KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ INŽINERIJA (P170B400)

Application Layer (L7) Firewall component

HRPI analyzer

Atliko ELITNET1:

Kazimieras Buškus, IFF-6/9 gr.

Šarūnas Andrijauskas, IFF-6/7 gr.

Robertas Strazdauskas, IFF-6/7 gr.

Tomas Jurevič, IFF-6/1 gr.

Priėmė:

lekt. Andrej Ušaniov

TURINYS

1. Bendra informacija 3

2. Trečias sprintas 3

2.1. Pagrindinės užduotys 3

3. Testavimas 3

3.1. Naudojama programinė įranga 3

3.2. Testavimo planas bei rezultatai 4

4. Išvados 5

# Bendra informacija

* Tvarkaraštis:

Semestro projektas: antradienis 15:30

Programų sistemų inžinerija: nelyg. sav. trečiadienis 11:00

* Programos *wiki:* https://github.com/sarand2/ELITNET1/wiki/
* Programos kūrimo backlog‘as:

**Email**: gaming\_fun@gmail.com

**Password**: labasrytas

* Komandos narių pareigos pagal atliekamus darbus:

Robertas Strazdauskas: GUI kūrimas ir duomenų bazė

Kazimieras Buškus: HRPI logika ir multiproceso įgyvendinimas

Šarūnas Andrijauskas: DDoS atakų generavimas ir multiproceso įgyvendinimas

Tomas Jurevič: paketų iš tinklo plokštės rinkimas

* Programos logika remiasi šiuo moksliniu straipsniu. [1]. Tongguang Ni, Xiaoqing Gu, Hongyuan Wang, and Yu Li, “Real-Time Detection of Application-Layer DDoS Attack Using Time Series Analysis,” Journal of Control Science and Engineering, vol. 2013, Article ID 821315, 6 pages, 2013. doi:10.1155/2013/821315

# Trečias sprintas

## Pagrindinės užduotys

Trečiajame sprinte buvo siekiama iki galo užbaigti visus sistemos komponentus ir juos sujungti į vieną visumą. Pagrindiniai darbai buvo:

* sukurti multiprocesinio paketų apdorojimo logiką
* pritaikyti HRPI entropijos skaičiavimą treniravimo duomenis realaus laiko veikimui
* sukurti efektyvius metodus dedančius bei imančius informaciją į/iš duomenų bazės
* sukurti minimalistinę GUI programos realaus laiko veikimui pavaizduoti

## Retrospektyvos rezultatai

* Dirbti poroje prie probleminių kodo dalių pasirodė ne tik lengviau, bet ir efektyviau.
* Iš anksto nesusipažinus su sudėtingesnių programos metodų testavimu, programiniai testai pasirodė užimsią, per daug laiko, jų atsisakymas šiuo sprintu baigus projektą atrodo pagrįstas.
* Kuriant GUI nuspręsta: geriau be daugiau galimybių suteikiantis *framework* ar *template*, o tas, kuris turi daugiau pavyzdžių.
* Derinant programą dėl *memory leak,* pirmiausia reiktų atsižvelgti į išorinių, mažiau pažįstamų API kvietinius.

# Testavimas

## Naudojama programinė įranga

* „Wireshark“ paketų gaudymo programa testuoti
* „Pylint“ statinei Python logikos kodo analizei
* „Resource“, „Objgraph“ Python moduliai *memory leak* diagnozuoti
* Chrome (version 66) bei Mozilla naršyklės (version 60) GUI komponentų vaizdavimui tikrinti

