KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Simonas Jusas

Aktyvaus procesų stebėjimo kompiuteryje programinės įrangos kūrimas ir tyrimas

Magistro darbas

Darbo vadovas
prof. E. Bareiša

Kaunas, 2007
KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Simonas Jusas

Aktyvaus procesų stebėjimo kompiuteryje programinės įrangos kūrimas ir tyrimas

Magistro darbas

Recenzentas
Doc. A. Linkevičius
2007-05-

Darbo vadovas
prof. E. Bareiša
2007-05-

Atliko
IFM-1/2 gr. stud.
Simonas Jusas
2007-05-25

Kaunas, 2007

SUMMARY

Desktop security software design and research

In this paper computer security issues are considered. The malware types and principles of activity are presented. There is also overview of antivirus technology.

The main idea of this paper is that active computer's processes observation and whitelisting should improve computer security. The principles of antivirus architecture were adapted to actively observe computer processes and let run only good ones. There was investigation of program needs for computer resources in experimental part and results are very promising. However it wasn't much better then antivirus software and still needs a lot of research to do, it may be the technology of the future.

TURINYS

1. ĮVADAS

„Tikrai saugi sistema yra ta, kuri yra išjungta iš elektros, uždaryta švininiame kambaryje saugomame ginkluotų sargybinių.“ Gene Spafford, kompiuterių saugumo ekspertas.

Turbūt nedaug kas atkreipė dėmesį, jog už penkių mėnesių pirmasis virusas švęs savo 24 gimtadienį [1]. Šis paprastas faktas parodo, kad jau seniai kompiuterių saugumas yra didžiulė problema. Be abejonės šiandien virusai nėra tokie kaip prieš 20 metų, per tą laiką gerokai padidėjo ne tik jų kokybė, bet ir kiekybė. Todėl siekiant užtikrinti kompiuterio saugumą, reikia tobulinti jau esamus sprendimus bei ieškoti naujų.

Šis darbas pradedamas nuo dabar kompiuteriui iškylančių grėsmių analizės. Nagrinėjami kenkėjiškos programinės įrangos tipai bei analizuojamos jų veiklos. Tuomet pristatomi populiariausi kovos su kenkėjiška programine įranga įrankiai – antivirusai. Nors antivirusų efektyvumas sprendžiant problemas yra didelis, tačiau jų požiūris į problemos sprendimą nesikeičia jau daugelį metų. Galbūt tas problemas galima spręsti kitaip ir tai daryti nemažesniu efektyvumu? Šis klausimas privertė iškelti teiginį, jog virusų sąrašus pakeitus gerų procesų sąrašu, taip pat sėkmingai būtų galima užtikrinti kompiuterio saugumą.

Remiantis pagrindiniais antivirusų architektūros principais buvo suprojektuota ir sukurta aktyvaus procesų stebėjimo programinė įranga. Pakeitus virusų sąrašą gerų procesų sąrašu, pasikeitė programinės įrangos charakteristikos: nereikia saugoti ilgų sąrašų, nes gerų procesų yra kelis šimtus kartų mažiau, nereikia nuolatos skenuoti kompiuterio, tik naujo proceso startavimo metu. Šios idėjos leidžia sutaupyti kompiuterių resursus, kurie gali būti panaudoti vykdant kitą veiklą.

Be abejonės sukurtas produktas dar negali nukonkuruoti antivirusinės programinės įrangos, tačiau darbe pristatytos idėjos neturėtų būti pamirštos, tolesnis jų vystymas gali išsirutulioti į labai sėkmingą ir galingą kompiuterio apsaugos įrankį.

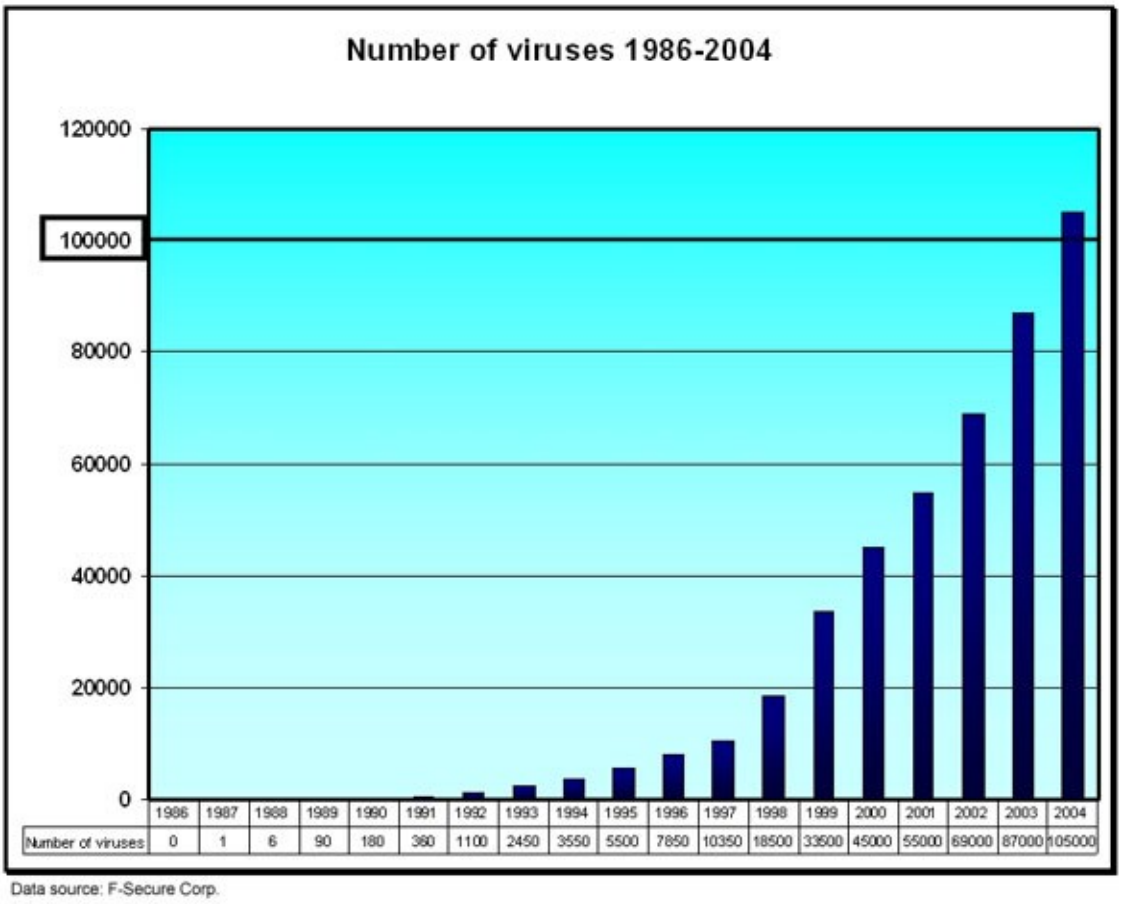
Ruošiant šį darbą buvo parengti keli straipsniai nagrinėjama tema, kuriuos publikavo tinklapis, skirtas verslo saugumui užtikrinti http://www.esecurity.lt.

Šio darbo metu sukurtas produktas buvo pristatytas jaunųjų mokslininkų darbų parodoje „KTU Technorama 2006“, vykusioje 2006 lapkričio 21-22d.

2. KOMPIUTERIO SAUGUMO PROBLEMOS IR JŲ SPRENDIMAI

2.1. Įžanga

Galima sakyti, kad kompiuterio saugumo problemos atsirado kartu su kompiuteriu. Pirmasis virusas buvo sukurtas 1983 metais [1], tiesa, jis dar nedarė jokios žalos, o tik pademonstravo patį principą – kompiuterius gali užpulti virusas. 1986 metais virusas „Brain“ pirmasis pradėjo plisti masiškai. Pandoros skrynia buvo atidaryta ir kiekvienais metais virusų pasirodydavo vis daugiau. F-secure [2] duomenimis 2004 metais virusų skaičius viršijo 100000, ar ilgai teks laukti kol virusų skaičius viršys milijoną? [3]



1 pav. virusų skaičius pagal F-secure [2]

2.2. Viruso apibrėžimas

Virusas yra maža programėlė, kuri keičia kompiuterio veiklą. Virusas turi atitikti du [4] kriterijus:

- Jis privalo įvykdyti save. Labai dažniai atlikdamas veiklą, virusas taip pat įdeda savo kopiją į kitą failą.
- Jis privalo padauginti save. Dažniausiai virusai užkrečia kitus failus, įrašydami į juos savo kodą, jie taip pat gali užkrėsti kitus kompiuterius esančius tinkle.

Vieni virusai yra sukurti gadinti programas, trinti failus ar net suformatuoti kietąjį diską. Kiti tik dauginasi ir savo buvimą pareiškia per tekstines ar vaizdo žinutes. Nepriklausomai nuo viruso tipo jis vis tiek daro žalą, kadangi išnaudoja geroms programoms reikalingus resursus. Neprognozuojamas ir nestandartinis virusų elgesys dažnai priveda prie sistemos lūžimų. Virusai taip pat yra programos ir juose taip pat pasitaiko riktų (ang. bug), kurie yra sistemos lūžimo priežastis.

Yra pagrindiniai penki virusų tipai:

- Failų užkretėjai – tokie virusai užkrečia kitus failus, dažniausiai taikoma į vykdomuosius .com, .bat, .exe. Jie gali užkrėsti kitus failus, skaitant programą iš diskelio, CD ar USB. Daugelis tokio tipo virusų pastoviai yra kompiuterio atmintyje. Kai tik užkrečiama atmintis, joje esantis virusas pradeda užkrėsti visus paleidžiamus failus.
- Užkrovimo sektoriaus (ang. Boot sector) virusai – užkrečia užkrovimo sektorių, kurį turi kietieji diskai (ir diskeliai). Visi diskai turi nedideles programas, kurios yra paleidžiamos diskui pradedant darbą. Užkrovimo sektoriaus virusai infekuoja tas programas, ir sėdi atmintyje kol kompiuteris dirba, jeigu darbo metu prijungiamas koks nors išorinis informacijos šaltinis virusas bando jį užkrėsti.
- Pagrindinio užkrovimo sektoriaus (ang. Master boot sector) virusai – veikia kaip ir užkrovimo sektoriaus virusai, pagrindinis skirtumas, kur pats virusas slepia savo kodą. Jeigu NTFS tipo sistema bus užkrėsta tokio tipo virusu kompiuteris nesikraus.

- Daugialypis virusas – tokie virusai save kombinuoja failų užkretėjų ir užkrovimo sektoriaus virusų savybes, todėl juos sunku išgydyti. Jeigu užkrovimo sektorius bus išgydytas nuo viruso, o kompiuteryje esantys failai ne, tai virusas iš naujo užkrės užkrovimo sektorių arba atvirkščiai.
- Macro virusai – užkrečia duomenų failus, dažniausiai tokie virusai būna parašyti Visual Basic kalba ir keliauja Microsoft Office failuose: Word, Excel, PowerPoint ir Access, tačiau gali užkrėsti ir kitų tipų duomenų failus. Kadangi šito tipo virusus lengva sukurti, jų yra daugiausiai.

2.3. Virusų veikimo principai

Kompiuteriniai virusai yra mažos programos, kurios slapta prasibrauna į kompiuterį ir siekia padaryti jam žalos [5]. Virusai kopijuoja save ir stengiasi prasibrauti į kitus kompiuterius siųsdami save el. laiškais su priedais arba tinklų per paliktus atvirus prievadus (ang. port) bei bendrintus (angl. shared) katalogus. Yra įvairių virusų tipų: vieni iškarto pasinaudoja adresų knygute ir išsiunčia visais rastais adresas, kiti sėdi pasislėpę ir po truputį gadina duomenis.

Tobulėjant antivirusinei programinei įrangai, virusų kūrėjai taip pat nesėdėjo vietoje. Jie dabar stengiasi paslėpti [6] savo viruso kodą kitų programų failuose. Labai dažnai pačio viruso kodą įdeda failo pabaigoje (anksčiau dėdavo pradžioje), o pradžioje įrašo tik komandą, kuri turėtų peršokti į failo galą ir įvykdyti ten rastas komandas

1998 metais pasirodė IDEA.6155 virusas, nors jis nepaplito plačiai po internetą, tačiau sukėlė didelį šurmulį tarp antivirusų kūrėjų, kadangi tai buvo pirmasis virusas, kuris kodavo pats save. Tai buvo naujos virusų koncepcijos pristatymas. Virusas save slėpė po trijų lygių kodais. Pirmasis lygis naudojo mutacijos variklį užkoduoti save, o raktas buvo paslėptas, pačio viruso kode. Antrasis lygis neturėjo kodo, tačiau jis gana greitai buvo atšifruotas pasitelkus Brute Force algoritmą, kadangi slaptažodis labai lengvas. Trečias lygis užkoduotas pačio viruso autoriaus sukurtu kodavimo algoritmu, o raktas buvo padalintas į dvi dalis ir paslėptas viruso kode. Tokiam virusui užsikrauti reikia nemažiau 5 sekundžių, jeigu kompiuteryje yra antivirusinė naudojanti euristinį skanavimo varikliuką, t.y. bando emuliuoti kompiuterio aplinką norėdama patikrinti, kaip veikia failas, tai virusui užsikrauti virtualioje aplinkoje gali prireikti net pusantros valandos. O dėl padidėjusių kompiuterio resursų naudojimo, vartotojas greičiausiai išjungtų antivirusinę programinę įrangą.

Kita mada buvo slėpti virusų kodą keičiant jo išvaizdą – taip vadinami polimorfiniai virusai. Tokie virusai turi turėti nedidelę kodo dalį, kurią vadina „galvą“. „Galva“ atsakinga už tai, kad virusas būtų sėkmingai atkoduotas, likusio kodo dalis yra paslepiama kitos programoje, kad antivirusai lengvai jos neaptiktų, į kodą primaišomas atsitiktinis skaičius įvairių konstantų. Dar vėliau atsirado metamorfiniai virusai, kurie iš tiesų sugebėdavo pakeisti savo kodą keliaudami iš vieno kompiuterio į kitą. Nors metamorfinis virusas, kaip ir polimorfinis, užkoduoja save, bei į kodą gali pridėti konstantų, kad būtų sunkiau atpažinti, vis tik metamorfiniai virusai taip pat turi turėti nedidelę kodo dalį vadinama „galva“.

Virusų kūrėjai ne tik stengiasi paslėpti savo kodą, bet taip pat imasi veiksmų prieš antivirusinę programinę įrangą. Vienas iš tokių pavyzdžių yra įmantrių ir egzotiškų komandų įtraukimas į kodą, kurių antivirusinė programinė įranga nesugeba suprasti. Antras - jau minėto IDEA.6155 viruso užkrovimas vyksta, taip ilgai, kad antivirusinė programinė įranga pagalvotų, jog tai normali programa ir jį praleistų. Trečias būdas, tai patikrinimas, ar virusas veikia tikroje sistemoje ar antivirusinės programos emuliuojamoje. Pavyzdžiui Magistr virusas nekenkdavo kompiuteriui jeigu jame nebūdavo interneto. Dar įdomesnis 2002 metais atrastas W32.Simile, jis ne tik yra metamorfinis, tačiau kartu su savimi 14000 asmeblerio kodo eilutėse turi ir visą metamorfizmo varikliuką. Jis taip pat gali pereiti nuo Windows sistemos prie Linux. Tačiau labiausiai gąsdinanti jo savybė yra ta, kad jis prieš startuodamas atsitiktinai spėja, ar šį kartą jam dekoduotis ir įvykdyti save, ar atidėti kitam kartui. Tokio tipo virusai gali išbūti žymiai ilgiau nepastebėti antivirusų kūrėjų.

2.4. Virusų pavadinimų kūrimo taisyklės

Kai virusas „VBS/VBSWG.J“ plačiai pasklido internetu užkrėsdamas gausybę kompiuterių žiniasklaidą jį pakrikštijo „AnnaKournikova“, kadangi virusas atkeliaudavo su JPEG failu, kuriame atseit būdavo garsios tenisininkės nuotraukos. Jeigu tą patį failą užkrėstą virusu nusiustumėte antivirusų kūrėjams gautumėte skirtingus atsakymus [7, 8, 9]. Ar tai reiškia, kad tas failas buvo užkrėstas keliais virusais? Ne. Yra tam tikros virusų pavadinimo taisyklės, kurių susitarė laikytis antivirusinės programinės įrangos kūrėjai. Pirmą kartą 1991 metais [10] susitarė dėl tokios formulės:

- Šeimos_vardas.Grupės_vardas.Pagrindinė_versija.Minorinė_versija[:Papildomos_žymės]

Sutartis draudžia vartoti kompanijos vardą, prekės ženklus, bet kokių gyvų žmonių vardus, nebent įrodyta, jog tas žmogus viruso autorius. Tai paaiškina, kodėl ne antivirusų kūrėjai, o žiniasklaida suteikia skambius vardus virusams. Vardų kūrimo konvencija nori apsaugoti nuo autorinių teisių ir kitų pažeidimų, juk viruso pavadinimas kokios nors garsios kompanijos vardu, gali pridaryti didelių nuostolių tai kompanijai.

Tačiau dažniausiai tenka išgristi nesudėtingus antivirusų kompanijų sukurtus vardus, bet lengvus žodžius pvz. "Arual" (atvirkščiai "Laura"), "Golni" ("Winlog" apverstas ir sutrumpintas) arba "Nimda" ("Admin" apverstas). Iš kur atsiranda tokie žodžiai? Dažniausiai, nuo pirmos versijos užkrečiamojo failo pavadinimo [11] arba el. laiško antraštės, su kuria pradėjo plisti kenkėjas.

Atsiradus naujiems virusams buvo keletas bandymų atnaujinti 1991 susitarimą, sėkmingiausias badymas 1999 metai pasiūlytas Gerald Scheidl, Ikarus Software [12].

Išnagrinėkime keletą pavyzdžių ką viruso pavadinimas gali pasakyti apie patį virusą:

W32.Gokar.A@mm [13]

W32 reiškia, kad virusas veikia visuose 32 bitų Windows operacinėse sistemose (Windows 95, 98, Me, XP, 2000, 2003, NT). Taip pat galima sutikti W98 arba W2K nurodančia konkrečią operacinę sistemą pirmu atveju Windows 98, antru Windows 2000.

A žymi versija, reiškia, kad tai pirmas virusas iš šios šeimos. Kiti virusai gali susilaukti daug versijų pvz. W32.Bagle.BB@mm arba net W32.Gaobot.BOW.

@mm santrumpa angliškų žodžių mass mailing, kas reiškia, jog virusas atkeliavęs į pašto dėžutę išsiunčia save visais rastais pašto dėžutėje adresais.

JS.KakWorm.E@m

JS reiškia, jog virusas parašytas JavaScript kalba.

E – tai penktos kartos virusas, tai reiškia, jog viruso kodas buvo patobulintas jau 5 kartus. Kitą virusų versiją nebūtinai turi sukurti pats viruso autorius, gali būti taip, jog netgi visas versijas sukūrė skirtingi žmonės.

@m žymi žodžius slow mailer, kas reiškia, kad toks virusas išsiunčia vieną laišką vienu metu iš adresų knygutės, todėl virusas plinta žymiai lėčiau.

W97M.Pacol.A

W97M reiškia Word 97 macro virusas.

A - reiškia, kad tai pirmas virusas iš šios šeimos.

Daugiau apie virusų pavadinimus galima apskaityti Symantec puslapyje arba virusų konvekcijos dokumente [14].

2.5. Kita kenkėjiška programinė įranga.

Iki šiol aptarėme tik virusus, bet virusai, deja, nėra vienintelė blogybė.

Kenkėjiška programa (ang. Malware) – bendrinis pavadinimas visai programinei įrangai, kuri atlieka kompiuteryje pakeitimus savo naudai be vartotojo sutikimo [15]. Virusai irgi yra kenkėjiškos programos.

Kirminai (ang. worm) kaip ir virusai yra sukurti pakenti kompiuteriui [16], taip pat kuria savo kopijas, tačiau priešingai nei virusai, jiems įvykdyti nereikalinga, kad žmogus paleistų kokią nors programą. Kirminai dažniausiai keliauja, kaip Microsoft Word arba Excel macro komandos [4] ir neturi savo vykdomojo pradinio failo.

Trojanai (ang. trojan) - tai tokios programos, kurios apsimeta, jog yra labai reikalingos ir vertingos, tačiau iš tiesų taip nėra [16, 4]. Priešingai nei virusai ar kirminai, jos pačios nesidaugina. Trojanuose yra paslėptas kodas, kurį įvykdžius gali būti sugadinta sistema ar prarasti duomenys. Norint kad trojanas atsidurtų kompiuteryje, jį reikia atsisiųsti, jis gali atkeliauti su laišku kaip priedas.

Keilogeriai (ang. keylogger) – programinė įranga skirta stebėti ir fiksuoti visus klaviatūros paspaudimus, kartais savo užfiksavimus keilogeriai siunčia programos autoriui el. paštu. Taip ne tik pavagiama slapta, asmeninė informacija iš žmogaus, bet taip pat surandami jo slaptažodžiai.

