aplankuose. Nepageidaujami laiškai gali būti sunaikinti arba atmesti, o kai vartotojas neegzistuoja, *Sieve* interpretatorius gali nusiųsti automatinį atsakymą. Kalba naudojama *Sieve* scenarijams rašyti yra paprasta, išplečiama ir nepriklausoma nuo sistemos, be to ji yra saugi, nes kitaip nei dauguma kitų el. pašto filtravimo scenarijų kalbų, *Sieve* neleidžia vartotojams

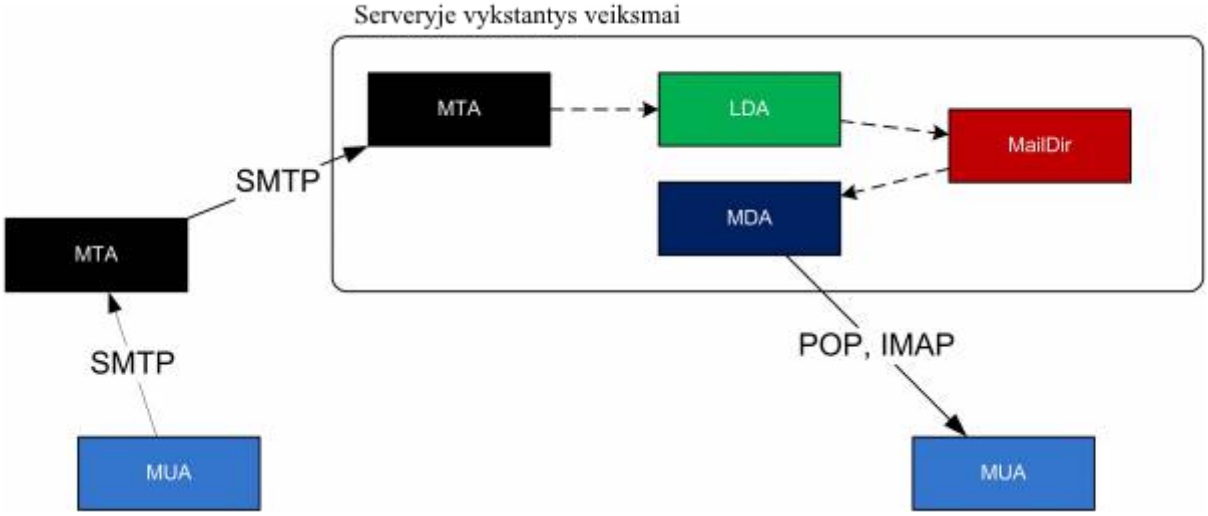
vykdyti savavališkų programų, o tai yra ypač naudinga apsaugai nuo pilno priėjimo prie pašto saugyklos. Kuriant *Sieve* kalbą esmė buvo sukurti ją tokią, jog būtų neįmanomas sudėtingesnių ir grėsmingų struktūrų vykdymas, todėl ji skirta rašyti paprastiems el. pašto filtrams.

Naudojant *ManageSieve* protokolą vartotojai gali įkelti savo *Sieve* scenarijus nuotoliniu būdu, be tiesioginės prieigos prie failų sistemos, t.y. nėra reikalo naudoti *FTP* ar *SCP* protokolų. Dėl papildomo stabilumo *ManageSieve* serveris visada įsitikina, jog įkelti scenarijai yra teisingi, tokiu būdu išvengiama klaidų el. pašto pristatyme. *Pigeonhole* projektas suteikia *Sieve* palaikymą kaip įskiepis *Dovecot* ir yra vadinamas *LDA* (angl. *Local Delivery Agent*), o *ManageSieve* protokolas yra pasiekiamas kaip atskira tarnyba, šalia *POP3* ir *IMAP* tarnybų ir leidžia konfigūruoti el. pašto apdorojimą. Konfigūravimo galimybės parodytos paveiksle 1.3.7.1.



1.3.7.1 pav. *ManageSieve* grafinė sąsaja *RoundCube* pašto programoje

Sieve palaikomas ir el. pašto klientuose kaip „Mozilla Thunderbird" ar „Kmail". Veikimo schema pavaizduota 1.3.7.2 paveiksle, kuris parengtas remiantis „queret.net" iliustracija [38]:



1.3.7.2 pav. Veikimo schema. Pigeonhole vykdomas LDA dalyje [38]

Paveikslo 1.3.7.2 terminų paaiškinimas:

- *MUA* (angl. *Mail User Agent*) – pašto klientas, pvz. „Mozilla Thunderbird", „Microsoft Outlook" ar „Kmail";
- *MTA* (angl. *Message Transfer Agent*) – pašto serveris, šiuo atveju *Exim*, atsakingas už laiškų siuntimą ir priėjimą iš kitų serverių. Teikia *SMTP* paslaugą (angl. *Simple Mail Transfer Protocol*, liet. *paprastas pašto perdavimo protokolas*);
- *LDA* (angl. *Local Delivery Agent*) – atsakingas už laiškų išsaugojimą jiems skirtuose kataloguose, po to kai MTA juos apdoroja, šiuo atveju už tai atsakingas *Dovecot LDA*. Tą patį darbą gali atlikti ir MTA;
- *Maildir* – saugoja kiekvieną laišką atskirame faile;
- *MDA* (angl. *Mail Delivery Agent*) teikia *IMAP*/*POP3* paslaugas, šiuo atveju *Dovecot*.

Sieve kalbos palaikymas el. pašto filtravimui *Dovecot IMAP* serveryje neteikiamas kartu su *cPanel* valdymo skydu, tačiau teikiamas kartu su *Plesk*.

1.3.8. Automatinis kenkėjiškų failų skanavimas *FTP* ir *PHP* įkėlimuose

Prieglobos (angl. *hosting*) paslaugas teikiančios įmonės labai dažnai susiduria su kenkėjiškų failų sukeltais padariniais, kaip:

- Neprivilegijuota prieiga prie sistemos failų;
- Brukalo (angl. *spam*) siuntimas elektroniniu paštu panaudojant *PHP mail()* funkciją;
- Konfidencialių duomenų atskleidimas.

ClamAV antivirusinė programa turi sąsają pateikti failams skanuoti, tad turint reikiamus įrankius atsiranda galimybė pateikti antivirusinei programai įkeliamus failus, o gavus atsakymą, kad failas yra infekuotas – neleisti jo saugoti. *FTP* serveriui *ProFTPd* yra sukurtas modulis *mod_clamav*, kuris bendrauja su *ClamAV* procesu naudojant *TCP* sąsają. *Pure-FTPd FTP* serveris siūlo galimybę įvykdyti bet kokį vykdomą failą po failų įkėlimo, tad žinant įkeliamų failų vardus galima po jų įkėlimo nurodyti *ClamAV* procesui patikrinti jų turinį, o jei turinys kenkėjiškas – pašalinti failą iš sistemos. *PHP* neturi reikiamo funkcionalumo vykdyti veiksmus įkeliamiems failams, todėl tam bus pasitelkiamas *PHP* įskiepis *suhosin*, kuris šią galimybę suteikia, ir, panašiai kaip *Pure-FTPd FTP* serveris, leidžia naudoti vykdomuosius failus *PHP* įkėlimams, tačiau tikrinimo procesas, priešingai nei *Pure-FTPd* atveju, vyksta įkėlimo metu, o ne po jo.

Automatinio kenkėjiškų failų skanavimo *FTP* ir *PHP* įkeliamuose failuose galimybės nesiūlo nei *Plesk,* nei *cPanel* valdymo skydai.

1.3.9. Įeinančio į serverį brukalo blokavimas *SMTP* metu

Daugelis brukalo filtravimo technologijų skanuoja laiškus, kurie patenka į el. pašto pristatymo programos (angl. *MTA*) eilę, tačiau nėra efektyvu skanuoti visus įeinančius laiškus ir juos traukti į el. pašto eilę laiškų pristatymui. Dalis brukalo siuntėjų nenaudoja tinkamos konfigūracijos, tad yra keletas išskirtinų bruožų, kuriuos galima tikrinti *SMTP* metu [31]. Brukalo siuntėjas dažnai neturi:

- *rDNS* (angl. *reverse DNS*), lietuviškas atitikmuo – atvirkštinė paieška, *IP* adresui, iš kurio siunčiami laiškai. Tai yra atvirkštinis procesas *DNS* paieškai, kuris randa *IP* adresui priskirtą domeno vardą;
- *SPF* (angl. *Sender Policy Framework*), lietuviškas atitikmuo – siuntėjo strategijos sistemos, *DNS* įrašas būna netinkamai sukonfigūruotas arba išvis neegzistuoja. Jei domenas neturi tinkamo *SPF* įrašo, tuomet jis gali būti naudojamas kenkėjiškų ar brukalo laiškų siuntimui iš nutolusių serverių;
- *DKIM* (angl. *DomainKeys Identified Mail*), lietuviškas atitikmuo – domeno raktu tvirtinami laiškai, neegzistuoja. *DKIM* laiškų pasirašymas suteikia galimybę padidinti siunčiamų laiškų patikimumo rodiklį.

Taip pat siuntėjai dažnai būna pakliuvę į juoduosius sąrašus, tad *RBL* (angl. *Real-time Blackhole List*) filtrų naudojimas taip pat padeda sulaikyti įeinantį brukalą.

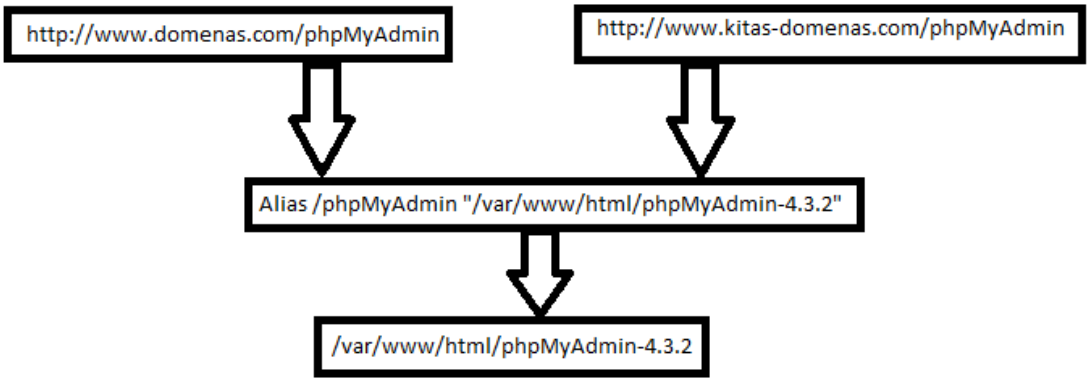
Panaudojus išskirtus brukalo siuntėjo bruožus, galima *SMTP* metu patikrinti ar siuntėjas atitinka reikalavimus, ar ne. Vieno reikalavimo neatitikimas suteiktų tam tikrą kiekį taškų, o pasiekus maksimalų jų skaičių el. pašto laiškas būtų iškart atmestas, dar net nepatekęs į laiškų eilę.

Laiškų atmetimo *SMTP* metu galimybės nesiūlo nei *cPanel,* nei *Plesk* valdymo skydai.

1.3.10. Saugus *PHP* vykdymas

Prieglobos paslaugas teikiančios įmonės dažnai nukenčia dėl *PHP* pažeidžiamumų, nes yra galimybė vartotojams nuskaityti failus, prie kurių priėjimo jie neturėtų turėti. Nors ir būna naudojama *open_basedir* apsauga, tačiau yra daugybė būdų jai apeiti. Vienas dažniausių būdų yra *exec(), shell_exec()* ir *system()* funkcijų naudojimas *PHP* programose, tad reikėtų suteikti galimybę šių funkcijų naudojimą apriboti.

Sekanti saugumo spraga yra susijusi su *Apache SuExec* moduliu. Normaliam *PHP-FastCGI* veikimui *SuExec* modulis yra reikalingas, tačiau dėl jo apribojimų yra rizikuojama saugumu. Modulio konfigūracija negali būti atliekama *Apache* „<Directory>” kontekste, tad jis yra konfigūruojamas „<VirtualHost>” kontekste. Kadangi prieglobos paslaugų teikėjai dažnai siūlo įvairias el. pašto programas, *phpMyAdmin* produktą, naudojamą *MySQL* duomenų bazės valdymui ir kitas paslaugas, jie failus saugoja ne kiekvieno vartotojo namų kataloge, o bendroje srityje, tarkime /var/www/html. Kad prieiga būtų galima per vartotojo domeną, reikia naudoti *Alias* direktyvą *Apache* konfigūracijoje. *Alias* veikimo schema pavaizduota 1.3.10.1 paveiksle:



1.3.10.1 pav. *Alias* direktyvos veikimas *Apache* žiniatinklio serveryje

Pagal 1.3.10.1 paveikslą, domenas.com ir kitas-domenas.com priklauso skirtingiems vartotojams, tad *PHP* yra vykdomas domeno vartotojo vardu, kaip parodyta 1.3.10.2 paveiksle.

```
<VirtualHost 127.0.0.1:80 >
        ServerName www.test.com
        ServerAlias www.test.com test.com
        ServerAdmin webmaster@test.com
        DocumentRoot /home/test/domains/test.com/public_html
        <IfModule !mod_ruid2.c>
                SuexecUserGroup test_vartotojas test_vartotojas
        </IfModule>
        .......
</VirtualHost>
```

1.3.10.2 pav. „SuexecUserGroup" direktyvos veikimas

Konfigūracijai veikiant pagal 1.3.10.2 paveiksle aprašytą logiką, naršant http://www.test.com/phpMyAdmin, efektyvus vartotojo *ID*, kurio teisėmis vykdomas *PHP*, būtų test_vartotojas. Tai reiškia, kad test_vartotojas privalo turėti skaitymo (kai kuriose žiniatinklio aplikacijose kaip el. pašto programoje ir rašymo) teises prie tam tikrų failų. Tokiu būdu suteikiama galimybė bet kuriam vartotojui turėti priėjimą prie failų, kuriuose gali būti el. pašto paskyrų duomenys (el. pašto adresai, konfigūracija), matyti kai kurių duomenų bazių slaptažodžius ir t.t. Kad viso to išvengti, reikia turėti galimybę „SuexecUserGroup" nustatyti „<Directory>" kontekste, kaip pavaizduota 1.3.10.3 paveiksle.

```
<Directory /var/www/html>
    SuexecUserGroup saugus_vartotojas saugus_vartotojas
</Directory>
```

1.3.10.3 pav. Saugus „SuexecUserGroup" direktyvos veikimas

Toks konfigūracijos metodas nėra numatytas *Apache* žiniatinklio serveryje, todėl jo negalima naudoti. Reikėtų parašyti *Apache* kodo patobulinimą (angl. *patch*) šiai problemai išspręsti. Kol kas ši problema nėra išspręsta nei viename valdymo skyde.

1.3.11. Saugus *Cronjob* periodinių užklausų vykdymas

Periodinių užklausų (*Cronjob*) vykdymas yra plačiai paplitęs tarp prieglobos paslaugų naudotojų, tačiau šios užklausos yra vykdomos vartotojo vardu, aplinkoje, kuri nėra izoliuota (angl. *chroot*). Tai reiškia, kad vartotojas turi galimybę lengvai paleisti kenkėjiškus procesus (pvz. brukalo siuntimo programą, *DDoS* atakas vykdančią programėlę, nuskaityti kitų vartotojų failus dėl netinkamų prieigos teisių ir kt.). Net globaliai apsaugojus *PHP Apache* žiniatinklio serverio konfigūracijoje, vartotojas gali pakeisti vykdomuosius *PHP* nustatymus, nes užklausos nebūtų vykdomos per žiniatinklio serverį. Tam atlikti pakanka naudoti „-c" *PHP* vykdymo vėliavėlę ir nurodyti savo sukurtą *PHP* konfigūracijos failą.

Kad būtų užkirstas kelias OS esančių vykdomųjų failų naudojimui, bei kitų aplankų skaitymui, reikia, kad vartotojo aplinka būtų izoliuota. Tikslo pasiekimui reikia sukurti tarpinę programėlę, per kurią būtų vykdomos užklausos (angl. *wrapper*). Kiekvieno periodinių užklausų failo viršuje galima pakeisti *SHELL* aplinką naudojant „SHELL=/path/to/chroot_shell_binary". Vartotojų virtualių failų sistemoms sukurti galima panaudoti *bind mounts* [32] funkcionalumą, išvengiant vykdomųjų falų dubliavimo vartotojų namų kataloguose. Tokiu būdu visi vykdomieji failai, kuriuos būtų galima naudoti izoliuotoje aplinkoje, galėtų būti saugomi vienoje disko vietoje, taip sutaupant disko vietos, palengvinant virtualios failų sistemos administravimą ir įrankio, leidžiančio lengvai pridėti ar pašalinti naujus vykdomuosius failus, kūrimą.

Įrankio kūrimą, kuris nukopijuos reikiamus vykdomuosius failus į virtualią failų sistemą, apsunkina įvairių bibliotekų naudojimas, kurios yra būtinos tų vykdomųjų failų vykdymui. Tačiau šią problemą galima bandyti išspręsti naudojant operacinės sistemos įrankį *ldd* (angl. *List Dynamic Dependencies*), kuris atspausdina visas vykdomojo failo naudojamas bibliotekas, tad galima apdoroti *ldd* išvestį ir nukopijuoti reikiamas bibliotekas į virtualią failų sistemą.

1.3.12. Saugi *CustomBuild* grafinė sąsaja su galimybe iškviesti automatines saugumo funkcijas

CustomBuild grafinė sąsaja kuriama kaip *DirectAdmin* valdymo skydo įskiepis (angl. *plugin*). Tam panaudojama *PHP* programavimo kalba, kuri yra viena populiariausių programavimo kalbų saityno turiniui kurti. *Plesk* ir *cPanel* valdymo skydai, teikiantys atnaujinimo galimybę per žiniatinklį, vykdo užklausas privilegijuoto „root“ vartotojo vardu. Tai nėra saugu, nes jei būtų atrasta saugumo spraga, neprivilegijuotas vartotojas įgytų *root* teises. Šiai problemai spręsti *DirectAdmin* valdymo skyde grafinė sąsaja veiks neprivilegijuoto vartotojo vardu, o tik reikiamos komandos bus perduodamos tarpininkaujančiai *C* programėlei (angl. *wrapper*), kuri bus vykdoma root vartotojo vardu, tačiau galimų komandų aibė bus apribota, o pati tarpininkaujanti *C* programa galės iškviesti tik *CustomBuild* scenarijaus failą. Tokiu atveju net ir atradus saugumo spragą būtų galima vykdyti tik ne kenkėjiškas komandas, tokias kaip programinės įrangos atnaujinimo, o vykdymo teisės bus apribotos *CustomBuild* scenarijumi.

Pateikimui naršyklėje bus naudojama *jQuery JavaScript* biblioteka, *HTML* kompiuterinė žymėjimo kalba, *BootStrap* karkasas grafinei sąsajai. Saugumo funkcijas bus galima iškviesti vieno mygtuko paspaudimu (pvz. tam, kad įdiegti ir sukonfigūruoti aplikacijų lygio ugniasienę, pakaks paspausti mygtuką „Install ModSecurity“).

1.4. Naudojamų saugos priemonių populiariausiuose valdymo skyduose apibendrinimas

Valdymo skydai prieglobos paslaugų teikėjams yra reikalingi pilnam aplinkos paruošimui ir paslaugų automatizavimui. Jie automatizuoja visus procesus įskaitant PĮ diegimą, atnaujinimą ir konfigūravimą, vartotojų paskyrų kūrimą, grafinę sąsają galutiniams naudotojams, kur galima kurti *FTP*, el. pašto, duomenų bazių paskyras, pridėti domenus, *SSL* sertifikatus, valdyti *DNS* zonas, kurti periodinius *Cronjob* veiksmus ir pan. Valdymo skydas leidžia administratoriams ir galutiniams naudotojams valdyti visas paslaugas įrankio pagalba.

Išanalizavus populiariausius valdymo skydus *cPanel*, *Plesk* ir *DirectAdmin* pastebėta, kad *DirectAdmin* konkurentų produktuose stinga apsaugos priemonių (žr. 1.4.1 lentelę). Dėl šios priežasties dalis prieglobos paslaugų teikėjų yra neatsparūs įvairių tipo atakoms.

1.4.1 lentelė. Konkurentų analizė

Kriterijai	*cPanel*	*Plesk*	*DirectAdmin*
Apsauga nuo išeinančio brukalo	-	-	bus įgyvendinta
Apsauga nuo įeinančio brukalo	atliekama ne SMTP metu	atliekama ne SMTP metu	bus įgyvendinta apsauga SMTP metu
Automatinis kenkėjiškų failų skanavimas *FTP* ir *PHP* įkėlimuose	-	-	bus įgyvendinta

Event MPM palaikymas Apache	✔	-	bus įgyvendinta
Grafinė sąsaja PĮ valdymui ir konfigūravimui	✔	✔	bus įgyvendinta
Let's Encrypt SSL sertifikatų palaikymas	–	✔	bus įgyvendinta
LiteSpeed palaikymas	✔	-	bus įgyvendinta
Nginx palaikymas	-	-	bus įgyvendinta
Nginx ir *Apache* kombinacijos palaikymas	-	✔	bus įgyvendinta
PHP-FPM ir *PHP-FastCGI* palaikymas	tik FastCGI	tik FastCGI	bus įgyvendinta
PHP vykdymo sauga	dalinis įgyvendinimas	dalinis įgyvendinimas	dalinis, bet bus ištaisytos *Apache* projektavimo klaidos ir sukurta saugi *PHP* konfigūracija
Sieve protokolo palaikymas *IMAP* (*Dovecot*)	-	✔	bus įgyvendinta
Saugus *cronjob* periodinių užklausų vykdymas	✔	✔	bus įgyvendinta
Žiniatinklio aplikacijų apsauga prieš brutualios jėgos atakas	apsauga tik valdymo skydui	apsauga tik valdymo skydui	bus įgyvendinta

Lietuvoje ar užsienyje nėra sukurto nė vieno produkto skirto *DirectAdmin* valdymo skydui, kuris išspręstų aptartas problemas. Vienokiu ar kitokiu būdu integravus galimus patobulinimus į CustomBuild įrankį, kuris naudojamas DirectAdmin valdymo skyde, pažeidžiamumų riziką galima ženkliai sumažinti.

1.5. Siekiamo sprendimo apibrėžimas

Nors egzistuoja alternatyvus kelias tikslui pasiekti, t.y. sukurti esamoms sistemoms *CustomBuild* ir *DirectAdmin* papildomus įskiepius, kurie užtikrintų reikiamą saugumo lygį, tinkamiausias sprendimas užsibrėžtiems tikslams įgyvendinti yra sukurti patobulintą sistemą *CustomBuild 2.0*, kurioje atsirastų minėti patobulinimai. Šis sprendimas yra tinkamiausias dėl to, jog sistemų papildymas įskiepiais reikalautų žymiai daugiau pastangų ir atneštų mažiau naudos dėl sekančių priežasčių:

1. *DirectAdmin* valdymo skydas naudoja daug vidinių patikrinimų dėl įdiegtų komponentų tam, kad būtų atliktas teisingas konfigūravimas po vartotojo pridėjimo. Patikrinimams yra kreipiamasi į *CustomBuild* options.conf failą ir kai kuriuos sisteminius kelius iki failų, todėl dėl glaudaus susietumo būtų sudėtinga trečiajai šaliai atitikti visus reikalavimus, o naujų funkcijų pridėjimas reikalauja pakeitimų ir pačiame *DirectAdmin* kode

2. Dėl saugumo, suderinamumo ir stabilumo nėra leidžiama kurti *CustomBuild* įskiepių, kurie papildytų *CustomBuild* kodą. Tad diegimas reikalautų kodo įterpimo į esamus *CustomBuild* scenarijų failus
3. Būtų sudėtinga pritraukti didelę potencialių įskiepių naudotojų auditoriją.