## Testavimo planas bei rezultatai

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testo ID**  **(GUI)** | **Scenarijus** | **Abstraktūs žingsniai** | **Tikėtinas rezultatas** | **Gautas rezultatas** | **Praėjo/nepraėjo** |
| **TEG01** | Įvedami neteisingi prisijungimo duomenys GUI | 1. Paleisti GUI naršyklėje 2. Įvesti neteisingus prisijungimo   duomenys | Neleidžiama prisijungti ir matyti GUI | Neleidžiama prisijungti | **Praėjo** |
| **TEG 02** | Į GUI įvesti istorijos intervalą ateityje arba tolimoje praeityje | 1. Paleistį GUI naršyklėje 2. Į laukus įvesti ateities/praeities intervalą | GUI grafike nevaizduoja traffic‘o  parametrų | Traffic‘o parametrai nevaizduojami | **Praėjo** |
| **TEG 03** | GUI turi atvaizduoti atpažintą atakos traffic‘ą | 1. Paleisti GUI naršyklėje 2. Paleisti ataką vykdantį scipt‘ą | GUI grafe turi po 10-20 sekundžių pradėti vaizduoti ataką reiškainčius taškus | Vaizduojami DDoS traffic‘o parametrai | **Praėjo** |
| **TEG 04** | GUI pasirenkant *Live* tab‘ą, turi būti vaizduojami realaus laiko traffic‘o parametrai | 1. Paleisti GUI naršyklėje 2. Pasirinkti *Live graph* skiltį | Vaizduojamas realaus laiko (su atsilikimu 1-2 s.) traffic‘o parametrų grafikas | Vaizduojami dabartinio traffic‘o parametrai | **Praėjo** |
| **TEG05** | GUI pasirenkant *History* tab‘ą, turi būti vaizduojami pasirinkto intervalo parametrai | 1. Paleisti GUI naršyklėje 2. Pasirinkti *History* skiltį 3. Pagal 3 įvedimo formas(data, valanda, valandos 5 minučių intervalas) pasirinkti norimą | Vaizduojamas pasirinkto intervalo traffic‘o parametrų grafikas | Vaizduojami pasirinkto intervalo traffic‘o parametrai | **Praėjo** |
| **TEG06** | GUI be anomalijų veikia ant Chrome 66, Mozilla 60 naršyklių | 1. Paleisti GUI iš atitinkamos naršyklės 2. Bandyti TEG01 TEG02 TEG03 TEG04 TEG05 atvėjus 3. Stebėti įvedimo/išvedimo komponentų anomalijas | Vaizduojamas GUI skirtingose naršyklėse skiriasi tik įvedimo formomis, anomalijų nerasta | Veikia kaip buvo tikėtina | **Praėjo** |
| **Testo ID**  **(traffic script)** | **Scenarijus** | **Abstraktūs žingsniai** | **Tikėtinas rezultatas** | **Gautas rezultatas** | **Praėjo/nepraėjo** |
| **TED01** | Paleistas DDoS/Normal sciptas IP adresų kiekiu bei iš vieno IP siunčiamų requestų kiekiu atitinka parametrus 10% ribose | 1. Paleisti traffic scriptą su norimais parametrais 2. Stebėti traffic‘ą *Wireshark* pagalba | *Wireshark* pateikiami duomenys nenukrypsta nuo parametrų per daugiau nei 10 % | Priklausomai nuo vykdymo laiko, pastebėtos didesnės nei scenarijuje nurodytos paklaidos | **Dalinai**  (priklauso nuo *proxy* serverių apkrautumo) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testo ID**  **(HRPI logic)** | **Scenarijus** | **Abstraktūs žingsniai** | **Tikėtinas rezultatas** | **Gautas rezultatas** | **Praėjo/nepraėjo** |
| **TEH01** | HRPI entropijos stebėjimo procesas sugeba atskirti DDoS bei normalų traffic‘ą | 1. Paleisti traffic scriptą su norimais parametrais 2. Stebėti traffic‘ą *Wireshark* pagalba 3. Stebėti traffic‘o klasifikavimą GUI, serveryje | GUI po 10-20 s. pradeda signalizuoti atakuojantį traffic‘ą, serveris vykdo nurodytus attack scriptus | Laikas, po kurio pradedama signalizuoti ataka, priklauso nuo proxy serverių apkrautumo, tačiau tiek GUI vaizdavimas, tiek script iškvietimai vykdomi teisingai | **Praėjo** |
| **Testo ID**  **(multiprocess queue)** | **Scenarijus** | **Abstraktūs žingsniai** | **Tikėtinas rezultatas** | **Gautas rezultatas** | **Praėjo/nepraėjo** |
| **TEM01** | Iš NIC requestai turi būti kaupiami teisingai bei neperkraunant atminties pagal paskaičiuotus dabartinius atakos requestų kiekių maksimumus (max 600 req./s). | 1. Paleisti traffic scriptą su norimais parametrais 2. Stebėti serverio resursų naudojimą kol vykdoma ataka 3. Stebėti request‘ų pasiskirstymą pagal *Wireshark* 4. Stebėti sistemos apkrautumą *Resource, Objgraph pagalba* | Serverio resursai (ypač RAM) neturi būti perkrauti atakos atveju, request‘ų sąrašai turi atitikti *Wireshark* | Dabartinėmis atakos sąlygomis, multiprocesinės eilės apdorojimas yra teisingas, bei našus | **Praėjo** |
| **Testo ID**  **(bendras)** | **Scenarijus** | **Abstraktūs žingsniai** | **Tikėtinas rezultatas** | **Gautas rezultatas** | **Praėjo/nepraėjo** |
| **TEB01** | Kodas testuojamas statinės analizės įrankiu | 1. Paleisti/įdiegti įrankį 2. Stebėti pranešimus | Kodas neturi kritinių klaidų, įspėjimų, refactorinimo siūlymų, kodo kartojimosi bendras kiekis (eilutėmis) turi neviršyti 10 % viso kodo | Įrankis rodo įspėjimus kodo refactorinimui, tačiau riba neviršijama | **Praėjo** |

# Išvados

Trečiuoju sprintu komanda baigė projektinį darbą. Projektas laikomas įgyvendintu, nes pavyko teisingai suprogramuoti pagrindinio komponento funkciją: programa geba atskirti tarp normalaus traffic’o ir atakos traffic’o būsenos.

Neįgyvendinti reikalavimai liko GUI dalyje: negalima pasirinkti platesnių intervalų traffic istorijos peržiūrėjimui, nėra *logout* funkcijos, *login* atliekama tik programiniame lygmenyje. Taip pat programa iki galo neištestuota dėl patirties trūkumo, laikas skirtas pagrindinių komponentų funkcionalumui įgyvendinti.

Matavimą, jog programos sudarytas ir naudojamas *Support vector machine* algoritmo taikymas testavimo metu atpažino ~93 % atakos traffic’o parametrų laikome projekto sėkmės rodikliu.