Šnipinėjimo programinė įranga (ang. spyware) – priešingai nei virusai, šnipinėjimo programinės įrangos dažniausiai nežaloja kompiuterių [15], o prasibrovusios pasislepia kompiuterio atmintyje ir atlieka pakeitimus, stebi vartotojo darbą, vagia informaciją, užkraudinėja tam tikrus puslapius, arba keičia paieškos rezultatus. Ši programinė įranga nesugeba pati atkeliauti iki vartotojo kompiuterio, dažniausiai įdiegiama kartu su kitomis programomis, pvz. anksčiau labai populiari programa KaZaa turėdavo labai daug šnipinėjimo programinės įrangos priedų [17]. Kitas tokios programinės įrangos įdiegimo būdas yra, kai užėjus į kokį nors puslapį išmetamas pranešimas pranešantis, jog reikia įsidiegti papildomas tvarkykles (ang. driver), kad puslapis būtų matomas teisingai. Šnipinėjimo programinės įrangos darbas dažniausiai pasireiškia sulėtėjusiais kompiuterio tempais.

Reklaminė programinė įranga (ang. adware) – programos specialiai sukurtos pristatyti nepageidaujama reklamą vartotojams [15]. Labai dažnai pasireiškia netikėtai iššokančiais langais.

Į kompiuterį patenka kartu su kita įdiegiama programine įranga [18]. Teoriškai įvairiose šalyse priimti vienokie ar kitokie įstatymai draudžiantis įdiegti reklaminę programinę įrangą be vartotojo sutikimo, tačiau dažniausiai vartotojai sutinka su licenzija jos neskaitę.

Botnetai (ang. botnet) – tai kompiuteriai, užkrėsti tam tikra programine įranga leidžiančia perimti tų kompiuterių kontrolę [19]. Botnetai dažniausiai yra valdomi per IRC arba P2P tinklus [20]. Dažnai botnetai pasitelkiami kitos blogos programinės įrangos įdiegimui pvz. reklaminės, arba nepageidaujamų laiškų siuntimui bei įvairioms atakoms.

2.6. Kenkėjiškos programinės įrangos palyginimas

1 lentelėje pateikiama visų kenkėjų palyginimas.

1 lentelė. Skirtumai tarp kenkėjiškos programinės įrangos

Tipas	Veikimo vieta	Aktyvavimas	Plitimo būdai	Veikla	Kita
Virusas	Kopijuoja save į kitus failus.	Plitimui reikalinga žmogaus pagalba, jis nesugeba paleisti savęs, todėl į pagalba pasitelkia įvairius gražius paveiksliukus ir t.t	Dažniausiai plinta el. laiškais su priedais. Taip pat duomenų laikmenomis: diskeliais, CD, USB. Plitimo greitis nėra didelis.	Gadina failus, gali neatstatomai sugadinti kietąjį diską	Mėgsta slėptis, gali būti užkoduoti metamorfiniai, polimorfiniai.
Kirminas	Sėdi kompiuterio atmintyje.	Nereikia vartotojo pagalbos norint paleisti. Yra gana savarankiški.	Keliauja internetu, ieškodamas gretimų kompiuterių tinkle, nestandartiniu neuždarytų	Išnaudoja programinės įrangos klaidas siekdami sutrikdyti normalų vartotojo	Dažniausiai nesislepia, kartais gali būti užkoduoti, polimorfiniai, metamorfiniai

			prievadų. Taip pat gali keliauti el. paštu. Plinta labai greitai.	darbą.	
Trojanas	Apsimeta gera programa, tačiau fone atlieka kokią nors blogą veiklą.	Vartotojas pats turi paleisti. Gali įsirašyti į startuojančių programų sąrašą.	Vartotojas turi pats parsisiųsti. Gali plisti P2P tinklais.	Suteikia kompiuterio valdymą trečiosioms šalims	Dažniausiai slepiasi
Keylogeris	Sėdi atmintyje, savarankiška programa.	Turi paleisti vartotojas. Gali įsirašyti į startuojančių programų sąrašą.	Vartotojas turi pats parsisiųsti	Fiksuoja visus klaviatūros klavišų paspaudimus.	Dažniausiai slepiasi
Šnipinėjimo pį	Naršyklės priedai, kitų programų priedai, savarankiškos programos.	Vartotojas turi pasileisti pats.	Vartotojas įsidiegia su kitomis programomis. Gali plisti P2P, IRC tinklais.	Stebi varotojo pomėgius, lankomus puslapius, gautą informaciją perduoda trečioms šalims. Keičia startinius puslapius.	Kartais slepiasi
Reklaminė pį	Kitų programų priedai.	Vartotojas turi pasileisti pats.	Vartotojas dažniausia įsidiegia su kitomis programomis.	Rodo nepageidauja mas reklamas	Kartais slepiasi
Botnetai	Sėdi	Vartotojas turi	Vartotojas turi	Sujungia	Dažniausiai

	atmintyje. Savarankiška programa.	pasileisti pats Gali įsirašyti į startuojančių programų sąrašą.	parsisiųsti. Gali plisti P2P, IRC tinklais.	kompiuterius į tinklą, suteikia priėjimą įdiegti kitoms kenkėjiškoms programoms.	slepiasi

Praktikoje labai sunku priskirti blogą programą konkrečiai kategorijai, kadangi dažniai jos derina kelių programų savybes. Pavyzdžiui, kirminas atklydęs į kompiuterį jame palieka trojaną, kad jo autorius galėtų sukurti botnetą, arba paleisti keilogerį. Botnetai labai dažnai yra kuriami, tam kad būtų galima uždirbti pinigus diegiant reklaminę arba šnipinėjimo programinę įrangą.

Seniau buvo labai populiarūs įvairūs virusai, kurie naikindavo įvairius kompiuterio darbui reikalingus failus, ar netgi formatuodavo kietuosius diskus. Tačiau blogiukai pasimokė – virusas paplinta žymiai plačiau, jeigu aukos kompiuteriui pakenkė tik nestipriai, tačiau nenutraukė jo darbo. Dabar daug populiaresni yra kirminai, techniniu požiūriu jie nedaro žalos kompiuteriui, tačiau praktiškai vis tiek parazituoja kompiuterio atmintyje, eikvoja resursus, be to dažnai su savimi pasiima ir kitų blogybių savybes. Kirminai paplinta žymiai greičiau, kadangi jų plitimui nereikia žmogaus įsikišimo. Tuo tarpu viruso atveju yra būtina, kad jį kas nors suaktyvintų, todėl reikia labai pasistengi ir pasitelkti visas įtikinėjimo įmantrybes, pridėti kokį nors failą ar užrašą, kuris galėtų sudominti vartotoją ir paskatintų aktyvuoti virusą.

2.7. Antivirusų veikimo principai

Antivirusai – programos, skirtos aptikti virusams ir kitai kenkėjiškai programinei įrangai [21]. Pagrindiniai antivirusų veikimo būdai:

- Tikrinimas pagal virusų žodyną
- Euristinis arba įtartinos veiklos aptikimas
- CRC kodo tikrinimas
- Smėlio dėžė arba elgesio blokavimas.

Didžiosios dalies antivirusinės programinės įrangos pagrindinis instrumentas yra būtent tikrinimas pagal žodyną. Žodyne yra kaupiami virusų parašai. Programai startuojant antivirusas sutikrina, ar ji neturi viruso požymių, ir leidžia startuoti. Šis metodas pagrinde derinamas su skanavimu budėjimo režime [22], kadangi budėjimo režime antivirusas turi veikti greitai ir neapkrauti perdaug sistemos, nes priešingu atveju vartotojas ją paprasčiausiai išjungs. Budėjimo režime antivirusas tikrina pradedančias arba baigiančias darbą programas, el. pašto gautus laiškus. Užkrėsta virusu programa nebūtinai jį aktyvuoja darbo pradžioje, tai gali būti atliekama ir programai baigiant darbą.

Pagrindinis šio metodo trūkumas tai, kad virusų žodyną reikia nuolatos atnaujinti. Pasirodžius naujam virusui, jo parašas atsiranda žodyne ne iš karto. Tai užtrunka, kadangi antiviruso kūrėjai turi išanalizuoti gautą virusą, atlikti testus ir aptikti tam tikrus bendrus bruožus, pagal kuriuos galimą tą virusą identifikuoti.

Įtartinos veiklos tikrinimas atliekamas imituojant programos kvietimą ir žiūrint kaip jinai elgsis (dinamine euristika). Tai daug laiko ir resursų reikalaujantis darbas, todėl jis derinimas su kompiuterio skanavimu vartotojui paprašius. Paprastai žmonės linkę susitaikyti su tuo, kad kompiuterio skanavimas nuo virusų užtruks ilgai ir sunaudos labai daug sistemos resursų. Tačiau toks skanavimas negarantuoja labai gerų rezultatų, kadangi nauji virusai darosi vis gudresni ir nebūtinai iš karto aktyvuojasi atvykę į sistemą, kiti virusai bando patikrinti ar tai nėra kvietimo imitacija.

Kitas euristinis metodas, seniau buvęs labai populiarus, yra taikomas asemblerio kalba parašytų virusų aptikimui. Toks būdas dar vadinamas statine euristika, kadangi tikrinamos asemblerio komandos ir žiūrima, ar jos neatlieka kokių nors keistų veiksmų, pavyzdžiui komanda patikrinanti, ar failas yra vykdomasis. Statinei euristikai reikia žymiai mažiau resursų ir laiko atlikti, todėl jinai labiau mėgstama antivirusų kūrėjų nei naujesnioji - dinaminė.

Norint patikrinti viruso parašą, reikia tame faile surasti kažkokią simbolių eilutę, tačiau metamorfiniai virusai mėgsta slėptis kituose failuose, patys save užkoduodami ir kiekvieną kartą evoliucionuodami savo kodą. Tokiu atveju surasti parašą nėra lengva. Anksčiau labai populiari, tačiau dabar beveik nebenaudojama technologija - CRC kodo tikrinimas [23]. CRC kodo pati idėja buvo patikrinti, ar failas nėra su klaida, be to šis kodas labai greitai paskaičiuojamas, tačiau virusų kūrėjai pastebėjo, jog tam tikros klaidos panaikina viena kitą ir netruko išnaudoti šio atradimo savo naudai.

Smėlio dėžės [24] principas yra skirtas dideliems tinklams, kadangi pagrindinė programa sukasi serveryje ir tikrina kas nori patekti į tinklą. Iš pirmo žvilgsnio šis metodas gali

pasirodyti panašus į euristinį skanavimą, tačiau euristinio tikrinimo metu yra imituojamas tikros sistemos veikimas stipriai apribojant resursus ir laiką, todėl kai kurie virusai sugeba apgauti šias apsaugas. Elgesio blokavimo metu visoms programoms leidžiama veikti realiu laiku išnaudojant visus laisvus kompiuterio resursus ir blokuojant tik prieš bandymus vykdyti žalinga sistemai veiklą. Smėlio dėžės stebėjimas apima:

- Bandymus atidaryti, peržiūrėti, sukurti, ištrinti, modifikuoti failus.
- Bandymus suformatuoti diską ir kitas žalingas diskui operacijas.
- Bandymus modifikuoti vykdomuosius failus, skriptus.
- Bandymus keisti sistemos nustatymus.
- Bandymus siusti el. laiškais ir el. žinutėmis vykdomuosius failus.
- Bandymus inicijuoti prisijungimus prie tinklo.

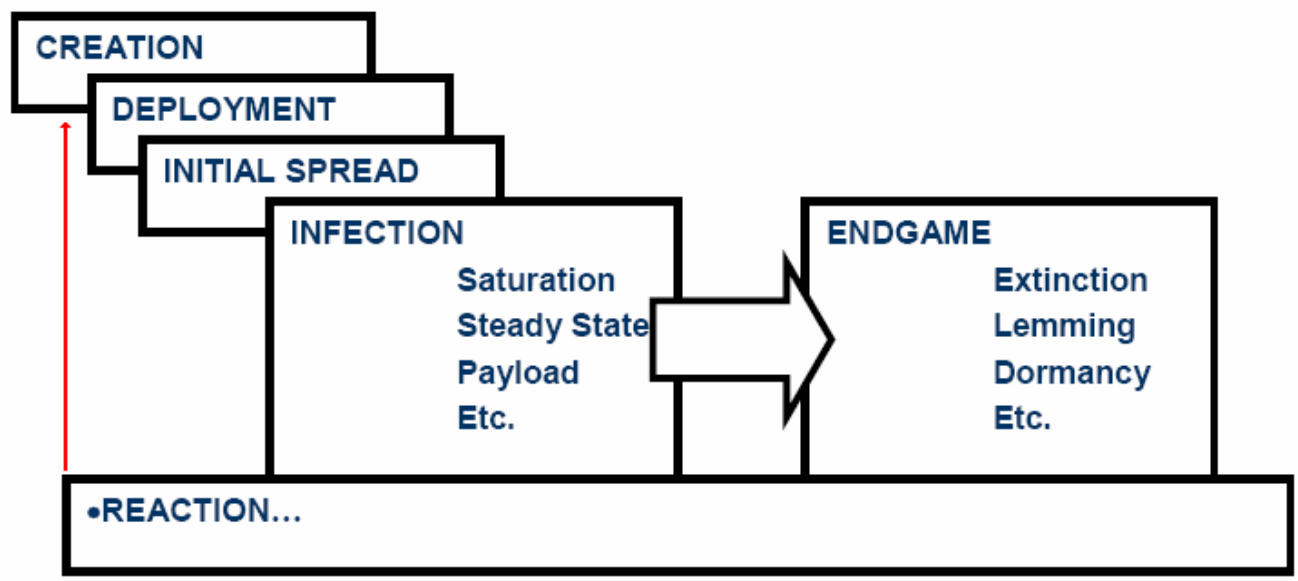
Šis metodas turi du pagrindinius trūkumus: visų pirma, jam reikia daug resursų, antra, virusas gali atakuoti tai kas neįtraukta taisyklėse pvz. failus, kurie vadinami nesvarbiais ir juos gadinti. Vis tik, kaip parodė praktika, jis atlieka savo darbą: First Citizens Banke [25], kuriame įdiegta tokio tipo saugumo sistema buvo užfiksuota neaiški veikla, kurią programa pavadino "Microsoft Directory Transversal Attack.", kitą dieną tas kodas buvo identifikuotas antivirusų kūrėjų kaip Code Red virusas.

2.8. Antivirusų architektūra

Antivirusų kūrėjai nėra linkę dalintis savo sistemų architektūra, ir tam galima nurodyti netgi kelias priežastis:

- Jie kuria komercinę programinę įrangą, ir architektūros atskleidimas gali padėti konkurentams.
- Virusų kūrėjai išanalizavę antivirusinės programinės įrangos architektūrą pradėtų kurti specialius ją gadinančius kenkėjus.

Kad geriau būtų suprantamas antivirusų veikimo principas 2 paveikslėlyje pateiktas viruso gyvenimo ciklas.



2 pav. Viruso gyvavimo ciklas [26].

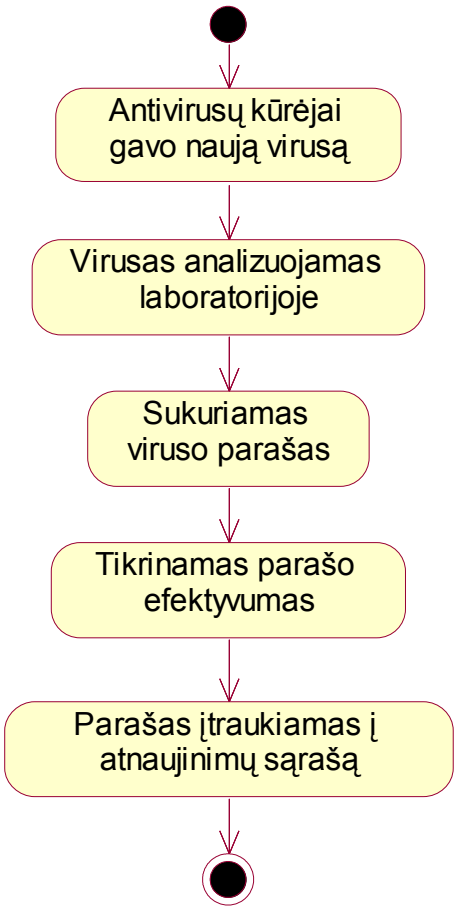
Viruso gyvavimo ciklas susideda iš 5 dalių

- Sukūrimo – sukuriamas pats virusas.
- Dislokavimo – tai gali būti viruso paleidimas į laisvę, arba jo kodo paskelbimas kokiame nors internetiniame puslapyje. Antru atveju peržiūrėjęs kodą, koks nors naujokas norės pažiūrėti kaip jis veikia ir pasileis (inicijuos sklidimą).
- Sklidimas – šios fazės metu virusas plinta, bando susirasti aukos kompiuterį.
- Užkrėtimas – šios fazės metu, virusas jau sėdi aukos kompiuteryje ir gali atlikti įvarius veiksmus: gadinti sistemą, išmesti trojanus bei kitą programinę įrangą, dėl kurios jis buvo sukurtas arba tiesiog laukti kokio nors tinkamo momento savęs realizavimui pvz. laiko bombos w95.CIH
- Pabaiga – šiuo atveju gali atsitikti įvairių dalykų: pasikeitė programinė įranga, buvo ištaisytos spragos programinėje įrangoje, visi vartotojai atsisiuntė antivirusinės programos atnaujinimus ir t.t.

Eilutė apačioje „reaction“ apimanti kelias fazes žymi antivirusinių kūrėjų darbą. Viskas prasideda, tuomet kai antivirusų kūrėjai gauną viruso pavyzdį ir ima analizuoti. Tai nereiškia, kad šita fazė prasideda iš karto sukūrus virusą, ji gali prasidėti bet kurios viruso gyvavimo fazės metu, pavyzdyje pavaizduotas bendrasis atvejis. Taip pat norėtųsi atkreipti dėmesį, jog net mirus

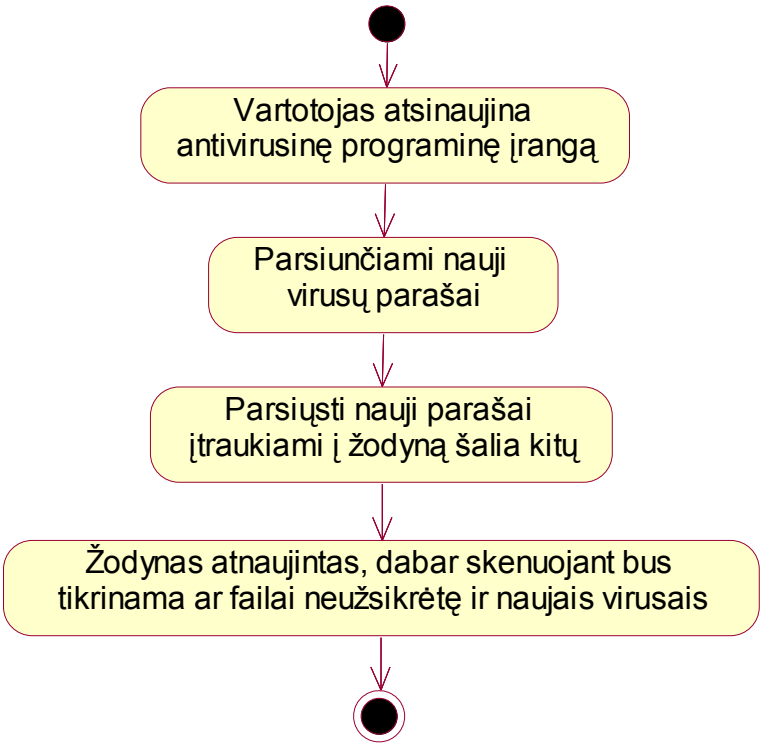
virusui, antivirusų kūrėjams ši fazė nesibaigia, viruso parašas būna įtrauktas į žodyną ir toliau skenuojant failus tikrinama ar nėra užsikrėtusiu tuo virusu.

3 paveikslėlyje vaizduojama viruso parašo sukūrimo diagrama. Labai dažnai vartotojai neįsivaizduoja, kad egzistuoja tokia dalis.



3 pav. Viruso parašo sukūrimo ciklas.