Taigi, racionaliausias sprendimas būtų tobulinti *CustomBuild 2.0* sistemą. Kadangi *CustomBuild* automatizuoto PĮ diegimo, valdymo ir atnaujinimo sistema buvo parašyta *bash* scenarijų kalba, todėl ir patobulinta versija bus rašoma ta pačia kalba. Ši kalba yra labai lengvai suprantama sistemų administratoriams, kurie ir bus pagrindiniai naudotojai. Internetinės grafinės sąsajos valdymui yra naudojama *PHP* kalba, tad sąsajos elementų papildymas taip pat bus rašomas *PHP* kalba. *PHP* programavimo kalba yra labiausiai paplitusi programuojant saityno turinį.

1.6. Analizės išvados

Atlikus *cPanel* ir *Plesk* valdymo skydų analizę buvo pastebėta, kad skiriama nepakankamai dėmesio užtikrinti didesniam saugumui. Iš to galima daryti išvadą, kad yra palankus metas įgyvendinti 1.3 skyriuje aptartas technologijas *DirectAdmin* valdymo skyde, tokiu būdu pritraukiant naujų vartotojų, kurie yra susirūpinę savo saugumu, arba vartotojų (duomenų centrų, prieglobos paslaugas teikiančių įmonių), kurie skiria daug lėšų žmogiškiesiems resursams, kad būtų pašalinamos ir stabdomos įvairios atakos prieš juos.

Taip pat pastebėta, kad konkurentai neturi planų artimiausių metu imtis visų 1.3 skyriuje išvardintų priemonių įgyvendinimo. Tikriausiai taip yra todėl, kad daugiau išteklių ir investicijų yra skiriama naujam funkcionalumui pateikti, marketingui ir reklamai.

Dėl *DirectAdmin* valdymo skydo apribojimų jo išeities tekstų redagavimui bei *CustomBuild* saugumo, suderinamumo ir stabilumo aspektų, aptartus sprendimus nuspręsta įgyvendinti kuriant ne papildomus įskiepius esamai automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo sistemoms *CustomBuild*, tačiau redaguoti sistemos išeities kodą.

Metodai bei programinė įranga, kuriuos planuojama į kuriamą sistemą integruoti, analizės dalyje yra detaliai aptarti. Metodai ir programinė įranga yra ganėtinai nauji ir kol kas dauguma jų nėra naudojami *cPanel* ar *Plesk* valdymo skyduose. Integravus aptartus sprendimus, tikimasi ženkliai padidinti saugumą serveriuose, kuriuose naudojamas *DirectAdmin* valdymo skydas.

2. AUTOMATIZUOTO PĮ DIEGIMO, ATNAUJINIMO IR KONFIGŪRAVIMO SISTEMOS CUSTOMBUILD PAPILDYMO PROJEKTAS

2.1. Reikalavimų specifikacija

2.1.1. Projekto apribojimai

Sistema veiks visose *Linux* ir *Unix* šeimos sistemose (distribucijose), kurias palaiko *CustomBuild* programa, t.y.:

- *RedHat Enterprise* / *CentOS* 5.x 32/64-bit, 6.x 32/64-bit, 7.x 64-bit;
- *FreeBSD* 8.x 64-bit, 9.x 32/64-bit, 10.x 64-bit;
- *Debian* 6.x 32/64-bit, 7.x 32/64-bit, 8.x 64-bit (ir *Ubuntu*, sukurtu šių sistemų pagrindu).

CustomBuild palaiko ir senesnės šių operacinių sistemų versijos, tačiau saugos funkcijų papildymai jose nebus bandomi, nes senos *OS* yra nerekomenduojamos naudoti dėl gyvavimo ciklo pabaigos (angl. *EOL*).

Sistemai įdiegti reikalinga naujai įdiegta bet kuri iš palaikomų operacinių sistemų (šiuo atveju sistemą leidžiama įdiegti *DirectAdmin* diegimo metu), arba sistema su jau įdiegtu *DirectAdmin* valdymo skydu. Diegimui taip pat reikia interneto ryšio su atvirais prievadais, reikalingais programinei įrangai veikti. Ugniasienėje pakankama atverti šiuos prievadus: 20 (*FTP* duomenų perdavimas), 21 (*FTP* kontrolė), 25 (*SMTP*), 53 (*DNS*), 80 (*HTTP*), 110 (*POP3*), 143 (*IMAP*), 443 (*HTTPS*), 465 (*SMTP* naudojant *SSL*), 587 (*SMTP*), 993 (*IMAPS*), 995 (*POP3S*), 2222 (*DirectAdmin*), 3306 (*MySQL*).

2.1.2. Funkciniai reikalavimai

Kuriamas automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo sistemos *CustomBuild* papildymas turi pateikti sekantį funkcionalumą:

- saugus visų palaikomų žiniatinklio serverių konfigūravimas, siekiant kuo mažesnės sistemos apkrovos (procesoriaus, operatyviosios atminties, standaus disko operacijų, tinklo) ir apsisaugojant potencialiai grėsmingų veiksmų;
- saugus *PHP* vykdymas ir konfigūravimas;
- išeinančio iš serverio brukalo prevencija;
- įeinančio į serverį brukalo prevencija;
- automatinis skanavimas dėl kenkėjiškų failų el. pašto laiškuose, žiniatinklio serverio perduodamuose failuose, failų įkėlimuose naudojant *FTP* ir *PHP*;
- saugus *Cronjob* periodinių užklausų vykdymas;
- saugi *CustomBuild* grafinė sąsaja su galimybe iškviesti automatines saugumo funkcijas.

Šiuos reikalavimus įgyvendinančios funkcijos pateiktos 2.2.2 skyriuje.

2.1.3. Nefunkciniai reikalavimai

Kuriamas automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo sistemos *CustomBuild* papildymas turi tenkinti sekančius nefunkcinius reikalavimus:

- stabilumas – PĮ galėtų nepertraukiamai veikti ilgą laiko tarpą, serveris galėtų aptarnauti daugiau užklausų, PĮ diegimas būtų sklandus;
- saugumas – diegimui reikalingos privilegijos būtų saugiai eskaluojamos, papildymai padidintų sistemos saugumą;
- paprastumas – naudoti automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo sistemą tiek komandinėje eilutėje, tiek grafinėje sąsajoje.

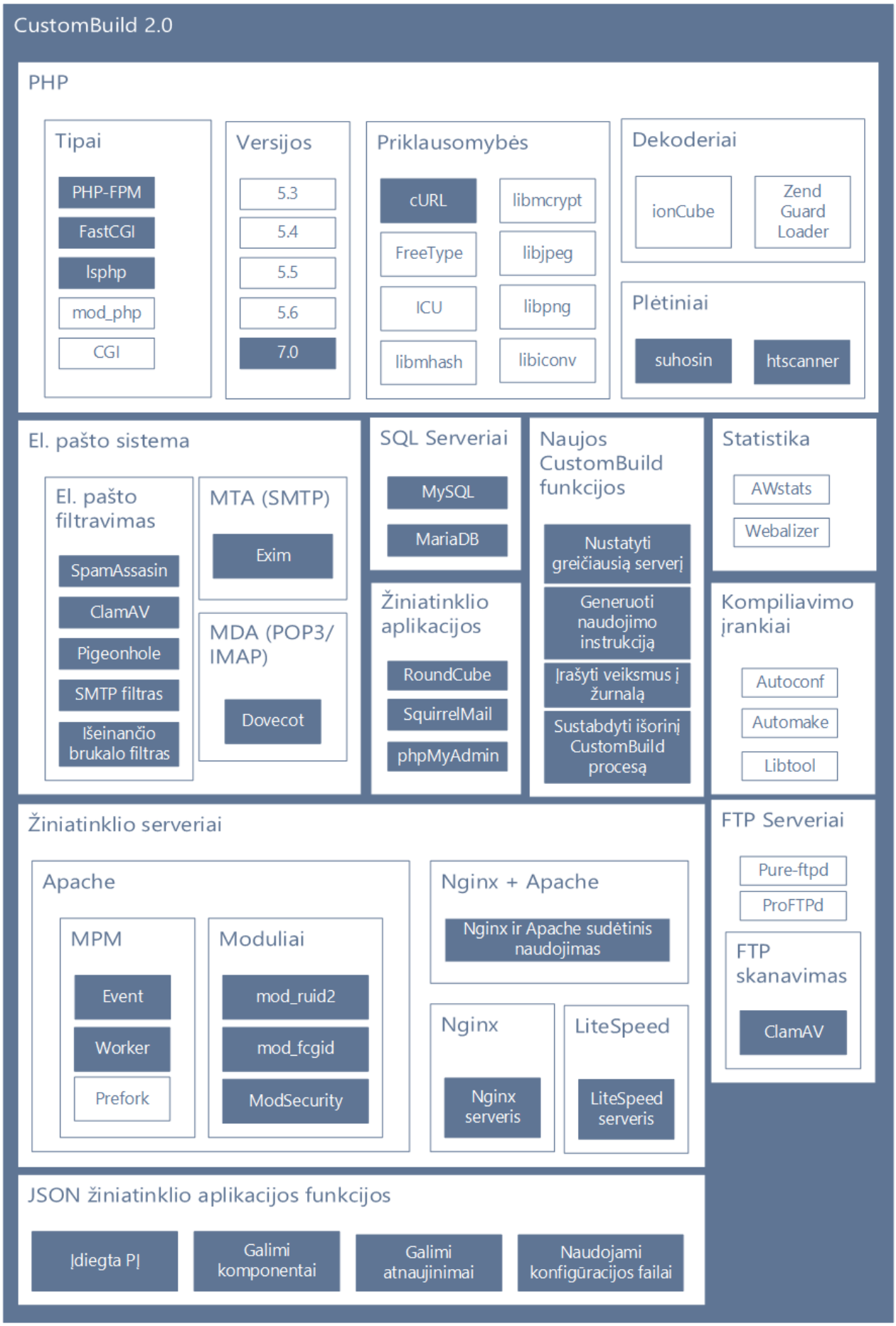
2.1.4. Techninė specifikacija

Be palaikomos OS nėra būtina jokia papildoma programinė įranga, tačiau svarbu, kad DirectAdmin valdymo skydas būtų diegiamas pagal oficialias instrukcijas. Minimalūs techninės įrangos reikalavimai sutampa su *DirectAdmin* valdymo skydo techninės įrangos reikalavimais. Rekomenduojamas bent 500 MHz taktiniu dažniu veikiantis procesorius, 1 GB operatyviosios atminties, 2 GB *swap* tipo atminties, 2 GB laisvos disko vietos po OS diegimo, 32 arba 64 bitų architektūros procesorius.

2.2. Sistemos projektas

2.2.1. Sistemos architektūra

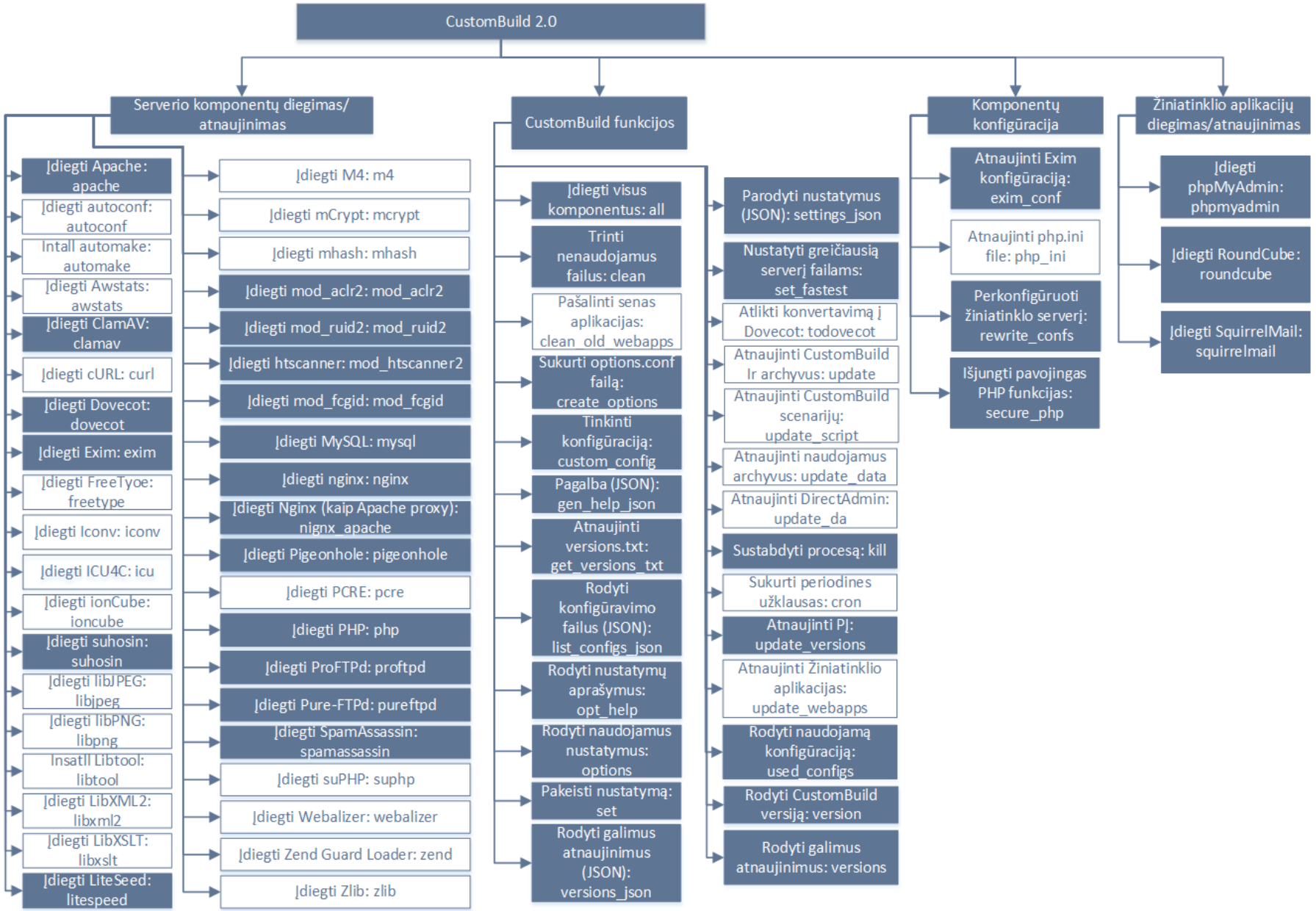
Apibendrintas sukurto produkto modelis, nurodant svarbiausias sudėtines jo dalis, pavaizduotas 2.2.1.1 paveiksle. Komponentai, kuriuos reikalinga redaguoti, yra pažymėti pilka fono spalva. Balta fono spalva nurodo funkcionalumą, kurio keisti nereikės. 2.2.1.1 paveiksle nurodytas ne visas, tačiau pagrindinis sistemos funkcionalumas ir komponentai.



2.2.1.1 pav. Apibendrintas sistemos modelis su siūlomais papildymais

2.2.2. Sistemos funkcijų hierarchijos diagrama

Funkcijų diagrama yra pavaizduota 2.2.2.1 paveiksle. Pilkame fone – funkcijos, kurias reikalinga koreguoti, baltame – jau esamos. Žemiausioje grandyje esantys elementai aprašo galimus veiksmus, o po dvitaškio – galimus „./build“ scenarijaus argumentus.



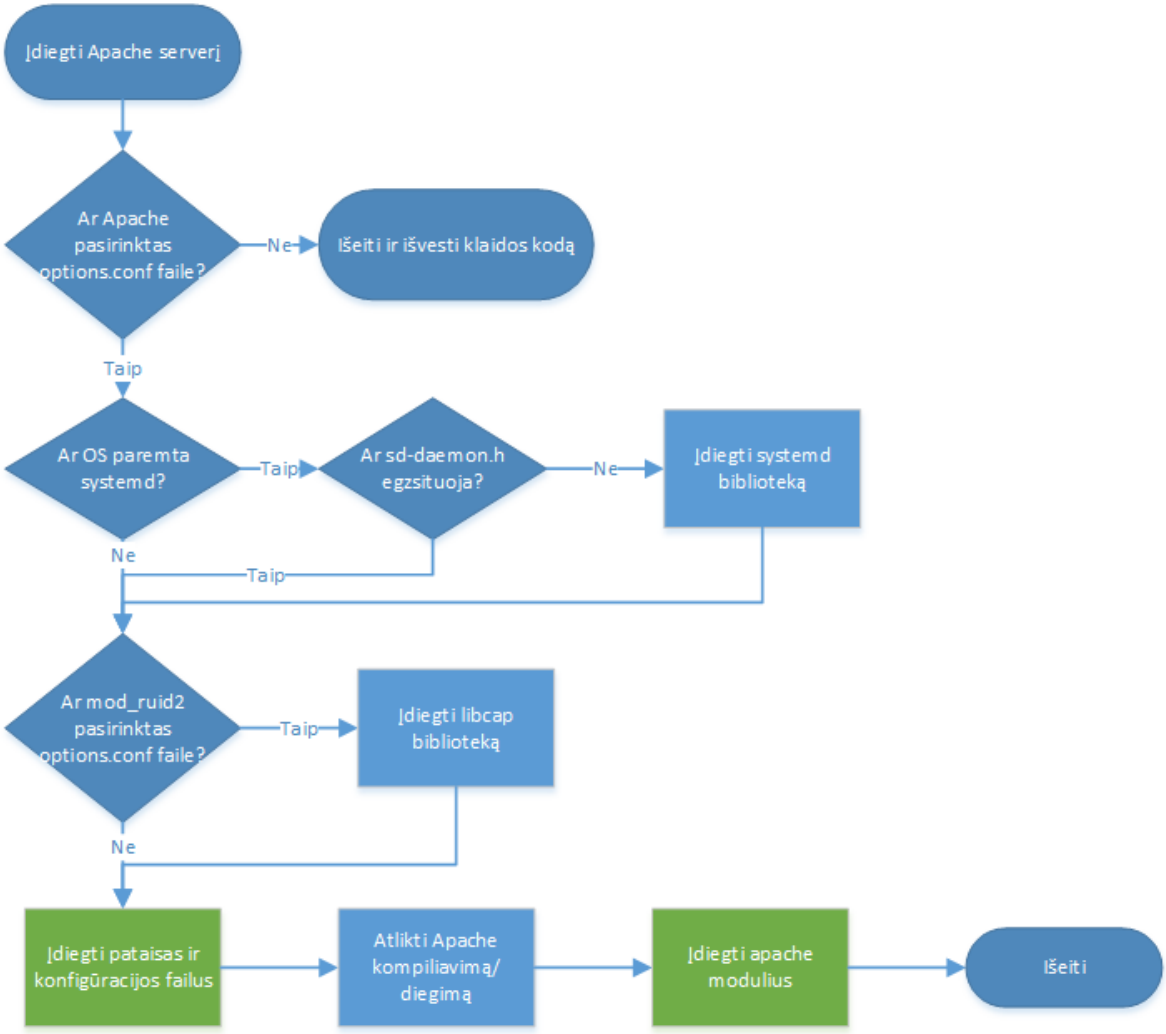
2.2.2.1 pav. Sistemos funkcijų hierarchijos diagrama

2.2.3. Numatomi pakeitimai pagal sistemos funkcijų hierarchijos diagramą

Apache komponento diegimo metu reikia:

- užtikrinti *ModSecurity* diegimą ir konfigūravimą;
- užlopyti (angl. *patch*) *mod_suexec* modulį, kad konfigūracija būtų palaikoma „<Directory>" konfigūracijos bloke, tokiu būdu suteikiant procesams vartotojų privilegijas;
- parinkti saugius protokolus šifruotiems sujungimams (prievadu 443);
- parinkti konfigūraciją pagal nutylėjimą tokią, kad būtų atlaikoma kuo daugiau lygiagrečių (angl. *concurrent*) sujungimų.

Modifikavimo reikalaujančios funkcijos yra pažymėtos žalia spalva 2.2.3.1 paveikslėlyje.



2.2.3.1. pav. *Apache* diegimo blokinė schema

ClamAV komponento diegimo metu reikia:

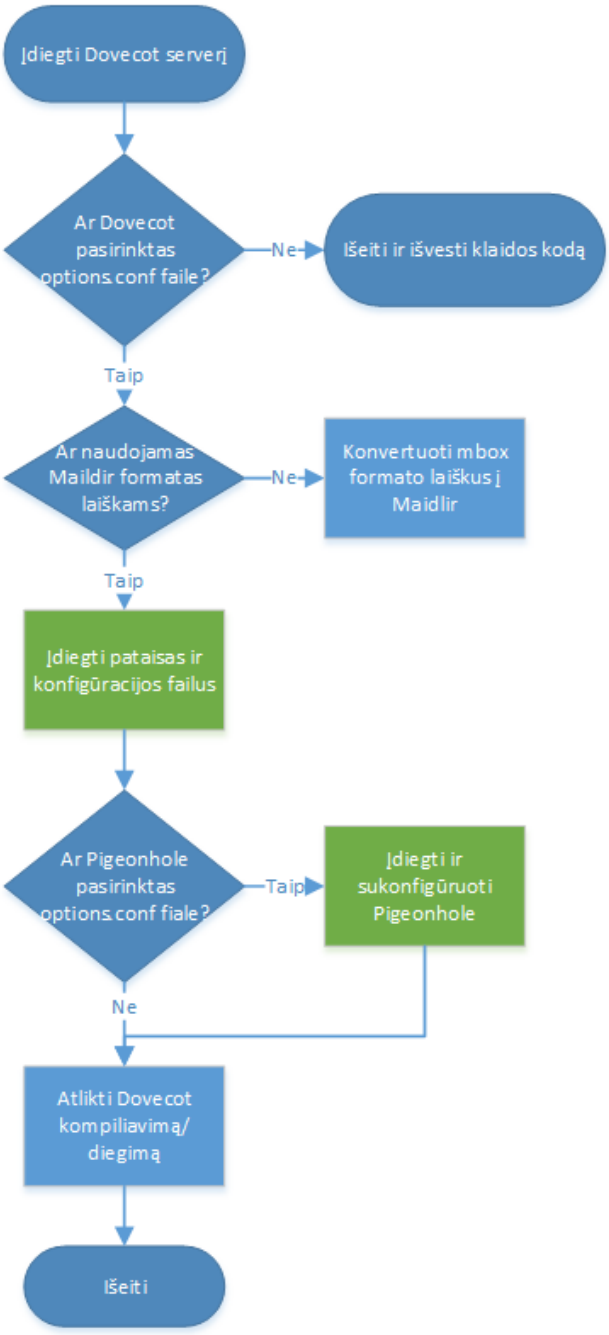
- užtikrinti tinkamą konfigūraciją žiniatinklio serverio perduodamiems failams skanuoti, tam panaudojant *ModSecurity* aplikacijų lygio ugniasienę;
- užtikrinti tinkamą konfigūraciją *PHP* perduodamiems failams skanuoti, tam panaudojant *suhosin PHP* modulį;
- užtikrinti tinkamą konfigūraciją *FTP* perduodamiems failams skanuoti, tam panaudojant *ProFTPd* arba *Pure-FTPd* funkcionalumą;

- užtikrinti tinkamą konfigūraciją failams, perduodamiems el. paštu, skanuoti.

