|  |
| --- |
| Харківський національний університет радіоелектроніки |
| (повна назва вищого учбового закладу) |
| Кафедра «Електронних обчислювальних машин» |
| (повна назва кафедри) |

КУРСОВИЙ ПРОЕКТ

|  |  |
| --- | --- |
| по курсу: | Системне програмне забезпечення |
|  | (назва дисципліни) |
| на тему: | Розробка програмного забезпечення для синтаксичного |
|  | аналізу файлів логів веб-сервера Apache |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Студента (ки) | 3 | | | курсу | | КІ-17-3 | групи |
|  | направлення підготовки | | | | Комп'ютерна інженерія | | | |
|  | спеціальності | | Системне програмування | | | | | |
|  | Антонова Д. О. | | | | | | | |
|  | (прізвище та ініціали) | | | | | | | |
|  | Керівник | Фомічов О. О. | | | | | | |
|  | cт. вик. каф. ЕОМ | | | | | | | |
|  | (посада, учбове звання, наукова ступінь, прізвище та ініціали) | | | | | | | |
|  | Національна шкала | | | |  | | | |
|  | Кількість балів: | | |  | | Оцінка: ECTS | |  |
|  |  | | | | | | | |

м. Харків – 2019 рік

РЕФЕРАТ

Пояснювальна записка до курсової роботи містить 32 с., 10 розділів, 4 додатки, список із 6 літературних джерел.

Об'єктом дослідження є программа, створена для аналізу файлув логів веб-сервера Apache.

В результаті виконання курсового проекту було отримано додаток, який повністю задовольняє поставленому завданню і доповнює його. Були вивчені різні можливі способи вирішення поставленого завдання, способи створення графічного інтерфейсу та написані алгоритми.

  Цільовою аудиторією цієї програми є люди, які хочуть ознайомитися з логами, провести їх фільтрацію і дослідити на безпеку.

Програма написана в середовищі Visual Studio на мові С# з використанням Microsoft.NET Framework.

ЛОГИ, ВЕБ-СЕРВЕР, БЕЗПЕКА, ВІКНО, ФІЛЬТРАЦІЯ, ФОНОВИЙ РЕЖИМ, ПОШТА

ЗМІСТ

ВСТУП..............................................................................................................................6

1 ПОСТАНОВКА ЗАДАЧІ ТА СФЕРА ЇЇ ВИКОРИСТАННЯ....................................7

2 ТЕОРЕТИЧНІ ВІДОМОСТІ………………………………………….........................8

2.1 Платформа .Net ………………………………………………………………8

2.2 Мова C#..........................…………………………………………………….10

2.3 Лог-файли …………………………………………………………………...11

3 ТЕХНІЧНЕ ЗАВДАННЯ……………………….........................................................13

4 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМИ…………...................14

4.1 Структура програми ………………………………………………………..14

4.2 Опис алгоритму роботи програми................................................................15

4.3 Структури даних.............................................................................................16

4.4 Перелік методів...............................................................................................17

5 ОПИС ПРОГРАМИ…………………..……………………………...........................20

6 ПОСІБНИК СИСТЕМНОГО ПРОГРАМІСТА…………………………………….25

7 ПОСІБНИК ОПЕРАТОРА…………………………………………………………..26ВИСНОВКИ....................................................................................................................27

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.......................................................................28

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ..........................................................................29

ДОДАТОК Б ВИГЛЯД ПРОГРАМИ ПРИ ЗАПУСКУ…............................................30

ДОДАТОК В ВИГЛЯД ПРОГРАМИ ПІД ЧАС РОБОТИ...........................................31

ДОДАТОК Г ВВЕДЕННЯ ПОШТИ……………………..............................................32

ВСТУП

Завданням для даної роботи є розробка програмного забезпечення, призначеного для аналізу логів веб-сервера Apache.

Усі сервера мають журнал з записами о подіях. Тому такий клас програм доволі популярний в сфері аналізу таких записів и подальшої роботи з ними, адже він дозволяє провести аналіз, фільтрацію и ввести подальший аналіз у фоновий режим.

В даній курсовій роботі описується, як програма дозволяє фільтрувати, аналізувати і входити у фоновий режим, продовжуючи слідкувати за файлом і аналізувати його. Окрім цього було реалізовано відправлення письма на пошту користувача у разі точного виявлення порушника.

Програма може використовуватися як засіб роботи, аналізу і фільтрації логів у домашніх цілях.

1 ПОСТАНОВКА ЗАДАЧІ ТА СФЕРА ІЇ РЕАЛІЗАЦІЇ

Програма повинна мати вікно, за допомогою якого користувач буде контактувати з програмою

Програма має надавати користувачу можливість фільтрувати записи у log файлі за датою, ім’ям файлу, ip запросу, типу результата, та виводити на екран останні n записів.

Програма, в залежності від обраних фільтрів виводить на екран запити, які задовільняють цим фільтрам, або весь файл, якщо жоден із фільтрів не був обраний.

Також повинен бути реалізований фоновий режим виконання программи. Користувач повинен мати можливість зупинити чи закрити програму у будь

який момент часу.