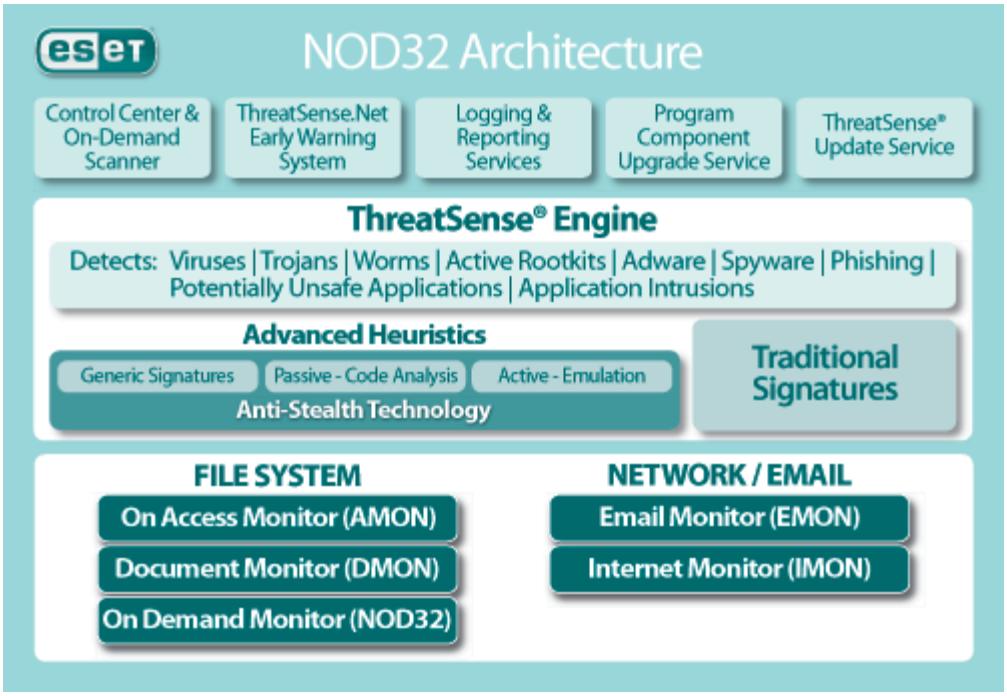
Ši fazė baigiasi viruso parašo įtraukimu į atnaujinimų sąrašą. 4 paveiksliukas analizuoja antrąją - viruso parašų gyvenimo ciklo dalį – kai jis patenka į vartotojo kompiuterį.



4 pav. Virusų žodyno atnaujinimas

Kompanijos teigia, jog nauji virusų parašai sukuriami iki 2 valandų nuo virusų patekimo į jų laboratorijas, tačiau kada atnaujinimai pasiekia vartotojus pasakyti negalima. Symantec Norton AntiVirus automatiškai atnaujinimų ieško kas 4 valandas, tačiau dalis vartotojų gali būti neįsijungę automatinių atnaujinimų, tokių atveju atnaujinai gali vėluoti mėnesiais.

Dabar detaliau paanalizuosime antrąją dalį t.y. antivirusinės programinės įrangos veiklą kompiuteryje ir kaip joje įsitvirtinę virusų parašai. Nors daugelis kompanijų ir slepia savo produktų architektūras, tačiau šiame darbe vis tik bus pateikta keletas pavyzdžių ir pabandyta išanalizuoti kiek galima plačiau. Analizavimas pradedamas nuo Eset kuriamo antiviruso NOD32 architektūros pateiktos 5 paveikslėlyje:



5 pav. NOD32 antiviruso architektūra [27]

Kaip galima matyti iš paveikslėlio pagrindiniai komponentai šioje antivirusinėje programoje yra skirti: darbui su failų sistema, darbui su tinklu skirti komponentai, ThreatSence Engine ir dar 5 komponentai pačios antivirusinės valdymui. Pabandysiu viską apžvelgti plačiau.

Control Center & On-demand Scaner – pagrindinių programos nustatymų valdymas, žvalgytuvo (ang. scanner) iškvietimo galimybė.

ThreatSence.Net Early Warning System – įspėjimų pateikimo vartotojui posistemis.

Logging & Reporting Servises – žurnalų vedimo ir ataskaitų teikimo posistemis.

Program Component Upgrade Service – programos komponentų atnaujinimo posistemis.

ThreatSence Update Service – viruso žodyno atnaujinimui bei priežiūrai skirtas posistemis.

Amon – atmintyje veikiantis žvalgytuvas, kuris patikrina, ar failai nėra užkrėsti prieš pradedant jais naudotis.

Dmon – skenuoja Microsoft Office programų kuriamus failus pasitelkdamas Microsoft API. Pagrindinis tikslas – gaudyti macro virusus.

Nod32 – Failų skanavimas pagal pareikalavimą, galima skenuoti tam tikrus nurodytus failus arba direktorijas. Galima atlikti planinius skenavimus.

Emon – Atmintyje sėdintis žvalgytuvas tikrinantis visą HTTP bei POP3 protokolais į kompiuterį patenkančią informaciją.

Imon – Skenuoja visą įeinančią ir išeinančią informaciją pasitelkdamas specialias sąsajas, pvz. Microsoft Outlook Microsoft Exchange.

ThreatSence Engine visos dalys plačiau buvo pakomentuotos ankščiau (žr. skyrelyje 1.7 Antivirusų veikimo principai), todėl dabar tik trumpai pristatysiu.

Traditional Signatures – virusų žodynas.

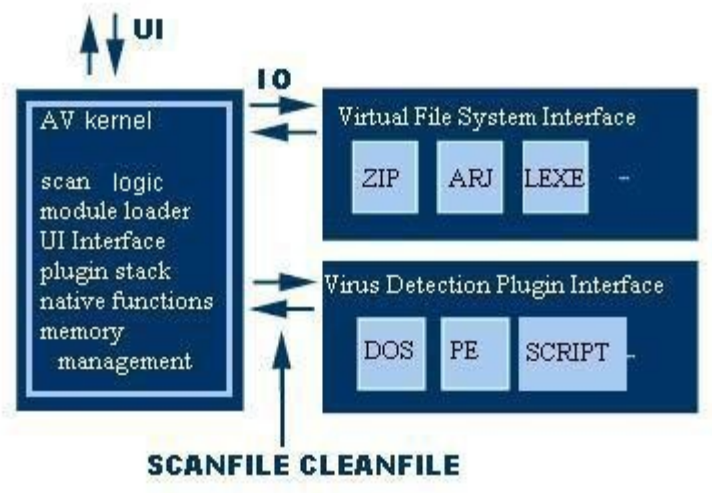
Visi advanced Heuristics dalyje pažymėti metodai yra skirti aptikti ir dezinfekuoti dar nežinomus virusus. Taip pat ši dalis gaudo polimorfinius virusus.

Generic signatures – bendriniai parašų šablonai, dažniausiai sutinkamų kenksmingų virusų naudojamų komandų sąrašas.

Pasive – Code Analysis – statinė euristika, failo nagrinėjimas asemblerio lygyje ieškant komandų, galinčių priklausyti virusui.

Active – Emulation – dinaminė euristika, failo vykdymas virtualioje aplinkoje žiūrint kaip jis elgsis.

6 paveiksliukas detalizuoja skenerio veikimo principą

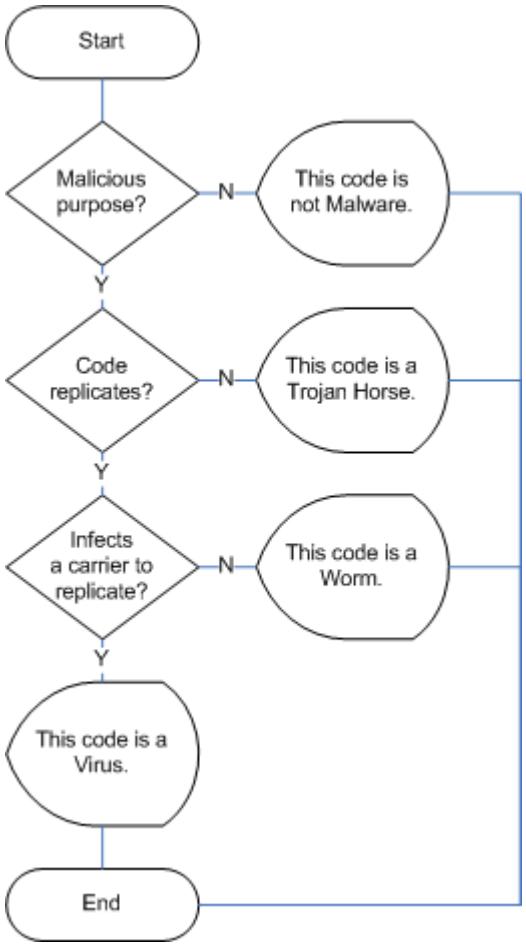


6 pav. Skenerio architektūra [28]

Skeneris susideda iš trijų pagrindinių dalių: kernelio, virusų aptikimo sąsajos ir virtualios failų sistemos sąsajos. Kernelio tikslas yra užkrauti reikiamus modulius bei nukreipti skenuojamus failus ten kur reikia. Virtuali failų sistema reikalinga, kai norima tikrinti failų archyvus, antivirusas negali jų išpakuoti į diską, nes jeigu jame yra virusas, tai reikštų jo aktyvavimą, kad taip neatsitiktų, antivirusas susikuria savo virtualią failų sistemą, kurioje ir įvyksta failų išpakavimas. Ši dalis pavyzdyje sudaryta iš trijų modulių, kiekvienas yra skirtas

skirtingo tipo archyvams analizuoti ir vienu metu užkraunamas tik vienas modulis – tas kurio reikia, šitaip pagreitinamas programos veikimas. Virusų aptikimo sąsaja taipogi sudaryta iš trijų modulių, kiekvienas jų reikalingas aptikti skritingo tipo virusams: DOS – DOS sistemos virusams, PE – Windows sistemos virusams, SCRIPT – macro virusams, kurie dažniausiai būna parašyti Basic kalba.

7 paveiksliukas vaizduoja patį skanavimo procesą



7 pav. Virusų identifikavimo algoritmas [29]

Start vietoje yra paduodamas failas, pirmiausiai patikrinama ar jis turi kažkokių įtartinų ketinimų, jeigu ne – skenavimas tuo ir baigiasi. Jeigu taip, reikia nustatyti, kokio tipo pavojų gali sukelti failas, kad parinkti tinkamas apsisaugojimo priemones. Pirmiausiai patikrinama ar rasta blogybė sugeba save dauginti, jeigu ne, tai ji yra trojanas. Tuo tarpu tiek virusai, tiek kirminai plinta dauginimosi būdu. Tačiau esminis skirtumas yra tai, jog virusas turi būtinai užkrėsti kokį nors failą, kad galėtų įvykdyti kenksmingą veiklą.

Reiktų atkreipti dėmesį, jog grafas tikrina tris dalykus: ar nėra trojanas, ar nėra kirminas, ar nėra virusas. Tačiau kenkėjiškos programinės įrangos yra ir daugiau. Ilgą laiką antivirusai apskritai negaudė nei šnipinėjimo, nei reklaminės programinės įrangos. Šiuo metu jau po truputį ir ją įtraukia į savo sąrašus, tačiau vis tik efektyvumo kol kas dar trūksta.

2.9. Problemos su antivirusais

Kai kurie specialistai tvirtina, jog antivirusų skanavimo principai yra pasenę, nes vadovaujasi senu Niutono principu: kiekvienam veiksmui yra tokio pačio dydžio atoveiksmis [30]. Pagrindinės antivirusinės programinės įrangos problemos:

- **Virusų parašų problema**. Atsiradus naujam virusui, pirmiausiai antivirusų kūrėjai turi gauti jo kopiją, tai lengviausia dalis. Antras etapas dekompiliuoti virusą ir pažiūrėti, ką jis daro. Tačiau kartais pasitaiko užkoduotų virusų ir tuomet belieka vienintelis būdas sukurti priešnuodžius – stebėti viruso elgseną. Kai pagaliau antivirusų gamintojai supranta, kaip veikia virusas, yra sukuriamas jo parašas ir tuomet seka testavimo fazė. Jokie atnaujinimai negali būti išleisti neatlikus testavimo, būtina įsitikinti, jog sukurtas parašas tikrai veikia ir nežaloja kitos programinės įrangos.

  Kuo toliau, tuo naujų virusų, atsiradusių per metus, skaičius vis didėja, o tai reiškia, kad antivirusų kūrėjai turi greičiau išleisti atnaujinimus vienam virusui ir tuoj pat griebtis kito. Ten kur padidinamas greitis paprastai nukenčia kokybė, virusų žodyno atnaujinimas gali sukelti visos sistemos lūžimą [31], o to niekas nenori.
- **Virusų pavadinimų problema**. Kita problema tai, kad skirtingi antivirusų kūrėjai linkę duoti skirtingus pavadinimas tiems patiems virusams [7, 8, 9] pvz. "Norton AntiVirus - W32.Novarg.A@mm“ , „RAV AntiVirus - Win32/Mydoom.A@mm“, „GroupShield for Exchange - W32/Mydoom@MM“, „BorderWare MXtreme Mail Firewall - I-Worm.Novarg“, „InterScan - WORM_MIMAIL.R“, „Antigen - MyDoom.A@m (Norman) worm“, „McAfee - W32/Mydoom@MM“ yra to paties viruso pavadinimas.

- **Etikos problema**. Taip pat reiktų apsvarstyti ir etikos problemą – antivirusų kūrėjai norėdami neatsilikti ir visuomet pateikti atnaujinimus laiku, naršo virusų kūrėjų svetaines, todėl kartais kylą klausimas ar jie nesamdo virusų kūrėjų? Kita medalio pusė yra ta, kad priešnuodžių kūrimas visgi yra verslas, ir kartas nuo karto pasigirsta vienų antivirusų kūrėjų kaltinimai kitiems, jog jie žinojo apie virusą, tačiau nepasidalino informacija su savo kolegomis arba specialiai delsė prieš pranešdami.
- **Vartotojų problema**. Vis tik didžiausia problema, kad kompiuterių vartotojai nesilaiko elementarių saugumo reikalavimų [30, 32], jie neatnaujina savo programinės įrangos [33], kol ji pati neužsikrečia. Norėtųsi pabrėžti, jog aš čia kalbu ne apie antivirusų žodynų reguliarų atnaujinimą, kas turėtų būti savaime suprantama, bet apie visos, o ypač Windows operacinės sistemos atnaujinimus. Kirminai dažniausia išnaudoja neatnaujintos programinės įrangos spragas.

2.10. Alternatyvus sprendimo būdas

Be jau mano minėtų problemų iškyla dar viena, kurią geriausiai apibūdina tokie žodžiai:

„Šiuo metu yra daugiau virusų parašų žodyne, nei failų paprastame kompiuteryje.“ Andrew Jaquith, vyriausias analitikas the Yankee Group. [3]

„Saugumo kompanijos, kurios kuria atnaujinimus savo produktas teoriškai turi sukurti milijoną kenkėjiškos programinės įrangos parašų savo klientams.“ Yuval Ben-Itzhak, Technologijų direktorius , Finjan Inc. [34]

Nors antivirusų kūrėjai vis dar atkakliai laikosi virusų žodynų, tačiau vis daugiau kompiuterių saugumo specialistų tą metodą kritikuoja. Jeigu nenaudoti viruso žodyno tai ką daryti tuomet?

Kalbant apie kompiuterių tinklų saugumą [35] ten jau senai žinomi du ugniasienes konfigūravimo būdai:

- Leisti viską išskyrus tai ko negalima
- Drausti viską išskyrus tai kam galima

Nors kartais dar pasitaiko ir pirmas būdas, bet pakankamai retai, kadangi jis yra kvailas. Iki 1992 metų buvo žinoma 15 būdų, kaip nulaužti kompiuterį per tinklą ir todėl juos buvo

galima užblokuoti, visam kitam leidžiant veikti, tačiau po 1992 metų įvairių skirtingų atakų skaičius pradėjo drastiškai augti, internetas pasiekė masinį vartotoją. To pasekoje pirmo ugniasienės konfigūravimo būdo buvo atsisakyta. Ir masiškai pradėtas vartoti antrasis.

Jeigu pažiūrėtume į dabartinį kompiuterio saugumą pamatytume analogišką situaciją: virusų skaičius auga drastiškai, nors antivirusų kūrėjai dar spėja pateikti parašus laiku, tačiau dėl jų kiekio antivirusinės labai sulėtina kompiuterio darbą. Ir jau dabar darosi aišku, kad greitai bus peržengta ta riba, kai antivirusų kūrėjai nespės kurti parašų, o kompiuteryje esantis žodynas bus perdaug didelis ir trukdys stabiliam darbui su kompiuteriui. Tai kodėl gi neperėmus idėjų iš tinklų saugumo ir nepritaikius jų asmeninio kompiuterio saugumui?

Įdiegus gerų procesų sąrašą vietoj blogų:

- Nereiktų gilintis kokio tipo kenkėjas puola kompiuterį, supaprastėtų programų architektūra.
- Nereiktų ilgai kurti parašų, analizuoti veikimo principų, dekompiliuoti kodo, nes patys parašai būtų skaičiuojami paprasčiau.
- Neliktų nesusipratimų dėl pavadinimų, kadangi pavadinimus suteikia programinės įrangos kūrėjas.
- Nekiltų etikos problemų, nes niekas neslėptų gerų programų nuo konkurentų.

2.11. Procesų klasifikacijos paaiškinimas

Antivirusų kūrėjai kenkėjus skirto įvairias kategorijas, analogiškai stebint procesus, taip pat reiktų juos rūšiuoti, bent jau į gerus ir visus kitus. Siekiant padidinti aiškumą paprastam vartotojui buvo sukurtos 5 procesų kategorijos:

- Sisteminiai – tai procesai kuriuos išjungus, Windows operacinė sistema, gali nulūžti, pradėti rodyti klaidų pranešimus arba šiaip elgtis neprognozuojamai ir nestabiliai. Primygtiniai nerekomenduojama jų išjungti. Taip pat nereiktų šitos kategorijos maišyti su standartiniu Windows vartotoju System, jeigu procesas priklauso vartotojui System, tai dar nereiškia, kad jo negalima išjungti, dar daugiau, tai dar nereiškia, kad tas procesas geras.

- Geri – tai procesai, kurie dažniausiai priklauso vartotojo paleistoms programoms. Paleistos programos nebūtiniai gali turėti sąsają pvz. Rtvscan.exe procesas priklauso Norton AntiVirus programos rezidentiniam skeneriui. Taip pat viena programa nebūtinai paleidžia vieną procesą pvz. On-Screen Keyboard (standartine Windows programa) paleidžia du procesus: osk.exe ir msswchx.exe
- Blogi – tai procesai, kurie gali padaryti žalą sistemai. Tokių sistemoje būti neturėtų.
- Žinomi yra procesai, kurie normaliomis aplinkybėmis nėra geri, tačiau žmonės dėl įvairių priežasčių juos naudoja. Pvz. keilogeriai – paprastai šią programa reiktų laikyti kenksminga, kadangi jinai seka visą klaviatūra vedamą informaciją, tačiau kartais tėvai norėdami sukontroliuoti savo vaikus, stebėti kokiuose puslapiuose jie lankomi, jiems nežinant, paleidžia keilogerius.
- Nežinomi – programinė įranga nuolatos yra kuriama, todėl neišvengiama, nepriklausomai nuo kokio dydžio bus centrinė duomenų bazė, pasitaikys procesų, kurie bus nežinomi. Nežinomo statusas nėra amžinas, jis gali pasikeisti į vieną iš anksčiau minėtų keturių statusų.

Vienu metu procesas gali priklausyti, tik vienai kategorijai. Kad paprastam vartotojui būtų aiškiau klientinėje programoje kiekviena kategorija atitinka viena spalvą: sisteminiai – tamsiai žali, geri - šviesiai žali, blogi – raudoni, žinomi – geltoni, nežinomi – balti.

2.12. Išvados

- Šiame skyriuje buvo apžvelgta kenkėjiška programinė įranga, pristatytas jos apibrėžimas bei tipai: virusai, kirminai, trojanai, keilogeriai, reklaminė bei šnipinėjimo.
- Pristatyta antivirusų architektūra bei paaiškintas jų veikimo principas. Parodytos dabartinių antivirusų problemos: parašų sukūrimo, virusų pavadinimų, etikos ir vartotojų išprusimo.
- Pristatyta aktyvaus procesų stebėjimo metodo idėja bei procesų skirstymas į: sisteminius, gerus, blogus, žinomus, nežinomus.

3. AKTYVAUS PROCESŲ STEBĖJIMO PROGRAMOS PROJEKTAVIMO ESMINIAI ASPEKTAI

3.1. Projekto prielaidos ir tikslai

Projekto idėja yra pasinaudoti panašių į antivirusinės programos veikimo modeliu pakeičiant vieną esminę dalį: vietoj blogų programų skaičiuoti geras programas.

Prielaidos leidžiančios realizuoti šią idėją:

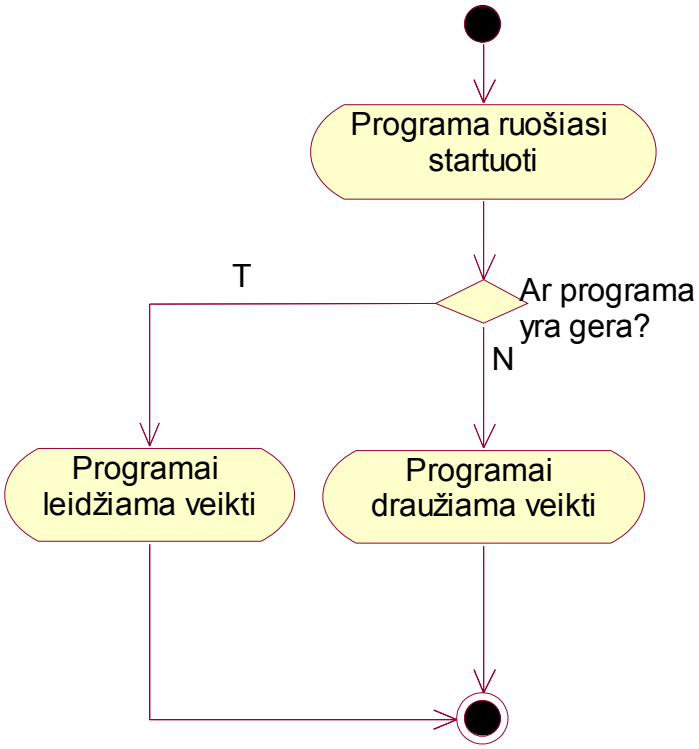
- Windows sistemos veikimas pagrįstas procesais.
- Kiekvienas procesas turi jį sukūrusį vykdomąjį failą.
- Gerų programų daug mažiau nei blogų.