Dovecot komponento diegimo metu reikia:

- naudoti *LMTP* (angl. *Local Mail Transfer Protocol*) protokolą vietiniams laiškų pristatymams;
- perduodamiems laiškams atlikti skanavimą panaudojant *pigeonhole* modulį (*Sieve* kalbą ir *ManageSieve* protokolą);
- pritaikyti modulinį konfigūracijos failų naudojimą, vietoje vieno bendro konfigūracijos failo viskam;
- suteikti galimybę vartotojams įdiegti *SSL* sertifikatus jų naudojamiems domenams.

Modifikavimo reikalaujančios funkcijos yra pažymėtos žalia spalva 2.2.3.2 paveikslėlyje.



2.2.3.2. pav. *Dovecot* diegimo blokinė schema

Exim komponento diegimo metu reikia:
- vietiniams laiškų pristatymams naudoti *LMTP* (perduoti juos *Dovecot LMTP* servisui);
- *SMTP* metu naudoti el. pašto skanavimą ir atmesti potencialiai kenkėjiškus laiškus, jiems net nepatekus į pristatomų laiškų eilę;
- įdiegti išeinančio iš serverio brukalo prevenciją;
- suteikti galimybę vartotojams įdiegti *SSL* sertifikatus jų naudojamiems domenams.

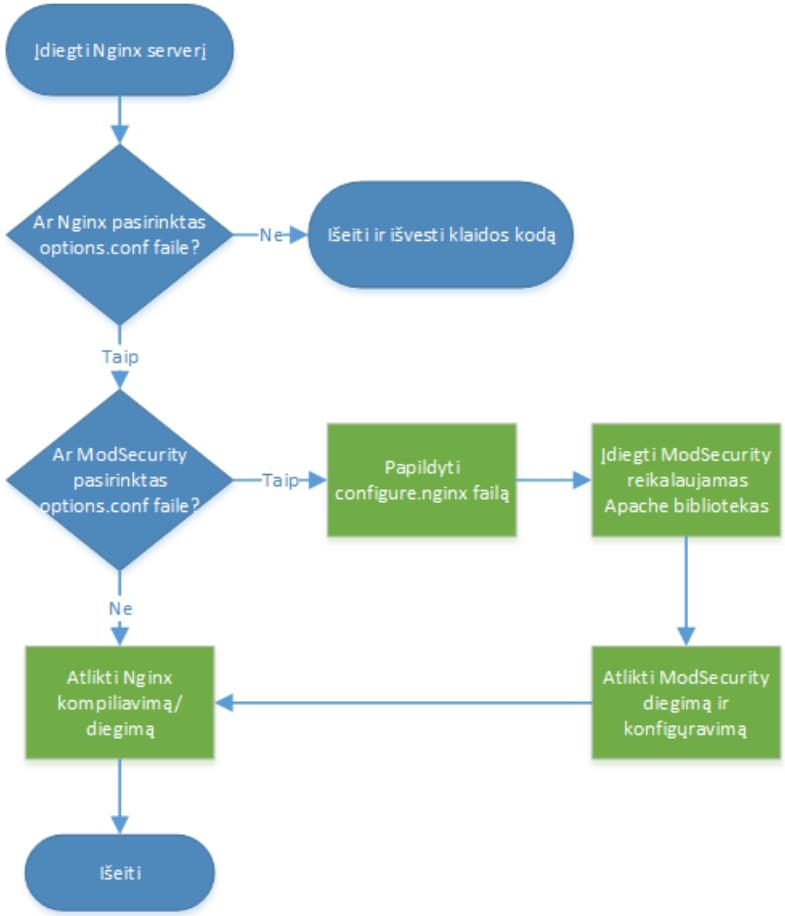
LiteSpeed žiniatinklio serverio diegimo metu reikia:
- užtikrinti *ModSecurity* diegimą ir konfigūravimą;
- parinkti konfigūraciją pagal nutylėjimą tokią, kad būtų atlaikoma kuo daugiau lygiagrečių (angl. *concurrent*) sujungimų.

Apache modulio *mod_fcgid* diegimo metu reikia:
- atlikti automatinį *Apache* „virtualhosts" konfigūravimą tokį, kad tam tikrose direktorijose vykstantis procesas turėtų kito vartotojo teises, tokiu būdų suteikiant saugias privilegijas visiems vartotojams vykdyti žiniatinklio aplikacijas, tokias kaip *RoundCube* ar *phpMyAdmin* per *Alias* direktyvą.

MySQL/*MariaDB* komponento diegimo metu reikia:
- atlikti saugų *MySQL*/*MariaDB* konfigūravimą pagal nutylėjimą;
- suteikti *MariaDB* diegimo galimybę.

Nginx komponento diegimo metu reikia:
- užtikrinti *ModSecurity* diegimą ir konfigūravimą;
- parinkti konfigūraciją pagal nutylėjimą tokią, kad būtų atlaikoma kuo daugiau lygiagrečių (angl. *concurrent*) sujungimų.

Modifikavimo reikalaujančios funkcijos yra pažymėtos žalia spalva 2.2.3.3 paveikslėlyje.



2.2.3.3. pav. *Nginx* diegimo blokinė schema

Pigeonhole komponento diegimo metu reikia:
- atlikti automatinį pigeonhole konfigūravimą.

PHP komponento diegimo metu reikia:
- diegiant PHP kaip *PHP-FPM*, užtikrinti izoliavimo (angl. *chroot*) galimybę;
- palaikyti PHP 7.0 versiją.

ProFTPd komponento diegimo metu reikia:
- suteikti galimybę įkeliamus failus skanuoti panaudojant *ClamAV* antivirusinę programą;
- suteikti galimybę automatiniam *sFTP* protokolo palaikymui, atliekant reikiamų modulių diegimą ir konfigūravimą.

Pure-FTPd komponento diegimo metu reikia:
- suteikti galimybę įkeliamus failus skanuoti panaudojant *ClamAV* antivirusinę.

phpMyAdmin komponento diegimo metu reikia:
- apsauga nuo brutualios jėgos (angl. *brute force*) atakos, registruojant nepavykusius prisijungimus į žurnalą ir automatiškai blokuoti IP po tam tikro kiekio neteisingų prisijungimų.

RoundCube komponento diegimo metu reikia:
- apsauga nuo brutualios jėgos atakos, registruojant nepavykusius prisijungimus į žurnalą ir automatiškai blokuoti IP po tam tikro kiekio neteisingų prisijungimų.

SquirrelMail komponento diegimo metu reikia:

- apsauga nuo brutualios jėgos atakos, registruojant nepavykusius prisijungimus į žurnalą ir automatiškai blokuoti IP po tam tikro kiekio neteisingų prisijungimų.

2.2.4. Sistemos konfigūracijos failų medis

CustomBuild turi aplanką pavadinimu *configure*, kuriame yra saugojami kompiliavimo konfigūracijos failai ir servisų konfigūracijos failai. Kiekvieną iš jų naudotojai gali perrašyti įkeliant failą į *custom* aplanką, tokiu atveju CustomBuild skiria prioritetą failams iš šio aplanko, vietoje *configure* ir leidžia paprastai keisti konfigūravimo vėliavėles pagal poreikius. *CustomBuild 2.0* patobulinimas reikalaus papildymo arba pridėjimo naujų konfigūracijos failų. Žemiau pateikiamame failų medyje reikalaujantys modifikavimo failai yra pažymėti mėlynai, žaliai – nauji failai.

Konfigūracijos failų medis:

```
├── ap2
│   ├── conf
│   │   ├── extra
│   │   │   ├── httpd-fcgid.conf
│   │   │   ├── httpd-modsecurity.conf
│   │   │   ├── httpd-mpm.conf
│   │   │   └── httpd-ssl.conf
│   ├── configure.apache
│   └── configure.php70
├── clamav
│   └── runav.pl
├── dovecot
│   ├── conf
│   │   ├── imap_mail_plugins.conf
│   │   ├── ip.conf
│   │   ├── limits.conf
│   │   ├── lmtp.conf
│   │   ├── lmtp_mail_plugins.conf
│   │   ├── mail_max_userip_connections.conf
│   │   ├── mail_plugins.conf
│   │   ├── protocols.conf
│   │   └── ssl.conf
│   ├── conf.d
│   │   ├── 90-quota.conf
│   │   └── 90-sieve.conf
│   ├── configure.dovecot
│   └── dovecot.conf
├── fastcgi
│   ├── configure.php70
│   └── fcgid70.sh
├── fpm
│   ├── conf
│   │   └── php-fpm.conf.70
│   └── configure.php70
```

```
├── litespeed
│   └── configure.php70
├── nginx
│   ├── conf
│   │   ├── nginx-defaults.conf
│   │   ├── nginx-events.conf
│   │   ├── nginx_limits.conf
│   │   ├── nginx-modsecurity.conf
│   └── configure.nginx
├── nginx_reverse
│   ├── conf
│   │   ├── nginx-defaults.conf
│   │   ├── nginx-events.conf
│   │   ├── nginx_limits.conf
│   │   ├── nginx-modsecurity.conf
│   │   ├── nginx-proxy.conf
│   └── configure.nginx
├── opcache
│   └── opcache.ini
├── proftpd
│   ├── conf
│   │   └── proftpd.conf
│   └── configure.proftpd
├── pureftpd
│   ├── configure.pureftpd
│   └── pureftpd_uploadscan.sh
├── suhosin
│   ├── php_uploadscan.sh
│   └── suhosin.ini
├── suphp
    ├── configure.php70
```

2.3. Projektinės dalies išvados

Visi techninėje magistro darbo užduotyje bei analitinėje dalyje suformuluoti reikalavimai kuriamai sistemai buvo įvertinti:

- identifikuotos modifikavimo reikalaujančios funkcijos, kurios atlieka komponentų diegimą, atnaujinimą ir konfigūravimą: *Apache, ClamAV, Dovecot, Exim, PHP, mod_fcgid, LiteSpeed, MySQL/MariaDB, Nginx, Pigeonhole, ProFTPd, Pure-FTPd, phpMyAdmin, RoundCube, SquirrelMail*;
- grafinei sąsajai modifikuoti bus reikalingas tik esamo *CustomBuild* scenarijaus pakeitimas.

Atliekant sistemos projektavimą problemų neiškilo, tačiau ateityje, toliau tobulinant sistemą, būtų galima visose scenarijaus vietose grąžinti išėjimo kodą (angl. *exit code*). Tokiu būdu atsirastų galimybė trečiųjų šalių programoms ir *CustomBuild* sąsajai žinoti, ar nurodytos komandos buvo sėkmingai įvykdytos, ar ne, nes šiuo metu tai matoma tik iš išvesties, tačiau ne klaidos kodų pagalba. Taip pat ateityje būtų galima suteikti vartotojams galimybę įsidiegti *SSL* sertifikatus savo domenams, kuriuos naudotų *FTP* serveris (šiuo metu naudojamas bendras serverio sertifikatas).

3. AUTOMATIZUOTO PĮ DIEGIMO, ATNAUJINIMO IR KONFIGŪRAVIMO SISTEMOS PAPILDYMŲ REALIZACINIAI ASPEKTAI

3.1. Išeinančio iš serverio brukalo prevencija

Automatizuotam išeinančio iš serverio brukalo prevencijos sprendimui sukurti buvo panaudotos dvi idėjos:

- riboti kiek el. laiškų gali būti išsiųsta į neegzistuojančias el. pašto dėžutes per valandą, blokuojant paskyrą pasiekus limitą;
- analizuoti PHP scenarijų, iš kurių yra siunčiami laiškai, kodą, siekiant aptikti žinomus kenkėjiškus failus ir blokuoti el. laiškų siuntimą iš jų.

3.1.1. Laiškų siuntimo į neegzistuojančius el. pašto adresus ribojimas

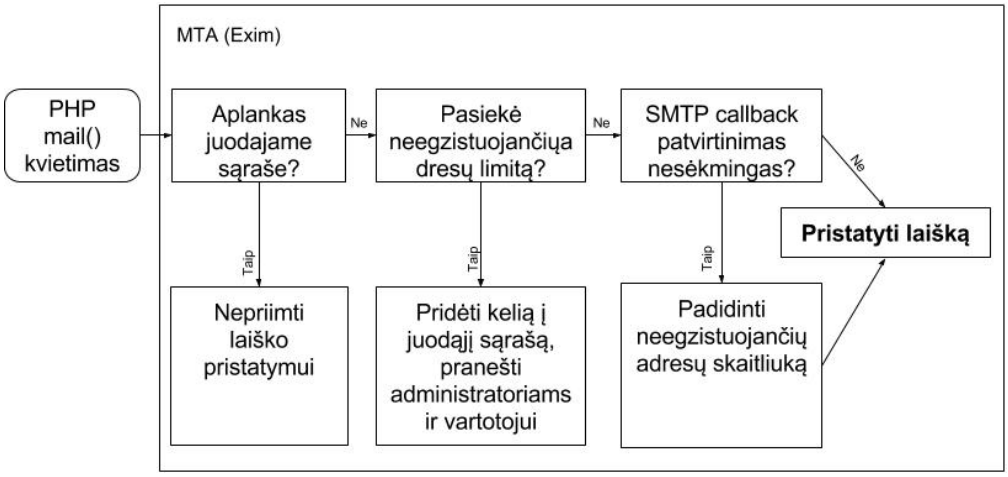
Programišiai, siunčiantys brukalą, dažnai naudoja didelius el. pašto adresų sąrašus, kurie turi dalį el. pašto adresų, kurie niekuomet neegzistavo, arba nebeegzistuoja. Idėja riboti, kiek laiškų gali būti išsiųsta į neegzistuojančias el. pašto dėžutes per atitinkamą laiko tarpą, nėra nauja. Diskusiją apie idėją internete jau 2010 metais iškėlė Andrew Hearn, kompanijos „Andrews & Arnold Ltd.", teikiančios el. pašto paslaugas, vadovas. Tačiau visos idėjos yra paremtos autentifikuotų *SMTP* vartotojų ribojimui ir blokavimui, bet ne el. laiškams siunčiamiems iš *PHP* scenarijų prieglobos paslaugas teikiančių serverių aplinkoje.

Sukurtas sprendimas leidžia nustatyti, kiek leidžiama išsiųsti laiškų į neegzistuojančius el. pašto adresus, todėl prieglobos paslaugas teikiančios įmonės gali optimizuoti nustatymus pačios, kad būtų griežtesnis arba laisvesnis limitavimas. Pagal nutylėjimą yra leidžiama išsiųsti 100 laiškų į neegzistuojančius el. pašto adresus, tuomet siuntimas iš aplanko, kuriame buvo galimai kenkėjiškas failas, yra blokuojamas tol, kol vartotojas arba administratorius paskyros neatblokuoja. Vartotojas turi galimybę atsiblokuoti pats todėl, kad jis gali pats imtis veiksmų išvalyti kenkėjiškiems failams ir atnaujinti turinio valdymo sistemą, kartu su jos naudojamais komponentais (angl. *plugins*). Kadangi informacinis pranešimas apie blokavimą siunčiamas tiek administratoriams, tiek vartotojui, jei vartotojas turės negerų ketinimų pastoviai atblokuojant paskyrą, administratoriams tai nebus sunku pastebėti ir pakartotinių pranešimų apie blokavimą. Pagal nutylėjimą parinktas limitas buvo nustatytas 100, nes bandymų metu su mažesniais limitais buvo gauta klaidingai teigiamų (angl. *false positives*) rezultatų, kurių didžioji dalis buvo iš el. parduotuvių, kurios siuntė naujienlaiškus į suklastotose registracijose (angl. *fake*) naudotus el. pašto adresus.

Kai limitas yra pasiekiamas, pilnas kelias iki aplanko, kuriame yra failas, yra automatiškai pridedamas į juodąjį sąrašą, saugomą faile. Pašto siuntimo programa (angl. *Mail Transfer Agent – MTA*) kiekvieną kartą prieš priimdama laišką siuntimui patikrina ar faile yra įtrauktas aplankas, iš kurio vykdomas *PHP* scenarijus, jei kelias iki aplanko randamas – laiškas yra atmetamas.

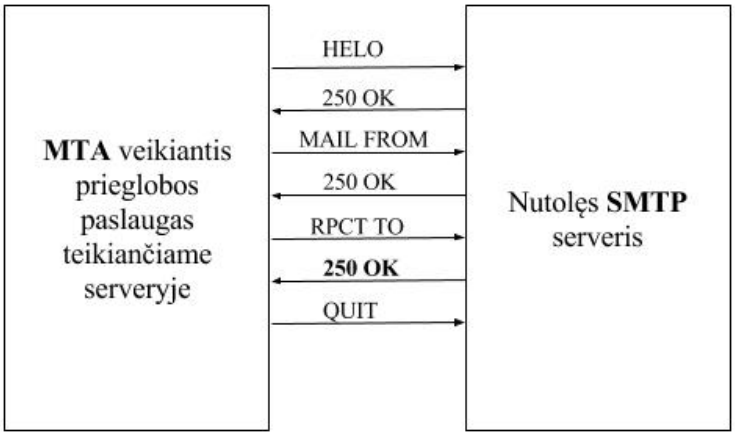
Gavėjo el. pašto adreso tikrinimas yra atliekamas naudojant *SMTP* atgalinio skambinimo tikrinimą (angl. *callback verification*), kuris yra atliekamas pačios pašto siuntimo programos (*MTA*) [33]. *DirectAdmin* valdymo skydas, kuriam buvo sukurta ši apsauga, veikia kartu su Exim pašto siuntimo programa. Kad patikrinti ar el. pašto adresas yra egzistuojantis, *MTA* jungiasi į nutolusį *SMTP* serverį (nustatytą *DNS MX* įrašuose), įvykdo standartinę *HELO*/*EHLO* komandą, reikalaujamą protokolo, išsiunčia *MAIL FROM* komandą be nurodyto adreso, tuomet išsiunčia *RCPT TO* komandą su el. pašto adresu, kuris turi būti patikrintas ir išeina naudojant komandą *QUIT*. Jei el. pašto adresas galioja (patikrinimas sėkmingas), nutolęs serveris į *RCPT* komandą atsako kodu 2xx, jei gaunamas atsakas su kodu 5xx, tai reiškia, kad el. pašto adresas negalioja (patikrinimas nesėkmingas). Gavus bet kurį kitą atsako kodą, sekantis pagal prioritetą serveris nurodytas *DNS MX* įrašuose yra tikrinamas (jei yra nurodytas daugiau nei vienas serveris *MX* įrašuose). Sėkmingas patikrinimas su atsako kodu 2xx negarantuoja, kad laiškas bus pristatytas sėkmingai adresatui, tačiau nesėkmingas patikrinimas su atsaku 5xx užtikrina, kad laiškas nebus pristatytas. Exim resursų taupymui dėl callback užklausų

naudoja spartinančią atmintinę (angl. *cache*). Sprendimo veikimo principas yra pavaizduotas 3.1.1.1 paveiksle.



3.1.1.1. pav. Veiksmų seka laiško pristatymui

Sukurtas sprendimas naudoja 10 sekundžių laukimo laiką (angl. *timeout*) ir *defer_ok* nustatymą [34], kuris nurodo *Exim MTA*, kad nesėkmingą susisiekimą su nutolusiu serveriu ir laikinas problemas traktuoti kaip sėkmingą patikrinimą *ACL* sąraše. Sėkmingas callback, kai iš nutolusio serverio, po komandos *RPCT TO* gaunamas atsakymas "250 OK", pavaizduotas paveiksle 3.1.1.2. Kai šioje vietoje gražinamas kodas 5xx, tuomet el. pašto adresas skaičiuojamas kaip neegzistuojantis.



3.1.1.2. pav. SMTP callback schema kai gavėjas egzistuoja

3.1.2. Kenkėjiškų *PHP* scenarijų identifikavimas

Anksčiau minėtas sprendimas yra sudedamoji viso sprendimo dalis išeinančio iš serverio brukalo prevencijai. Nors jis ir sustabdo dalį išeinančio brukalo, jį verta papildyti, nes kenkėjiški *PHP* failai vis dar turi galimybę išsiųsti nepageidaujamus el. laiškus kol nepasiekia 100 neegzistuojančių gavėjų adresų limito per valandą.

Atliekant kenkėjiškų *PHP* scenarijų kodo analizę prieglobos paslaugas teikiančiuose serveriuose, buvo pastebėta, kad didžioji dauguma scenarijų naudoja užkoduotą *PHP* kodą failuose, tam kad nebūtų lengvai aptikti vartotojų. Pavyzdžiui *base64_decode()* funkcija yra naudojama kodui iškoduoti, o gale įvykdomas *eval()* iškvietimas šiam kodui įvykdyti. Pasinaudojant šia savybe buvo sudarytas reguliariųjų reiškinių (angl. *regex*) sąrašas, pagal kurį galima lengvai identifikuoti ar *PHP* scenarijus yra įtartinas. Sudarius sąrašą buvo patikrinti 5264 tinklalapių failai, kad įsitikinti, jog šie reguliarieji reiškiniai neaptinka klaidingai teigiamų failų (angl. *false positives*). Visi surasti failai buvo

kenkėjiški. Didžiajai daugumai kenkėjiškų *PHP* scenarijų atpažinti pakanka reguliariųjų reiškinių, kurie randa failus, turinčius:

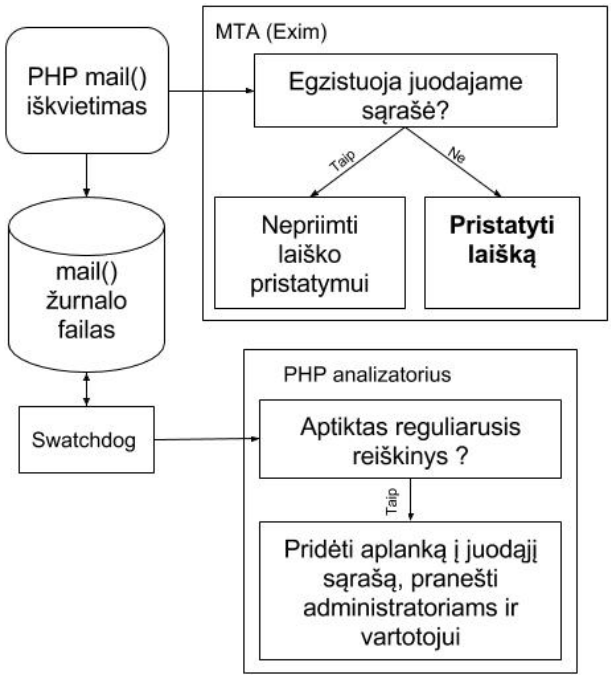
- *base64_decode()* ir *eval()* tame pačiame faile, mail() funkcijos iškvietimas paslėptas;
- daugiau nei 10 kartų panaudotas kintamasis *GLOBALS* ir kviečiamas *eval()* tame pačiame faile, *mail()* funkcijos iškvietimas paslėptas;
- *GLOBALS*, *substr()* ir *return* vienoje eilutėje, *mail()* funkcijos iškvietimas paslėptas;
- daugiau nei 10 kartų panaudota *chr()* funkcija, iškviestas *eval()* tame pačiame faile, *mail()* funkcijos iškvietimas paslėptas.