2 ТЕОРИТИЧНІ ВІДОМОСТІ

2.1 Платформа .Net

Платформа – це середовище, що забезпечує виконання програмного коду.

Фреймворк – це інфраструктура середовища виконання програм. Вона встановлює засоби взаємодії з операційною системою і прикладним програмним забезпеченням, методи доступу до баз даних і так далі.

Microsoft.Net це платформа, а NET Framework – інфраструктура цієї платформи, що містить два основних компоненти: Common Language Runtime (CLR) та .NET Framework Class Library (.NET FCL).

Common Language Runtime (CLR) – середовище, що відповідає за виконання керованого програмного коду і представляє служби, які спрощують процес розробки.

Керований код – код, який під час свого виконання має здатність використовувати служби CLR. Це дозволяє:

* керувати кодом, його загрузкою і виконанням;
* керувати пам’яттю;
* провіряти безпеку виконання;
* обробляти виключення;
* отримувати доступ до метаданих.

Фреймворк .Net представляє платформу для створення прикладного програмного забезпечення, основою якої є середовище Common Language Runtime (CLR), завдяки чому .Net підтримує декілька мов програмування: C#, VB.NET, C++, F# та інші. При компіляції код на будь-якій із цих мов компілюється в збірку на мові CIL (Common Intermediate Language), що дозволяє розробляти окремі модулі однієї програми на різних мовах.

Важливою особливістю платформи .Net є кросплатформеність, тобто можливість створювати програми, які можна буде виконувати на різних операційних системах. В цей список входить більшість сучасних ОС Windows (Windows 10/8.1/8/7/Vista), деякі ОС сімейства Linux, а також мобільні платформи Android та iOS.

Окрім цього можна виділити такі риси платформи .Net, як велику бібліотеку класів і різноманітність технологій. Наприклад, для роботи з базами даних призначена технологія ADO.NET, для створення графічних додатків технології WinForms, WPF та UWP, для створення веб-сайтів – ASP.NET і так далі.

Так як тема моєї курсової роботи зв’язана з IP адресою, вважаю необхідним уточнити її визначення.

IP-адреса - унікальна мережева адреса, необхідна для знаходження, передачі та отримання інформації від одного пристрою до іншого. IP-адреса присвоюється пристрою незалежно від величини мережі, до якої він підключений - це може бути як глобальний доступ в інтернет, так і локальна мережа між кількома пристроями.

2.2 Мова C#

Всі, хто знайомі з мовами С, С++ або Java, як правило, з легкістю будуть спроможні навчитися працювати на мові C#, адже вони досить сильно схожі. Але, незважаючи на це, C# представляє деякі додаткові можливості, що дозволяє спростити розробку та підвищити рівень безпеки і продуктивності таких програм. Наприклад тип значень Nullable, перерахування, делегати, лямбда-вирази, а також прямий доступ к пам’яті.

Як об’єктно-орієнтована мова, C# підтримує поняття інкапсуляції, наслідування, поліморфізму. Усі змінні та методи інкапсулюються в визначення класів. На відміну від С++, клас може наслідувати тільки один клас, але реалізовувати необмежену кількість інтерфейсів, тим самим виключаючи ромбовидне наслідування. Для методів, що перевизначають віртуальні методи, необхідне ключове слово override та final, щоб виключити випадкове повторне визначення.

Окрім основних об’єктно-орієнтованих принципів, мова С# спрощує розробку програмного забезпечення завдяки кільком новим конструкціям:

* делегати – конструкції, що підтримують покажчики на методи;
* властивості, що виступають в ролі методів доступу к закритим змінним;
* атрибути з метаданими о типах;
* інтегровані коментарі XML-документації;
* LINQ – механізм, що дозволяє роботи запроси на різні джерела даних.

Отже можна сказати, що процес побудови програм простіший ніж в С та С++, але більш гнучкий ніж в Java. Тут не існує окремих файлів заголовка, а методи та типи не потрібно об’являти в якомусь встановленому порядку. Окрім цього C# підтримує покажчики та поняття небезпечного коду, коли прямий доступ к пам’яті має дуже велике значення.

2.3 Лог-файли

Логи (лог-файли) - це файли, що містять системну інформацію роботи сервера або комп'ютера, в які заносяться певні дії користувача або програми. Іноді також вживається російськомовний аналог поняття - журнал.

Їх призначення - протоколювання операцій, що виконуються на машині, для подальшого аналізу адміністратором. Регулярний перегляд журналів дозволить визначити помилки в роботі системи в цілому, конкретного сервісу або сайту (особливо приховані помилки, які не виводяться при перегляді в браузері), діагностувати зловмисну ​​активність, зібрати статистику відвідувань сайту.

Є кілька видів логів:

- Список доступу. Показують, який користувач, в яку дату і час, з якої посиланням перейшов на ресурс і який була отримана відповідь. Дані записи допомагають знайти вразливе місце, якщо ресурс зламають.

- Список помилок. Показують помилки, що видаються при функціонуванні сайту або в процесі звернення до його деяких функцій. Тут є можливість відшукати і ліквідувати баг, що приводить до помилки.