Tikslai:

- Sukurti programą, kuri stebėtų sistemos veikiančius procesus.
- Programa turi galėti identifikuoti sistemoje veikiančius procesus.
- Programa turi leisti veikti tik geriems procesams sistemoje.
- Programa turi būti paprasta ir suprantama vartotojams.

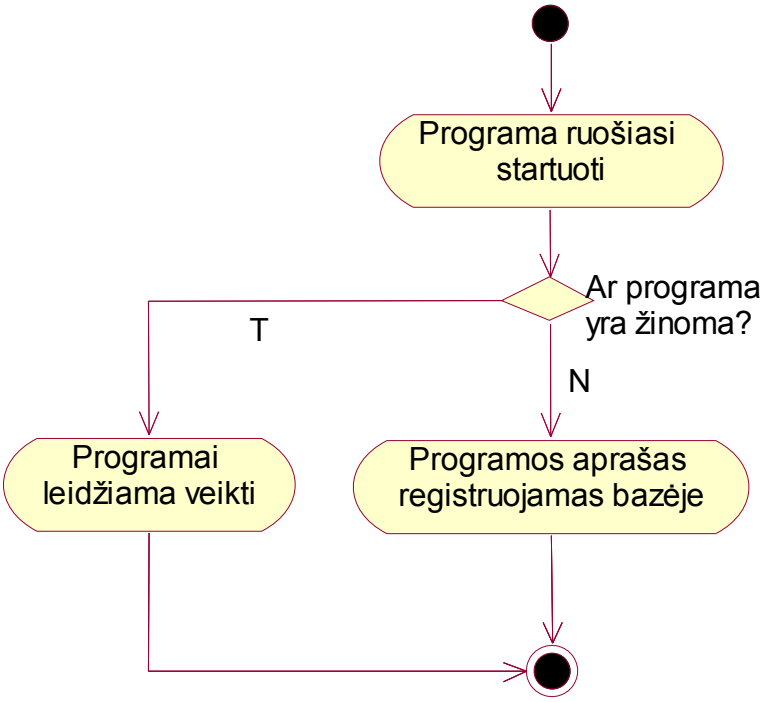
3.2. Procesų identifikavimas

Vienas iš reikalavimų yra, kad programa galėtų leisti veikti tik geroms programoms. Šis principas pavaizduotas 8 paveikslėlyje:



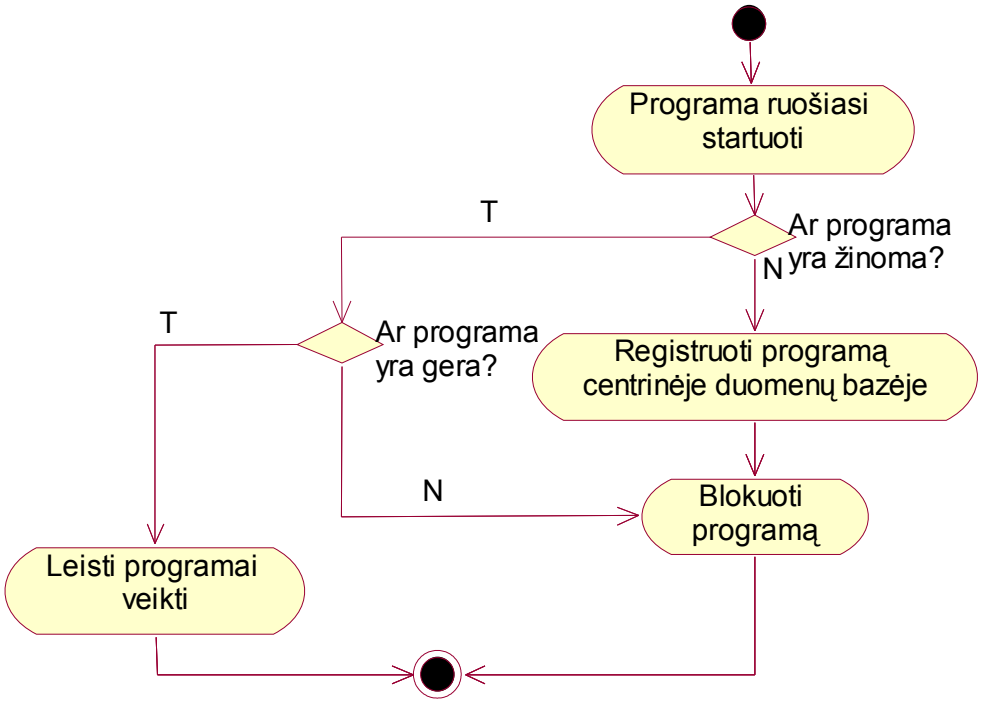
8 pav. Programos identifikavimas

Bandanti startuoti, programa patikrinama, jeigu gera – jai leidžiama startuoti, priešingu atveju neleidžiama. Klausimas: iš kur žinoti, programa gera ar bloga? Šiuo atveju reikės sukurti kažką panašaus į virusų parašus – gerų programų parašus. Taigi neišvengiamai bus reikalinga centrinė duomenų bazė, kurioje bus saugoma informacija apie geras programas, o klientinė dalis parsisiųs atnaujinimus. Sudarinėti sąrašą rankomis būtų ilgas, neefektyvus ir nuobodus procesas, todėl reikės į klientinę dalį įtraukti automatinį naujų procesų fiksavimą, patį procesą plačiau paaiškina 9 diagrama:



9 pav. Programos įtraukimas į duomenų bazę

Startuojant programai, pabandoma ją identifikuoti, jeigu pavyksta programai leidžiama veikti. Kitu atveju visa informacija apie procesą yra siunčiama į duomenų bazę. 8 ir 9 paveiksliukai yra panašūs, nors realizuoja du skirtingus funkcinius reikalavimus, todėl jie apjungiami į vieną ir programoje šis mechanizmas veiks taip kaip pavaizduota 10 diagramoje:

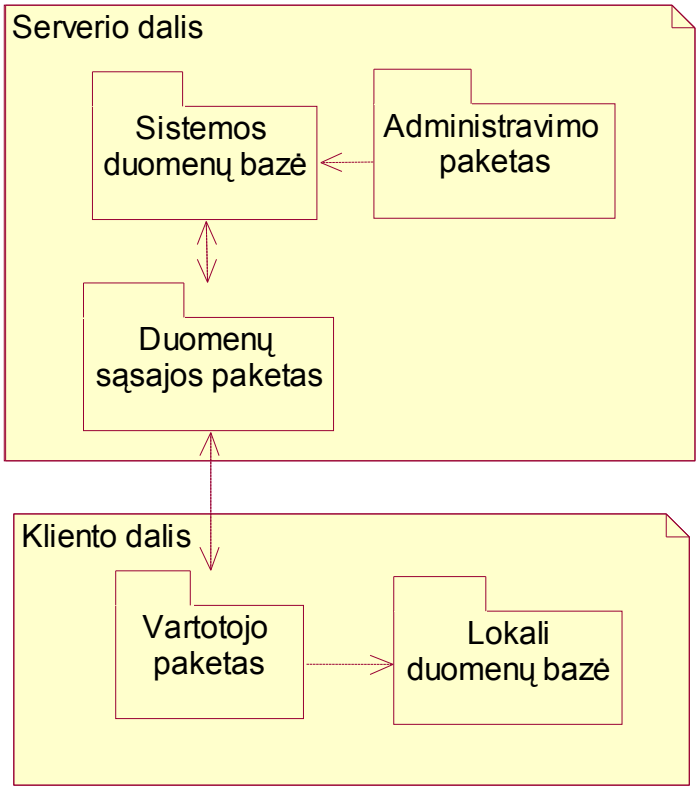


10 pav. Programos veiksmai ruošiantis startuoti naujam procesui

Pirmiausiai patikrinama ar programa yra sąraše, jeigu ne – ją reikia užregistruoti centrinėje duomenų bazėje ir jos veiklą blokuoti. Jeigu programa yra žinoma, dar nereiškia, kad jinai bus gera, nors ir pagrindinė idėja yra skaičiuoti geras programas, tačiau negalima užsimerkti radus kompiuteryje ir blogą. Taigi, dar vienas patikrinimas, ar programa gera. Ir tik tuo atveju jeigu programa gera jai leidžiama startuoti.

3.3. Bendra programos architektūra.

Kadangi jau aptarėme bendrus veikimo principus, dabar galima būtų aptarti bendrąjį architektūros vaizdą, kuris pavaizduotas 11 paveikslėlyje.



11 pav. Bendroji sistemos architektūros schema

Sistema susideda iš 5 pagrindinių posistemių: Sistemos administravimui reikalingų funkcijų, sistemos duomenų bazės, paketo, skirto užtikrinti bendradarbiavimą tarp serverinės dalies ir klientinės - vartotojo programos ir vartotojui skirtos lokalios duomenų bazės.

Kliento dalis – tai lokaliame kompiuteryje esanti nedidelė programėlė stebinti kompiuterio veiklą ir priklausomai pagal parametrus priimanti vieną ar kitą sprendimą. Ji yra atitikmuo antivirusinės programinės įrangos klientui. Lokali duomenų bazė – tai vieta, kur būtų saugomi gerų programų parašai. Tai virusų žodyno atitikmuo. Esminis skirtumas, jog virusų žodynas saugo visus žinomus virusus, todėl jis labai didelis, tuo tarpu čia bus saugomos tik konkrečiame kompiuteryje naudojamų programų parašai.

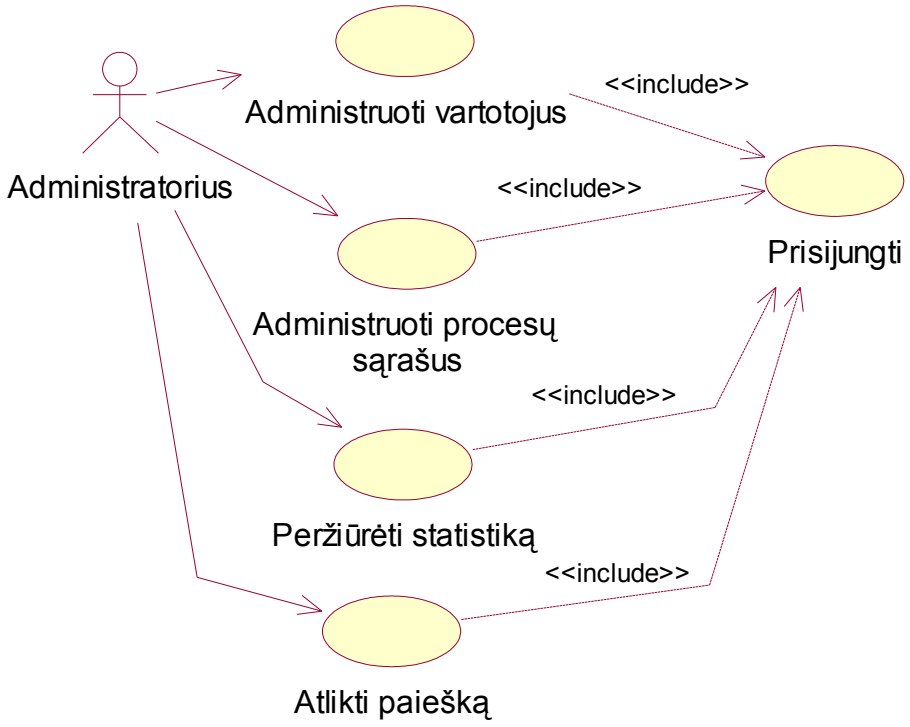
Serverio dalis – tai ta nematoma antivirusinių programų veikimo dalis, be kurios kliento kompiuterio saugumo užtikrinimo galimybės būtų ribotos. Duomenų sąsajos paketas yra skirtas klientinės programos ryšiui su serveriu. Jis turi atlikti dvi pagrindines funkcijas: naujų programų aprašymų priėmimą ir žinomų programų aprašymų pateikimą klientui.

Sistemos duomenų bazėje bus saugoma visa informacija apie geras programas.

Administravimo paketas yra skirtas prižiūrėti sistemos duomenų bazę. Norint kad informacija būtų kokybiška ir laiku pasiektų vartotojus, ją reikia nuolatos atnaujinti.

3.4. Panaudojimo atvejai

Kadangi sistema susideda iš dviejų dalių, tai ir panaudojimų atvejai yra atskiri abejoms dalims. Pirmiausiai išanalizuosime duomenų bazės administratoriaus galimybes, kurios pavaizduotos 12 paveikslėlyje.



12 pav. Internetinės dalies panaudojimo atvejai

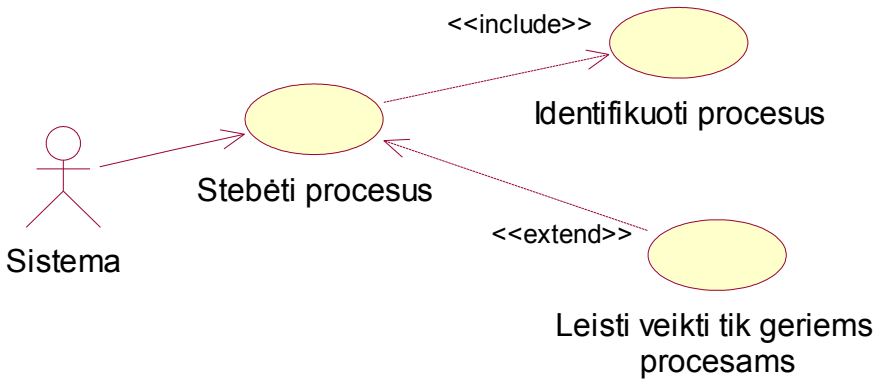
Internetinėje dalyje visi veiksmai galimi tik prisijungus prie sistemos.

Administruoti vartotojus – apima visą informaciją, susijusią su sistemos vartotojais: slaptažodžių/el. pašto keitimas, naujų vartotojų sukūrimas/ištrinimas/blokavimas.

Administruoti procesų sąrašus – apima visą informaciją, saugomą apie geras programas: aprašymų keitimas, naujų įtraukimas/pašalinimas iš sąrašo.

Peržiūrėti statistika – galima peržiūrėti dažniausiai naudojamų programų sąrašą bei dažniausiai užklaustų neidentifikuotų procesų sąrašą. Jeigu koks nors virusas atsirastų, jis greitai atsidurtų antrojo sąrašo viršuje.

Projekto tikslas – kiek galima mažiau apkrauti darbu vartotoją, todėl daugelį funkcijų atliks pati programa automatiškai be vartotojo įsikišimo. Klientinės programos veikimo principai pavaizduoti 13 paveikslėlyje.



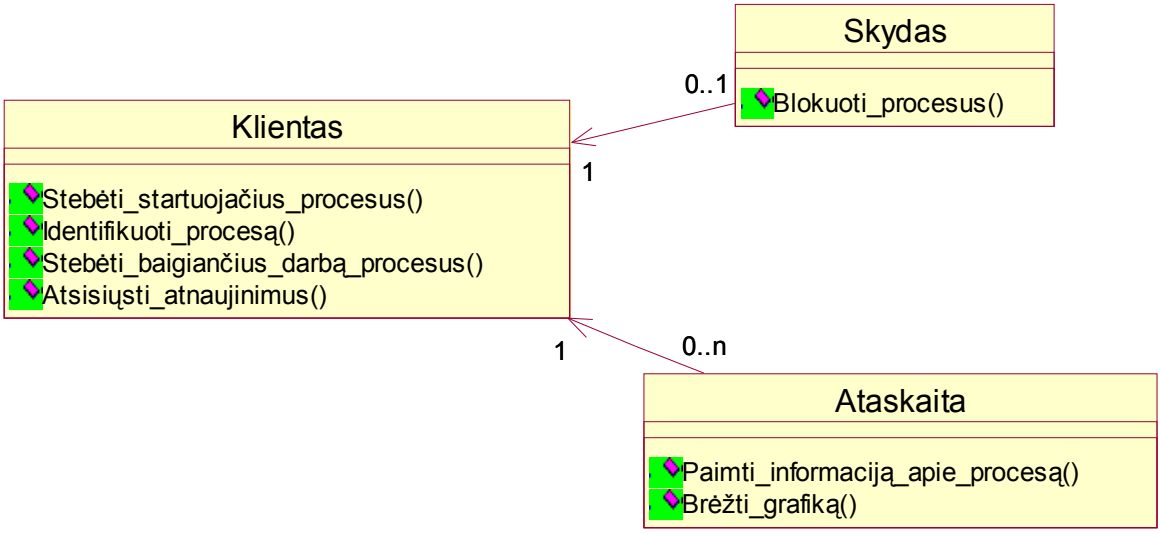
13 pav. Klientinės dalies panaudojimo atvejai

Sistema pati stebės visus procesus veikiančius sistemoje ir vartotojui pageidaujant galės pateikti detalią ataskaitą. Startavus naujam procesui programa visuomet jį bando identifikuoti. Proceso identifikavimas taip pat apima ir kreipimąsi į centrinę duomenų bazę bei proceso užregistravimą joje, jeigu informacija apie procesą nerasta.

Pagrindinė programos dalis yra procesų monitoringas, tačiau vartotojui pageidaujant programa gali blokuoti visus procesus, kurie nėra įtraukti į gerų procesų sąrašą. Prieš įjungiant šią programos funkciją reiktų porą savaičių leisti programai apsimokyti.

3.5. Bendras klasių diagramos vaizdas

Šiame skyriuje aptariama kliento klasių diagrama, kuri pavaizduota 14 paveikslėlyje.



14 pav. Apibendrinta klasių diagrama

Klasė „Klientas“ apima visą pagrindinę programos logiką: procesų stebėjimą, identifikavimą, bendravimą su serveriu, atnaujinimų pasiuntimą. Klasė „Skydas“ skirta procesų blokavimui, jinai naudojasi iš pagrindinės klasės gauta informacija. Klasė „Ataskaita“ parengia ataskaitą apie konkretų procesą bei leidžia stebėti, kaip konkretaus proceso resursų išnaudojimas kinta laike. Vienu metu galimų stebėti procesų skaičius neribojamas.

3.6. Duomenų saugojimo failo pavyzdys

Iš pradžių galvojau, kad geriausiai duomenis saugoti duomenų bazėje, bet tokiu atveju, klientinė programa būtų labai nedraugiška vartotojui, jeigu su ja kartu reiktų įsidiegti MySql ar kokią nors kitą duomenų bazių valdymo sistemą. Duomenis reikėjo saugoti kokiu nors formatu, kurį būtų galima lengvai apdoroti. Tam buvo pasirinktas XML failas tokiu formatu:

```
<?xml version='1.0' encoding='UTF-8'?>
<NewDataSet>
 <Table1>
 <Name>Proceso pavadinimas</Name>
 <Path>Proceso direktorija </Path>
 <MD5>Proceso md5 kodas</MD5>
```

<Status>Proceso statusas</Status>

<Comment>Proceso apibūdinimas</Comment>

<Count>Statistika</Count>

</Table1>

</NewDataSet>

Šio sprendimo trūkumai:

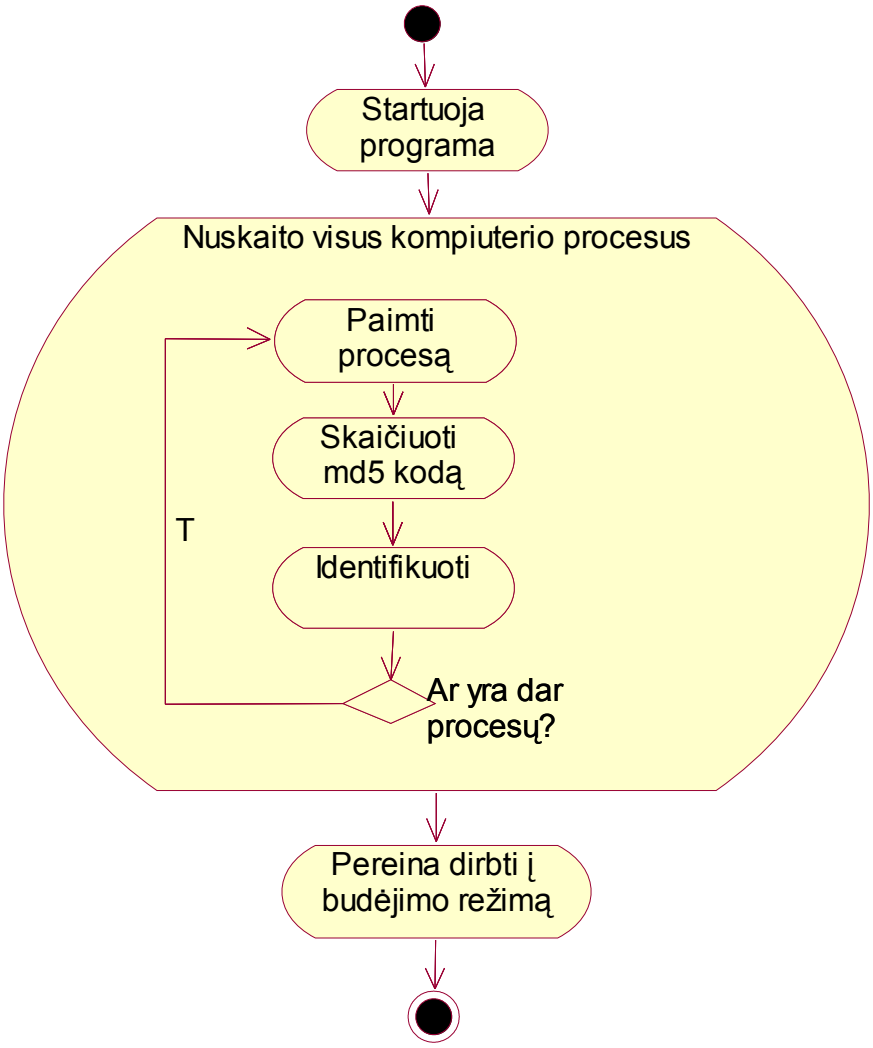
- XML failą lengva redaguoti, todėl kenkėjai gali save lengvai įtraukti į gerų procesų sąrašą. Galimas patobulinimas: koduoti visą arba bent dalį informacijos.