Atsiradus naujiems kodo maskavimo būdams, reguliariųjų reiškinių sąrašą galima lengvai papildyti.

Nors *PHP* suteikia galimybę tiesiog globaliai išjungti nenorimas funkcijas naudojant *disable_functions* nustatymą, visos šios funkcijos gali būti naudojamos ir ne kenkėjiškuose failuose, tačiau jos paprastai nebūna naudojamos paminėtomis kombinacijomis, ypač paslepiant *mail()* funkcijos iškvietimą. Todėl funkcijų uždraudimas neigiamai paveiktų visą prieglobos paslaugų serverio aplinką, sugadindamas kai kurių *PHP* scenarijų vykdymą. Be to, *eval()* uždrausti net nėra galimybių, nes tai yra *PHP* kalbos konstrukcija (angl. *language construct*), o ne paprastas funkcijos iškvietimas.

Efektyviam kenkėjiškų PHP scenarijų aptikimui naudojamas žurnalo failų (angl. *log file*) skanavimas. *PHP* suteikia realaus *mail()* funkciją kviečiančių scenarijų įrašymo į žurnalo failus galimybę [32], tokiu būdu galime lengvai patikrinti ar *PHP* scenarijus turi paslėptą mail() iškvietimą kode ir realiu laiku tikrinti ar failas yra kenkėjiškas. Teigti, kad *PHP* scenarijus, kuris išsiuntė el. laišką ir turi paslėptą *mail()* funkcijos iškvietimą, yra kenkėjiškas būtų neteisinga, nes yra galimybė užkoduoti *PHP* kodą naudojant komercinius įrankius, tokius kaip *ionCube* ar *Zend*. Jie yra naudojami tam, kad pardavinėjant programinę įrangą butų išvengta atviro programinio kodo atskleidimo.

Žurnalo failų skanavimui realiu laiku buvo panaudota *Swatchdog* (*Simple Log Watcher*) programinė įranga [33], kuris tik aptikus naują įrašą žurnalo faile iškviečia scenarijų *PHP* kodo patikrinimui. Jeigu randama, kad *mail()* funkciją iškvietė kenkėjiškas failas, pilnas kelias iki aplanko, kuriame yra failas, yra įtraukiamas į juodąjį sąrašą, taip blokuojant bet kokių el. laiškų siuntimą iš šio aplanko. Papildomai apie incidentą yra pranešama administratoriams ir galutiniams vartotojams. Principinė *PHP* kodo analizės schema yra pavaizduota 3.1.2.1 paveiksle.



3.1.2.1. pav. Principinė PHP kodo analizės schema

Kaip pavaizduota 3.1.2.1 paveiksle, kai *PHP* scenarijuje iškviečiama *mail()* funkcija, laiškas yra išsiunčiamas (jei neegzistuoja juodajame sąraše), o tuo pačiu metu įvykis apie išsiųstą laišką iškart atsiranda žurnalo faile. *MTA* ir *Swatchdog* veikia nepriklausomai vienas nuo kito, todėl techniškai negalima sustabdyti pirmojo laiško nuo išsiuntimo. Tačiau *Swatchdog* tikrina žurnalo failus realiuoju laiku, todėl, atsiradus naujam įrašui, jis iškart perduoda pilną kelią iki failo, kuris iškvietė *mail()* funkciją, scenarijui, kuris analizuoja *PHP* kodą. Jei scenarijus aptinka kenkėjišką failą (pagal reguliariuosius reiškinius), jis į juodąjį sąrašą prideda pilną kelią iki aplanko, kuriame saugomas failas. Sekantį kartą siunčiant el. laišką iš tame aplanke esančio *PHP* scenarijaus, *MTA* aptiks kelią iki aplanko juodajame sąraše ir neleis išsiųsti laiško.

Daugelis turinio valdymo sistemų (TVS) turi aplankus, kurie yra naudojami vartotojų failų įkėlimams (pvz. nuotraukų, filmukų, dokumentų ir pan.). Šie aplankai yra dažnas taikinys neapribotų failų įkėlimo atakoms [34]. Tam, kad dalinai būtų išspręstas aukščiau įvardinto metodo statiškumas, gali būti naudojamas globalus aplankų, iš kurių nebūtų galima siųsti el. pašto laiškų, sąrašas. Tokiu atveju *MTA* tikrintų ar pilnas kelias iki aplanko iš kurio yra siunčiami laiškai neturi reguliariųjų išraiškų atitikmenų globaliame blokuojamų aplankų sąraše. Jei atitikmuo randamas, laiško pristatymas būtų blokuojamas. Pavyzdžiui, „WordPress" turinio valdymo sistemoje būtų saugu blokuoti visus el. laiškų siuntimus panaudojant šias reguliariąsias išraiškas:

- ^.*/wp-content/cache.*
- ^.*/wp-content/uploads.*

3.2. Įeinančio į serverį brukalo prevencija

Automatizuotas įeinančio į serverio brukalo prevencijos metodas remiasi laiško antraščių, turinio ir domeno *DNS* įrašų analize *SMTP* sujungimo metu. Tokiu būdų galima atmesti laiškus dar prieš jiems patenkant į el. laiškų eilę, taip pat sutaupyti procesoriaus resursų dėl išankstinio patikrinimo prieš perduodant laišką turinio analizei. T.y. jeigu esame tikri, kad laiškas yra nepageidaujamas, jį galime atmesti tuo laiko momentu, kuomet tai yra nustatoma, prieš perduodant laišką skanuoti sudėtingesnei trečiųjų šalių programinei įrangai kaip *SpamAssassin* ar *ClamAV*.

Išanalizavus brukalo laiškų rinkinį, buvo sudarytas taškų rinkimu paremtas algoritmas laiškams atmesti. Jei pasiekiamas 100 taškų kiekis – laiškas atmetamas, jei pasiekiama 55 – papildomai vykdomas *SpamAssassin* skanavimas *SMTP* metu.

SMTP metu atliekami šie patikrinimai eilės tvarka:

- tikrinamas *SPF* įrašas, jei pagal *TXT* įrašą *DNS* zonoje siuntėjo IP neturi teisės siųsti laiškų – laiškas atmetamas, jei būsena yra *soft fail* (*DNS* zonoje naudojama *~all*), tuomet pridedama 30 taškų, jei SPF įrašas tinkamas – bendras taškų kiekis sumažinamas 30 taškų;
- tikrinamas atgalinis *DNS* (angl. *reverse DNS*) siuntėjo *IP* adresui, jei atgalinio *DNS* nėra, pridedama 100 taškų prie bendro taškų skaičiaus, jei atgalinis *DNS* yra tinkamas, taškų kiekis išlieka toks pat, o jei atgalinis *DNS* atitinka *DNS* zonoje esantį *A* tipo įrašą, tuomet bendras taškų kiekis sumažinamas 10 taškų;
- jei siuntėjo IP neatitinka siuntėjo domeno *MX* įraše nurodyto serverio, užlaikyti sujungimą 2 sekundes, tokiu būdu atmetant dalį brukalo siuntėjų;
- jei siuntėjo adresas aptinkamas realaus laiko juoduosiuose sąrašuose (angl. *RBL*), prie bendro taškų skaičiaus pridėti 100 taškų;
- tikrinamas *DKIM* įrašas, jei jis neteisingas arba el. laiškas nepasirašytas *DKIM* antraštėje, o *DNS* zonoje jei egzistuoja – pridėti 100 taškų, jei *DKIM* įrašo neegzistuoja *DNS* zonoje, taškų kiekio nekeisti, o jei *DKIM* teisingas – sumažinti bendrą taškų kiekį 20 taškų;
- jei laiško dydis yra mažesnis jei 200KB, o bendras taškų kiekis yra daugiau nei 55, vykdyti laiško turinio skanavimą naudojant *SpamAssassin*, jei *SpamAssassin* nustato, kad tai brukalas – pridedama 20 taškų prie bendro taškų skaičiaus, ir *SpamAssassin* suteiktų taškų kiekis padaugintas iš 10.

Bet kurie taškų kiekiai pagal poreikį yra keičiami konfigūracijos faile. Atlikus visus veiksmus patikrinamas bendras taškų kiekis, jei jis yra didesnis arba lygus numatytam – laiškas atmetamas, jei mažesnis – perduodamas tolimesniam apdorojimui ir patenka į el. pašto laiškų eilę serveryje.

3.3. Saugus *PHP* vykdymas

Pagal nutylėjimą *PHP* vykdyme yra leidžiamos pavojingos funkcijos, kurias naudojant galima vykdyti sistemos komandas, paleisti procesus, nuskaityti bet kurį vartotojo aplinkoje esantį failą. Pavojingi *PHP* scenarijai, vadinami *PHP Shell*, dažnai būna įkeliami pasinaudojant nutolusio failo įkėlimo ataka (angl. *RFI – Remote File Inclusion*), atradus pažeidžiamumą turinio valdymo sistemoje ar jos įskiepiuose. Atlikus populiarių PHP Shell analizę, buvo atrinktos šios funkcijos, kaip keliančios pavojų: *exec(), system(), passthru(), shell_exec(), escapeshellarg(), escapeshellcmd(), proc_close(), proc_open(), dl(), popen(), show_source()*. Nors funkcijos *escapeshellarg()* ir *escapeshellcmd()* pačios nėra pavojingos, tačiau jos palengvina automatišką kaitos ženklų naudojimą vykdomoms komandoms. Visos šios funkcijos patobulintame įrankyje *CustomBuild* gali būti uždraustos naudoti vieno mygtuko paspaudimu.

Sekanti problema, dėl kurios *PHP* vykdymas nėra saugus, kai *PHP* veikia naudojant *FastCGI* serverio sąsają, yra susijusi su *Apache SuEXEC* moduliu. Apache nesuteikia galimybės nurodyti, kurio vartotojo teisėmis turi būti *PHP* vykdymas tam tikroje direktorijoje. Dėl šios priežasties *Alias* direktyvose naudojami aplankai turi būti pasiekiami visiems vartotojams skaitymo ir vykdymo teisėmis, kas sukelia saugumo grėsmę. Šios problemos sprendimui reikėjo sukurti pataisą (angl. *patch*), kuri leistų „SuexecUserGroup” nustatymą naudoti „<Directory>” kontekste. Pataisos kodas yra pavaizduotas paveiksle 3.3.1.

```
--- httpd-2.4.10/modules/generators/mod_suexec.c.old    2011-12-05 01:08:01.000000000 +0100
+++ httpd-2.4.10/modules/generators/mod_suexec.c        2014-09-11 00:16:21.444000009 +0200
@@ -59,7 +59,7 @@
                                     const char *uid, const char *gid)
 {
     suexec_config_t *cfg = (suexec_config_t *) mconfig;
-    const char *err = ap_check_cmd_context(cmd, NOT_IN_DIR_LOC_FILE);
+    const char *err = ap_check_cmd_context(cmd, NOT_IN_LOCATION|NOT_IN_FILES);

     if (err != NULL) {
         return err;
@@ -116,7 +116,7 @@
 {
     /* XXX - Another important reason not to allow this in .htaccess is that
      * the ap_[ug]name2id() is not thread-safe */
-    AP_INIT_TAKE2("SuexecUserGroup", set_suexec_ugid, NULL, RSRC_CONF,
+    AP_INIT_TAKE2("SuexecUserGroup", set_suexec_ugid, NULL, RSRC_CONF|ACCESS_CONF,
       "User and group for spawned processes"),
     { NULL }
 };
--- httpd-2.4.10/support/suexec.c.old   2014-10-10 11:48:20.388000025 +0200
+++ httpd-2.4.10/support/suexec.c       2014-10-10 11:50:30.757000025 +0200
@@ -308,6 +308,7 @@
 #ifdef AP_SUEXEC_UMASK
         fprintf(stderr, " -D AP_SUEXEC_UMASK=%03o\n", AP_SUEXEC_UMASK);
 #endif
+        fprintf(stderr, " -D AP_PER_DIR=\"yes\"\n");
 #ifdef AP_UID_MIN
         fprintf(stderr, " -D AP_UID_MIN=%d\n", AP_UID_MIN);
 #endif
```

3.3.1 pav. *Apache SuEXEC* modulio pataisa

Kaip matoma iš paveikslo 3.3.1, pataisymai *Apache* kodui yra minimalūs. To priežastis – šis funkcionalumas galimai buvo planuojamas, tačiau neužbaigtas. Kadangi pats *SuEXEC* modulio kodas buvo nekeistas beveik 15 metų, net *Apache* kūrėjai negalėjo atsakyti kodėl dalis kodo šiam funkcionalumui realizuoti buvo pusiau išbaigtas, tačiau esantis kodas nei buvo pašalintas, nei užbaigtas

iki galo. Kad įsitikinti, jog šis sprendimas neatvers jokių naujų saugumo spragų, pataisos kodas buvo parodytas *Apache* kūrėjams, o iš jų buvo gautas patvirtinimas, kad kodas yra saugus naudoti.

3.4. Viešai pasiekiamų aplikacijų apsaugojimas nuo atakų

Viešai pasiekiamos aplikacijos kaip el. pašto programa *RoundCube* ar duomenų bazių valdymo įrankis *phpMyAdmin* suteikia galimybę prieglobos paslauga besinaudojantiems vartotojams, įmonės darbuotojams ar svetainės administratoriams, naršyklės lange prisijungti prie el. pašto, skaityti ir rašyti laiškus, administruoti *MySQL* duomenų bazę. Tačiau taip pat tai yra ir taikinys atakoms, nes žinant prisijungimo vardą (arba bandant jį atspėti) gali būti panaudota brutalios jėgos arba žodyno atakos, kad būtų sužinotas paskyros slaptažodis.

Kad būtų užkirstas kelias įsibrovimui panaudojant aukščiau paminėtas programas, reikia apriboti neteisingų prisijungimų per valandą kiekį, o kai jis būna pasiektas, blokuoti atakuotojo IP adresą. Tai patogiausia atlikti naudojant žurnalo (angl. *log*) failus, tačiau *phpMyAdmin* neturi galimybės identifikuoti neteisingų prisijungimų ir juos įrašinėti į žurnalo failus. Dėl šios priežasties reikėjo sukurti pataisą (angl. *patch*), kuri prideda papildomą kodą į *phpMyAdmin* paketą ir įrašinėja nesėkmingus prisijungimus į žurnalo failus. Pakeitimai buvo atlikti *libraries/logging.lib.php* faile, o *libraries/common.inc.php* iškviečiama reikiama funkcija. Pagrindiniai pakeitimai yra pavaizduoti 3.4.1 paveiksle.

```
function log_to_file($user, $status)
{
        $LOG_FILE=AUTH_LOG_FILE;

        if ($user == '')
        {
                $user = PMA_getenv('PHP_AUTH_USER');
        }

        if ($user == '')
                return true;

        //remove any ' characters from $user
        $user = urlencode($user);

        //check for logout
        if ($status == 'not authenticated')
        {
                if (isset($_GET['old_usr']) && isset($_SERVER['PHP_AUTH_USER']))
                {
                        if ($_GET['old_usr'] == $_SERVER['PHP_AUTH_USER'])
                        {
                                $status = 'logout';
                        }
                }
        }

        $log_str = date('M d H:i:s').":: pma auth user='$user' status='$status' ip='".$_SERVER["REMOTE_ADDR"]."'";

        $fp = fopen($LOG_FILE, 'a');
        if ($fp === false)
        {
                //log to apache error log instead
                error_log($log_str."\n");
                return;
        }

        fwrite($fp, $log_str."\n");
        fclose($fp);
}
```

3.4.1 pav. *phpMyAdmin* pataisos failas apsaugai nuo brutalios jėgos ir žodyno atakų

Paveiksle 3.4.1 pavaizduota funkcija įrašo nesėkmingus prisijungimus į žurnalo failą, o valdymo skydas *DirectAdmin* nuolat jį skanuoja ir blokuoja *IP* adresus, kurie atliko daugiau nei 5 nesėkmingus prisijungimus per valandą. Blokavimas atliekamas *iptables* ugniasienės pagalba.

Panašus metodas pritaikytas ir *RoundCube* el. pašto programai pasiekiamai per naršyklę, tačiau *RoundCube* pataisos sukurti nereikėjo, nes šis funkcionalumas jau yra realizuotas pagal nutylėjimą.

Bandymai atspėti slaptažodį yra aktuali problema ir prieglobos paslaugomis besinaudojančio vartotojo aplinkoje. Atakos gali sulėtinti viso serverio darbą arba padaryti jį nepasiekiamu dėl per didelės apkrovos. Jei sėkmingai atspėjami prisijungimo duomenys prie administratoriaus zonos, piktavaliai gali įkelti kenkėjiškų failų, parsisiųsti slaptus duomenis iš duomenų bazės, paleisti procesus serveryje. Kaip teigiama Imperva ataskaitoje „Web Apllication Attack Report #5", dažniausiai atakuojama turinio valdymo sistema yra „WordPress". Kadangi WordPress sistema pagal nutylėjimą nesaugo nesėkmingų prisijungimų žurnalo failuose, buvo nuspręsta naudoti apache arba *Nginx* serverio žurnalo failus. Juos analizuojant pastebėta, kad *HTTP* būsenos kodas sėkmingai prisijungus yra 302, nes vyksta peradresavimas į administratoriaus zoną. Jei prisijungimas nebūna sėkmingas, *HTTP* būsenos kodas yra 200, t.y. tiesiog rodomas tas pats prisijungimo langas. Pavyzdžiui:

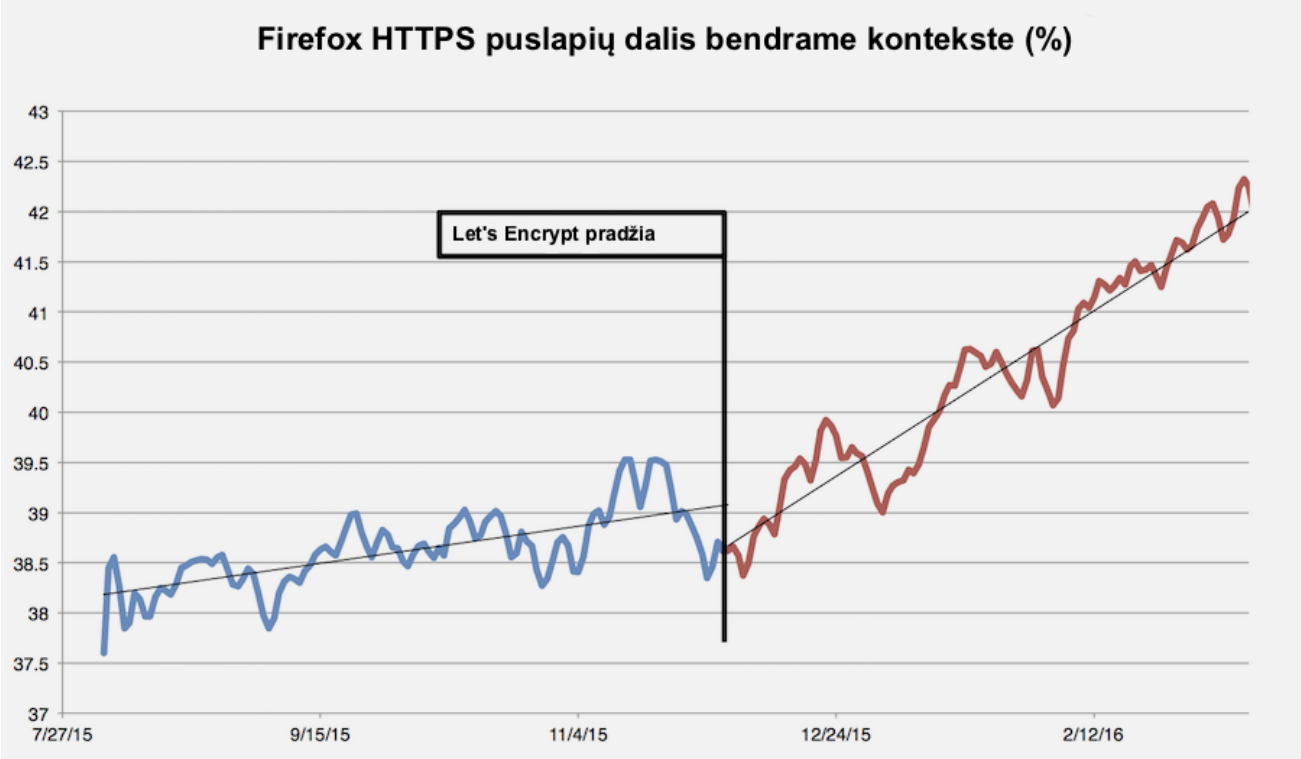
- ... "POST /wp-login.php HTTP/1.1" **302** 1151 ... – sėkmingas prisijungimas;
- ... "POST /wp-login.php HTTP/1.1" **200** 2024 ... – nesėkmingas prisijungimas.

Pagal *HTTP* būsenos kodą galima tiksliai identifikuoti nesėkmingus prisijungimus ir blokuoti su jais susijusius *IP* adresus, jeigu pasiekiamas 5 neteisingų prisijungimų per valandą limitas.

3.5. *Let's Encrypt SSL* sertifikatų diegimas

Daugelis interneto vartotojų ir svetainių savininkų nežino skirtumų tarp *HTTP* ir *HTTPS* protokolų, todėl dažnai naudojasi neapsaugotu ryšio kanalu tarp svetainės ir galutinio naudotojo. Apsikeičiant duomenimis atviru tekstu (angl. *plain-text*), jie gali būti lengvai nuskaitomi perdavimo metu. Tokiu būdu gali būti atskleidžiami slapti duomenys kaip kredito kortelės numeris, vartotojo slaptažodis ar kita su juo susijusi informacija. Naudojantis *HTTPS* protokolu visas ryšio kanalas yra šifruojamas, o išduodant *SSL* sertifikatą, domenas būna patvirtinamas, todėl mažėja apgaulės atvejų.

SSL sertifikatus išduoda sertifikavimo organizacija (angl. *certificate authority*). Prieglobos paslaugomis besinaudojantiems vartotojams yra sudėtinga patiems įsidiegti SSL sertifikatus, nes jie dažnai nesupranta viešojo rakto infrastruktūros veikimo. Be to, SSL sertifikatai būdavo mokami. Tačiau 2014 m. lapkričio mėn. 18 d. buvo paskelbtas pranešimas apie *Let's Encrypt* sertifikavimo organizaciją, kuri suteiks *SSL* sertifikatus nemokamai, o jų suteikimo metodas bus automatizuotas. Pirmasis nemokamas sertifikatas buvo išduotas 2015 m. spalio mėn. 19 d. *Let's Encrypt* organizacijos kūrėjai yra ne pelno organizacijos „Mozilla Foundation", „Electronic Frontier Foundation" ir Mičigano universitetas. Organizacijos tikslas – įgalinti visuotinį šifravimą WWW serveriuose. Per trumpą laikotarpį sertifikatų organizacija tapo trečia pagal išduotų sertifikatų kiekį, jų buvo išduota 1.8 milijono per beveik 5 mėnesius [31], o sertifikatai naudojami daugiau nei 3.8 milijono svetainių. Kadangi 85 proc. svetainių naudojančių *Let's Encrypt* sertifikatus niekuomet nenaudojo *HTTPS* protokolo, tai gerokai padidino bendrą svetainių, kurios naudoja šifruotą ryšio kanalą, kiekį. Remiantis Mozilla pateikiamais duomenimis iš Firefox naršyklę naudojančių vartotojų, šiuo metu daugiau nei 42 proc. apsilankymų svetainėse naudojamas saugus ryšys, palyginus su 38.5 proc. prieš *Let's Encrypt* įkūrimą. Šis skaičius pastoviai auga, siekdamas maždaug vieną procentą per mėnesį. *SSL* sertifikatus naudojančių tinklalapių kiekio augimas bendrame kontekste pavaizduotas paveiksle 3.5.1, paruoštame pagal „Mozilla" iliustraciją [34].