- Інші логи. Фіксують події в різних компонентах сервера: логи пошти сервера і т.п.

Якщо веб-сайт працює нормально, в штатному режимі, немає необхідності переглядати лог-файли. Але бувають випадки, коли сервер раптово перевантажується, ресурс піддається спаму, видає достаток помилок або виникають проблеми в ранжируванні в пошукових системах.

В такому випадку системні адміністратори або seo фахівці починають аналізувати відвідувачів, ідентифікувати доступ до файлів з боку сторонньої особи, а саме IP-адреса, звідки він був здійснений, після чого роблять відповідні висновки.

Список зберігаються в файлі access.log в системній папці будь-якого сервера, будь то Nginx, Apache або будь-який інший. Лог-файли відкриваються через текстові редактори. Кожен рядок тут відповідає не більш, ніж одному зверненню.

Програма, розроблена в ході цього проекту, на мою думку, підходить для використання аналізу у некомерційних, домашніх проектах. Адже вона має невеликий, але необхідний функціонал і зручна у використанні.

Також, вважаю необхідним окремо зупинитися на пояснені теорії щодо принципу роботи такого типу програм. Під терміном «аналіз на безпеку» розуміється виявлення великої кількості запитів до одного файлу (наприклад, login.php) з однієї ip адреси у короткий проміжок часу для того, щоб дізнатися і заблокувати порушника. Фоновий режим необхідний задля постійного стеження за роботою серверу і виявленню порушника в режимі реального часу.

3 ТЕХНІЧНЕ ЗАВДАННЯ

3.1 Вступ

Основна мета розробки – створити програмне забезпечення для синтаксичного аналізу файлів веб-сервера Apache.

Дане програмне забезпечення використовується в сферах веб-розробки та Seo маркетингу.

3.2 Призначення програми та вимоги до неї

Програма призначена для аналізу записів у acess.log із можливістю задати необхідні фільтри, або прочитати увесь файл, а далі здійснити перевірку на взлами.

Програма також повинна мати змогу працювати у фоновому режимі слідкування за файлом і сповістити користувача про взлам за допомогою Windows повідомлення, або листа на пошту.

Вимоги до функціональних характеристик:

* платформа .NET Framework;
* операційна система Microsoft Windows 10.

4 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМИ

4.1 Структура програми

Програма розроблена згідно з парадигмою ООП та має модульну структуру.

На цьому етапі вважаю важливим пояснити основну структуру програми та призначення кожного із класів:

* BackgroundWatcher.cs – реалізовує функціонал програми у фоновому режимі у іншому потоці.
* DangerousHTTPRequests.cs – зберігає небезпечні запити та реалізує механізми іх пошуку.
* FilteredRecords.cs – зберігає відфільтровані запити та реалізує функціі роботи з ними.
* LogFile.cs – основний клас. Зберігає інформацію про відкритий зараз лог-файл. Саме зі створення об’екта цього класу починається основна логіка програми.
* OneRecord.cs – представляє один запит. Має статичні і нестатичні функціі для роботи із ним.
* Form1.cs – головна форма програми.
* Form2.cs – форма вводу mail адреси.

4.2 Опис алгоритму роботи програми

Програма має доволі простий алгоритм роботи. Після запуску на екрані з’являється вікно у якому користувач повинен вибрати лог-файл. Далі користувач має змогу обрати необхідні йому фільтри, або не обирати жодного і натиснути на кнопку пошуку. У текстовому блоці будуть виведені усі результати пошуку у файлі, який вибрав користувач. Далі стає доступна кнопка «Аналіз на безпеку», після натискання якої з’являеться інформація про спроби несанкціонованного доступу у відфільтрованних запитах. Користувач також має змогу переглянути деяку інформацію стосовно самого файлу. Також є змога перевести програму у фоновий потік, який спершу прочитає і запам’ятає останні 20 записів и буде аналізувати збереження нових записів, видаляючі перший. У разі спроби несанкціонованного доступу користувачеві буде виведено повідомлення з попередженням і відправлено письмо на пошту, яку користувач може вказати.

Пошаговий алгоритм роботи програми:

Шаг 1: після запуску програми з’являється основне вікно.

Шаг 2: користувач повинен обрати лог-файл:

* Якщо файл обраний, то користувач переходить до пункту 3;
* Якщо ні, користувач не має змоги щось робити, окрім пункту 2;

Шаг 3: Обрання фільтрів, або їх відсутність;

Шаг 4: Натискання кнопки пошуку:

* Якщо фільтри з пункту 3 обрані, пошук буде виконуватися згідно с обраними фільтрами.
* Якщо фільтри з пункту 3 не обрані, у користувача буде запитано, чи дійсно він бажає переглянути увесь файл? Якщо так – весь файл буде прочитаний. Якщо ні – користувач повертається до пункту 3;

Шаг 5: Натискання кнопки «Аналіз на безпеку»:

* Якщо на пункті 3 були обрані фільтри, користувачеві у іншому текстовому блоці будуть виведені результати, або повідомлення, що загрози не виявлено.
* Якщо на пункті 3 не було обрано жодного з фільтрів, у користувача буде запитано, чи дійсно він бажає просканувати на загрозу увесь файл. Якщо користувач відповість «так», то результати будуть виведені у інший текстовий блок, або повідомлення «загрози не виявлено» у разі відсутності загрози. Якщо ні – користувач повертається до кінця шагу 4.