Šio sprendimo privalumai:

- XML failą lengva redaguoti.
- Galima peržiūrėti su standartinėmis priemonėmis Internet Explorer, notepad.
- Lengva keisti formatą, jeigu ateityje reiktu papildyti failą nauju lauku .
- Lengva nuskaityti ir atvaizduoti klientinėje programoje.
- Lengva pagaminti atskaitas iš XML failo.

3.7. Klientinės programos veikimo diagrama

Paleidus programą, jos startavimas užtrunka kelias sekundes, kad būtų galima suprasti kodėl taip vyksta pateikiu programos startavimo algoritmą 15 paveiksliukyje.



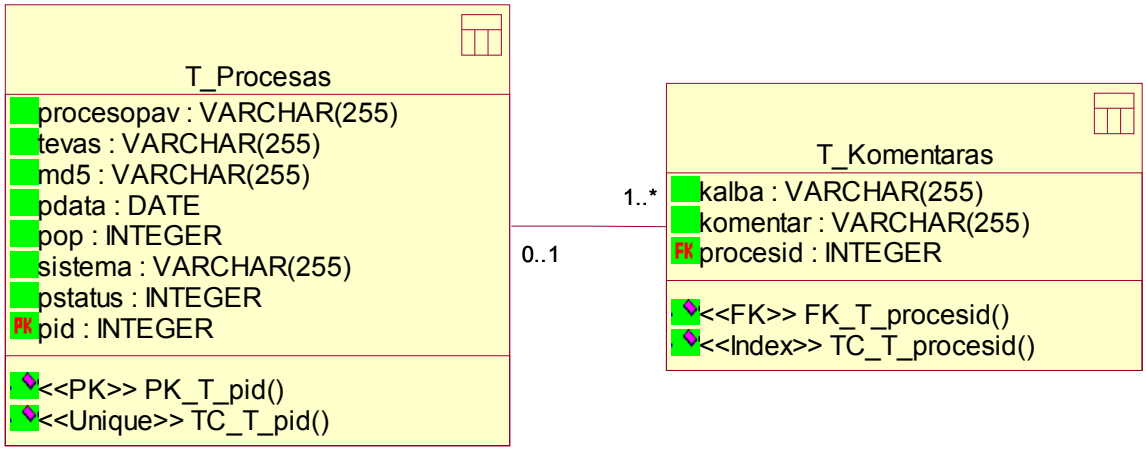
15 pav. Programos veikimo diagrama

Kaip matyti iš diagramos programa startavimo metu nuskaito visus kompiuteryje esančius procesus ir juos identifikuoja. Pirmą kartą programa startuoja gerokai ilgiau nei kitus, kadangi naujame kompiuteryje nėra sukurtas gerų programų sąrašas, todėl programa turi kontaktuoti su centrine duomenų baze, kad tą sąrašą sudarytų. Programa negali turėti standartinio sąrašo, nes neįmanoma surasti dviejų kompiuteriu su identiškomis konfigūracijomis.

Budėjimo režime programa atlieka panašius veiksmus kaip ir startavimo metu: kai tik pamato, jog naujas procesas bando pradėti darbą – jį identifikuoja.

3.8. Bendras duomenų bazės modelis

Serverio dalis atsakinga už duomenų saugojimą. Informacijos apie gerus procesus saugojimo duomenų lentelių schema pavaizduota 16 paveisklėlyje.



16 pav. Sistemos duomenų bazės modelis

Sistemos duomenų bazės esmė – kaupti informaciją apie procesus. Lentelėje „T_Procesas“ saugoma visa informacija apie procesą.

2 lentelė. Duomenų lentelės „T_Procesas“ laukų paaiškinimas

Laukas	Tipas	Ilgis	Komentaras
procesaspav	varchar	255	Proceso pavadinimas
tevas	varchar	255	Proceso tėvas
Md5	varchar	255	Proceso failo md5 kodas, skirtas proceso originalumui užtikrinti
pdata	date	-	Proceso užregistravimo data
pop	int	11	Proceso populiarumas, skaičiuojamas vartotojų užklausų apie procesą skaičius
sistema	varchar	255	Sistemos, kuriose veikia tas procesas
pstatus	int	11	Proceso statusas (blokuotas; aktyvuotas)
pid	int	11	Unikalus proceso identifikatorius, priskiriamas sistemos automatiškai. Lentelės raktas.

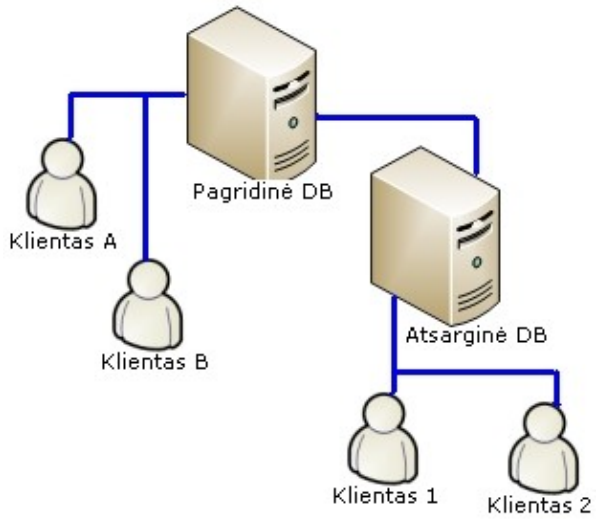
Lentelėje „T_Komentaras“ saugoma visa informacija apie procesų komentarus, komentarai apie vieną procesą gali būti keliomis kalbomis.

3 lentelė. Duomenų lentelės „T_Komentaras“ laukų paaiškinimas

Laukas	Tipas	Ilgis	Komentaras
procesid	int	11	Proceso identifikatorius. Išorinis raktas.
kalba	varchar	255	Komentaro kalba
komentar	text	-	Komentaras apie procesą.

3.9. Duomenų mainai tarp kliento ir serverio

Klientas ir serveris neišvengiamai turi kontaktuoti ir keistis duomenimis. Kadangi klientas ir serveris buvo realizuoti skirtingomis kalbomis (kliento pagrindas C#, serverio - PHP), tai reikėjo sprendimo, kuris sujungtų šias dvi technologijas. Duomenų mainams pasirinkau UDDI [36] standartą. Šis standartas buvo sukurtas specialiai kaip nepriklausomas šablonas duomenų mainams, jį palaiko tokios firmos kaip IBM, Microsoft ir daugelis kitų. Duomenų manai vyksta WSDL kalba SOAP protokolu. Šio sprendimo privalumas yra tas, jog mano duomenų baze gali naudotis ne tik mano klientas, bet ir trečių šalių sukurta programinė įranga. Taip pat šis sprendimas leidžia turėti kelias duomenų bazes, tarkim pagrindinį serverį ir atsarginę duomenų bazę. Atsarginė duomenų bazė save galėtų atnaujinti per tą pačią sąsają kaip ir klientas kiekvieną vakarą. Šią situaciją atvaizduoja 17 paveikslėlis.



17 pav. Duomenų mainų sąsajos galimybės

Naudojamas duomenų perdavimo būdas įgalino realizuoti storą klientą. Šio sprendimo trūkumai:

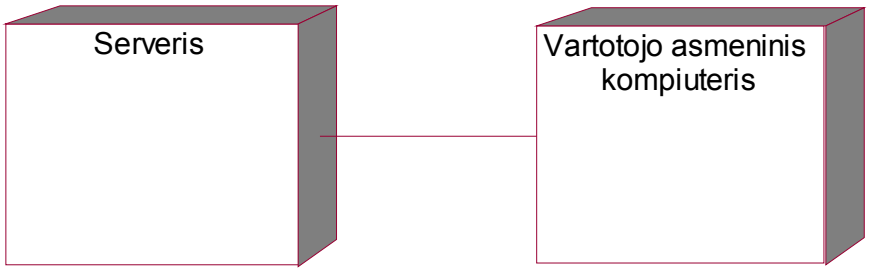
- Atnaujinus programą visiems vartotojams ją reikia įsidiegti iš naujo.

Šio sprendimo privalumai:

- Storas klientas nėra pasyvus stebėtojas, bet gali atlikti naudingą veiklą vartotojo kompiuteryje.
- Neapkrauna serverio resursų, nes visi skaičiavimai ir veiksmai atliekami kliento kompiuteryje.
- Gali suteikti daugiau informacijos apie procesus, kuri naudojant ploną klientą būtų neprieinama.

3.10. Išdėstymo vaizdas

18 paveikslėlyje pavaizduotas sistemos išdėstymo vaizdas. Ryšiui tarp serverio ir vartotojo palaikyti naudojamas UDDI [36] komponentas.



18 pav. Sistemos išdėstymo vaizdas

Minimalūs reikalavimai klientui: Windows XP SP2, Pentium III 500Mhz, 256MB RAM, 10MB vietos, internetas.

Reikalavimai serveriui: MySQL, Apache, PHP palaikymas

3.11. Statistinė projekto informacija

4 Lentelėje pateikiama apibendrinta realizuoto projekto statistika.

4 lentelė. Projekto statistika.

Kriterijus	Klientas	Serveris
Programavimo kalbų kodo eilučių skaičius		

PHP	0	1634
C#	2693	0
C++	189	0
Žymėjimo kalbų kodo eilučių skaičius		
XML	12*	0
XSL	23	0
WSDL	0	53
CSS	0	150
HTML	0	282
Viso	2917	2119
Kita statistika		
Klasių skaičius	5	3
Funkcijų skaičius	111	43

*kodas generuojamas automatiškai, standartiškai saugomas tik jo šablonas.

Šis projektas nėra standartinis savo idėja, todėl kaip matoma iš lentelės, teko priimti nestandartinius sprendimus ir į pagalbą pasitelkti daug kalbų. XML formatas gana universalus, todėl jo pagalba lengvai galima suderinti tokias iš pirmo žvilgsnio sunkiai suderinamas kalbas kaip PHP, C# bei C++ ir apjungti jas vienam projektui. Paprastai žmonės linkę daryti viską nuo pradžių iki galo viena kalba, kodėl to nebuvo galima daryti šiuo atveju? Visual Studio suteikia C# kalba komponentus tiesiogiai jungtis prie duomenų bazės, tačiau toks variantas nėra tinkamas saugumo sumetimais. Todėl lieka vienintelis sprendimas – kurti internetinį servisą. Šiuo atveju vėlgi būtų galima jį kurti su C#, kaip turbūt daugelis ir darytų, tačiau tuomet problema yra talpinimas. Internetiniam servisui parašytam su C#, būtinai reikia Windows Server 2003 IIS, tokios konfigūracijos talpinimai dažniausiai brangiai kainuoja, todėl aš pasirinkau PHP. Kad nereiktų galvoti savo taisyklių nusprendžiau realizuoti UDDI standartą [36]. Techniškai tai sudėtingas sprendimas, tačiau labai praktiškas.

3.12. Išvados

- Buvo suprojektuota ir sukurta programa, kuri identifikuoja procesus ir neleidžia startuoti blogiems.
- Dėl to kad pati programos idėja nėra standartinė, teko imtis nestandartinių sprendimų. Programa susideda iš dviejų dalių: storo kliento ir serverio. Kliento pagrindą sudaro C# kalba, o serverio – PHP.

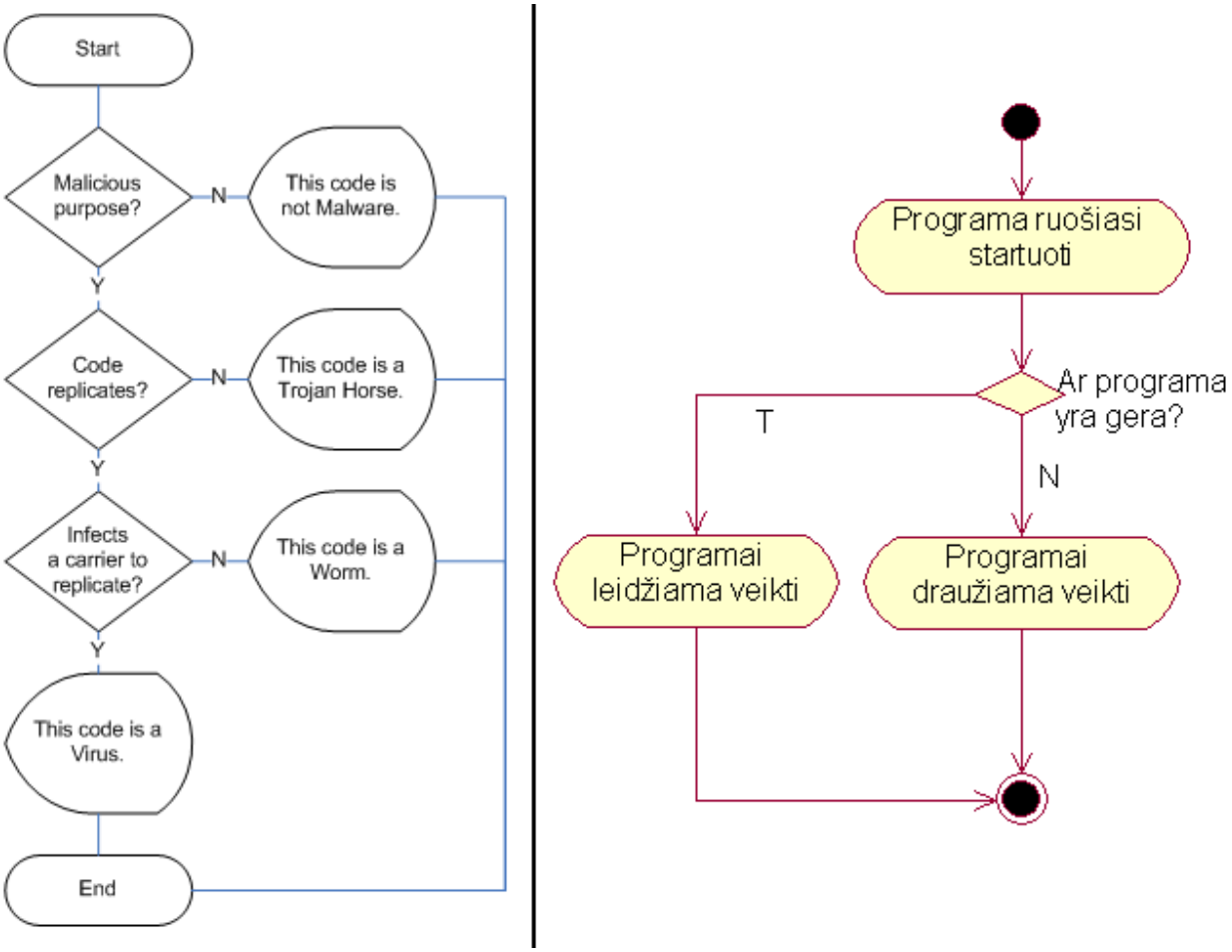
Duomenų mainams naudojamas UDDI standartas, kuris perduoda duomenis SOAP protokolu.

- Kuriant projektą, teko pasitelkti iš viso 8 skirtingas kalbas, parašyti daugiau nei 5000 eilučių kodo. Negalima sakyti, jog viskas yra idealu, yra trūkumų, kuriuos reiktų išspręsti ateityje planuojant tobulinti projektą.

4. AKTYVAUS PROCESŲ STEBĖJIMO PROGRAMOS KOKYBĖS TYRIMAS

4.1. Aktyvaus stebėjimo programos veikimo principų palyginimas su antivirusine programine įranga

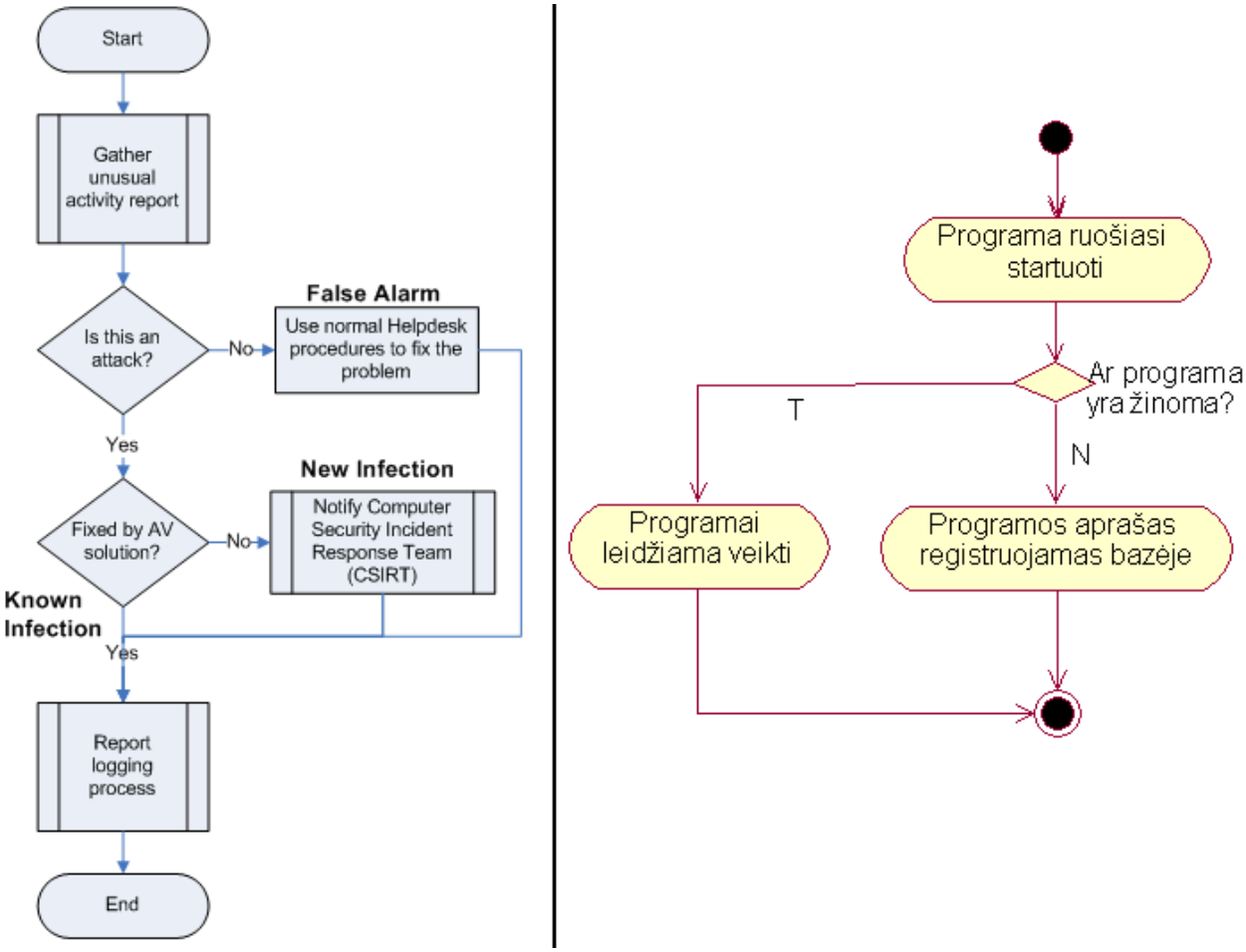
Pats aktyvaus stebėjimo programos veikimo principas labai panašus į antivirusinės programos, tai labai akyvaizdžiai iliustruoja 19 paveiksliukas:



19 pav. Antivirusinės [29] ir aktyvaus stebėjimo programų veikimo principo palyginimas

Antivirusinės programinės įrangos veikimo schema yra daug sudėtingesnė, kadangi jiems reikia nustatyti kenksmingos programos tipą. Tuo tarpu aktyvaus stebėjimo programa tiesiog skirsto procesus į gerus ir blogus, nesigilindama kokio tipo blogybė.