3.5.1 pav. *Firefox* naršyklėje lankomų tinklalapių dalis, naudojanti *SSL* sertifikatus [34]

Kad prieglobos paslaugomis besinaudojantys vartotojai galėtų įdiegti *Let's Encrypt SSL* sertifikatą vieno mygtuko paspaudimu *DirectAdmin* valdymo skyde, buvo sukurtas *Let's Encrypt* klientas, parašytas bash scenarijų kalba, kuris automatizuoja sertifikato užklausos, įdiegimo, konfigūravimo ir nuolatinio atnaujinimo procesus. Serverių administratoriai papildomai taip pat gali vieno mygtuko paspaudimu įdiegti *SSL* sertifikatą el. pašto šifruotam ryšio kanalui palaikyti (*SMTP*, *IMAP* ir *POP3* protokolai), *FTP* serveriui, *DirectAdmin* valdymo skydui (*HTTPS* prievadas 2222). Sertifikatai yra automatiškai atnaujinami kas 60 dienų, kaip rekomenduojama *Let's Encrypt* dokumentacijoje. Pagal nutylėjimą *Let's Encrypt* sertifikatai galioja 90 dienų.

3.6. Automatinis kenkėjiškų failų skanavimas *FTP* ir *PHP* įkėlimuose

Prieglobos paslaugas teikiančios įmonės daug laiko skiria kenkėjiškų failų sukeltais padariniams likviduoti, tai gali būti neprivilegijuota prieiga prie sistemos failų, brukalo siuntimas elektroniniu paštu, konfidencialių duomenų atskleidimas. Kad kenkėjiški failai nepatektų į sistemą, buvo sukurti metodai, kurie užtikrina failų skanavimą jų įkėlimo metu naudojant *FTP* arba *PHP*. Failų skanavimui pasirinkta *ClamAV* antivirusinė sistema. *PHP* įkeliamiems failams skanuoti buvo panaudotas suhosin *PHP* modulis, galintis vykdyti scenarijus failų įkėlimo metu, naudojant „suhosin.upload.verification_script" nustatymą. Pats *suhosin* modulis tiesiog neleidžia įkelti failo, jeigu patikrinimo scenarijus grąžina 0, kitu atveju įkėlimas leidžiamas. Scenarijaus kodas yra pavaizduotas 3.6.1 paveiksle.

```
#!/bin/sh

# Make the script CageFS compatible
if [ -e ~/.cagefs$1 ]; then
        USER_HOMEDIR=`echo ~`
        FILE=${USER_HOMEDIR}/.cagefs$1
else
        FILE=$1
fi

if [ -n "`/usr/local/bin/clamdscan --infected --no-summary ${FILE}`" ]; then
        echo 0
else
        echo 1
fi
exit 0
```

3.6.1 pav. Kenkėjiškų failų skanavimo scenarijus naudojantis *ClamAV*

Kadangi *CloudLinux* sistemoje naudojant *CageFS* vartotojų failų sistemų atskyrimo funkcija laikinoji „/tmp" direktorija yra saugoma vartotojo aplinkoje, scenarijus buvo parašytas taip, kad būtų pilnas suderinamumas izoliuotoje vartotojo aplinkoje.

Panašus metodas naudojamas ir kenkėjiškų failų aptikimui įkėlimo metu į *Pure-FTPd FTP* serverį. Pats *Pure-FTPd* serveris sukompiliuojamas naudojant „--with-uploadscript" konfigūravimo vėliavėlę, o scenarijus įterpiamas konfigūracijoje numatytoje vietoje. Skirtingai nei *Pure-FTPd*, *ProFTPd* atveju naudojamas *mod_clamav* modulis, pilnai automatizuojantis skanavimą.

3.7. Saugus *Cronjob* periodinių užklausų vykdymas

Vartotojo failų sistemos izoliavimui (angl. *chroot*) buvo parašytas scenarijus *bash* kalba, kuris pagal konfigūracijos failuose nurodytus vykdomuosius failus juos perkelia į bendrą aplanką sistemoje, kartu su vykdomųjų failų reikalaujamomis bibliotekomis. Bibliotekos yra randamos įvykdžius *ldd* kvietimą vykdomajam failui. Virtualizuotoje failų sistemoje taip pat sugeneruojami atitinkami *passwd, group* ir *shadow* failai, būtini sklandžiam OS veikimui izoliuotoje vartotojo aplinkoje. Kai kurios programos reikalauja virtualių įrenginių */dev* skirsnyje, todėl taip pat sukuriami */dev/random, /dev/tty, /dev/ptmx* ir */dev/zero* įrenginiai.

Vartotojų virtualių failų sistemoms sukurti buvo panaudotas *bind mounts* [32] funkcionalumas teikiamas *Linux* operacinės sistemos branduolio. Šis funkcionalumas egzistuoja nuo 2.4.0 *Linux* branduolio (angl. *kernel*) versijos. Naudojant *bind mounts* skaidinių prijungimo būdą galima nekopijuojant failų vieną aplanką sistemoje prijungti (angl. *mount*) kitoje failų sistemos vietoje. Tokiu būdu vartotojo aplanke */home/vartotojas* galima prijungti bendrame sistemos aplanke esančias direktorijas, kuriose pirmame šio skyrelio etape patalpinome vykdomuosius ir vartotojo aplinkos failus. Taip yra išvengiama nepatogaus (dėl vykdomųjų failų įtraukimo, atnaujinimo) ir daug disko vietos užimančio vykdomųjų falų dubliavimo vartotojų namų kataloguose

Tinkamai prijungus skaidinius reikia panaudoti tarpinę programėlę (angl. *wrapper*), per kurią būtų vykdomos komandinės eilutės užklausos. Šiam tikslui buvo panaudota kompanijos „Ixia Communications" tarpinė programėlė „chrootshell.c" [35], kuri parašyta *C* programavimo kalba.

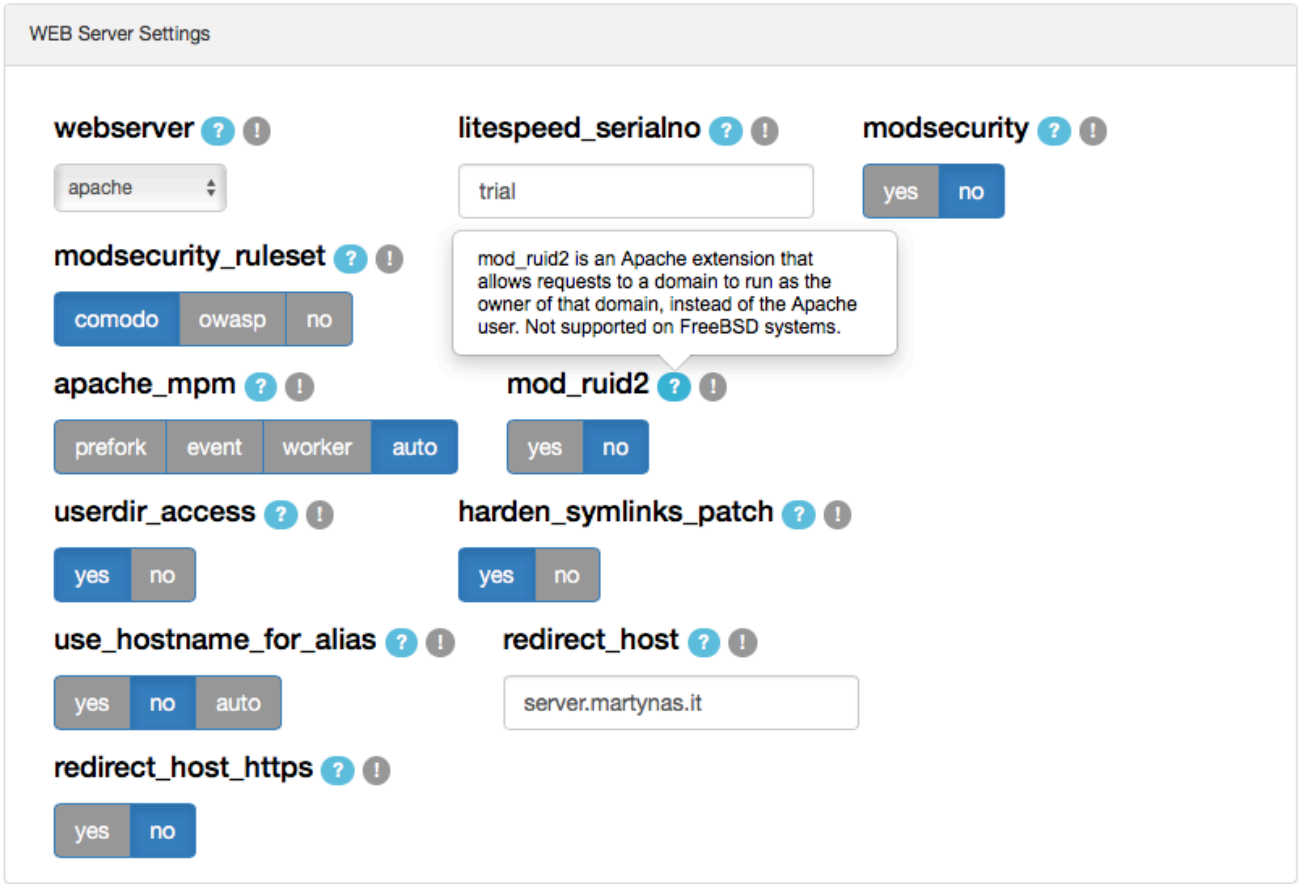
Kiekvieno periodinių užklausų failo viršuje pagal nutylėjimą bus pridedama *SHELL* aplinką keičianti eilutė „SHELL=/bin/jail". Tokiu būdu vykdant periodines *Cronjob* užklausas yra užkertamas kelias atlikti veiksmus išeinant už vartotojo aplanko ribų. Visi veiksmai yra atliekami izoliuotoje vartotojo failų sistemoje, kuri yra virtualizuota.

3.8. *Saugi* CustomBuild grafinė sąsaja su galimybe iškviesti automatines saugumo funkcijas

CustomBuild grafinė sąsaja kuriama kaip *DirectAdmin* valdymo skydo įskiepis (angl. *plugin*). Įskiepis parašytas *PHP* programavimo kalba. Kad būtų išvengta žalos net aptikus saugumo spragas, įskiepis yra vykdomas naudojant *cb_plugin* vartotojo teises. Atliekant veiksmus, įskiepis kreipiasi į

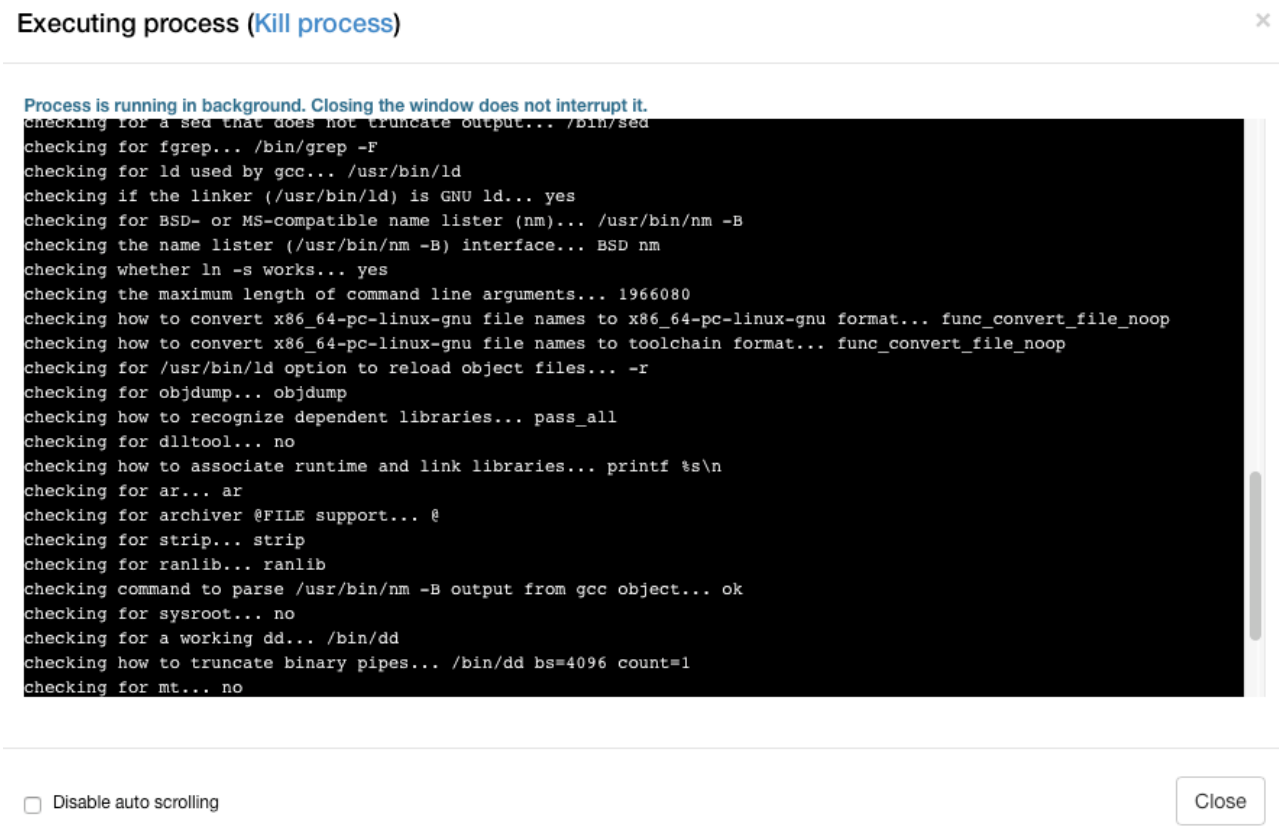
tarpinę programėlę (angl. *wrapper*), kuri leidžia vykdyti komandas kviečiant tik *CustomBuild* vykdomąjį failą, jam perduodant vykdymo argumentus. Prieš komandų vykdymą yra išvalomi aplinkos (angl. *environment*) kintamieji, o vykdant komandas tarpinė programėlė tikrina ar vykdytojas yra *cb_plugin* vartotojas. Pati tarpinė programėlė gali vykdyti *CustomBuild* komandas *root* vartotojo teisėmis, nes vykdomasis failas turi nustatytą *SUID* bitą, kuris leidžia vykdomąjį failą vykdyti jo savininko teisėmis. Įgyvendinus visas išvardintas priemonės, net aptikus saugumo spragą įskiepyje, nebūtų galima įgyti privilegijuotų teisių, o visi veiksmai būtų vykdomi neprivilegijuoto vartotojo *cb_plugin* teisėmis, todėl žalos sistemai nebūtų galima padaryti.

Pateikimui naršyklėje panaudota *jQuery JavaScript* biblioteka, *HTML* kompiuterinė žymėjimo kalba, *BootStrap* karkasas grafinei sąsajai. Visi veiksmai su konfigūracijos failais, sistemos nustatymais yra vykdomi per *CustomBuild* sistemoje sukurtą API sąsają. Galimų atlikti veiksmų pateikimas pagal esamus nustatymus taip pat yra pateikiamas *CustomBuild* sistemos per API sąsają. Dėl šių priežasčių *cb_plugin* vartotojui nėra reikalingos teisės prieiti ar redaguoti sistemoje esančius failus, be to, atsinaujinus *CustomBuild* programinei įrangai, nereikia atnaujinti įskiepio, nes jis atvaizduoja naujus elementus ir nustatymus automatiškai. Nustatymų lango, žiniatinklio serverio dalies, įskiepio ekrano kopija pateikiama paveiksle 3.8.1.



3.8.1 pav. *CustomBuild* įskiepio nustatymų lango dalies ekrano kopija

Kiekvienas aktyvus nustatymas yra pažymėtas mėlyna spalva, šalia nustatymų pavadinimų, kurie yra identiški nustatytiems konfigūracijos faile, yra detaliai nurodoma kam skirtas nustatymas ir kokia jo reikšmė pagal nutylėjimą. Programinės įrangos diegimo ir atnaujinimo metu yra rodoma išvestis vykdymo metu, kuri būtų matoma terminalo lange. Taip galima pastebėti kokia vykdymo eiga, matyti ar diegimas buvo sėkmingas. PĮ diegimo metu matomo lango ekrano kopija pateikiama paveiksle 3.8.2.



3.8.2 pav. *CustomBuild* įskiepio PĮ diegimo dalies ekrano kopija

PĮ diegimo ar atnaujinimo metu išjungus langą, kuriame matoma proceso eigos būsena, visa išvestis yra ir toliau saugoma į žurnalo (angl. *log*) failą ir gali būti peržiūrėta bet kuriuo metu, diegimo ar atnaujinimo procesas uždarius langą neužsibaigia, nes veiksmai yra vykdomi antrame plane (angl. *background*). Norint nutraukti vykdomą procesą, tai galima padaryti vieno mygtuko paspaudimu įskiepyje.

3.9. Realizacijos dalies išvados

Realizacijos etape buvo įgyvendinti visi projekto vykdymo pradžioje užsibrėžti tikslai. Patobulinta automatizuoto PĮ diegimo, atnaujinimo ir konfigūravimo sistema *CustomBuild*, palaikanti naujausias technologijas, siūlanti naujausias programinės įrangos versijas ir unikalius sprendimus, kurie anksčiau nebuvo naudojami. Žiniatinklyje pasiekiama grafinė sąsaja padaro įrankį labiau prieinamą nedaug apie sistemų administravimą išmanantiems naudotojams.

Ateityje sistemą būtų galima tobulinti išplečiant naujai sukurtus algoritmus, o atlikus plačios apimties apklausas dėl efektyviausių parametrų naujai sukurtuose sprendimuose, būtų galima juos taikyti pagal nutylėjimą. SSL sertifikatus galima pagal nutylėjimą įdiegti visiems naujai sukuriamiems domenams, o izoliuotą vartotojų failų sistemą ateityje galima panaudoti didesnėje dalyje procesų.

4. AUTOMATIZUOTO PĮ DIEGIMO, ATNAUJINIMO IR KONFIGŪRAVIMO SISTEMOS PAPILDYMŲ EKSPERIMENTINIS TYRIMAS

4.1. Išeinančio iš serverio brukalo prevencija

Sprendimo įvertinimui abu brukalo prevencijos metodai buvo išbandyti atskirai. Pasirinkti šie įvertinimo kriterijai:

- aptiktų bandymų išsiųsti brukalą kiekis;
- klaidingai teigiamų (angl. *false positive*) aptikimų kiekis.

El. laiškų išsiuntimo į neegzistuojančias el. pašto dėžutes kiekio ribojimas per valandą buvo pritaikytas atsitiktinai parinktame serveryje, kuris turi 688 prieglobos paslaugomis besinaudojančius vartotojus, ir 1801 domenus susietus su jais. Sprendimas buvo įdiegtas ir veikė septynias dienas, po to buvo patikrinti surinkti duomenys apie įvykdytus blokavimus ir laiką, kada blokavimas buvo įvykdytas.

Per 7 dienų laikotarpį buvo aptikta 10 bandymų siųsti brukalą iš 3 skirtingų prieglobos paslaugomis besinaudojančių vartotojų. Tai sudaro 0.44 proc. visų serveryje esančių vartotojų. Rezultatai yra pavaizduoti lentelėje 4.1.1. Aplanko stulpelyje galima pamatyti, kad kenkėjiški failai yra giliai paslepiami aplankų medyje, tokiu būdų sumažinant tikimybę, kad prieglobos paslaugomis besinaudojantis vartotojas savarankiškai jį suras. Be to, yra naudojami painūs failų pavadinimai, tokie kaip css.php ar blog.php, kurie vartotojui gali nesukelti įtarimo, nes panašūs failų vardai naudojami turinio valdymo sistemose ar jų įskiepiuose (angl. *plugins*). Datos stulpelyje nurodytas laikas kada buvo užblokuotas bandymas išsiųsti brukalą, taip pat matoma kiek blokavimų atliekama per tam tikrą laiko tarpą. Tikri domenų vardai yra paslėpti dėl konfidencialumo priežasčių.

4.1.1 lentelė. Užblokuotų kenkėjiškų PHP scenarijų sąrašas naudojant neegzistuojančių el. pašto adresų ribojimą

Aplankas	Data	Kenkėjiškas failas
/home/gertex/domains/secretdomain.pl/public_html/wp-content/uploads/wow-slider-plugin/9/tooltips	02 Nov 2015 12:08	css.php
/home/gertex/domains/secretdomain.pl/public_html/produkty	03 Nov 2015 03:46	blog.php
/home/technogrf/domains/secretdomain2.pl/public_html/wp-includes/pomo	03 Nov 2015 04:30	plugin74.php
/home/technogrf/domains/secretdomain2.pl/public_html/wp-includes/js/swfupload	03 Nov 2015 23:29	ini.php
/home/technogrf/domains/secretdomain2.pl/public_html/wp-admin/images	04 Nov 2015 20:34	global76.php
/home/gertex/domains/secretdomain.pl/public_html/wp-includes/js/mediaelement	05 Nov 2015 01:15	page26.php
/home/gertex/domains/secretdomain.pl/public_html/wp-content/uploads/2015/04	06 Nov 2015 08:09	test.php
/home/champion/domains/secretdomain3.pl/public_html/zarzadzanie	06 Nov 2015 14:42	error65.php
/home/gertex/domains/secretdomain.pl/public_html/cgi-bin	07 Nov 2015 03:31	error94.php
/home/technogrf/domains/secretdomain2.pl/public_html/wp-content/plugins/ubermenu/standard/styles	07 Nov 2015 06:19	help15.php

Visuose užblokuotuose aplankuose buvo aptikti failai turintys kenkėjišką turinį. Nebuvo aptikta nė vieno klaidingai teigiam aptikimo (angl. *false positive*). Tai leidžia daryti prielaidą, kad apsauga veikia tinkamai, o 100 neteisingų adresatų kiekis per valandą gali būti sumažintas, kad būtų sustabdyta dar daugiau bandymų išsiųsti brukalą. 1 bandymas siųsti brukalą iš 11 (9.1 proc.) nenaudojo *eval() PHP* kode. Nors tai yra nedidelis kiekis, tačiau jis yra pakankamas, kad serveris patektų į

juoduosius sąrašus ir duomenų centras imtųsi veiksmų izoliuoti serverį, iš kurio siunčiami laiškai (pvz. užblokuoti *IP* adresą ar *SMTP* prievadą). Faktas, kad 1 iš 11 bandymų išsiųsti brukalą savo kode nenaudojo *eval()* iškvietimo, patvirtina, kad reguliariųjų reiškinių *PHP* sąrašą reikės papildyti atsirandant vis naujiems metodams, o kad būtų užkirstas kelias nesant aptikimo metodui sąraše – reikia naudoti sudėtinį sprendimą, kuris aptinka tiek kenkėjiškus *PHP* scenarijus, tiek skaičiuoja kiek laiškų buvo bandoma išsiųsti neegzistuojantiems adresatams per valandą.