Після пункту 2 користувачеві стає доступна кнопка «перейти у фоновий режим». Розглянемо цю лінію подій:

Шаг 6: Натискання кнопки «перейти у фоновий режим»;

Шаг 7: Користувач може ввести свою пошту, або залишити поле пустим.

Шаг 8: Програма переводиться у фоновий режим:

* Якщо у пункті 7 користувач ввів свою пошту, у разі виявлення загрози користувач отримає Windows повідомлення, а також повідомлення на пошту;
* Якщо у пункті 7 користувач не ввів свою пошту, у разі виявлення загрози користувач отримає тільки Windows повідомлення.

4.3 Структури даних

При розробці програми досить важливою частиною є вибір структур даних, адже це може доволі сильно впливати на ефективність роботи програми. Я зупинив свій вибір на трьох структурах даних: List<T>, Dictionary<T,T>, а також на звичайних масивах.

Причина використання масивів – проста реалізація, швидка та зручна індексація, яка доволі часто використовується в коді, окрім цього, така структура даних займає малу кількість місця у пам’яті; List<T> використовується як основна структура даних через свою простоту та універсальність; Dictionary<T,T> використана для реалізації пар «Ip» - «Вірогідність». Розглянемо призначення основних полів цих структур даних у класах:

LogFile.cs

public List<string> wrongRecordsList – структура, що зберігає список неправильних запитів із лог-файлу;

public Dictionary<string, float> fileInfo – структура, що зберігає інформацію про файл;

FilteredRecords.cs

public List<OneRecord> FilteredRecordsList – зберігає відфільтрований список запитів із лог-файлу;

DangerousHTTPRequests.cs

private Dictionary<string, float> dangerousip – зберігає небезпечні IP в якості ключа і вірогідність їх небезпечності у якості значення;

4.4 Перелік методів

BackgroundWatcher.cs

public void LogWatcherON() – запускає фоновий режим.

public void LogWatcherOFF() – зупиняє фоновий режим.

private void WatcherChanged(object sender, FileSystemEventArgs e) – обробка події змін у файлі.

public void WriteDangerousRequest\_InBackgroundMode\_ToWindow(RichTextBox AnalysisTextBox) – записує небезпечні дані у текстовий блок.

private void fillFilteredRecords(int n = 20) ­– заповнює список запитів останніми 20 строками при ініціалізації обьекта.

private bool isLastRecordAdded() – перевіряє, була додана строка до файлу, чи ні.

static string ReadLastLine(string path) – зчитує останню строку.

public static bool IsFileClosed(string filepath, bool wait) – Перевіряє файл. Якщо він зайнятий, чекає його звільнення.

public static void makeNotify(NotifyIcon notifyIcon1, string text, int time = 1000) – створює Windows повідомлення.

public void SendEmail(string ip) – відправляє листа на пошту.

DangerousHTTPRequests.cs

public void AddIp(string ip, float probabilityOfDangerous) – додає новий елемент до списку небезпечних IP.

public void writeDangerousRequestsToWindow(System.Windows.Forms.RichTextBox AnalysisTextBox) – записує список небезпечних IP у текстовий блок.

public static bool isRecordLoginFailure(OneRecord \_record) – перевіряє, чи була спроба залогінитись помилковою.

public static bool isRecordDangerous(int numberOfRequests, float probabilityOfDangerous) – перевіряє, чи є IP небезпечним.

FilteredRecords.cs

public void AddRecord(OneRecord \_record) – додає новий запит до списку фільтрованих запитів.

public void deleteFirstRecord() – видаляє перший запит зі списку запитів.

public void WriteFilterRecordsToWindow(System.Windows.Forms.RichTextBox richTextBox1) – записує відфільтровані запити до текстового блоку.

public DangerousHTTPRequests AttackDetector() – виявляє із фільтрованного списку небезпечні записи.  
 LogFile.cs  
public void writeFileToWindow(System.Windows.Forms.RichTextBox richTextBox1) – виводить увесь файл в текстовий блок.

public void writeFileInfoToWindow(System.Windows.Forms.RichTextBox InfoTextBox) – виводить інформацію про файл у текстовий блок.

public FilteredRecords Filter(DateTime minDate = default(DateTime), DateTime maxDate = default(DateTime), string fileName = default(string), int resultType = 0, string ip = default(string), int lastRecords = -1) – фільтрує записи згідно з обраними користувачем фільтрами.

public DangerousHTTPRequests SecureFilterAndScanAllFile() – сканувати весь файл.

public static BackgroundWatcher MakeBackgroundWatcher(LogFile logFile, NotifyIcon notifyIcon1, MailAddress mail = null) – створює об’єкт для входу у фоновий режим.