Antivirusų kūrėjai turi didelę virusų parašų bazę savo serveriuose, analogiška internetinė bazė turi saugoti gerus procesus aktyvaus stebėjimo programos atveju. Vienas iš būdų kaip antivirusų kūrėjai kaupia tuos įrašus bazėje yra klientinėje programoje įmontuota įtartinų programų aptikimo ir antivirusų kūrėjų informavimo mechanizmas. Aktyvaus stebėjimo programoje realizuotas analogiškas mechanizmas, palyginimo schema pateikta 20 paveikslėlyje:



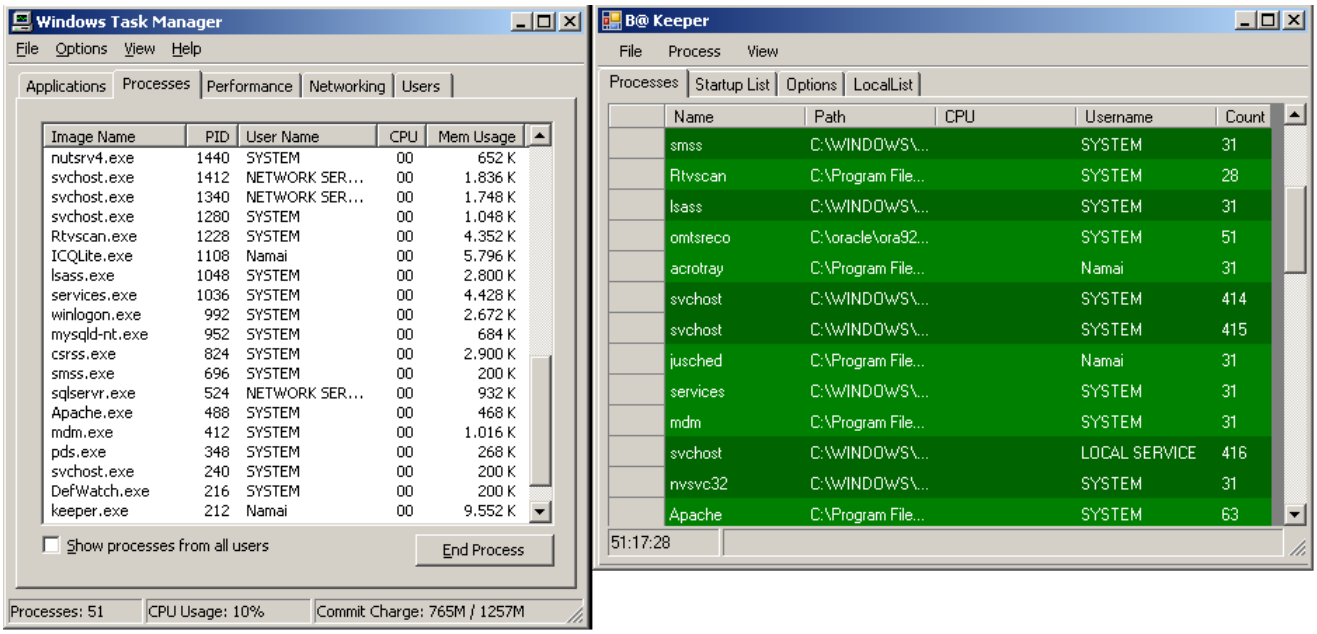
20 pav. Kenksmingos veiklos identifikavimas [29] ir naujų programų aptikimo palyginimas

Antivirusinė programa, aptikusi kenksmingą veiklą, pirmiausia bando pasitelkusi turimas priemones ją nukenksminti, ir jeigu nepavyksta, tuomet visą informaciją siunčią anitivirusų kūrėjams. Aktyvaus stebėjimo programa analogiškai elgsis su programomis, startavus

naujai programai pirmiausiai bandoma ją identifikuoti, nepavykus to atlikti, visa informacija apie naują programą registruojama internetinėje bazėje.

4.2. Aktyvaus stebėjimo programos grafinės sąsajos palyginimas su Task manager

Aktyvaus stebėjimo programos tikslas identifikuoti procesus, todėl visai nekeista, kad savo išvaizda ir funkcinėmis galimybėmis ji panaši į visiems vartotojams įprastą Task manager. Grafinės sąsajos palyginimas pateiktas 21 paveikslėlyje.

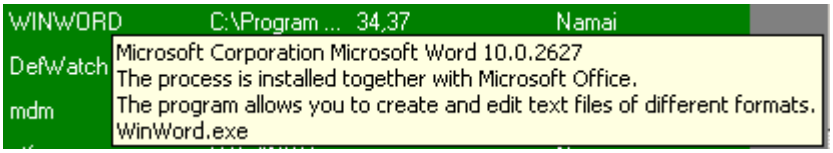


21 pav. Task manager ir aktyvaus stebėjimo programos grafinės sąsajos palyginimas

Kaip galima matyti iš 21 paveikslėlio sistemos atrodo labai panašiai. Standartiškai Task manager rodo tokius parametrus: proceso pavadinimą, proceso savininką, CPU, atminties apkrovimą. Aktyvaus procesų stebėjimo programa: proceso pavadinimą, direktoriją, CPU, proceso savininką. Tačiau pageidaujant abejuose programose galima įsidėti papildomų laukų, apie tai plačiau žiūrėti „4.3 Sistemos funkcionalumo palyginimas su Task manager“ skyriuje.

Pagrindinis skirtumas nuo Task manager, yra tas jog programa nuspalvina procesus priklausomai pagal jų tipus (apie procesų tipus bei spalvas žiūrėti skyrių 2.11 Procesų klasifikacijos paaiškinimas), tai padeda paprastam vartotojui geriau susiorientuoti sistemoje ir

suprasti kas joje veikia. Taip pat užėjus su rodykle ant proceso pavadinimo ir palaikius kelias sekundes programa pateiks proceso aprašymą, tai iliustruoja 22 paveiksliukas.



22 pav. Proceso komentaro pateikimas

4.3. Aktyvaus stebėjimo programos funkcionalumo palyginimas su Task manager

Jau aptarėme grafinės sąsajos panašumus, 5 lentelėje palyginamas funkcinis panašumas

5 lentelė. Task Manager ir projekto funkcijų palyginimas

Funkcija	Task Manager	Aktyvaus stebėjimo programa
Proceso pavadinimo rodymas	yra	yra
Proceso ID rodymas	yra	yra
Proceso išjungimas	yra	yra
Tėvo ID	nėra	yra
Tėvo pavadinimą	nėra	yra
Proceso savininko vardas	yra	yra
CPU naudojimas	yra	yra
CPU laikas	yra	yra
Atminties naudojimas	yra	yra
Piko atminties naudojimas	yra	yra
Virtualios atminties dydis	yra	yra
Puslapiavimo klaidos	yra	nėra
Puslapiuotos atminties dydis	yra	yra
Nepuslapiuotos atminties dydis	yra	yra
Maksimali atmintis išskirta procesui	nėra	yra
Minimali atmintis reikalinga procesui	nėra	yra
Vartotojo objektai	yra	nėra
Proceso prioritetas	yra	yra
Galimybė keisti proceso prioritetą	yra	yra
I/O rašymai	yra	nėra
I/O skaitymai	yra	nėra
Veikiančių programų rodymas	yra	nėra
Kompiuterio apkrautumo grafikai	Tik bendras visai sistemai	Tik kiekvienam procesui atskirai

Vartotojų rodymas	yra	nėra
Naujo proceso paleidimas	yra	yra
Proceso direktorijos rodymas	nėra	yra
Proceso direktorijos atidarymas	nėra	yra

Kaip matyti iš 5 lentelės aktyvaus procesų stebėjimo programa padengia daugelį Task manager funkcijų, taip pat prideda dar keletą papildomų, kurių tokio tipo programai reikėtų.

4.4. Aktyvaus stebėjimo programos funkcionalumo palyginimas su antivirusine programine įranga

Vis tik pagrindinis projekto tikslas yra pateikti alternatyvų požiūrį antivirusinei programinei įrangai, todėl neišvengiamai dalį funkcijų projektas turi padengti. Antivirusininės programinės įrangos ir aktyvaus procesų stebėjimo programos palyginimas pateiktas 6 lentelėje.

6 lentelė. Antiviruso ir aktyvaus procesų stebėjimo programos funkcijų palyginimas.

Funkcija	Antivirusinė programinė įranga	Projektas
Kompiuterio saugojimas	Virusų gaudimas	Leidimas veikti tik geriems procesams
Monitoringas	yra	yra
Konkretaus failo tikrinimas	yra	Tik vykdomuosius failus
Konkrečios direktorijos tikrinimas	yra	nėra
Automatinis atnaujinimas	yra	yra
Rankinis atnaujinimas	yra	yra
Aprašai	Tik virusų pavadinimai	Gerų procesų aprašai
Įvykių žurnalas	yra	nėra

Kaip galima pastebėti iš 6 lentelės, aktyvaus procesų stebėjimo programos padengia pagrindines antivirusinės programinės įrangos funkcijas. Vis tik aktyvaus procesų stebėjimo programos yra daug paprastesnė, kadangi neskirsto blogų procesų į potipius, taip pat jos tikslas nėra pagydyti sistemą. Antivirusinė programinė įranga saugo kompiuterį nuo virusų, ir, kai sistema būna užkrėsta, bando pagydyti. Tuo tarpu aktyvaus procesų stebėjimo programos leidžia sistemoje veikti tik geriems procesams, jeigu blogiems procesams nebus leista veikti, sistema nebus užkrėsta.

4.5. Aktyvaus stebėjimo programos trūkumai ir sprendimo būdai ateičiai

Aktyvaus procesų stebėjimo programa tikrina gerus procesus, tačiau pavyzdžiui javascript kalba aprašytas virusas nekuria naujo proceso, o veikia naršyklėje. Galimas patobulinimas – tikrinti procesus fredu lygiu, tokiu atveju būtų galima atskirti naršyklėje esantį eilinį puslapį, nuo javascript kenkėjo ir jį blokuoti.

Aktyvaus procesų stebėjimo programa nėra tinkama naudoti programuotojams, kadangi jie šiek tiek pataisę projektą vėl perkompiliuoja ir perkompiliuotas projektas, nėra analogiškas senam. Todėl aktyvaus procesų stebėjimo programa jo negali identifikuoti kaip gero. Galimas patobulinimas, tikrinti pagal tėvą, jeigu tėvas yra kompiliatorius, tuomet programą praleisti, bet tokiu atveju atsiranda saugumo spraga, kuria gali bandyti pasinaudoti kenkėjų kūrėjai

Gerų programų sąrašas lokaliame kompiuteryje saugomas XML formato faile, kurį gali redaguoti bet kas. Kenkėjų kūrėjai gali pasinaudoti tuo ir įrašyti savo programą, kaip gerą. Galimas patobulinimas – užkoduoti XML formato failą, bet tokiu atveju programos startavimas sulėtėtų.

4.6. Išvados

Tyrimo metu:

- Buvo palyginta antivirusinė ir aktyvaus procesų stebėjimo programinė įranga. Remiantis palyginimais nustatyta, kad antivirusinė programinė įranga realizuota žymiai sudėtingiau, nors esminiai principai yra tie patys.
- Aktyvaus procesų stebėjimo programa savo išvaizda labai primena Task Manager, taip pat padengia jo funkcionalumą, todėl vartotojams bus nesunku jį įsisavinti ir naudoti.
- Aktyvaus procesų stebėjimo programa padengia dalį antivirusinės programinės įrangos funkcijų, o konkrečiai tai aktyvų stebėjimą saugant kompiuterį nuo kenkėjų.

5. AKTYVAUS PROCESŲ STEBĖJIMO PROGRAMOS CHARAKTERISTIKŲ MATAVIMAS

5.1. Tiriamųjų kompiuterių charakteristikos

Nors programinė įranga buvo išbandyta ant daug įvairios konfigūracijos kompiuterių, tačiau eksperimentai buvo atliekami ant trijų:

Testuojamas1:

- Operacinė sistema: Windows 2003 SP1
- Procesorius: AMD Athlon XP 2600+, 1,91GHZ
- Atmintis: 512 RAM

Testuojamas2:

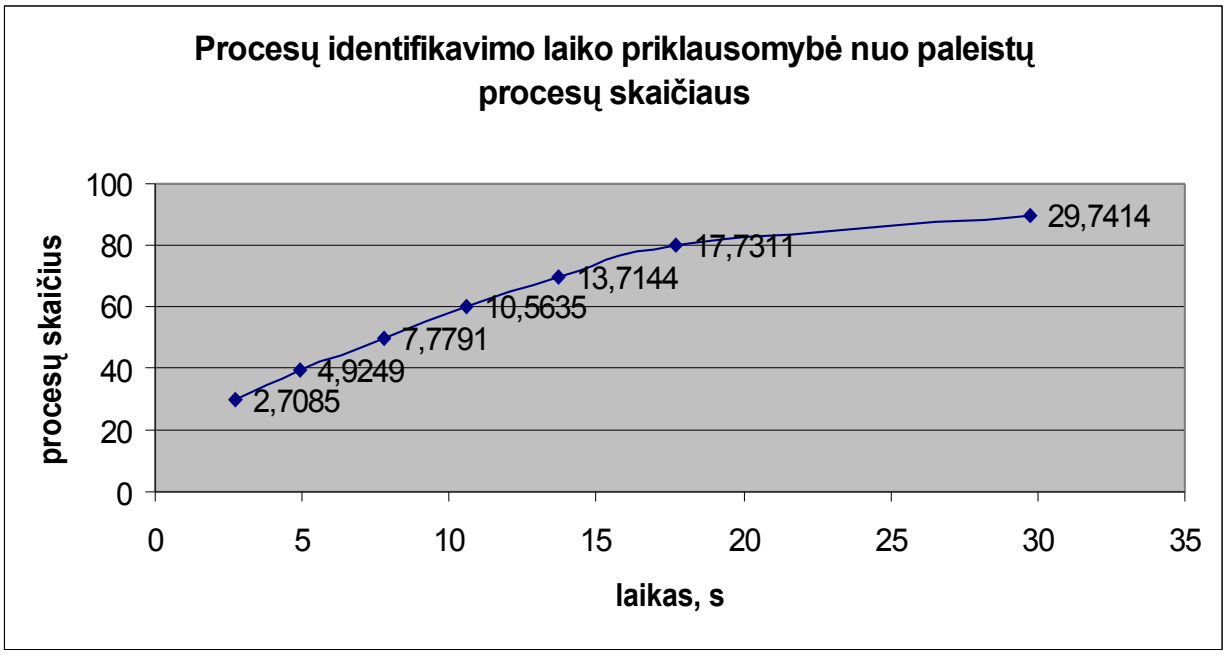
- Operacinė sistema: Windows XP SP2
- Procesorius: Intel Pentium M 2,00GHZ
- Atmintis 512 RAM

Testuojamas3:

- Operacinė sistema: Windows 2003 SP1
- Procesorius: Intel Pentium D, 3,40GHZ
- Atmintis: 512 RAM

5.2. Procesų identifikavimo laiko priklausomybė nuo procesų skaičiaus

Pirmiausiai norėjosi išsiaiškinti, kaip aktyvaus procesų stebėjimo programa apkrauna kompiuterio resursus. Jeigu programa perdaug apkrauna kompiuterį, tai vartotojai jos nenaudos. Kadangi vieno proceso startavimas trunka labai trumpai, jo laiko matavimas nėra tikslingas, todėl buvo pasirinktas momentas, kai programa startuoja, tokiu atveju jinai turi identifikuoti visus veikiančius sistemoje procesus. Aišku yra vienas niuansas – startuodama programa taip pat turi inicializuoti savo komponentus, tačiau į šį faktą nebus atsižvelgta. Matavimai buvo atlikti kompiuteryje *testuojamas1*, rezultatai pateikti 23 paveiksliukyje.

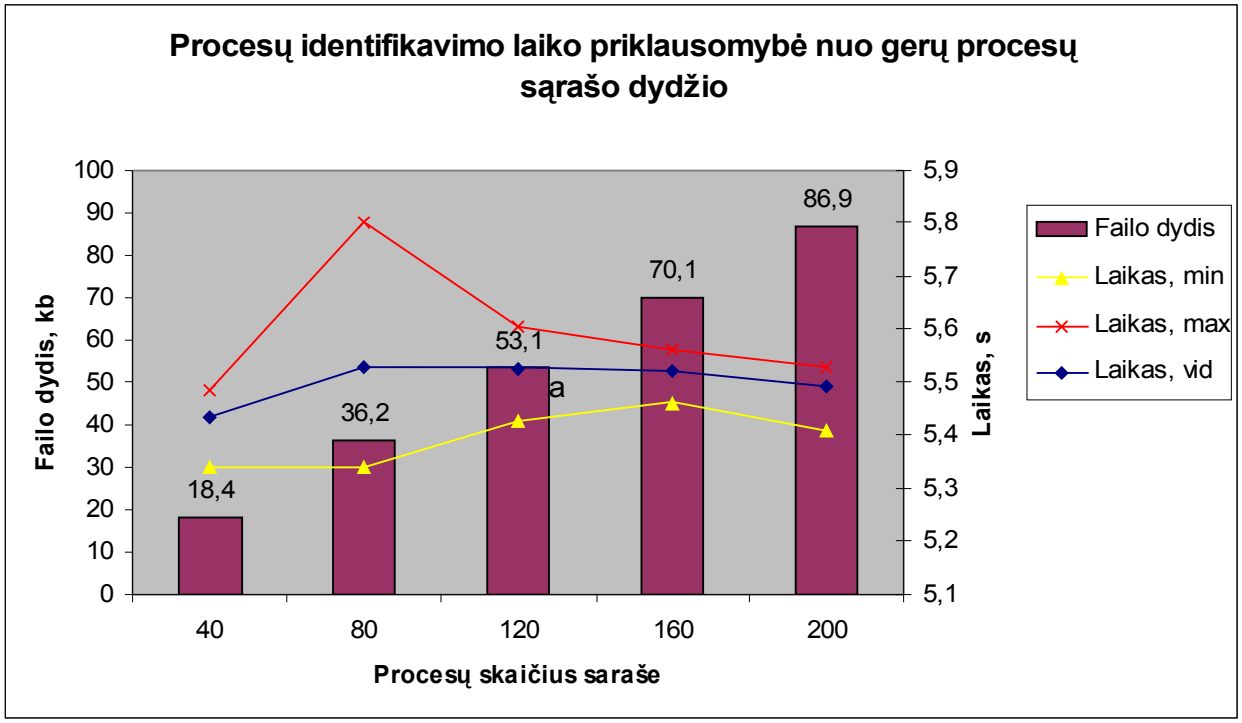


23 pav. Procesų identifikavimo laiko priklausomybė nuo paleistų procesų skaičiaus.

Grafike matome, esant 30 veikiančių procesų programa startuoja per nepilnas 3 sekundes, tačiau procesų skaičiui pasiekus 80 stipriai blogėja. Tai galima paaiškinti ne tik tuo, jog programai starto metu reikia nuskaityti daug procesų, bet ir tuo, jog daug procesų išnaudoja daug atminties bei CPU, ir programa turi tenkintis tuo kas lieka, todėl visi identifikavimo procesai šiek tiek užtrunka.

5.3. Procesų identifikavimo laiko priklausomybė nuo sąrašo dydžio

Naudojant programą ilgesnį laiką, procesų sąrašas vis ilgės. Labai tikėtina, kad pirmą kartą bus apie 40 procesų, tačiau po pusmečio skaičius gali išaugti iki 100. Šiuo tyrimu buvo bandoma nustatyti, kaip padidėjęs procesų sąrašas įtakoja sistemos veikimą. Vėlgi, kadangi matuoti vieno proceso laiką perdaug sudėtinga, buvo matuojamas programos startavimo laikas, kai sistemoje veikia 40 procesų. Matavimai buvo atlikti kompiuteryje *testuojamas1*, rezultatai pateikti 24 paveikliukyje.