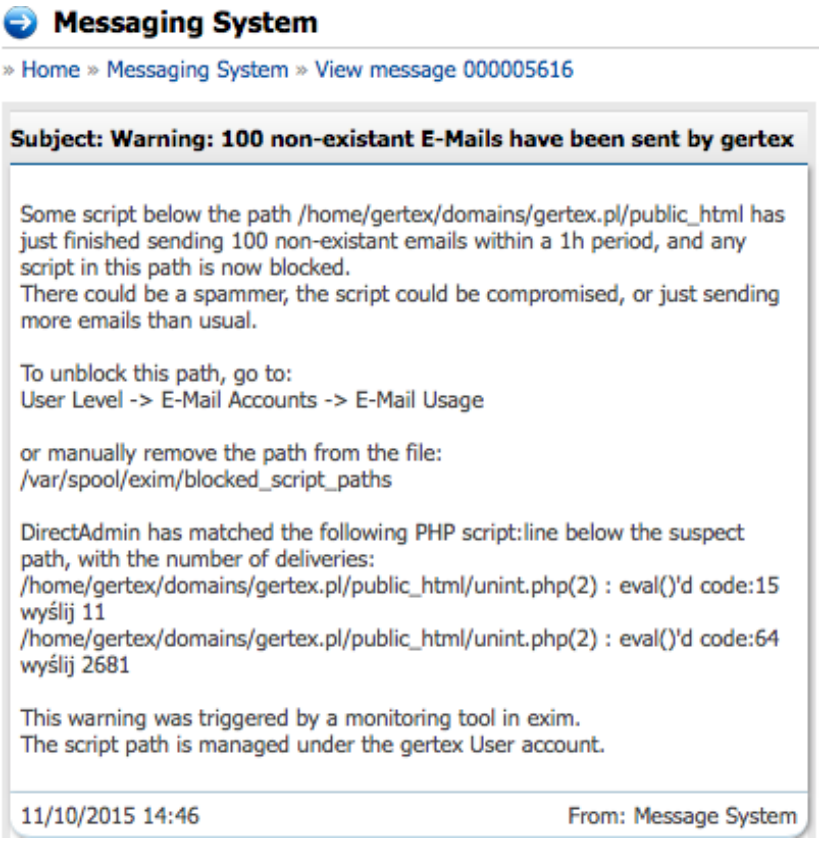
Kenkėjiškų *PHP* failų identifikavimo metodas buvo testuojamas kitame serveryje, todėl efektyvumo rezultatai nėra įtakojami prieš tai išbandyto metodo. Eksperimentui naudotame serveryje buvo 359 prieglobos paslaugas naudojantys vartotojai, kurie iš viso turėjo jiems priskirtus 1469 domenus. Bandymai taip pat buvo atliekami 7 dienas. Sprendimas sėkmingai aptiko ir užblokavo bandymus išsiųsti brukalą iš 3 vartotojams priklausančių aplankų. Tai sudaro 0.84 proc. visų vartotojų esančių serveryje. Rezultatai yra pavaizduoti lentelėje 4.1.2.

4.1.2 lentelė. Užblokuotų kenkėjiškų PHP scenarijų sąrašas identifikuojant kenkėjišką turinį

Aplankas	**Data**	**Kenkėjiškas failas**
/home/tagmaler/domains/secretdomain2.dk/public_html/wp-content/plugins/codestyling-localization	02 Nov 2015 15:19	ini.php
/home/vokalind/domains/secretdomain.dk/public_html/wp-content/themes/enfold/config-layerslider/LayerSlider/img	03 Nov 2015 08:37	help.php
/home/vokalind/domains/secretdomain.dk/public_html/wp-content/themes/twentythirteen/inc	03 Nov 2015 18:24	code25.php
/home/bedsteci/domains/secretdomain.lt/public_html/libraries/joomla/session	04 Nov 2015 22:13	default.php

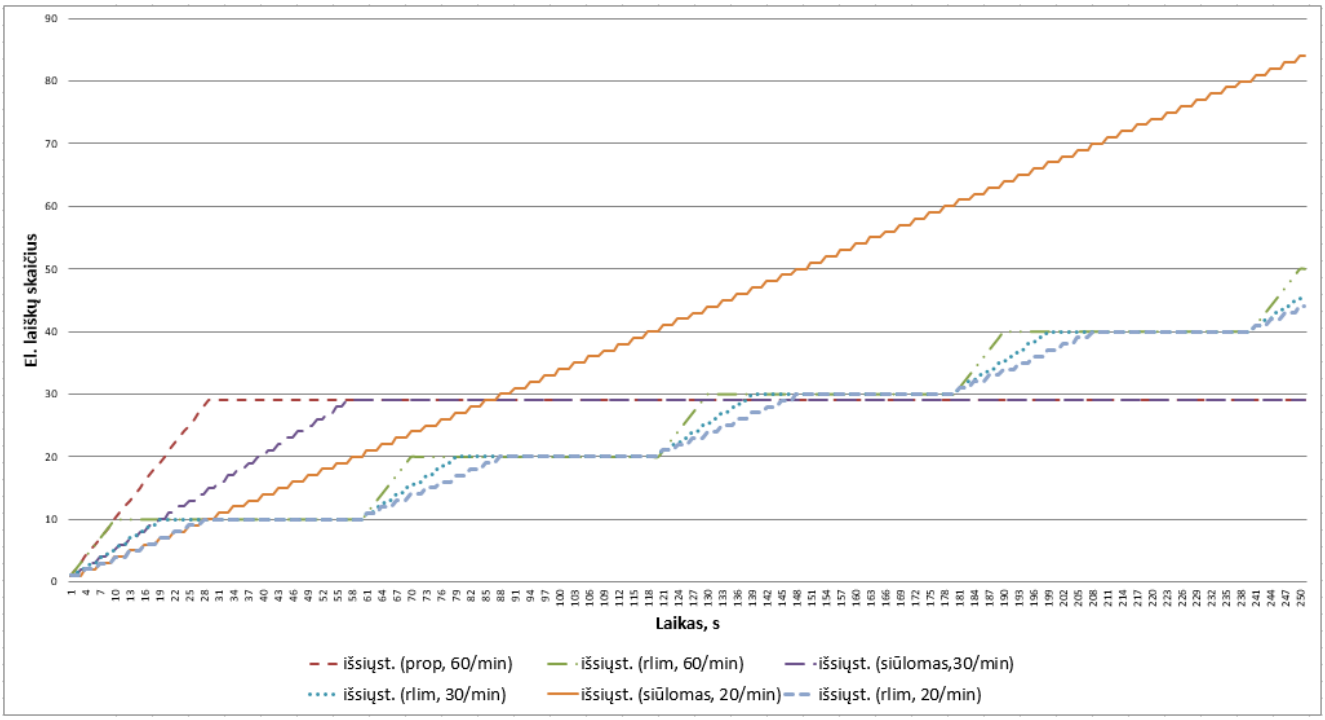
Kenkėjiškų *PHP* scenarijų identifikavimo metodas, kuris papildo siuntimo į neegzistuojančius adresatus metodą, taip pat neturėjo klaidingai teigiamų rezultatų.

Administratoriai ir galutiniai vartotojai buvo sėkmingai informuoti apie bandymus išsiųsti brukalą iš infekuotų tinklalapių. Kartu su pranešimu apie incidentą išsiunčiamos ir atblokavimo instrukcijos. Pranešimo pavyzdys yra parodytas paveiksle 4.1.1. Galutiniam vartotojui suteikti atblokavimo galimybę yra tikslinga, nes jis žinodamas apie incidentą gali pašalinti kenkėjiškus failus, atnaujinti turinio valdymo sistemą ir naudojamus įskiepius, o tada atblokuoti laiškų siuntimą. Tokiu būdų taip pat sutaupomas sistemų administratorių laikas. Labai retais atvejais vartotojai patys gali būti atsakingi už kenkėjiškus scenarijus ir brukalą, tačiau tokie atvejai gali būti labai greitai identifikuojami, nes sistemų administratoriai pakartotinai gautų pranešimus apie brukalą, siunčiamą iš to pačio sistemos vartotojo.



4.1.1 pav. Pranešimas *DirectAdmin* valdymo skyde apie aptiktus kenkėjiškus veiksmus

Metodo efektyvumas buvo palygintas su laiškų išsiuntimo per tam tikrą laiką limitu (angl. *rate-limit*) – plačiausiai naudojamu brukalo prevencijos būdų. Palyginimas buvo atliktas nustačius 10 laiškų per minutę limitą ir atskirai tikrinant kiekvieno metodo efektyvumą siunčiant skirtingus kiekius laiškų kas minutę. El. pašto adresų sąrašuose buvo atsitiktinių būdu įmaišyta 1/3 neegzistuojančių adresatų. Rezultatai yra matomi paveiksle 4.1.2.



4.1.2 pav. Siūlomo metodo ir *rate-limit* metodo palyginimas

Kaip matoma iš paveikslo 4.1.2, ribojimo per laiko tarpą (angl. *rate-limit*) metodas turi privalumų, kuomet siunčiamas mažas kiekis brukalo iš neatpažinto kenkėjiško *PHP* scenarijaus. Šiuo atveju neegzistuojančių adresatų kiekis per valandą nepasiekia metode nurodyto kiekio. Visais kitais atvejais sprendimas veikė taip pat gerai arba geriau nei *rate-limit* metodas.

Eksperimentiniai rezultatai parodė, kad nebuvo klaidingai teigiamų (angl. *false positive*) kenkėjiškų *PHP* scenarijų aptikimų, metodas veikė kaip tikėtasi ir tiksliai aptiko bei blokavo scenarijus, naudojamus siųsti brukalui iš prieglobos paslaugoms naudojamų serverių. Nors sprendimas efektyviai blokavo siuntimus ir užkrito kelią patekimui į juoduosius sąrašus, klaidingai teigiamų aptikimų nebuvimas leidžia daryti prielaidą, kad, priklausomai nuo aplinkos, gali būti taikomi dar griežtesni limitai el. laiškų siuntimui į neegzistuojančius adresatus.

4.2. Įeinančio į serverį brukalo prevencija

Sprendimo įvertinimui pagal surinktą nepageidaujamų el. pašto laiškų rinkinį buvo sudaryti alternatyvūs laiškai (su vienoda siunčiančio serverio konfigūracija), pakeičiant tik galutinį adresatą. Buvo išsiųsta 10 skirtingų el. laiškų ir tikrinamas atmetimo kiekis. 5 iš 10 laiškų buvo atmesti *SMTP* metu net nevykdant turinio skanavimo, pritaikant *DNS* požymių patikrinimus, o tai reiškia, kad vien taikant šį sprendimą galima ženkliai sutaupyti serverio resursus, išnaudojamus el. laiškų turinio analizei. 3 iš 10 laiškų buvo perduoti turinio analizei *SMTP* metu ir buvo atmesti, o 2 iš laiškų pateko į el. pašto laiškų eilę ir buvo pristatyti galutiniams adresatams, kadangi nebuvo pasiekta taškų riba atmetimui, o *SpamAssassin* turinio analizės įrankis net naudojant *DCC* ir *Razor2* metodus, neatpažino laiškų kaip brukalo. Vieno iš šių laiškų antraštė yra pavaizduota paveiksle 4.2.1.

```
Return-Path: <undertruck@unrotated48.win>
Delivered-To: michel@webawere.com
Received: from linux24.webawere.nl
     by linux24.webawere.nl (Dovecot) with LMTP id ncE/JMWIDldxAwAA/+K40w
     for <michel@webawere.com>; Wed, 13 Apr 2016 19:58:30 +0200
Return-path: <undertruck@unrotated48.win>
Received: from mail by linux24.webawere.nl with spam-scanned (Exim 4.86.2)
     (envelope-from <undertruck@unrotated48.win>)
     id 1aqP3b-00061Z-Vg
     for nic@webawere.com; Wed, 13 Apr 2016 19:58:30 +0200
X-Spam-Checker-Version: SpamAssassin 3.4.1 (2015-04-28) on linux24.webawere.nl
X-Spam-Level: *
X-Spam-Status: No, score=1.3 required=5.0 tests=HTML_MESSAGE,RDNS_NONE,
     SPF_HELO_PASS,SPF_PASS,T_REMOTE_IMAGE autolearn=no autolearn_force=no
     version=3.4.1
X-Spam-DCC: URT: linux24.webawere.nl 1060; Body=2 Fuz1=127 Fuz2=86
Received: from [104.148.109.31] (helo=unrotated48.win)
     by linux24.webawere.nl with esmtp (Exim 4.86.2)
     (envelope-from <undertruck@unrotated48.win>)
     id 1aqP3Z-0005zR-7y
     for nic@webawere.com; Wed, 13 Apr 2016 19:58:27 +0200
From: "CNN News" <undertruck@unrotated48.win>
Date: Wed, 13 Apr 2016 12:57:01 -0500
MIME-Version: 1.0
Subject: CNN Report Trump Drops a BOMB
To: <nic@webawere.com>
Message-ID: <gNhBOKWHNYp5AouGxL7E1YDQTCJ4wL0IJqoiSpbD3qI.elhsaCExJ95o3g1IBJIiin
Content-Type: multipart/alternative;
     boundary="------------963267903772270616351 96"
SPFCheck: Server passes SPF test, -30 Spam score
ReverseDNS: No reverse DNS for mailserver at 104.148.109.31, +100 Spam score
SpamTally: Final spam score: 70
X-Antivirus-Scanner: Clean mail though you should still use an Antivirus
```

4.2.1 pav. El. laiškas surinkęs 70 taškų įeinančio brukalo filtravime SMTP metu

Nors 4.2.1 paveiksle pavaizduoto laiško antraštėje matome, kad atgalinio *DNS* įrašo *IP* adresas neturėjo, tačiau el. laiško siuntėjas turėjo teisingai sukonfigūruotą SPF įrašą DNS zonoje ir nebuvo pakliuvęs į juoduosius sąrašus. Todėl galutinė taškų suma buvo 70. Tokių laiškų kiekio sumažinimui serveriuose būtų patartina įrankį sukonfigūruoti taip, kad galutinė taškų riba būtų žemesnė, o už *DKIM* įrašo neturėjimą zonoje būtų pridedamas papildomai nedidelis kiekis taškų, pvz. 15.

Eksperimentiniai rezultatai parodė, kad metodas veikė kaip numatyta ir atmetė el. laiškus, kurie užtikrintai buvo atpažinti kaip brukalas. Taip metodas sumažino į el. laiškų eilę patekusių laiškų kiekį nuo 10 iki 2, kurie turinio analizės metu nebuvo aptikti kaip brukalas ir pristatyti galutiniam gavėjui. Nors sprendimas pakankamai efektyviai aptiko brukalą, sistemų administratoriams patartina pagal poreikį ir gaunamo brukalo kiekį papildomai konfigūruoti papildomų taškų ribas, nustatant griežtesnius limitus negu nustatyta pagal nutylėjimą.

4.3. Saugus *PHP* vykdymas

Pagal numatytus reikalavimus, teisingai sukonfigūruotas *Apache* serveris turi vykdyti *PHP* kodą tam tikrame aplanke nurodyto vartotojo teisėmis. Eksperimentas *PHP* vykdymo teisėms išbandyti buvo atliktas naudojant *Apache* serverio konfigūraciją pavaizduotą 4.3.1 paveiksle.

```
Alias /phpMyAdmin /var/www/html/phpMyAdmin

<Directory /var/www/html>
        ...
        <IfModule mod_fcgid.c>
                FcgidWrapper /usr/local/safe-bin/fcgid56.sh .php
                SuexecUserGroup webapps webapps
                ...
        </IfModule>
</Directory>

<VirtualHost 92.222.207.16:80 [2001:41d0:2:7a8a:0:0:0:1]:80 >
        ServerName www.testing.hard.lt
        ServerAlias www.testing.hard.lt testing.hard.lt
        DocumentRoot /home/admin/domains/testing.hard.lt/public_html
        <IfModule !mod_ruid2.c>
                SuexecUserGroup admin admin
        </IfModule>
        ...
</VirtualHost>
```

4.3.1 pav. Apache konfigūravimas SuExec teisių nustatymui aplanke

Pagal konfigūraciją nurodyta 4.3.1 paveiksle, naršant *http://testing.hard.lt*, kodas turi būti vykdomas *admin* vartotojo teisėmis, tačiau apsilankius *http://testing.hard.lt/phpMyAdmin* kodas turi būti vykdomas *webapps* vartotojo teisėmis. Tokiu būdu failai saugomi vartotojo kataloge vykdymo metu neturės tinkamų teisių nuskaityti failams iš */var/www/html/phpMyAdmin* aplanko.

Bandymas buvo atliekamas vykdant *PHP* funkciją *exec()* ir kviečiant komandą *id*, kuri parodo vartotoją, jo grupę ir priskirtus *ID*.

Rezultatai įvykdžius *id.php* failą:

- vartotojo namų aplanke, įvykdžius per *http://testing.hard.lt/id.php*: uid=501(admin) gid=502(admin) groups=502(admin);
- /var/www/html/phpMyAdmin aplanke, įvykdžius per *http://testing.hard.lt/phpMyAdmin/id.php*: uid=500(webapps) gid=501(webapps) groups=501(webapps).

Pagal rezultatus matoma, kad sprendimas veikia sėkmingai ir leidžia nustatyti vartotojo vykdymo teises pasirinktiems aplankams, taip išvengiant perteklinių teisių suteikimo aplankams, kurie yra globalūs, ir pasiekiami naudojant *Alias* direktyvą *Apache* žiniatinklio serveryje.

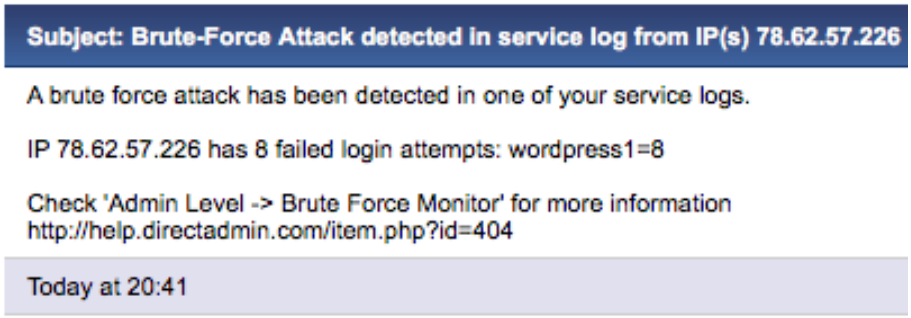
4.4. Viešai pasiekiamų aplikacijų apsaugojimas nuo atakų

Viešai pasiekiamų aplikacijų atsparumui brutalios jėgos ir žodyno atakoms išbandyti buvo imituojamos atakos *RoundCube* el. pašto klientui, *phpMyAdmin* duomenų bazių valdymo įrankiui ir *WordPress* turinio valdymo sistemai. Kiekvienam iš atvejų buvo atliekami daugiau nei 5 nesėkmingi prisijungimai iš to pačio IP adreso ir stebima ar gautas pranešimas apie *IP* adreso blokavimą. Paveiksle 4.4.1 matomi nesėkmingi prisijungimai (su *HTTP* būsenos kodu 200) prie *WordPress* turinio valdymo sistemos administratoriaus zonos.

```
78.62.57.226 - - [19/Apr/2016:20:36:59 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php?redirect_to=http%3A%2F%2Ftesting.hard.lt%2Fwp-
admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/601.
5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:37:35 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:36 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:41 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:43 +0200] "POST /wp-login.php HTTP/1.1" 200 1908
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:45 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:52 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
78.62.57.226 - - [19/Apr/2016:20:40:56 +0200] "POST /wp-login.php HTTP/1.1" 200 1935
 "http://testing.hard.lt/wp-login.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11
_4) AppleWebKit/601.5.17 (KHTML, like Gecko) Version/9.1 Safari/601.5.17"
```

4.4.1 pav. Nesėkmingi prisijungimai prie *WordPress* administratoriaus zonos

Aptikus daugiau nei 5 nesėkmingus prisijungimus, *DirectAdmin* valdymo skydas automatiškai užblokuoja atakuotojo *IP* adresą ir atsiunčia pranešimą apie bandymą įsibrauti. Pranešime nurodomas IP adresas ir tikslas (šiuo atveju *WordPress*). Pranešimo pavyzdys matomas paveiksle 4.4.2.



4.4.2 pav. *DirectAdmin* pranešimas apie nesėkmingus prisijungimus prie *WordPress* administratoriaus zonos

phpMyAdmin atveju, kuomet nepavykę prisijungimai yra saugomi */var/www/html/phpMyAdmin/log/auth.log* žurnalo faile, failas yra analizuojamas *DirectAdmin* valdymo skydo, o aptikus 5 ar daugiau nesėkmingų prisijungimų per valandą yra blokuojamas *IP* adresas ir siunčiamas analogiškas pranešimas į pavaizduotą 4.4.2 paveiksle. Pavyzdinis žurnalo failo turinys, kuomet buvo atlikti 6 nesėkmingi prisijungimai, pavaizduotas 4.4.3 paveiksle.

```
Apr 19 20:50:26:: pma auth user='bandoma_db' status='mysql-denied' ip='78.62.57.226'
Apr 19 20:50:31:: pma auth user='bandoma_db1' status='mysql-denied' ip='78.62.57.226'
Apr 19 20:50:35:: pma auth user='bandoma_db2' status='mysql-denied' ip='78.62.57.226'
Apr 19 20:50:37:: pma auth user='bandoma_db2' status='mysql-denied' ip='78.62.57.226'
Apr 19 20:50:41:: pma auth user='bandoma_db2' status='mysql-denied' ip='78.62.57.226'
Apr 19 20:51:18:: pma auth user='bandoma_db' status='mysql-denied' ip='78.62.57.226'
```

4.4.3 pav. Nesėkmingi prisijungimai prie *phpMyAdmin* duomenų bazės valdymo įrankio

Paveiksle 4.4.3 pavaizduoti bandymai atspėti *bandoma_db, bandoma_db1* ir *bandoma_db2* duomenų bazių vartotojų slaptažodžius. Visi jie buvo atlikti iš to pačio *IP* adreso, kuris po nesėkmingų bandymų buvo užblokuotas.

4.5. *Let's Encrypt SSL* sertifikatų diegimas

Let's Encrypt kliento veikimas buvo išbandytas testing.martynas.it domenui. Visų pirma buvo tikrinama ar sertifikato išdavimas ir konfigūravimas vyksta teisingai, tuomet buvo patikrintas automatinio sertifikato atnaujinimo kas 60 dienų veikimas. Vartotojo lygmenyje domenui testing.martynas.it buvo pasirinktas *Let's Encrypt SSL* sertifikato diegimas. *DirectAdmin* valdymo skydo atsakas apie *Let's Encrypt* sertifikato diegimą pavaizduotas paveiksle 4.5.1.