OneRecord.cs

public static bool IsRecordCanBeCreated(string \_logString) – перевіряє валідність запису у файлі.

public DateTime ConvertDateToDateFormat(string \_string) – конвертує текстову дату до формату DateTime.

public bool isRecordMinDateValid(DateTime min) – фільтр запису по даті «від».

public bool isRecordMaxDateValid(DateTime max) – фільтр запису по даті «до».

public bool isRecordFileNameValid(string \_name) – фільтр запису по імені файла.

public bool isRecordsResultTypeValid(int \_type) – фільтр запису по типу результата.

public bool isRecordIPValid(string \_ip) – фільтр запису по IP.

5 ОПИС ПРОГРАМИ

Основною метою було реалізувати влучні алгоритми роботи програми. Інтерфейс було виконано за допомогою Win Forms.

Приклад 5.1 ілюструє конструктор класу, який є основним і саме з нього починається робота програми, класу LogFile. Пошук відбувається тільки по файлам з розширенням «.log». Клас зберігає шлях і ім’я файлу разом, а також кожен із цих параметрів окремо. Також створюється і надалі заповнюється словник, який буде зберігати системну інформацію про файл, яку користувач може переглянути в окремому текстовому блоці.

public LogFile()

{

OpenFileDialog openFileDialog1 = new OpenFileDialog() { Filter = "Текстовые файлы(\*.log)|\*.log" };

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

fileName = openFileDialog1.FileName;

onlyFilePath = openFileDialog1.FileName;

onlyFileName = openFileDialog1.SafeFileName;

onlyFilePath = onlyFilePath.Replace(onlyFileName, "");

}

fileInfo = new Dictionary<string, float>();

long size = 0;

if (fileName != null)

{

FileInfo file = new System.IO.FileInfo(fileName);

size = file.Length;

}

else

return;

fileInfo.Add("Размер файла", (int)size);

isBackgroundWatcherON = false;

}

Приклад 5.1 – Конструктор класу LogFile

Після створення об’єкту класу LogFile нам потрібно виконувати якісь операції із ним, наприклад, вивести на екран весь файл. Приклад 5.2 показує фрагмент виведення на екран користувача даних, які містяться у файлі, який був обраний користувачем. Також було враховано і вірішено можливість потрапляння до лог-файлу якоїсь некоректної інформації. У такому разі цей запис не виводиться на екран, а відправляється до списку некоректних записів і надалі буде доступним.

while ((line = sr.ReadLine()) != null)

{

lineNumbers++;

if (OneRecord.IsRecordCanBeCreated(line))

{

outputLineNumbers++;

OneRecord record = new OneRecord(line);

richTextBox1.Text += String.Format("[{0}] {1}\n", lineNumbers, record.logString);

}

else

{

wrongRecords++;

wrongRecordsList.Add(line);

}

}

Приклад 5.2 – фрагмент методу виведення файлу на екран

Якщо користувач має бажання використати якісь фільтри, а не виводити увесь файл, то визивається метод, який активує прапори фільтрів, якщо користувач їх обрав. Фрагмент коду цього методу описується у прикладі 5.3.

FilteredRecords filteredList = new FilteredRecords();

bool isMinDateFilterActive = false;

if (minDate != default(DateTime))

isMinDateFilterActive = true;

bool isMaxDateFilterActive = false;

if (maxDate != default(DateTime))

isMaxDateFilterActive = true;

bool isNameFilterActive = false;

if (fileName != default(string))

isNameFilterActive = true;

bool isResultFilterActive = false;

if (resultType != 0)

isResultFilterActive = true;

bool isIpFilterActive = false;

if (ip != default(string))

isIpFilterActive = true;

bool isLastRecordsFilterActive = false;

if (lastRecords > -1)

isLastRecordsFilterActive = true;

Приклад 5.3 – Встановлення прапорів фільтрів

Дані, за якими проходитиме фільтрація разом із прапорами передаються у приватний метод, який реалізує увесь функціонал фільтрації і повертає список відфільтрованих записів спочатку до відкритого методу фільтрації, а він в свою чергу – звідки був визваний. Фрагмент цього методу описаний у прикладі 5.4.

if (isMinDateFilterActive)

if (!record.isRecordMinDateValid(min))

allFiltersOK = false;

if (IsMaxDateFilterActive)

if (!record.isRecordMaxDateValid(max))

allFiltersOK = false;

if (isNameFilterActive)

if (!record.isRecordFileNameValid(\_name.Trim()))

allFiltersOK = false

if (isResultFilterActive)

if (!record.isRecordsResultTypeValid(\_resultType))

allFiltersOK = false;

if (isIpFilterActive)

if (!record.isRecordIPValid(\_ip.Trim()))

allFiltersOK = false;

Приклад 5.4 – Фрагмент приватного методу фільтрації

Метод, який перевіряє запис у файлі на можливість створення об’єкту класу OneRecord, який представляє один запис із файлу, який обрав користувач є статичним і представлений у прикладі 5.5. Реалізація зроблена за допомогою регулярного виразу .

public static bool IsRecordCanBeCreated(string \_logString)

{

string logEntryPattern = "^([\\d.]+) (\\S+) (\\S+) \\[([\\w:/]+\\s[+\\-]\\d{4})\\] \"(.+?)\" (\\d{3}) (\\d+)";