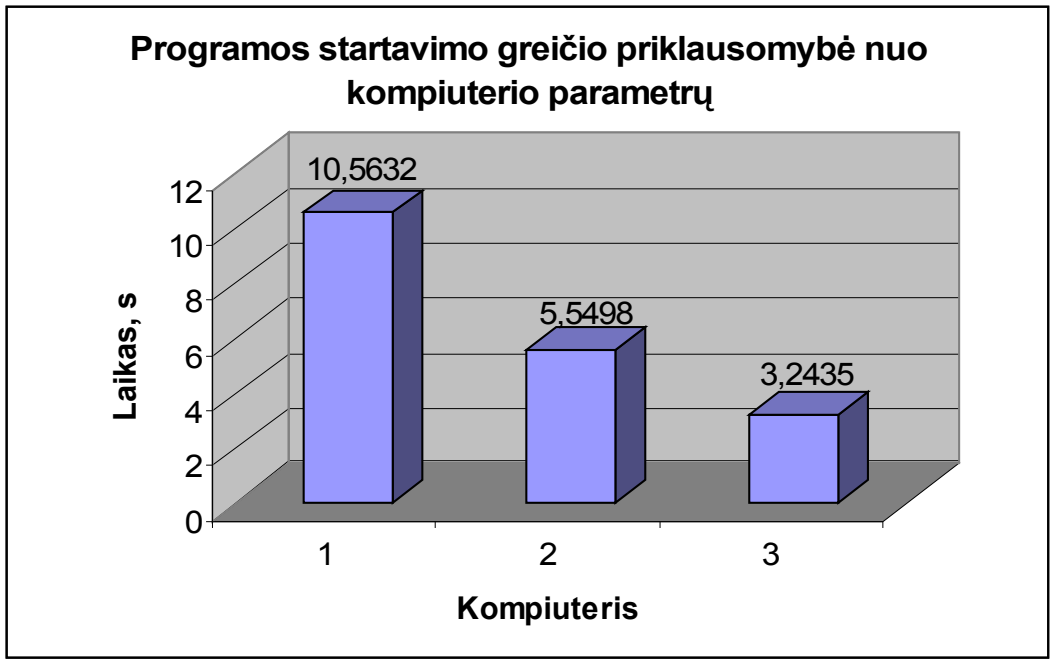


24 pav. Procesų identifikavimo laiko priklausomybė nuo gerų procesų sąrašo dydžio

Šioje diagramoje pateikti net keturi grafikai. Stulpelinė diagrama žymi failo didėjimą kilobaitais priklausomai nuo procesų skaičiaus tame faile. Tiesinės diagramos vaizduoja tris programos startavimo grafikus sekundėmis: lėčiausias, vidurkis, greičiausias. Remiantis grafiku galima būtų padaryti išvada, jog 200 procesų sąrašą apsimoka turėti labiau nei 80, tačiau tokia išvada būtų neteisinga. Programos startavimo laikas esant tokiomis mažomis reikšmėmis visiškai nepriklauso nuo sąrašo dydžio, galbūt kokią nors priklausomybę būtų galima pastebėti esant 10 kartų didesnėmis reikšmėms, tačiau tokio dydžio sąrašai yra nerealūs ir normaliomis sąlygomis niekada neturėtų būti sukurti. Vienintelis dalykas, nuo kurio tikrai priklauso programos užsikrovimo ir darbo greitis tai yra laisvos atminties kiekis ir procesoriaus dažnis. Kodėl ties 80 maksimali reikšmė yra gerokai labiau išsišokusi? Kai kurie procesai tam tikru metu atlieka įvairius veiksmus, pavyzdžiui Outlook tikrina ar nėra gautų naujų laišku arba Word atlieka automatines failo užsaugojimo operacijas. Todėl tam tikrais laiko momentais gali pasitaikyti netikėti ir neprognozuojami sistemos resursų sumažėjimai, ko pasekmės ir atsispindi grafike, tačiau būtent tai ir įrodo, jog programos greitis priklauso tik nuo laisvų sistemos resursų ir nepriklauso nuo gerų procesų sąrašo ilgio.

5.4. Procesų identifikavimo laiko priklausomybė nuo kompiuterio resursų

Skyriuje „5.3 Procesų identifikavimo laiko priklausomybė nuo sąrašo dydžio“ buvo iškelta hipotezė, jog programos veikimas priklauso tik nuo sistemos resursų ir nepriklauso nuo gerų procesų sąrašo ilgio. Norint patvirtinti šią hipotezę, buvo atlikti programos startavimo laiko matavimai ant skirtingų kompiuterių, rezultatai pateikti 25 paveiksliukyje.



25 pav. Programos startavimo greičio priklausomybė nuo kompiuterio parametrų

25 paveikslėlyje pateiktų kompiuterių charakteristikos:

1) Testuojamas1

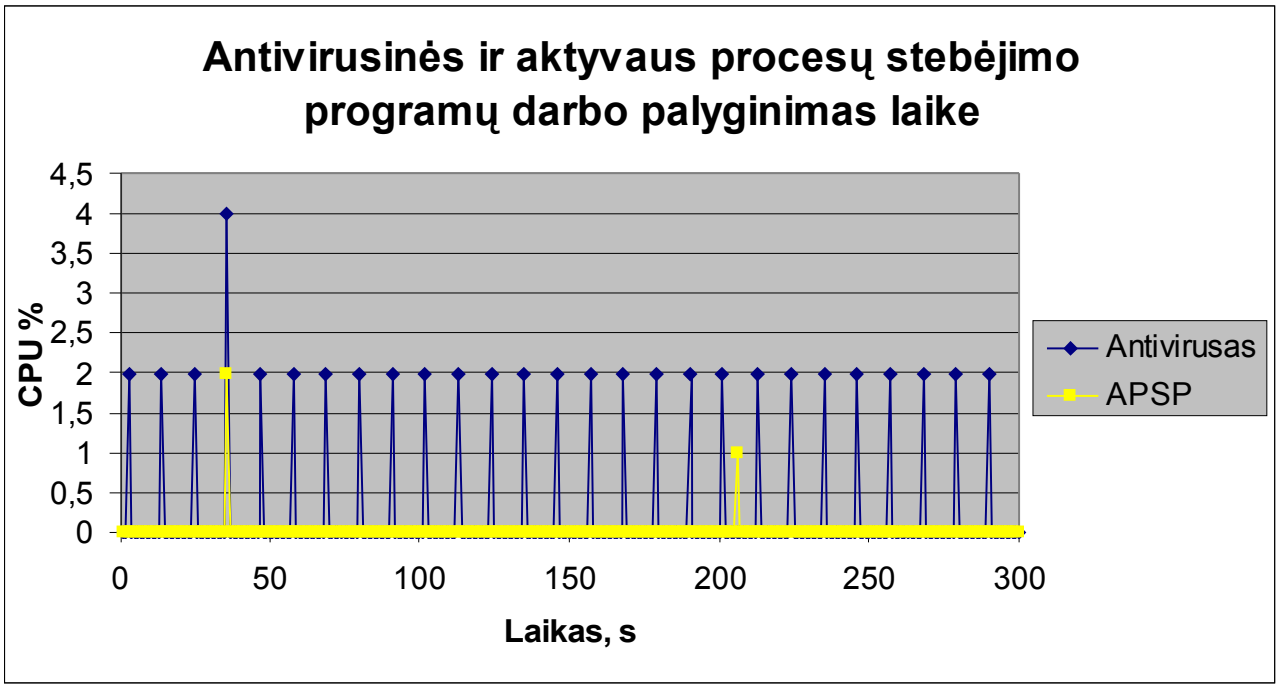
2) Testuojamas2

3) Testuojamas3

Visi matavimai buvo atlikti kompiuteryje paleidus 60 procesų. Kaip matyti iš rezultatų, kuo greitesnis kompiuteris, tuo programa užsikrauna greičiau.

5.5. Aktyvaus procesų stebėjimo programos palyginimas su antivirusu

Neužtenka pristatyti kitokį požiūrį į kompiuterių saugumą, reikia dar įrodyti, jog jis efektyvumu nenusileidžia jau esamiems. Todėl 26 paveikslėlyje pavaizduotas grafikas parodo CPU apkrovimą 5 minučių laikotarpyje.



26 pav. Antivirusinės ir aktyvaus procesų stebėjimo programų darbo palyginimas laike.

Kaip matyti iš grafiko antivirusinė programinė įranga reguliariai maždaug kas 11 sekundžių vykdo įvairus patikrinimus ir tai užima apie 2% CPU. Tuo tarpu Aktyvaus procesų stebėjimo programa (APSP), 5 minučių laikotarpyje matyti tik du šuoliai. Pirmuoju atveju startavo Outlook Express programa, antruoju – Notepad. Aktyvaus procesų stebėjimo programa neatlieka jokios veiklos, kol procesas nepradeda arba nebaigia darbo. Taigi, aktyvių kompiuterio resursų akivaizdžiai naudoja mažiau.

5.6. Populiariausi kompiuterio procesai

Pats populiariausias procesas yra svchost.exe. Šis procesas atsakingas už darbą su DLL failais, kompiuteryje visuomet veikia keli svchost.exe procesai, kompiuteryje testuojamas1 - 9. Windows XP sistemoje paprastai būna 6. Šis procesas dar ypatingas tuo, kad Windows žino

tiksliai kiek jų turi veikti, todėl vieną priverstinai išjungus, po kelių sekundžių jis vėl pasileis iš naujo.

Kiti sisteminiai procesai yra paleidžiami tik vieną kartą vienam vartotojui, todėl jų skaičius analogiškas vartotojų prisijungimų skaičiui.

Kiti populiariausi procesai yra kiekvienam vartotojui individualūs ir priklauso nuo jo darbo stiliaus. Labai tikėtina, kad tarp jų bus: naršyklė, teksto redaktorius, muzikos grojimo programa, pokalbių programos.

5.7. Kita statistika apie procesus

Duomenų bazėje pirmasis procesas užregistruotas 2006 spalio 4 d. *lsass.exe Microsoft Corporation LSA Shell 5.2.3790.0.* lsass.exe procesas atsakingas už Microsoft Windows saugumo mechanizmą. Konkrečiai atsako už prisijungimus ir vartotojų teisias. Išjungus šį procesą sistema gali nulūžti arba atjungti visus vartotojus, kurie tuo metu yra prisijungę.

Iš viso užregistruota daugiau nei 600 procesų, iš kurių apie 50 identifikuoti kaip sisteminiai. Kodėl tiek daug? Sistema buvo bandoma ant Windows XP bei Windows 2003, skirtingose sistemose visi procesai yra skirtingi nors pavadinimai yra tokie patys arba panašus, taip pat kai kurie sisteminiai procesai yra modifikuojami atnaujinimų, todėl Windows XP SP1 ir SP2 kai kurie procesai gali skirtis.

Pirmasis užregistruotas procesas vienodas Windows XP ir Windows 2003 sistemoms jusched.exe yra *Sun Microsystems, Inc. Java(TM) 2 Platform Standard Edition binary 5.0.60.5* Konkrečiai pats procesas yra Java Update varikliukas skirtas sekti, ar nėra Java programinės įrangos atnaujinimų ir jeigu yra parsiųsti tuos atnaujinus.

Gerų procesų sąrašas kompiuteryje labai priklauso nuo vartotojo darbo stiliaus. Kompiuteryje *testuojamas1* programai dirbant aktyvaus stebėjimo rėžime ilgiau nei mėnesį laiko buvo sudarytas sąrašas iš 65 skirtingų programų.

Minimalus procesų skaičius *testuojamas1* sistemai užsikrovus yra 38, *testuojamas2* – 34. Taip yra todėl, kad šiuose kompiuteriuose užsikrovimo metu nestartuoja jokia papildoma programinė įranga. Vartotojai dažnai mėgsta startavimo metu užsikrauti įvairias pokalbių programas, naršykles, todėl pas juos dažniausiai po užsikrovimo procesų skaičius yra gerokai didesnis.

Minimalus sisteminių procesų skaičius kompiuteryje *testuojamas1* vos tik jam užsikrovus yra 16. Tai nėra daug, kadangi net 9 priklauso svchost.exe.

Normalaus darbo rėžimo metu kompiuteryje *testuojamas1* vidutiniškai būna tarp 41 - 49 procesų. Tai yra darbo rėžimas, kai nešvaistoma nei savo laiko, nei kompiuterio resursų nepaleidžiant nereikalingos programinės įrangos. Kompiuteryje *testuojamas3*, kuris atlieka serverio vaidmenį, todėl prie jo nuolatos būna prisijungę 2 – 3 vartotojai procesų skaičius svyruoja tarp 60 - 75, reiktų pastebėti, jog tame kompiuteryje paleisti daugiau procesų nelabai įmanoma, nes laisvų resursų kiekis yra minimalus ir jų vos užtenka darbiniam rėžimui palaikyti.

Kopijuojat/Įdedant kokį nors paveiksliuką į MS Word failą visuomet pasileidžia MSpaint.exe procesas, kuris taip pat yra atsakingas už visiems gerai žinomos Windows Paint programos veikimą. Kopijuojat/Įdedant grafiką į MS Word failą pasileidžia graph.exe procesas.

On-Screen Keyboard (standartine Windows programa) paleidžia du procesus: osk.exe ir msswchx.exe, Taip pat Skype pokalbių programa standartiškai paleidžia du procesus: Skype.exe, SkypePM.exe.

Pagal Microsoft programų versijas galima tiksliai pasakyti kokią operacinę sistemą ir kiek atnaujinimų yra įdiegęs žmogus. Pavyzdžiui, svchost.exe Md5 kodas yra *ca8e6441930b54a8b8210061ce5fcce7*, sistemos pateikiama informacija yra *Microsoft Corporation Generic Host Process for Win32 Services 5.2.3790.1830*. Pagal šituos duomenis galima pasakyti, jog procesas priklauso Windows 2003 Server SP1 sistemai. Jeigu tas pats svchost.exe Md5 kodas yra *0f7d9c87b0ce1fa520473119752c6f79*, sistemos pateikiama informacija yra *Microsoft Corporation Generic Host Process for Win32 Services 5.1.2600.2180*. Procesas priklauso Windows XP SP2 sistemai

5.8. Išvados

Eksperimentų metu buvo nustatyta:

- Procesų identifikavimo laikas blogėja esant daugiau paleistų procesų, tačiau to pagrindinė priežastis sumažėjas laisvų resursų kiekis, tenkantis aktyvaus procesų stebėjimo programai.
- Procesų identifikavimo laikas nepriklauso nuo gerų programų sąrašo dydžio, jis priklauso tik nuo procesoriaus greičio ir laisvos atminties kiekio.
- Aktyvaus procesų stebėjimo programinė įranga apkrauna procesorių darbu tik naujo proceso startavimo metu, tuo tarpu antivirusinė programinė įranga reguliariai skenuoja kompiuterį.
- Gerų programų sąrašas yra individualus kiekvienam vartotojui. Eksperimento metu buvo sukurtas sąrašas iš 65 procesų, kitiems vartotojams šis sąrašas gali būti šiek tiek didesnis, tačiau vis tiek tai kelis šimtus kartų mažesnis sąrašas nei dabar siūlomas antivirusių gamintojų.
- Dažniausias procesas sistemoje yra svchost.exe, toliau seka vartotojo naudojama programinė įranga.
- Buvo sudaryta centrinė duomenų bazė su daugiau nei 600 programų aprašymų.

6. IŠVADOS

Šiame darbe:

- Apžvelgta kenkėjiška programinė įranga, pristatytas jos apibrėžimas bei tipai: virusai, kirminai, trojanai, keilogeriai, reklaminė bei šnipinėjimo.
- Pristatyta antivirusų architektūra bei paaiškintas jų veikimo principas. Parodytos dabartinių antivirusų problemos: parašų sukūrimo, virusų pavadinimų, etikos ir vartotojų išprusimo.
- Pristatyta aktyvaus procesų stebėjimo metodo idėja.
- Remiantis antivirusų architektūros pagrindiniais principais suprojektuota ir sukurta aktyvaus procesų stebėjimo programinė įranga, kuri identifikuoja procesus ir neleidžia startuoti blogiems.
- Aktyvaus procesų stebėjimo programos grafinė sąsaja labai primena Task manager, taip pat padengia jo funkcionalumą, todėl vartotojams bus nesunku jį įsisavinti ir naudoti.
- Aktyvaus procesų stebėjimo programa padengia dalį antivirusinės programinės įrangos funkcijų, o konkrečiai tai aktyvų stebėjimą saugant kompiuterį nuo kenkėjų.
- Buvo sudaryta centrinė duomenų bazė su daugiau nei 600 programų aprašymų.

Eksperimentų metu nustatyta:

- Procesų identifikavimo laikas blogėja esant daugiau paleistų procesų, tačiau to pagrindinė priežastis yra sumažėjas laisvų resursų kiekis tenkantis aktyvaus procesų stebėjimo programai.
- Procesų identifikavimo laikas nepriklauso nuo gerų programų sąrašo dydžio, jis priklauso tik nuo procesoriaus greičio ir laisvos atminties kiekio.
- Aktyvaus procesų stebėjimo programinė įranga apkrauna procesorių darbu tik naujo proceso startavimo metu, tuo tarpu antivirusinė programinė įranga reguliariai skenuoja kompiuterį.

7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

Brute Force – algoritmas išbandantis visas įmanomas kombinacijas, kol randamas teisingas sprendimas.

CD (Compact disc) - kompaktinis diskas, informacijos laikmena.

CPU (Central Processing Unit) – kompiuterio pagrindas, vieta kur atliekami skaičiavimai.

CRC (Cyclic Redundancy Check) – algoritmas skirtas patikrinti failo vientisumą.

KaZaa – P2P protokolu dirbanti duomenų apsikeitimo programa.

NTFS (New Technology File System) – failų sistema pirmą karta panaudota Windows NT operacinėje sistemoje. Dabar tapo standartu.

RAM (random-access memory) – virtualios atminties kiekis kompiuteryje, atmintis naudojama aktyviems programos veiksmams atlikti.

SOAP – duomenų apsikeitimo standartas pagrįstas XML kalba.

UDDI (Universal description, discovery and integration) - XML pagrindų sukurtas standartas duomenims apsikeisti.

USB (Universal Serial Bus) - šiuo atveju kalbama apie atminties laikmeną.

XML (Extensible Markup Language) – žymių kalba skirta apibūdinti įvairius duomenų formatus. Šios kalbos esmė, kad kiekvienas gali apsirašyti savo norimą žymę.

WSDL (Web Services Description Language) – XML pagrindų sukruta kalba skirta duomenų mainams tarp interneto servisų.

8. LITERATŪROS SĄRAŠAS

[1] Computer viruses now 20 years old, *BBC News* [interaktyvus] 2003 lapkritis. [žiūrėta 2007-04-16] Prieiga per internetą: Prieiga per internetą: http://news.bbc.co.uk/2/hi/technology/3257165.stm

[2] F-Secure Corporation's Data Security Summary for 2004, *F-secure*. [interaktyvus] [žiūrėta 2007-04-16] Prieiga per internetą: http://www.f-secure.com/2004/

[3] MCMILLAN Robert. McAfee sees 400,000 virus definitions by 2008. *Network World* [interaktyvus] 2006, birželis. [žiūrėta 2007-04-24] Prieiga per internetą: http://www.networkworld.com/news/2006/070606-mcafee-sees-400000-virus-definitions.html

[4] What is the difference between viruses, worms and Trojans? *Symantec* [interaktyvus] [žiūrėta 2007-04-25] Prieiga per internetą: http://service1.symantec.com/SUPPORT/nav.nsf/docid/1999041209131106

[5] What are computer viruses? [žiūrėta 2007-04-25] Prieiga per internetą: http://viruscenter.freedom.net/html/what_are_computer_viruses_.html

[6] HAYES Bill. Who Goes There? An Introduction to On-Access Virus Scanning, Part Two *Security Focus* [interaktyvus] 2002, rugsėjis. [žiūrėta 2007-04-25] Prieiga per internetą: http://www.securityfocus.com/infocus/1626

[7] FORNO Richard. Anti-virus industry: white knight or black hat? *Security Focus* [interaktyvus] 2004, vasaris. [žiūrėta 2007-04-18] Prieiga per internetą: http://www.securityfocus.com/news/8057

[8] MARTIN Brian, Anti-Virus Companies: Tenacious Spammers [interaktyvus] 2004, sausis [žiūrėta 2007-04-18] Prieiga per internetą: http://attrition.org/security/rant/av-spammers.html

[9] Virustotal - internetinis virusų skeneris. *Esecurity* [interaktyvus] 2007 gegužė. [žiūrėta 2007-05-17] Prieiga per internetą: http://www.esecurity.lt/article/2064.html

[10] RAIU Costin. A Virus by Any Other Name: Virus Naming Practices. *Security Focus* [interaktyvus] 2002, liepa. [žiūrėta 2007-04-30] Prieiga per internetą: http://www.securityfocus.com/infocus/1587

[11] VAMOSI Robert. What's in a virus's name? Everything you need to know! *ZDNet* [interaktyvus] 2002, vasaris. [žiūrėta 2007-04-30] Prieiga per internetą: http://review.zdnet.com/4520-6033_16-4206809.html

[12] SCHEIDL Gerald. Virus Naming Convention. *Ikarus Software* [žiūrėta 2007-04-30] Prieiga per internetą: http://members.chello.at/erikajo/vnc99b2.txt

[13] Deciphering virus names [žiūrėta 2007-04-30] Prieiga per internetą: http://www.geekgirls.com/basic_virus_guide.htm

[14] Virus naming conventions. *Symantec* [interaktyvus] [žiūrėta 2007-04-30] Prieiga per internetą: http://www.symantec.com/avcenter/vnameinfo.html

[15] How Spyware Works. *How Stuff Works* [žiūrėta 2007-04-26] Prieiga per internetą: http://computer.howstuffworks.com/spyware.htm

[16] What is a trojan, virus and worm. *Windows Guide* [žiūrėta 2007-04-25] Prieiga per internetą: http://windows.gumph.org/content/antivirus/what-is-a-trojan-virus-worm.html

[17] PIERCY Matt. The Spyware Threat And How To Deal With It. *Net Security* [interaktyvus] 2004, lapkritis. [žiūrėta 2007-04-29] Prieiga per internetą: http://www.net-security.org/article.php?id=746

[18] Adware. *Spyware Guide* [žiūrėta 2007-04-26] Prieiga per internetą: http://www.spywareguide.com/term_show.php?id=13

[19] BotNet. *Spyware Guide* [žiūrėta 2007-04-26] Prieiga per internetą: http://www.spywareguide.com/term_show.php?id=58

[20] HIGGINS Kelly. Black Hat: Botnets Go One-on-One. *Dark Reading* [interaktyvus] 2007 vasaris. [žiūrėta 2007-04-29] Prieiga per internetą: http://www.darkreading.com/document.asp?doc_id=117924

[21] How does anti-virus software work? [žiūrėta 2007-04-16] Prieiga per internetą: http://www.antivirusworld.com/articles/antivirus.php

[22] HAYE Bill. Who Goes There? An Introduction to On-Access Virus Scanning, Part One *Security Focus* [interaktyvus] 2002, rugsėjis. [žiūrėta 2007-04-17] Prieiga per internetą: http://www.securityfocus.com/infocus/1622

[23] HAYE Bill. Who Goes There? An Introduction to On-Access Virus Scanning, Part Two *Security Focus* [interaktyvus] 2002, rugsėjis. [žiūrėta 2007-04-17] Prieiga per internetą: http://www.securityfocus.com/infocus/1626

[24] NACHENBERG Carey. Behavior Blocking: The Next Step in Anti-Virus Protection *Security Focus* [interaktyvus] 2002, kovas. [žiūrėta 2007-04-17] Prieiga per internetą: http://www.securityfocus.com/infocus/1557

[25] MESSMER Ellen. Behavior blocking repels new viruses. *Network World* [interaktyvus] 2002, sausis. [žiūrėta 2007-04-19] Prieiga per internetą: http://www.networkworld.com/news/2002/0128antivirus.html

[26] BRIDWELL Larry, YANEZA Jaime. Virus Longevity—Virus Lifecycle, Tarptautinės konferencijos pranešimų medžiaga. AVAR, Sidnėjus (Australija), 2003 .