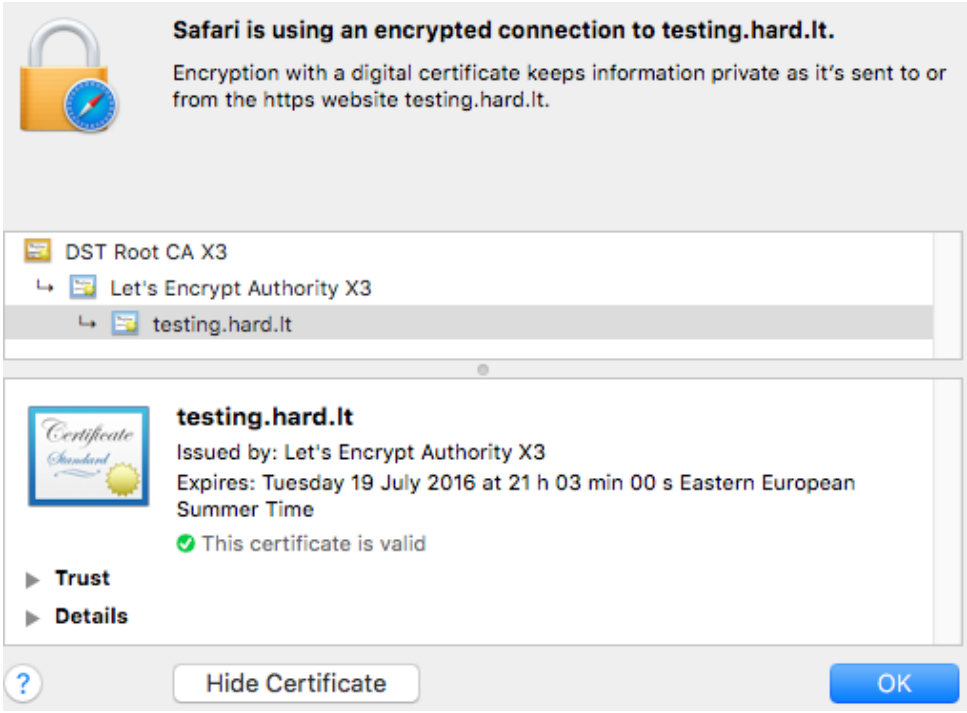
Certificate and Key Saved.

Details

Getting challenge for testing.hard.lt from acme-server...
Waiting for domain verification...
Challenge is valid.
Getting challenge for www.testing.hard.lt from acme-server...
Waiting for domain verification...
Challenge is valid.
Generating 2048 bit RSA key for testing.hard.lt...
openssl genrsa 2048 > "/usr/local/directadmin/data/users/admin/domains/testing.hard.lt.key.new"
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Certificate for testing.hard.lt has been created successfully!

4.5.1. Sėkmingas *Let's Encrypt SSL* sertifikato diegimas

Po sėkmingai sugeneruoto sertifikato vienu paspaudimu panaudojant *DirectAdmin* valdymo skydą, buvo patikrinta *Nginx* ir *Apache* serverių konfigūracija domenui *testing.hard.lt*, užtikrinanti *HTTPS* ryšio šifravimą naudojant *Let's Encrypt* sertifikatą. Neaptikus klaidų konfigūracijoje buvo patikrinta *Exim* ir *Dovecot* konfigūracija el. pašto saugai užtikrinti. Informacija apie sėkmingai naudojamą *SSL* sertifikatą domenui *testing.hard.lt* yra pateikiama paveiksle 4.5.2.



4.5.2. Informacija apie išduotą *Let's Encrypt SSL* sertifikatą

Taip pat buvo išbandytas automatinis sertifikatų atnaujinimas, kuris sėkmingai suveikė po 60 dienų.

4.6. Automatinis kenkėjiškų įkėlimų blokavimas *FTP* ir *PHP*

Bandymui atlikti buvo pasirinkti 5 kenkėjiški failai: „Eicar" standartinis antivirusinės programos bandymo failas, „C99 PHP Shell" ir „R57 PHP Shell", skirti *bash* komandų vykdymui naudojant *PHP* sąsają, ir du *PHP* failai, kurie buvo naudojami brukalui siųsti, aptikti 4.1 skyriuje atlikto bandymo metu. Plačiai naudojami kenkėjiški failai buvo atpažinti, o jų įkėlimas naudojant *FTP* protokolą buvo blokuotas. Sėkmingas „C99 PHP Shell" failo blokavimas matomas paveiksle 4.6.1.

```
Status:     Starting upload of /Users/macbookpro/Desktop/c99.php
Status:     Retrieving directory listing of "/domains/custombuild.eu/public_html"...
Command:    TYPE I
Response:   200 Type set to I
Command:    PASV
Response:   227 Entering Passive Mode (176,31,211,16,139,132).
Command:    MLSD
Response:   150 Opening BINARY mode data connection for MLSD
Response:   226 Transfer complete
Command:    TYPE A
Response:   200 Type set to A
Command:    PASV
Response:   227 Entering Passive Mode (176,31,211,16,140,158).
Command:    STOR c99.php
Response:   150 Opening ASCII mode data connection for c99.php
Response:   550-Virus Detected and Removed: Win.Trojan.C99-15
Response:   550 c99.php: Operation not permitted
Error:      File transfer failed
```

4.6.1 pav. Blokuotas kenkėjiškas failas *FTP* failo įkėlimo metu

Brukalui siųsti naudojami failai nebuvo atpažinti kaip kenkėjiški, todėl jų įkėlimas nebuvo blokuotas. Antivirusinės programos nėra pajėgios jų identifikuoti, nes brukalui siųsti naudojamų kenkėjiškų *PHP* scenarijų turinys yra pastoviai kintantis. Nepaisant to, kad antivirusinė programa neaptiko paskutinių dviejų failų kaip kenkėjiškų, didesnę žalą galintys sukelti kenkėjiški failai buvo aptikti, o tai sumažina riziką patirti nuostolius dėl kenkėjiškų veiksmų. Be to, neaptikta dalis failų gali būti sėkmingai identifikuoti anksčiau aptartuose sprendimuose.

4.7. Saugus *Cronjob* periodinių užklausų vykdymas

Išbandymui ar virtuali failų sistema yra izoliuota buvo atlikta eilė bandymų, kurių rezultatai parodė, kad jokie kiti ne vartotojui priklausantys failai nėra matomi. Buvo išbandytas *exec()* kvietimas *PHP* aplinkoje, kuris kreipėsi į */home* ir */usr/bin* aplankuose esančius failus ir vykdomuosius failus. Grąžintas rezultatas buvo tiesiog vartotojui priskirti failai, jokių pašalinių aplankų ir failų nebuvo atvaizduota. Taip pat buvo išbandyti sukurti produktai „unchroot.c" ir „unchroot.pl", padedantys išeiti iš izoliuotos aplinkos, tačiau jie nesuveikė. Tai leidžia teigti, kad vartotojo izoliuota aplinka yra saugi, o *Cronjob* periodinių užklausų vykdymas tapo saugesnis. Ši virtuali aplinka taip pat gali būti panaudota izoliuoti vartotojo *SSH* sesijai, *PHP* interpretatoriaus vykdymui.

4.8. Eksperimentinės dalies išvados

Eksperimento etape metodai buvo išbandyti ar veikia korektiškai, pagal įvertinimo kriterijus, nustatytus kiekvienam iš metodų. Įgyvendinti sprendimai veikė kaip tikėtasi, tačiau realioje prieglobos serverių aplinkoje, kad metodai veiktų efektyviau, gali reikėti nustatyti griežtesnes ribines vertes brukalo prevencijai sukurtų sprendimų konfigūracijos failuose. Koreguojant brukalo prevencijai skirtų algoritmų naudojamas el. laiškų atmetimą įtakojančias ribines vertes, lygiagrečiai turėtų būti vykdoma klaidingai teigiamų (angl. *false positive*) rezultatų patikra.

Eksperimentinių tyrimų rezultatai parodė, kad PĮ diegimo, atnaujinimo ir konfigūravimo sistema *CustomBuild 2.0* yra tinkama naudoti veikiančiuose (angl. *production*) prieglobos paslaugoms naudojamuose serveriuose.

5. DARBO REZULTATAI IR IŠVADOS

1. Atlikus populiariausių prieglobos (angl. *hosting*) paslaugoms teikti skirtų valdymo skydų analizę nustatyta, kad valdymo skydai PĮ diegimui naudoja sisteminius paketus, todėl negali užtikrinti greito PĮ atnaujinimų tiekimo, praplėtimų ir tinkinimo galimybių. Tačiau naudojant sisteminius paketus sutaupomi PĮ diegimo laikas ir žmogiškieji resursai, reikalingi paketų valdymo programinei įrangai kurti.
2. Atlikus pažeidžiamumų prieglobos paslaugas teikiančių serverių aplinkose analizę, identifikuotos saugos spragos, susijusios su mažu aplinkų atsparumu *DDoS* atakomis, brukalo siuntimu, galimybe vykdyti kenkėjiškus veiksmus serveriuose. Nustatyta, kad nei viename populiariausių prieglobos (angl. *hosting*) paslaugoms teikti skirtų valdymo skydų pažeidžiamumo klausimai nėra iki galo išspręsti, todėl dalis prieglobos paslaugų teikėjų yra neatsparūs įvairių tipo atakoms.
3. Išanalizavus galimus prieglobos paslaugas teikiančių serverių aplinkų pažeidžiamumus, darbe pateikti pasiūlymai, aprašantys kaip naudoti, konfigūruoti, diegti ir atnaujinti PĮ, kad prieglobos paslaugos būtų teikiamos saugiau. Identifikuotos modifikavimo reikalaujančios funkcijos, kurios atlieka komponentų, tokių kaip *Apache, ClamAV, Dovecot, Exim, PHP, mod_fcgid, LiteSpeed, MySQL/MariaDB, Nginx, Pigeonhole, ProFTPd, Pure-FTPd, phpMyAdmin, RoundCube* ir *SquirrelMail*, diegimą, atnaujinimą ir konfigūravimą. Grafinei sąsajai modifikuoti reikalingas tik esamo *CustomBuild* scenarijaus pakeitimas.
4. CustomBuild 2.0 kūrimo metu suprojektuotas naujų komponentų *ClamAV*, *suhosin*, *LiteSpeed*, *mod_aclr2*, *mod_ruid2*, *mod_htscanner2*, *mod_fcgid*, *Nginx*, *Pigeonhole, ModSecurity* palaikymas, modifikuotos dabartinių komponentų (*Apache*, *Dovecot*, *Exim*, *PHP*, *ProFTPd*, *Pure-FTPd*) diegimą atliekančios funkcijos ir jų pagal nutylėjimą įdiegiami konfigūracijos failai.
5. Pagal projektą realizuotas *DirectAdmin* įskiepis, prisitaikantis prie *CustomBuild 2.0* programinės įrangos teikiamo funkcionalumo, tad atsinaujinus *CustomBuild 2.0* funkcionalumui, įskiepio atnaujinti nereikia. Tai realizuota įskiepyje apdorojant *CustomBuild 2.0* scenarijaus sugeneruotas *JSON* formato ataskaitas. *DirectAdmin* įskiepis veikia neprivilegijuoto vartotojo *cb_plugin* teisėmis, todėl rizika dėl įsibrovimo į sistemą per įskiepį sumažėja iki minimumo.
6. Darbe pasiūlytas išeinančio iš serverio brukalo valdymo būdas, paremtas el. laiškų, išsiųstų į neegzistuojančias el. pašto dėžutes per valandą, kiekio analize ir ribojimu bei kenkėjiškų *PHP* scenarijų identifikavimu.
7. Eksperimentiniai rezultatai parodė, kad pasiūlytas būdas pateikia geresnius rezultatus nei plačiai naudojamas *rate-limit* metodas ir turi privalumų kai siunčiamas mažas brukalo kiekis iš neatpažinto kenkėjiško *PHP* scenarijaus. Taip pat realiose prieglobos serverių aplinkose, pasiūlyto būdo efektyvumo pagerinimui gali reikėti nustatyti griežtesnes, el. laiškų atmetimą įtakojančias, ribines vertes. Koreguojant šias vertes, lygiagrečiai turėtų būti vykdoma klaidingai teigiamų (angl. *false positive*) rezultatų patikra.
8. Darbe pateikti ir *CustomBuild 2.0* realizuoti kiti su saugesniu PĮ naudojimu, konfigūravimu, diegimu ir atnaujinimu susiję pasiūlymai buvo ištestuoti ir eksperimentiškai patikrinti. Gauti eksperimentiniai rezultatai leidžia daryti prielaidą, kad darbo autoriaus pasiūlymų įgyvendinimas eliminuoja didelę dalį analizėje aprašytų pažeidžiamumų ir padaro prieglobos paslaugas teikiančių serverių aplinką saugesne. Įgyvendinti pasiūlymai leidžia ne tik efektyviai valdyti el. pašto srityje kylančias problemas dėl brukalo, bet taip pat užkerta kelią gauti prieigą prie įvairių serverio resursų, blokuoja kenkėjiškų failų įkėlimą naudojant *FTP* ir *PHP,* sprendžia PĮ našumo, lygiagretumo, operatyvios atminties naudojimo klausimus *DDoS* atakų atvejais, suteikia perduodamų duomenų šifravimo galimybę.
9. Darbe pasiūlyti sprendimai buvo įdiegti prieglobos paslaugas teikiančių įmonių „WebaWere Internet Solutions B.V.” ir „Precept UK Ltd.” serveriuose. Jų taikymo rezultatai parodė, kad užtikrinama ne tik saugesnė aplinka, bet ir yra sumažinamas darbo jėgos poreikis, reikalingas

nuolat atsirandančioms problemoms spręsti (pvz. brukalo siuntimo šaltinio suradimas, jo izoliavimas ir pranešimas paslaugų naudotojui).

6. LITERATŪRA

[1] Netcraft Ltd., „April 2016 Web Server Survey“. [Tinkle]. Available: http://news.netcraft.com/archives/category/web-server-survey/. [Kreiptasi 24 balandžio 2016]

[2] Parallels IP Holdings GmbH, „Documentation“. [Tinkle]. Available: http://www.parallels.com/products/plesk/documentation/. [Kreiptasi 05 gruodžio 2015]

[3] cPanel Inc., „Overview“. [Tinkle]. Available: http://cpanel.net/cpanel-whm/. [Kreiptasi 05 gruodžio 2015]

[4] Symantec Corporation, „2016 Internet Security Threat Report”. [Tinkle]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf. [Kreiptasi 25 balandžio 2016]

[5] The PHP Group, „FastCGI Process Manager (FPM)“ . [Tinkle]. Available: http://www.php.net/manual/en/install.fpm.php. [Kreiptasi 15 gruodžio 2015]

[6] CloudLinux Inc., „What is CloudLinux OS?”. [Tinkle]. Available: https://www.cloudlinux.com/all-products/product-overview/cloudlinuxos. [Kreiptasi 25 balandžio 2016]

[7] Apache Software Foundation, „Apache Module mod_proxy_fcgi”. [Tinkle]. Available: https://httpd.apache.org/docs/2.4/mod/mod_proxy_fcgi.html. [Kreiptasi 25 balandžio 2016]

[8] Q-Success, „Usage of web servers broken down by ranking”. [Tinkle]. Available: http://w3techs.com/technologies/cross/web_server/ranking. [Kreiptasi 26 balandžio 2016]

[9] D. Remi, „A little holiday present: 10,000 reqs/sec with Nginx!”. [Tinkle]. Available: http://blog.webfaction.com/2008/12/a-little-holiday-present-10000-reqssec-with-nginx-2/. [Kreiptasi 25 gruodžio 2015]

[10] World Wide Web Consortium, „Persistent Connections”. [Tinkle]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html. [Kreiptasi 25 balandžio 2016]

[11] Andrey Belov, „Module for Apache 2.x which automates serving static content with NGINX local proxy”. [Tinkle]. Available: https://github.com/defanator/mod_aclr2. [Kreiptasi 26 balandžio 2016]

[12] I. Sysoev, „Nginx”. [Tinkle]. Available: http://nginx.org/en/. [Kreiptasi 21 gruodžio 2015]

[13] C. Nedelcu, „Nginx HTTP Server“, Packt Publishing Ltd., p. 217-263, 2010.

[14] W. Reese, „Nginx: the High-Performance Web Server and Reverse Proxy“. [Tinkle]. Available: http://www.linuxjournal.com/article/10108. [Kreiptasi 21 gruodžio 2015]

[15] LiteSpeed Technologies Inc., „PHP Hello World Benchmark“. [Tinkle]. Available: http://www.litespeedtech.com/products/litespeed-web-server/benchmarks/php-hello-world. [Kreiptasi 11 gruodžio 2015]

[16] Apache Software Foundation, „Apache MPM event“. [Tinkle]. Available: http://httpd.apache.org/docs/2.4/mod/event.html. [Kreiptasi 10 sausio 2015]

[17] I. Ristic, „Apache security : [the complete guide to securing your Apache web server]“, O'Reilly, p. 124-151, 2005.

[18] Kaspersky Lab, „Spam and phishing in Q2 2015” [Tinkle]. Available: https://securelist.com/analysis/quarterly-spam-reports/71759/spam-and-phishing-in-q2-of-2015/. [Kreiptasi 27 lapkričio 2015]

[19] WhiteHat, „Website Security Statistics Report” [Tinkle]. Available: https://www.whitehatsec.com/statistics-report/featured/2015/05/21/statsreport.html. [Kreiptasi 27 lapkričio 2015]

[20] Imperva, „Web Application Attack Report #5” [Tinkle]. Available: http://www.imperva.com/docs/hii_web_application_attack_report_ed5.pdf. [Kreiptasi 28 lapkričio 2015]

[21] Apache Software Foundation, „What Apache SpamAssassin Is” [Tinkle]. Available: http://svn.apache.org/repos/asf/spamassassin/branches/3.4/README. [Kreiptasi 28 lapkričio 2015]
[22] SPF Council, „What is SPF” [Tinkle]. Available: http://www.openspf.org/FAQ/What_is_SPF. [Kreiptasi 29 lapkričio 2015]
[23] D. Crocker, „DKIM Frequently Asked Questions” [Tinkle]. Available: http://www.dkim.org/info/dkim-faq.html. [Kreiptasi 29 lapkričio 2015]
[24] Google, „About DMARC“ [Tinkle]. Available: https://support.google.com/a/answer/2466580?hl=en. [Kreiptasi 29 lapkričio 2015]
[25] Hetzner, „Terms and Conditions” [Tinkle]. Available: https://www.hetzner.de/es/hosting/legal/agb. [Kreiptasi 1 gruodžio 2015]
[26] OVH, „Specific Terms and Conditions on the rental of a dedicated server” [Tinkle]. Available: https://www.ovh.com/us/support/termsofservice/Special_conditions_for_dedicated_server.pdf. [Kreiptasi 1 gruodžio 2015]
[27] cPanel Inc., „How to Prevent Email Abuse” [Tinkle]. Available: https://documentation.cpanel.net/display/CKB/How+to+Prevent+Email+Abuse. [Kreiptasi 1 gruodžio 2015]
[28] Parallels Inc., „Protection from Outbound Spam” [Tinkle]. Available: http://download1.parallels.com/Plesk/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=71349.htm. [Kreiptasi 1 gruodžio 2015]
[29] cPanel Inc., „Scan Outgoing Mail” [Tinkle]. Available: https://documentation.cpanel.net/display/ALD/Scan+Outgoing+Mail. [Kreiptasi 2 gruodžio 2015]
[30] S. Bosch, „Pigeonhole: Overview“. [Tinkle]. Available: http://pigeonhole.dovecot.org/. [Kreiptasi Kreiptasi 21 gruodžio 2015]
[31] O. Comber, „Easy, reliable spam fighting with Exim“. [Tinkle]. Available: http://olicomber.co.uk/blog/b/Easy,_reliable_spam_fighting_with_Exim. [Kreiptasi 15 lapkričio 2015]
[32] 1H Ltd, „Bind mounts”. [Tinkle]. Available: http://docs.1h.com/Bind_mounts. [Kreiptasi 26 balandžio 2016]
[33] University of Cambridge, „Exim Callout Verification” [Tinkle]. Available: http://www.exim.org/exim-html-current/doc/html/spec_html/ch-access_control_lists.html#SECTcallver. [Kreiptasi 7 gruodžio 2015]
[34] A. Greenberg, „A Scheme to Encrypt the Entire Web is Actually Working” [Tinkle]. Available: http://www.wired.com/2016/04/scheme-encrypt-entire-web-actually-working/. [Kreipasi 10 balandžio 2016]
[35] Ixia Communications, „chrootshell.c” [Tinkle]. Available: http://kegel.com/crosstool/current/chrootshell.c. [Kreipasi 27 balandžio 2016]
[36] Denis Frolov, „Configuring nginx as a front-end to apache” [Tinkle]. Available: https://www.opennet.ru/base/net/nginx_frontend_apache.txt.html. [Kreipasi 27 balandžio 2016]
[37] Florian Baumann, „Apache 2.4 und mpm_event” [Tinkle]. Available: https://noqqe.de/blog/2012/05/28/apache-2-dot-4-und-mpm-event/. [Kreipasi 27 balandžio 2016]
[38] Yoann Queret, „ Mail: How-to” [Tinkle]. Available:http://wiki.queret.net/docs/mail. [Kreipasi 27 balandžio 2016]

7. PRIEDAI

7.1. priedas. Straipsnis

Prevention of outbound unsolicited electronic mail

M. Bendorius[1], I. Lagzdinyte-Budnike[1], K. Paulikas[1], A. Budnikas[2]
[1]*Department of Applied Informatics, Kaunas University of Technology, Studentu St. 50–402a, LT-51368 Kaunas, Lithuania, phone: +370 652 19840*
[2]*Department of Telecommunications, Kaunas University of Technology, Studentu St. 50–424, LT-51368 Kaunas, Lithuania, phone: +370 652 21950*
ingrida.lagzdinyte@ktu.lt

***Abstract*—A novel outbound unsolicited electronic mail prevention solution is presented, which can be used in shared web hosting environments. It combines several new ideas, including the rate of non-existent recipient email addresses, the content of the file which called the mail function and automatic actions after detection of suspicious activity to isolate and block it. A complete solution was designed and implemented, all the algorithms were described and presented in the article. The evaluation was done by testing the efficiency of outbound unsolicited electronic mail detection and the number of incidents blocked.**

***Index Terms*— Electronic mail, message systems, unsolicited electronic mail.**

I. INTRODUCTION

Even in 2015, unsolicited electronic mail, also known as SPAM, not just continues to plague businesses and users, but also accounts for a noticeable amount of traffic in global networks. According to an international software security group Kaspersky Lab [1], in Q2 2015, the proportion of spam in email traffic was 53.4%.

A variety of different methods exist to send unsolicited electronic mail, however, one of the most popular ways is sending SPAM through existing vulnerable websites on web hosting servers. The reason behind this trend is the fact that 86% of all websites have at least one serious vulnerability-[2]. The report by WhiteHat Security also notes that most of the time the sites contain more than just a single vulnerability. A provider of cyber and data security products Imperva confirms WhiteHat indications in "Web Apllication Attack Report #5" [3] and additionally notes the increase of 24% in Remote File Inclusion (RFI) attacks and WordPress as the most common target for Content Management System (CMS) attacks.

Today's world is mostly concentrated on incoming spam filtering solutions like the most popular open source solution SpamAssassin [4], greylisting, blacklists, SMTP-time filtering and research on spam filters. Even though they help lowering the amount of received spam in customer's mailboxes, not all of the messages are successfully identified as spam. Moreover, it does not change the amount of spam in global networks.