Dictionary<int, string> HTTPResultTypes = OneRecord.GetHTTPResultValidTypes();

Match theMatch = Regex.Match(\_logString, logEntryPattern);

if (theMatch.Success && HTTPResultTypes.ContainsKey(Int32.Parse(theMatch.Groups[6].Value)))

{

return true;

}

else

{

return false;

}

}

Приклад 5.5 – Метод перевірки запису на можливість створення об’єкту

Також клас OneRecord має метод конвертації дати із файлу до формату дати класу DateTime мови С#, що необхідно для фільтрації за датою. Реалізація цього методу наведена у прикладі 5.6.

public DateTime ConvertDateToDateFormat(string \_string)

{

\_string = String.Concat(\_string.Substring(0, \_string.IndexOf(':')), ' ', \_string.Substring(\_string.IndexOf(':') + 1));

DateTime date = DateTime.Parse(\_string);

return date;

}

Приклад 5.6 – Метод конвертації дати

Фільтровані записи утворюють об’єкт класу FilteredRecords, який має метод, представлений у прикладі 5.7, функція якого – виводити фільтровані дані на екран користувача.

if (FilteredRecordsList.Any())

{

int lineNumbers = 0;

foreach (OneRecord \_record in FilteredRecordsList)

{

lineNumbers++;

richTextBox1.Text += String.Format("[{0}] {1}\n", lineNumbers, \_record.logString);

}

}

else

{

richTextBox1.Text = "Filtered list is empty";

}

Приклад 5.7 – Метод виводу фільтрованих даних

Клас FilteredRecords має також метод, який реалізує пошук запитів, які можуть нести загрозливий характер. Для того щоб уникнути випадків, коли не зрозуміло, чи є череда запитів загрозливою, чи ні, мною був реалізований алгоритм, згідно з яким кожен запит, який отримував незадовільну відповідь від сервера, намагаючись, наприклад, залогінитися, потрапляє під певні підозри і IP, з якого цей запит був відправлений починає перевірятися далі. Якщо надалі зустрічаються аналогічні випадки в певну кількість часу цей IP отримує статус можливого злочинця, який є відсотковою величиною, залежної від кількості запитів. Невеликий фрагмент цього методу представлено у прикладі 5.8.

float probabilityOfDangerous = (float)100 \* numberOfRequests / 15;

if (probabilityOfDangerous > 100)

probabilityOfDangerous = 100;

if (DangerousHTTPRequests.isRecordDangerous(numberOfRequests, probabilityOfDangerous))

{

if (dangerousRequests.DangerousIp.ContainsKey(ip))

{

if (dangerousRequests.DangerousIp[ip] < probabilityOfDangerous)

{

dangerousRequests.DangerousIp[ip] = probabilityOfDangerous;

}

}

else

dangerousRequests.AddIp(ip, probabilityOfDangerous);

}

Приклад 5.8 – Фрагмент методу пошуку злочинних запитів

Для активного фонового режиму була реалізована функція відправлення повідомлення на пошту, яка використовує класи MailAddress та SmtpClient, яка описана в прикладі 5.9.

public void SendEmail(string ip)

{

MailAddress from = new MailAddress("logvnvlyzer@gmail.com", "log VnVlyzer");

MailAddress to = mailAddress;

MailMessage m = new MailMessage(from, to);

m.Subject = "На вашем ресурсе обнаружен вредитель!";

m.Body = String.Format("<h2>Мы поймали на вашем ресурсе розбийныка!<br>Его ip - <p style=\"color:red\">{0}</p>", ip);

m.IsBodyHtml = true;

SmtpClient smtp = new SmtpClient("smtp.gmail.com", 25);

smtp.Credentials = new NetworkCredential("logvnvlyzer@gmail.com", "a\_qwerty");

smtp.EnableSsl = true;

smtp.Send(m);

}

Приклад 5.9 – Функція відправлення повідомлення на пошту

6 ПОСІБНИК СИСТЕМНОГО ПРОГРАМІСТА

Програма не використовує сторонніх бібліотек та не має інсталятора. Для запуску програми доцільно використовувати виконуючий файл Coursework\_main.exe, який знаходиться у папці Debug. Окрім цього для використання програми необхідна платформа .Net Framework та операційна система Windows 10. Усі необхідні для запуску файли та бібліотеки знаходяться в папці з проектом.

Програма написана в інтегрованому середовищі Microsoft Visual Studio 2019. Для її компіляції потребується транслятор з цієї мови та ОС Windows 10. Технічна документація написана з використанням програмного забезпечення Microsoft Word 2016.

Програма була написана та откомпільована на комп’ютері з наступною конфігурацією Intel Core i5-7200U CPU @ 2.50GHz та операційною системою Microsoft Windows 10.