[27] NOD32 Antivirus Software. *Eset* [interaktyvus] [žiūrėta 2007-05-09] Prieiga per internetą: http://www.eset.com/products/index.php

[28] RIAU Costin. OpenAV: Developing Open Source AntiVirus Engines. *Security Focus* [interaktyvus] 2002, gruodis. [žiūrėta 2007-05-09] Prieiga per internetą: http://www.securityfocus.com/infocus/1650

[29] Antivirus Defense-in-Depth Guide. *Microsoft* [žiūrėta 2007-05-09] Prieiga per internetą: http://www.bizforum.org/whitepapers/microsoft-5.htm

[30] SCHMEHL Paul. Past its Prime: Is Anti-Virus Scanning Obsolete? *Security Focus* [interaktyvus] 2002, balandis. [žiūrėta 2007-04-19] Prieiga per internetą: http://www.securityfocus.com/infocus/1562

[32] ZENKIn Denis. Protecting Your Workplace: 10 Anti-Virus Rules. *Security Focus* [interaktyvus] 2001, vasaris. [žiūrėta 2007-04-19] Prieiga per internetą: http://www.securityfocus.com/infocus/1288

[31] STURGEON Will. CA antivirus deletes Windows component *Silicon.com* [interaktyvus] 2006, rugsėjis. [žiūrėta 2007-04-19] Prieiga per internetą: http://software.silicon.com/security/0,39024655,39162022,00.htm?r=2

[32] GORDON Jason. Lessons Learned from Virus Infections. *Security Focus* [interaktyvus] 2004, spalis. [žiūrėta 2007-04-29] Prieiga per internetą: http://www.securityfocus.com/infocus/1804

[34] WESTERVELT Robert. Attackers hide malicious code using new method. *Search Security* [interaktyvus] 2007, sausis. [žiūrėta 2007-05-10] Prieiga per internetą: http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci1238045,00.html

[35] MORRISDALE. The Six Dumbest Ideas in Computer Security. [interaktyvus] 2005, rugsėjis. [žiūrėta 2007-05-10] Prieiga per internetą: http://www.ranum.com/security/computer_security/editorials/dumb/

[36] The Universal Description, Discovery and Integration (UDDI) protocol. [žiūrėta 2007-05-20] Prieiga per internetą: http://www.uddi.org/

9. PRIEDAI

Kenkėjų tipai 2007-05-25 (http://www.esecurity.lt/article/2083.html)

Simonas Jusas

Anglai kompiuterių kenkėjams apibūdinti dažniausiai naudoja žodį malware, tuo tarpų lietuvių kalboje mes dažniausiai girdime žodį „virusas“. Tačiau virusas yra tik vienas iš kenkėjiškų programų tipų. Šiame straipsnelyje norėčiau pristatyti visus pagrindinius kenkėjų tipus ir parodyti jų esminius skirtumus.

Virusai yra mažos programėlės, kurios keičia kompiuterio veiklą. Virusai savo egzistavimui turi užkrėsti kitus failus arba kompiuterio užkrovimo sektorių (ang. boot sector). Tačiau virusai niekada negali įvykdyti savęs, juos turi paleisti žmogus. Todėl visuomet į pagalba yra pasitelkiama psichologija. Visai nesenai Skype programa plitęs virusas „Sandra“ kvietė pažiūrėti merginos nuotraukų.

Kirminai (ang. worm) kaip ir virusai yra sukurti pakenti kompiuteriui, taip pat kuria savo kopijas, tačiau priešingai nei virusai, jiems įvykdyti nereikalinga, kad žmogus paleistų kokią nors programą. Kirminai dažniausiai keliauja, kaip Microsoft Word arba Excel macro komandos ir neturi savo vykdomojo pradinio failo.

Kartais sakoma, jog kirminai yra tik virusų potipis, nes daugelius savo savybių jie yra panašus. Dažniausiai pilna įvairiomis informacijos pernešimo laikmenomis, seniau diskeliais, dabar kompaktinis diskais, USB atmintinėmis, el. laiškais. Mėgsta save užkoduoti, kad antivirusų kūrėjams, būtų sunkiau aptikti. Tuo tarpu kiti kenkėjai dažniausia plinta IRC ir kitų pokalbių programų pagalba, taip pat P2P tinklais ir dažniausiai savęs neslepia.

Trojanai (ang. trojan) - tai tokios programos, kurios apsimeta, jog yra labai reikalingos ir vertingos, tačiau iš tiesų taip nėra. Priešingai nei virusai ar kirminai, jos pačios nesidaugina. Trojanuose yra paslėptas kodas, kurį įvykdžius gali būti sugadinta sistema ar prarasti duomenys. Norint kad trojanas atsidurtų kompiuteryje, jį reikia atsisiųsti, jis gali atkeliauti su laišku kaip priedas.

Keilogeriai (ang. keylogger) – programinė įranga skirta stebėti ir fiksuoti visus klaviatūros paspaudimus, kartais savo užfiksavimus keilogeriai siunčia programos autoriui el. paštu. Taip ne tik pavagiama slapta, asmeninė informacija iš žmogaus, bet taip pat surandami jo slaptažodžiai.

Šnipinėjimo programinė įranga (ang. spyware) – priešingai nei virusai, šnipinėjimo programinės įrangos dažniausiai nežaloja kompiuterių, o prasibrovusios pasislepia kompiuterio atmintyje ir atlieka pakeitimus, stebi vartotojo darbą, vagia informaciją, užkraudinėja tam tikrus

puslapius, arba keičia paieškos rezultatus. Ši programinė įranga nesugeba pati atkeliauti iki vartotojo kompiuterio, dažniausiai įdiegiama kartu su kitomis programomis, pvz. anksčiau labai populiari programa KaZaa turėdavo labai daug šnipinėjimo programinės įrangos priedų. Kitas tokios programinės įrangos įdiegimo būdas yra, kai užėjus į kokį nors puslapį išmetamas pranešimas pranešantis, jog reikia įsidiegti papildomas tvarkykles (ang. driver), kad puslapis būtų matomas teisingai. Šnipinėjimo programinės įrangos darbas dažniausiai pasireiškia sulėtėjusiais kompiuterio tempais.

Reklaminė programinė įranga (ang. adware) – programos specialiai sukurtos pristatyti nepageidaujama reklamą vartotojams. Labai dažnai pasireiškia netikėtai iššokančiais langais. Į kompiuterį patenka kartu su kita įdiegiama programine įranga. Teoriškai įvairiose šalyse priimti vienokie ar kitokie įstatymai draudžiantis įdiegti reklaminę programinę įrangą be vartotojo sutikimo, tačiau dažniausiai vartotojai sutinka su licenzija jos neskaitę.

Botnetai (ang. botnet) – tai kompiuteriai, užkrėsti tam tikra programine įranga leidžiančia perimti tų kompiuterių kontrolę. Botnetai dažniausiai yra valdomi per IRC arba P2P tinklus. Dažnai botnetai pasitelkiami kitos blogos programinės įrangos įdiegimui pvz. reklaminės, arba nepageidaujamų laiškų siuntimui bei įvairioms atakoms.

Praktikoje labai sunku priskirti blogą programą konkrečiai kategorijai, kadangi dažniai jos derina kelių programų savybes. Pavyzdžiui, kirminas atklydęs į kompiuterį jame palieka trojaną, kad jo autorius galėtų sukurti botnetą, arba paleisti keilogerį. Botnetai labai dažnai yra kuriami, tam kad būtų galima uždirbti pinigus diegiant reklaminę arba šnipinėjimo programinę įrangą.

Vis tik manyčiau jog vertėtų pasistengti kenkėjus vadinti tikraisiais vardais. Ne viskas yra virusas, kas gadina jūsų kompiuterį.

Virusų pavadinimai 2007-05-14 (http://www.esecurity.lt/article/2067.html)

Simonas Jusas

Jeigu kada nors lankėtės pas būrėją, ar šiaip kokį šarlataną, tai jis turbūt jums pasiūlė atspėti jūsų ateiti iš delno, veido, vardo ar dar ko nors. Galbūt palyginimas nelabai geras, tačiau profesionalui viruso pavadinimas gali pasakyti tikrai labai daug. Norėčiau jums paaiškinti virusų vardų kūrimo taisykles.

Kai virusas „VBS/VBSWG.J“ plačiai pasklido internetu užkrėsdamas gausybę kompiuterių žiniasklaidą jį pakrikštijo „AnnaKournikova“, kadangi virusas atkeliaudavo su JPEG failu, kuriame atseit būdavo garsios tenisininkės nuotraukos.

Jeigu tą patį failą užkrėstą virusų nusiustumėte antivirusų kūrėjams gautumėte skirtingus atsakymus. Ar tai reiškia, kad tas failas buvo užkrėstas keliais virusais? Ne. Yra tam tikros virusų pavadinimo taisyklės, kurių susitarė laikytis antivirusinės programinės įrangos kūrėjai. Pirmą kartą 1991 metais susitarė dėl tokios formulės:

Šeimos_vardas.Grupės_vardas.Pagrindinė_versija.Minorinė_versija[:Papildomos_žymės]

Sutartis draudžia vartoti kompanijos vardą, prekės ženklus, bet kokių gyvų žmonių vardus, nebent įrodyta, jog tas žmogus viruso autorius. Tai paaiškina, kodėl ne antivirusų kūrėjai, žiniasklaida suteikia skambius vardus virusams. Vardų kūrimo konvencija, nori apsaugoti nuo autorinių teisių ir kitų pažeidimų, juk viruso pavadinimas kokios nors garsios kompanijos vardu, gali pridaryti didelių nuostolių tai kompanijai.

Tačiau dažniausiai tenka išgristi nesudėtingus antivirusų kompanijų sukurtus vardus, bet lengvus žodžius pvz. "Arual" (atvirkščiai "Laura"), "Golni" ("Winlog" apverstas ir sutrumpintas) arba "Nimda" ("Admin" apverstas). Iš kur atsiranda tokie žodžiai? Dažniausiai, nuo pirmos versijos užkrečiamojo failo pavadinimo [27] arba el. laiško antraštės sukuria pradėjo plisti kenkėjas.

Atsiradus naujiems virusams buvo keletas bandymų atnaujinti 1991 susitarimą, sėkmingiausias badymas 1999 metai pasiūlytas Gerald Scheidl, Ikarus Software.

Iš nagrinėkime keletą pavyzdžių ką viruso pavadinimas gali pasakyti apie patį virusą:

W32.Gokar.A@mm [28]

W32 reiškia, kad virusas veikia visuose 32 bitų Windows operacinėse sistemose (Windows 95, 98, Me, XP, 2000, 2003, NT). Taip pat galima sutikti W98 arba W2K nurodančia konkrečią operacinę sistemą pirmu atveju Windows 98, antru Windows 2000.

A žymi versija, reiškia, kad tai pirmas virusas iš šios šeimos. Kiti virusai gali susilaukti daug versijų pvz. W32.Bagle.BB@mm arba net W32.Gaobot.BOW.

@mm santrumpa angliškų žodžių mass mailing, kas reiškia, jog virusas atkeliavęs į pašto dėžutę išsiunčia save visais rastais pašto dėžutėje adresais.

JS.KakWorm.E@m

JS reiškia, jog virusas parašytas JavaScript kalba.

E – tai penktos kartos virusas, tai reiškia, jog viruso kodas buvo patobulintas jau 5 kartus. Kitą virusų versija nebūtinai turi sukurti pats viruso autorius, gali būti taip, jog netgi visas versijas sukūrė skirtingi žmonės.

@m žymi žodžius slow mailer, kas reiškia, kad toks virusas išsiunčia vieną laišką vienu metu iš adresų knygutės, todėl virusas plinta žymiai lėčiau.

W97M.Pacol.A

W97M reiškia Word 97 macro virusas.

A - reiškia, kad tai pirmas virusas iš šios šeimos.

Tikiuosi šis trumpas paaiškinimas jums padės aiškiau susigaudyti virusų pasaulyje. Dabar galėsite ir jūs paburti draugams jų kompiuterių ateitį iš viruso pavadinimo.

Ar antivirusai miršta? 2007-05-14 (http://www.esecurity.lt/article/2066.html)

Simonas Jusas

Antivirusai – programos, skirtos aptikti virusams ir kitai kenkėjiškai programinei įrangai. Pirmoji antivirusinė programa pasirodė 1988 ir buvo skirta aptikti ir pašalinti tik vieną virusą. Tokią antivirusinę programą galėjo sukurti vienas žmogus, nuo to laiko daug kas pasikeitė, antivirusai tapo labai sudėtingomis programomis. Ir vis tik manyčiau, jog jau neužkaltų ta diena kai antivirusus teks nurašyti į šiukšlių dėžę. Kad suprastumėte, jog mano teiginiai nėra išpiršto laužti pirmiausiai išsiaiškinkime kaip dabartinis antivirusas veikia.

Pagrindiniai antivirusų veikimo būdai:

• Tikrinimas pagal virusų žodyną

• Euristinis arba įtartinos veiklos aptikimas

• CRC kodo tikrinimas

Didžiosios dalies antivirusinės programinės įrangos pagrindinis instrumentas yra būtent tikrinimas pagal žodyną. Žodyne yra kaupiami virusų parašai. Programai startuojant antivirusas sutikrina, ar ji neturi viruso požymių, ir leidžia startuoti. Šis metodas pagrinde derinamas su skanavimu budėjimo režime, kadangi budėjimo režime antivirusas turi veikti greitai ir neapkrauti perdaug sistemos, nes priešingu atveju vartotojas ją paprasčiausiai išjungs. Budėjimo režime antivirusas tikrina el. pašto gautus laiškus, pradedančias arba baigiančias darbą programas. Užkrėsta virusu programa nebūtinai jį aktyvuoja darbo pradžioje, tai gali būti atliekama ir programai baigiant darbą.

Įtartinos veiklos tikrinimas atliekamas imituojant programos kvietimą ir žiūrint kaip jinai elgsis (dinamine euristika). Tai daug laiko ir resursų reikalaujantis darbas, todėl jis derinimas su kompiuterio skanavimu vartotojui paprašius. Paprastai žmonės linkę susitaikyti, su tuo kad kompiuterio skanavimas nuo virusų užtruks ilgai ir sunaudos labai daug sistemos resursų. Kitas euristinis metodas, seniau buvęs labai populiarus, yra taikomas asemblerio kalba parašytų virusų aptikimui. Toks būdas dar vadinamas statine euristika, kadangi tikrinamos asemblerio komandos ir žiūrima, ar jos neatlieka kokių nors keistų veiksmų, pavyzdžiui komanda patikrinanti, ar failas yra vykdomasis. Statinei euristikai reikia žymiai mažiau resursų ir laiko atlikti, todėl jinai labiau mėgstama antivirusų kūrėjų nei naujesnioji - dinaminė.

Anksčiau labai populiari, tačiau dabar beveik nebenaudojama technologija - CRC kodo tikrinimas. CRC kodo pati idėja buvo patikrinti, ar failas nėra su klaida, be to šis kodas labai

greitai paskaičiuojamas, tačiau virusų kūrėjai pastebėjo, jog tam tikros klaidos panaikina viena kitą ir netruko išnaudoti šio atradimo savo naudai.

Jums jau turbūt aišku, kodėl CRC kodo tikrinimas, buvo išbrauktas iš antivirusinių programų ginklų arsenalo, dabar pakalbėkime apie kitų metodų trukumus.

Statinės euristikos pagrindinis trūkumas, jog dažnai failuose, būna daug eilučių, ir nėra tiek daug laiko, kad būtų analizuojamas visas kodas, todėl skaitomas ribotas eilučių kiekis. Virusų kūrėjai sugalvojo apeiti šį metodą įterpdami viruso kodą, ne failo pradžioje, bet pabaigoje, ir kur nors kode įrašydami „jump“ komandą, iškviečiančią virusą.

Dinaminės euristikos trūkumas – resursų ribotumas. Kadangi imituojant sistemos veikimą virusui prieinamas tik ribotas resursų kiekis, tai virusų kūrėjai sugebėjo rasti būdus, kaip pasitikrinti jie veikia tikroje ar virtualioje sistemoje. Pavyzdžiui IDEA.6155 viruso užkrovimas vyksta, taip ilgai, kad antivirusinė programinė įranga pagalvotų, jog tai normali programa ir jį praleistų. Taip yra todėl, kad virusas save užkoduoja trijų lygių kodais besislėpdamas nuo antiviruso, o atkodavimas reikalauja labai daug resursų. Ant paprasto kompiuterio šio viruso užsikrovimas trunka apie 5 sekundes, tuo tarp virtualioje sistemoje gali užtrukti iki pusantros valandos. Kitas virusas - Magistr nekenkdavo kompiuteriui jeigu jame nebūdavo interneto. Dar įdomesnis 2002 metais atrastas W32.Simile, jis ne tik yra metamorfinis, tačiau kartu su savimi 14000 asmeblerio kodo eilutėse turi ir visą metamorfizmo varikliuką. Jis taip pat gali pereiti nuo Windows sistemos prie Linux. Tačiau labiausiai gąsdinanti jo savybė, kad jis prieš startuodamas atsitiktinai spėja ar šį kartą jam dekoduotis ir įvykdyti save, ar atidėti kitam kartui.

Pats seniausias metodas, kurio virusų kūrėjai vis dar atkakliai laikosi – virusų parašų tikrinimas. Pirmasis šio metodo trūkumas, jog naujieji virusai dažnai būna polimorfiniai t.y. kiekvieną kartą jie keičia kodą, todėl labai sunku jį aptikti ir sukurti konkretų parašą. O ir pats parašų sukūrimas nėra toks paprastas. Visų pirma antivirusų kūrėjai turi gauti tą virusą, taigi labai tikėtina, jog dalis sistemų jau bus užkrėstos. Antra - viruso parašo sukūrimas labai priklauso nuo pačio kenkėjo sudėtingumo, jeigu virusas tikrai sudėtingas, tai parašo sukūrimas, gali užtrukti nuo poros valandų iki poros dienų. Galu gale tuos parašus reikia atnaujinti ir nevisi vartotojai skuba tai padaryti.

Ir paskutinis vinis į antivirusinę programinę įrangą: šiuo metų yra daugiau nei 200,000 žinomų virusų, arba kitaip sakant daugiau įrašų apie virusus nei failų paprastame kompiuteryje. Ir kai tikrinama nuo virusų reikia patikrinti pagal visus parašus. Jus dar stebina, kodėl jūsų kompiuteris su antivirusu veikia taip lėtai?

Apibendrinant, galima pasakyti, jog dabartinė virusų ir antivirusų kūrėjų kova primena, pelės ir katės žaidimą. Jeigu žinote animacinį filmuką „Tom&Jerry“, tai turbūt puikiai pamenate, kas laimi tas animacines kovas...