Manuscript received April XX, 20XX; accepted April XX, 20XX.
This research was funded by a grant (No. XXX-00/0000) from the Research Council of Lithuania. This research was performed in cooperation with the Institution.

To filter spam, major email service providers check SPF [5] and DKIM [6] records [7], but that does not help if spam is sent from a PHP script in a web hosting account, configured with correct SPF and DKIM records. That is why attackers send SPAM from infected websites.

To lower the amount of spam in global networks and consequences caused, outbound spam prevention solutions must be used. In addition to, using outbound SPAM prevention provides other benefits for shared web hosting providers, including non-blacklisted IP addresses, good server reputation, less emails in mail queue, resulting in better quality of services for the end customer. Data centers usually have strict Terms of Service (ToS) for spam and take serious actions when they detect outbound spam from their network to save the reputation of their entire network. Actions may include null routing of the client's IP addresses [8] if outbound spam from the servers is detected, resulting in no availability of other services like HTTP, FTP or DNS, or a block for SMTP port 25 [9], resulting in no availability of SMTP service, which prevents delivery of normal email messages from shared web hosting servers.

II. TECHNIQUES TO PREVENT OUTBOUND SPAM

One of the most popular technique to prevent outbound spam on shared web hosting servers is rate limiting, offered by two of the most popular web hosting control panels cPanel [10] and Plesk [11]. The technique limits the number of email messages that could be sent in a particular amount of time for a specific email address or domain. It is effective with strict limits set, but has several disadvantages. Firstly, if a customer reaches the email rate limit set without sending any spam messages, all the additional outgoing messages will fail and the customer will be blocked from sending more mails for a particular amount of time. Secondly, when the period of time expires for the limit, the customer is able to send the same amount of messages again, including the spam messages. For example, if the rate limit is set to 50 messages per hour, when the hour limit expires, the customer can send another 50 messages. Thirdly, even if administrators are notified about customers who reached the limit, they do not indicate if that was a spam incident or not, and requires system administrators to review the incident manually.

Another approach offered by cPanel is outbound mail filtering using Apache SpamAssassin [12], however due to high resource usage, many false positives and insufficient

protection they do not suggest using it on production servers.

III. PROPOSED SOLUTION

Considering the experience in shared web hosting sphere and knowing most common ways used to send unsolicited electronic mail, we are offering a complete solution which automatically detects and blocks the attack. Our goal is to block email delivery from infected websites. The solution consists of several separate ideas: A) limiting the number of message submits to nonexistent email addresses per hour, B) analyzing the PHP scripts which were used to send mails. Both of the solutions ensure blocking further attempts to send mail from affected accounts or scripts directly in MTA.

A. Rate limiting of attempts to send emails to nonexistent mail addresses

Attackers usually use a large list of email addresses having a number of email addresses which never existed or do not exist anymore. The idea of rate limiting of attempts to send emails to nonexistent recipient email addresses is not new, it has been discussed on the Internet in 2010 by Andrew Hearn, CEO of Andrews & Arnold Ltd, providing email services for the masses, and some other places. However, they focus on authenticated SMTP users, and no complete solution was found for emails sent from PHP scripts on shared web hosting environments.

Our proposed solution allows to set custom rate limiting period and the number of nonexistent recipients, that way shared web hosting companies can tune the settings up for them to be stricter or more tolerant. In our further experiments the default limit was set to delivery of 100 emails to nonexistent email addresses in 1 hour until the delivery from a specific path gets blocked. The default limit was set to 100, because from various tests we have got some false-positives with a lower number, mainly from electronic shops having fake registrations for newsletters.

When the limit is reached, the full script execution path is added to the list of denied paths file automatically, the end-user (legal shared web hosting account owner) and system administrators get notified about the incident. Without a manual action from administrators or the end customers it is not possible to unlock the path, so attackers cannot send more emails even when the rate limiting period (1 hour) ends.

Recipient verification is based on SMTP callback verification offered by Mail Transfer Agents (MTA) [13]. In our case Exim was used as an MTA. To verify a sender address it connects to a remote SMTP server (set in DNS MX records), executes a standard HELO command required by the protocol, submits an empty MAIL FROM field and fills the RCPT TO field with the address to be tested, then it quits with a QUIT command. The remote server response to the RCPT command is a 2xx code meaning that verification succeeded, or a 5xx code, which means that verification failed. For any other condition the next host set in DNS MX records is tried (if there are set any). A successful callback does not guarantee a successful delivery, but a failing callout guarantees that delivery would fail. To save the resource usage by address verification, Exim caches the result of callbacks. The solution is represented in fig. 1.

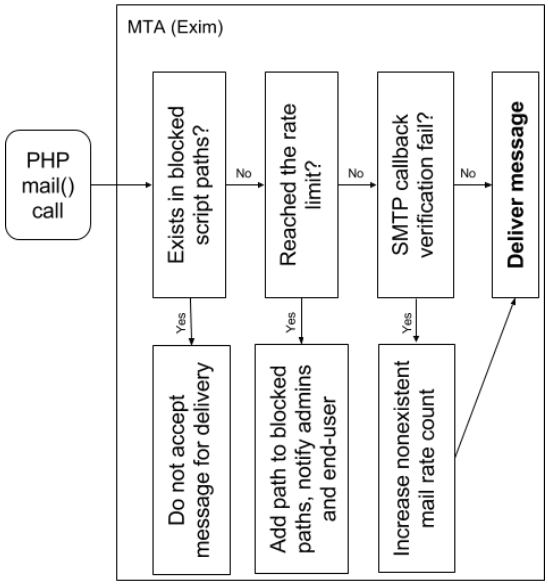


Fig. 1. Rate limiting solution for non-existent mails scheme

The proposed solution uses a time limit of 10 seconds for a timeout and a defer_ok setting [14], which makes the check to treat a failure of contacting any host or any other kind of temporary problem as success by the ACL. The structure of the callback (also known as callout) is represented in fig. 2.

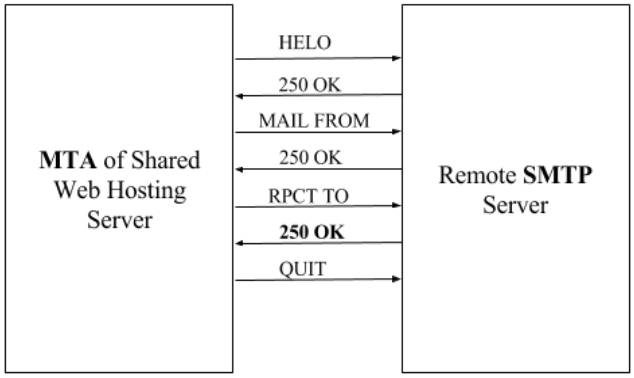


Fig. 2. SMTP callback scheme with a correct recipient

As seen in fig. 2., the response in bold after RPCT TO command returned 2xx code, meaning a verification success. The proposed solution only counts the email addresses that return code 5xx, meaning a verification failure.

B. IDENTIFICATION OF MALICIOUS PHP SCRIPTS

The previously described solution reflects only a part of the complete solution. Even though the previous solution stops many of the spam attempts, it still can be complemented. Attackers still have a way to send up to 100 unsolicited electronic mails from a directory on the system.

To reduce the number of mails sent a PHP code analysis of scripts which used the *mail()* function can be done. By analyzing spam scripts on a shared web hosting server, it was noted that most of the infected scripts use an encoded PHP code inside. In such case a *base64_decode()* PHP function is called to decode the code and *eval()* PHP language construct is used to execute the code. 5264 virtual hosts on different dedicated servers were scanned to see if there are any occurrences of *mail()*, *base64_decode()* and *eval()* usage in the same PHP script and it was found that none of normal PHP scripts had this combination used in a single PHP file.

Despite the fact PHP allows simply disabling PHP functions in PHP configuration file, *base64_decode()* and *eval()* are used in normal scripts too, so they could not be simply disabled, because that could break a number PHP applications on shared web hosting servers. In addition to, *eval()* is a language construct, so it is not possible to disable it in disable_functions PHP configuration setting.

To effectively detect scripts having *eval()* and *base64_decode()* in the same file, which was used to send mails using *mail()*, log scanning must be used. The reason behind that – the files are encoded using *base64_encode()* and that allows attackers to hide a *mail()* function call in their PHP scripts. However, PHP offers live logging setting for the usage *mail()* function [15], and this feature allows us to know which PHP script called the *mail()* function, then it can be simply scanned for the usage of *base64_encode()* function and *eval()* language construct. We cannot simply say that a PHP script, which has it's source code encoded and *mail()* function hidden, is a malicious script, because it is possible to encode PHP source code using ionCube or Zend encoders, and they help commercial solutions to legally hide their PHP source code.

Our proposed solution uses an application named Swatchdog (Simple Log Watcher) [16] to scan the logs in real time and parse the scripts having the *mail()* calls. If the file is detected to have *base64_decode()* and *eval()* inside, it's added to the list of denied paths and administrators together with the end-user are notified as shown in fig. 3.

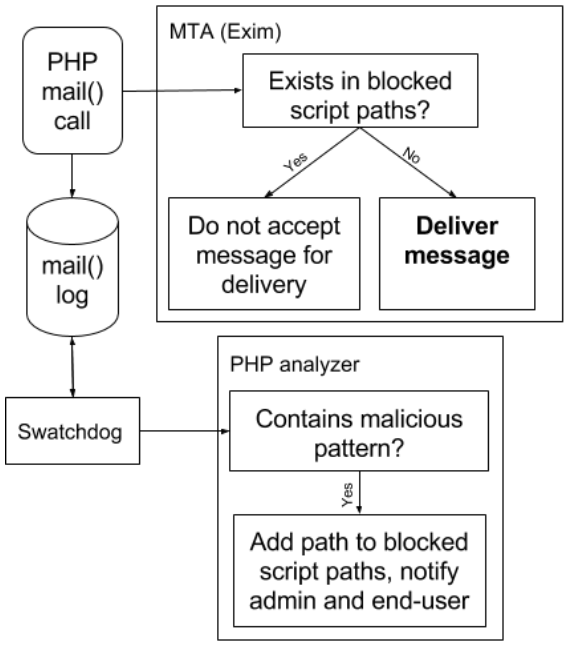


Fig. 3. PHP code analysis scheme

As seen in fig. 3, when PHP mail() function call happens, it instantly logs the usage of the function and submits the mail to MTA at the same time. This is the main reason why we cannot technically stop the first mail from being sent, because MTA and Swatchdog work independently from each other. However, Swatchdog checks the log in real time and passes the full path to the script, which analyses PHP code. If the script detects *eval()* and *base64_decode()* function in the script, which was used to sent the mail, it adds the path to the list of paths that are not allowed to send any mails from the system. The second try to send a mail using PHP will fail, because MTA will detect that the path is not allowed to send the mail and will refuses to deliver it.

When the automatic detection and blocking of spam delivery was started, a few more patterns used to hide PHP files were found and added to automatic blocking script:

- more than 10 occurrences of variable GLOBALS, *eval()* function called in the same file, *mail()* function hidden;
- GLOBALS, substr, return used on the same line, *mail()* function hidden;
- more than 10 occurrences of *chr()* function, *eval()* function called in the same file, *mail()* function hidden.

Even though the regular expression list for detection of malicious PHP scripts detected all the files that were used to send unsolicited electronic mail, we are sure that more patterns exist and even more ways to hide PHP code will be created in the future. The main drawback of this method is the need to constantly update the static list of regular expressions for the detection of malicious PHP scripts that are used to send spam.

Most of the time content management systems (CMS) have folders that are used for user-level file uploads. These folders are a common target for unrestricted file upload attacks [17]. To partially solve the main drawback of the method mentioned above, a global blocked script paths file can be used. That way MTA would check if the path from which the mails are sent has no matches in the regular expression list of paths. If a match is detected, delivery would be blocked. For example, it would be safe to block email sends with the following regular expressions for WordPress CMS:

- ^.*/wp-content/cache.*
- ^.*/wp-content/uploads.*

IV. EVALUATION

To evaluate the solution, both of the spam prevention solutions were tested separately.

Our qualitative characteristics for evaluation criteria were:

- detection of malicious attempts to send spam;
- number of false-positive detections.

The rate limiting of sends to non-existent email addresses solution was installed to a random production server having 688 shared web hosting accounts and 1801 domains setup on it. The solution was installed and left for a week to run. Then a file having the list of blocked paths was checked to see when and what was blocked.

In a period of 7 days, 10 attempts to send spam were blocked from 3 different web hosting accounts. That accounts for 0.44% of all the customers. The results are shown in table 1. From the directory names seen in directory column we can see that malicious files are usually placed deeply in subdirectories, so that it would be harder to detect them for the end-customer. In addition to, the name of the infected files is usually set to confuse the end-customers too, because file names like css.php or blog.php do not look strange, and it could be believed that the files are provided by the content management system or it's plugins. The date shown in the table indicates when the spam attempt was detected and

blocked, to evaluate the quantity of scripts blocked in a period of time. Real domain names were changed in the table for confidentiality purposes.

TABLE I. LIST OF BLOCKED PATHS

Directory	Date	Infected file
/home/gertex/domains/secretdomain.pl/public_html/wp-content/uploads/wow-slider-plugin/9/tooltips	2 Nov 2015 12:08	css.php
/home/gertex/domains/secretdomain.pl/public_html/ /produkty	3 Nov 2015 03:46	blog.php
/home/technogrf/domains/secretdomain2.pl/ /public_html/wp-includes/pomo	3 Nov 2015 04:30	plugin74.php
/home/technogrf/domains/secretdomain2.pl/ /public_html/wp-includes/js/swfupload	3 Nov 2015 23:29	ini.php
/home/technogrf/domains/secretdomain2.pl/ /public_html/wp-admin/images	4 Nov 2015 20:34	global76.php
/home/gertex/domains/secretdomain.pl/ /public_html/wp-includes/js/mediaelement	5 Nov 2015 01:15	page26.php
/home/gertex/domains/secretdomain.pl/public_html/ /wp-content/uploads/2015/04	6 Nov 2015 08:09	test.php
/home/champion/domains/secretdomain3.pl/public_html/zarzadzanie	6 Nov 2015 14:42	error65.php
/home/gertex/domains/secretdomain.pl/ /public_html/cgi-bin	7 Nov 2015 03:31	error94.php
/home/technogrf/domains/secretdomain2.pl/ /public_html/wp-content/plugins/ubermenu/standard/styles	7 Nov 2015 06:19	help15.php

All of the blocked paths had malicious content inside and there were no false positives. That indicates that the protection works and that the threshold of 100 sends to non-existent emails per-hour could be lowered to catch more spam attempts. 1 attempt to spam of 11 (9.1%) used no *base64_decode()* and *eval()* in their PHP code. Even though it is a low amount, it is usually enough for the server to get blacklisted and for the actions to be taken from the data center (null route the IP address or block SMTP port). The fact confirms that there is a need of a complete combined solution, which detects both the usage of malicious PHP scripts and rate limit of sends to non-existent email addresses.

The identification of malicious PHP scripts method was tested on a separate server to get the efficiency results without any influence from the previous detection method. The server had 359 active shared webhosting accounts owning 1469 domains. Test results were also taken from a time period of a week. It was found that the spam affected 3 different customers and the spam attack was blocked. That accounts for 0.84% of all the customers on the server. The results from blocked paths file are shown in table 2.

TABLE 2. LIST OF BLOCKED PATHS

Directory	Date	Infected file
/home/tagmaler/domains/secretdomain2.dk/ /public_html/wp-content/plugins/codestyling-localization	2 Nov 2015 15:19	ini.php
/home/vokalind/domains/secretdomain.dk/ /public_html/wp-content/themes/enfold/config-layerslider/LayerSlider/img	3 Nov 2015 08:37	help.php
/home/vokalind/domains/secretdomain.dk/ /public_html/wp-content/themes/twentythirteen/inc	3 Nov 2015 18:24	code25.php
/home/bedsteci/domains/secretdomain.info/ /public_html/libraries/joomla/session	4 Nov 2015 22:13	default.php

The identification of malicious PHP scripts method, which complements the first method of rate limiting sends to non-existent email accounts, also had no false positives and malicious scripts were found in all the paths reported.

Administrators and end-customers were successfully informed about the attempts to send unsolicited electronic mails from affected websites. Instructions for unblocking the path were also provided. One of the notifications is shown in fig. 4. It is beneficial to inform the end-customer about the incident and allow them to unblock themselves, because they can clean their websites up by themselves, upgrade their content management system and plugins used with it, without any actions done from system administrators. In very rare cases it could be the customers themselves responsible for the spam, however, such cases can be quickly identified, because administrators would repeatedly receive messages from the system about the spam originating from the customer.

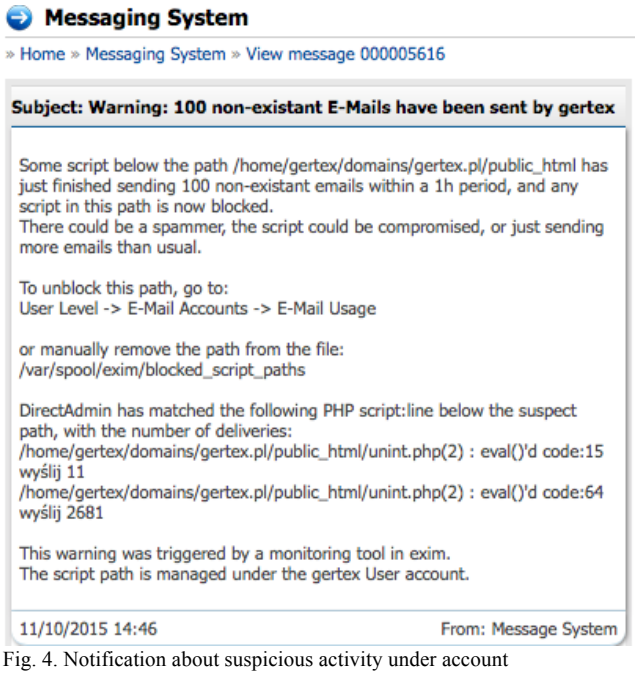


Fig. 4. Notification about suspicious activity under account

The efficiency of our proposed solution was also compared with rate limiting - the most widely used outbound unsolicited mail prevention method. Comparison was done by setting the rate limit to 10 emails/minute and checking the efficiency of every method by sending different amount of emails every minute. We randomly added 1/3 non-existent email addresses

to the list of recipients. The results are seen in fig. 5.

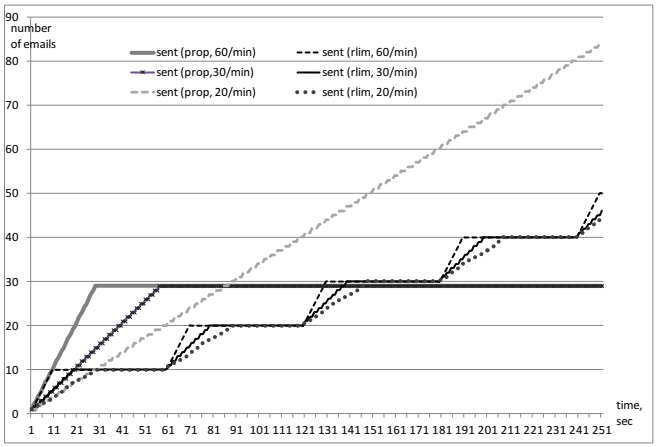


Fig. 5. Comparison of proposed solution and rate limit method

As seen in fig. 5 above, rate limit method has the advantage of preventing the outbound spam when it is sent from unrecognized PHP script and the amount of spam messages is low. That way the threshold of non-existent email addresses per hour in our proposed solution is not reached. In all the other cases our proposed solution behaved equally or better than the rate limit method.

V. CONCLUSIONS

Unsolicited electronic mail is still an actual world-wide problem, having a proportion of 53.4% in global email traffic. With a growth of the number of unsolicited electronic mails from infected websites, there is an urgent need to replace a widely used rate limiting solution, which simply describes how many emails can be sent in a particular amount of time, because of the problems faced by shared web hosting end-customers.

In this work a novel solution to identify and prevent outbound spam was offered, which ensures more modern protection. A complete solution was created and used for the evaluation of the product on production servers with a success.

Experimental results have shown no false-positive reports and an accurate detection of malicious scripts used to send unsolicited electronic mails in shared web hosting servers. Even though the solution successfully blocked spam attempts and prevented the servers of getting blacklisted, that indicates that stricter rate limits for email sends to non-existent email addresses could be chosen, according to the environment. In addition to, it might be worth combining conservative rate-limit method with the proposed solution to prevent even more outbound unsolicited mail in situations with unrecognized PHP scripts and high-quality spam lists with minority of non-existent email addresses.

REFERENCES

[1] Kaspersky Lab, “Spam and phishing in Q2 2015” [Online]. Available: https://securelist.com/analysis/quarterly-spam-reports/71759/spam-and-phishing-in-q2-of-2015/

[2] WhiteHat, “Website Security Statistics Report” [Online]. Available: https://www.whitehatsec.com/statistics-report/featured/2015/05/21/statsreport.html

[3] Imperva, “Web Application Attack Report #5” [Online]. Available: http://www.imperva.com/docs/hii_web_application_attack_report_ed5.pdf

[4] Apache Software Foundation, “What Apache SpamAssassin Is” [Online]. Available: http://svn.apache.org/repos/asf/spamassassin/branches/3.4/README.

[5] SPF Council, “What is SPF” [Online]. Available: http://www.openspf.org/FAQ/What_is_SPF

[6] D. Crocker, “DKIM Frequently Asked Questions” [Online]. Available: http://www.dkim.org/info/dkim-faq.html.

[7] Google, “Authenticate email with DKIM“ [Online]. Available: https://support.google.com/a/answer/174124?hl=en

[8] Hetzner, “Terms and Conditions” [Online]. Available: https://www.hetzner.de/es/hosting/legal/agb.

[9] OVH, “Specific Terms and Conditions on the rental of a dedicated server” [Online]. Available: https://www.ovh.com/us/support/termsofservice/Special_conditions_for_dedicated_server.pdf.

[10] cPanel Inc., “How to Prevent Email Abuse” [Online]. Available: https://documentation.cpanel.net/display/CKB/How+to+Prevent+Email+Abuse.

[11] Parallels Inc., “Protection from Outbound Spam” [Online]. Available: http://download1.parallels.com/Plesk/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=71349.htm.

[12] cPanel Inc., “Scan Outgoing Mail” [Online]. Available: https://documentation.cpanel.net/display/ALD/Scan+Outgoing+Mail.

[13] University of Cambridge, “Exim Callout Verification” [Online]. Available: http://www.exim.org/exim-html-current/doc/html/spec_html/ch-access_control_lists.html#SECTcallver.

[14] University of Cambridge, “Exim Additional Parameters for Callouts” [Online]. Available: http://www.exim.org/exim-html-current/doc/html/spec_html/ch-access_control_lists.html#CALLaddparcall.

[15] The PHP Group, “Runtime Configuration” [Online]. Available: http://php.net/manual/en/mail.configuration.php.

[16] Todd Atkins, “Simple Log Watcher” [Online]. Available: http://sourceforge.net/projects/swatch/.

[17] The Sans Institute, “Web Application File Upload Vulnerabilities” [Online]. Available: https://www.sans.org/reading-room/whitepapers/testing/web-application-file-upload-vulnerabilities-36487.