7 ПОСІБНИК ОПЕРАТОРА

7.1 Інструкція користувача

Програма призначена для аналізу log файлу сервера Apache. Тобто для фільтрації запитів за такими критеріями: дата, ім’я файлу, типу результату, IP-адресою джерела запросу, а також кількістю останніх записів. Програма підтримує фоновий режим, під час якого постійно перевіряються і аналізуються нові записи. У разі виявлення порушника, програма відображає Windows повідомлення, якщо в настройках Windows підтримуються повідомлення. У разі сто відсоткового потрапляння IP до статусу злочинних, програма може висилати на пошту листа, у якому є IP злочинця. Програма під час виконання не повинна викликати якісь повідомлення до оператора, але у випадку помилки може викликати виключення та повідомити користувача. На даному етапі під час роботи та тестування програми таких повідомлень не було, проте при виникненні даної ситуації слід звернутися до розробника для корегування помилок.

7.2 Технічні вимоги

Програму рекомендується виконувати на операційній системі Windows 10 з встановленою платформою .Net Framework.

Мінімальні системні вимоги:

* Процесор – Pentium 1 GHz;
* Графічний адаптер, що підтримує режим 1600х900;
* ОП – 512 Мб;
* Вільне місце на жорсткому диску – 100 Мб.

ВИСНОВКИ

У цій роботі була виконана поставлена задача – створення програми на тему синтаксичний аналіз файлів логів веб-серверу Apache, а також додані можливості, які виходять за рамки поставленої задачі, такі як фоновий режим і взаємодія з поштою користувача.

За допомогою цієї програми можна базово аналізувати лог-файли серверу Apache. Крім того, програма не займає багато місця та не вимоглива до встановленого програмного забезпечення. Проте вона має певну проблему із інтерфейсом користувача.

Можливе подальше розвинення проекту в напрямках:

* додавання функції запам’ятовування колись помічених зловмисних IP;
* удосконалення інтерфейсу користувача;
* рефакторінг коду для оптимізації роботи програми;
* перенесення усіх обчислень у хмари;
* вдосконалення алгоритму пошуку зловмистного IP;

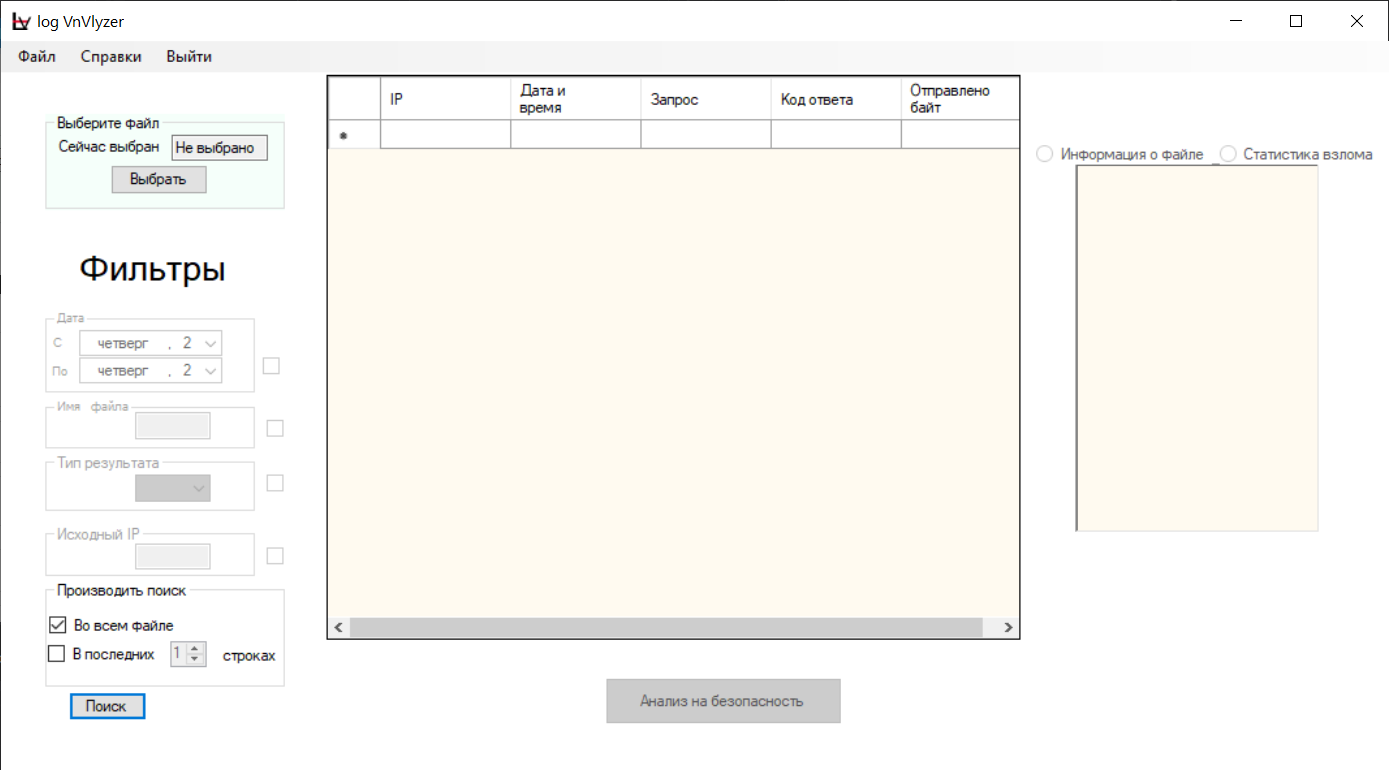
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скит Дж. C# для профессионалов. Тонкости программирования, 3-е издание/ Пер. с англ. — 3 е изд., исправ. —. СПб. : Издательство «Вильямс» ; 2014. — 608 стр. :
2. MSDN. Create a UI with XAML [Електронний ресурс] – Режим доступу: https://msdn.microsoft.com/en-us/library/windows/apps/mt228349.aspx
3. Windows Dev Center. Windows.UI.Xaml.Shapes [Електронний ресурс] – Режим доступу: <https://docs.microsoft.com/en-us/uwp/api/windows.ui.xaml.shapes>
4. MSDN. Controls, layouts, and text [Електронний ресурс] – Режим доступу: https://msdn.microsoft.com/en-us/library/windows/apps/mt228348.aspx
5. Полное руководство по языку программирования С# [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/tutorial/>
6. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Frame work 4.5 на языке C#. Мастер класс. / Пер. с англ. — 4 е изд., исправ. —. СПб. : Издательство «Питер» ; 2017. — 896 стр. :

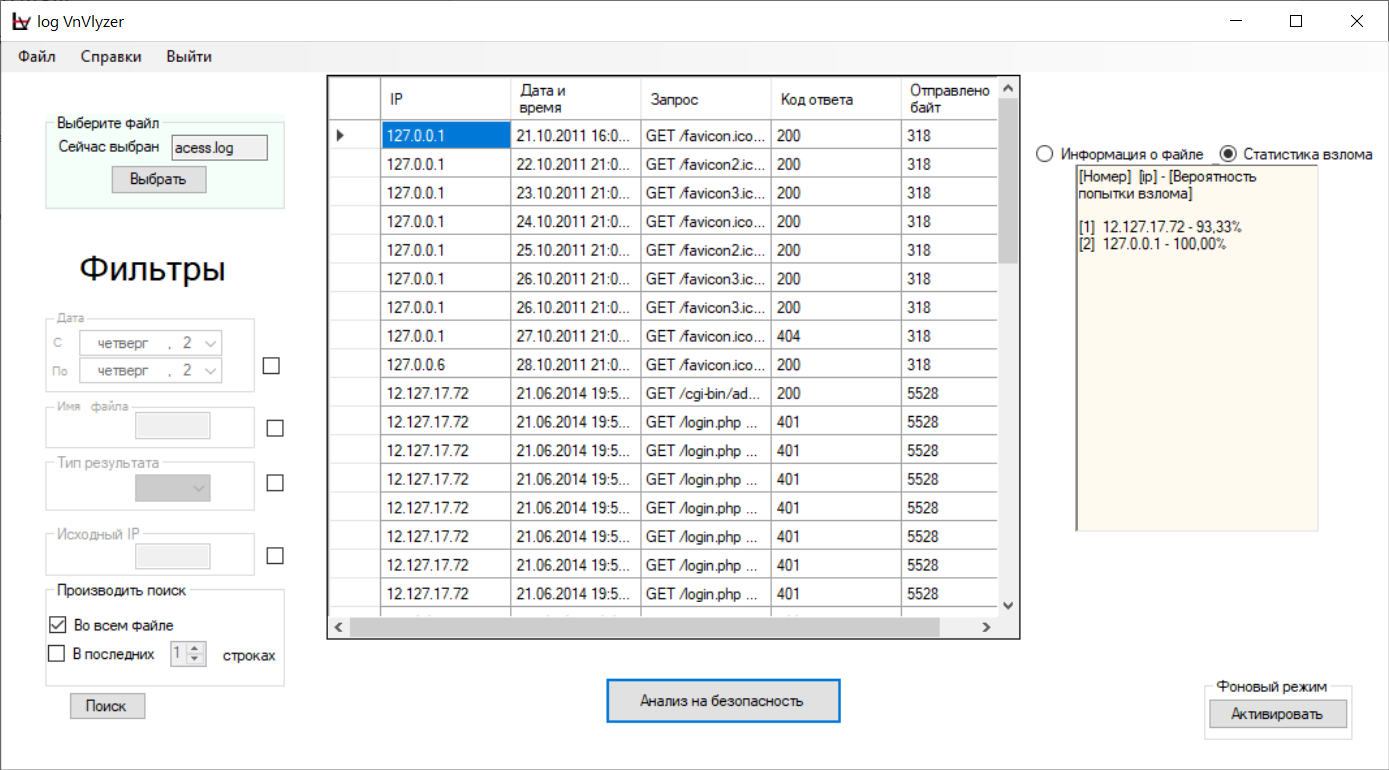
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

Лістинг програми знаходиться на диску.

ДОДАТОК Б ВИГЛЯД ПРОГРАМИ ПРИ ЗАПУСКУ



ДОДАТОК В ВИГЛЯД ПРОГРАМИ ПІД ЧАС РОБОТИ



ДОДАТОК Г ВВЕДЕННЯ ПОШТИ

