**Средний уровень**

**Подозрительная активность на сервере**

static void ServerAnalyzer()

{

/\* Задача выполнена, но классы CorrectService и FishingService не используются и созданы формально. Нужно понять как именно будут вводиться данные. Сейчас предполагается, что данные вводятся в консоль до тех пор, пока вводится пустая строка \*/

List<string> input = new List<string>();

string str;

do

{

str = Console.ReadLine();

input.Add(str);

} while (str != "");

ServerLogAnalyzer analyzer = new ServerLogAnalyzer(input);

List<string> l = analyzer.ProcessingServerLogs();

foreach (string line in l) Console.WriteLine(line);

}

interface IWriteServiceActivity

{

void PrintData();

}

class CorrectService : IWriteServiceActivity

{

public void PrintData()

{

Console.WriteLine("!");

}

}

class FishingService : IWriteServiceActivity

{

public void PrintData()

{

Console.WriteLine("!");

}

}

public class ServerLogAnalyzer

{

List<string> input;

public ServerLogAnalyzer(List<string> inputData)

{

input = inputData;

}

public List<string> ProcessingServerLogs()

{

var strings = new List<string>();

var lst = new List<LogRecord>();

foreach (string s in input)

{

string[] data = s.Trim(new char[] {' ', '<', '>'}).Split(" ");

if(data.Length == 3 &&

Regex.IsMatch(data[0], @"^service=""\d\*""") &&

Regex.IsMatch(data[1], @"^data=""\w\*""") &&

Regex.IsMatch(data[2], @"^action=""\w\*"""))

{

string serviceNumber = data[0].Substring("service=".Length).Trim('"');

string dataString = data[1].Substring("data=".Length).Trim('"');

string action = data[2].Substring("action=".Length).Trim('"');

LogRecord rec = new LogRecord(serviceNumber, dataString, action);

if (int.Parse(serviceNumber) >= 10000 && int.Parse(serviceNumber) <= 99999 && Regex.IsMatch(dataString, "^[A-Z]{9}$") && (action == "read" || action == "write"))

{

lst.Add(rec);

}

}

}

SortedDictionary<string, List<int>> readWriteCounter = new SortedDictionary<string, List<int>>(new StringComparer()); //ключ, список из целых

foreach(LogRecord r in lst)

{

if (!readWriteCounter.ContainsKey(r.ServiceNumber))

{

readWriteCounter.Add(r.ServiceNumber, new List<int>() { 0, 0 });

}

if(r.Action == "read") readWriteCounter[r.ServiceNumber][0] += 1;

if(r.Action == "write") readWriteCounter[r.ServiceNumber][1] += 1;

}

if (readWriteCounter.Count > 0)

{

foreach (KeyValuePair<string, List<int>> kv in readWriteCounter)

{

if (kv.Value[1] \* 100 / (kv.Value[1] + kv.Value[0]) >= 75) strings.Add($"Alert! {kv.Key} has suspicious activity");

else strings.Add($"{{\"service\":\"{kv.Key}\":\"read\":{kv.Value[0]}:\"write\":{kv.Value[1]}}}");

}

}

else strings.Add("none");

return strings;

}

}

class StringComparer : IComparer<string>

{

public int Compare(string a, string b)

{

return a.CompareTo(b);

}

}

class LogRecord

{

public string ServiceNumber {get; set;}

public string DataString { get; set; }

public string Action { get; set; }

public LogRecord(string serviceNumber, string dataString, string action)

{

ServiceNumber = serviceNumber;

DataString = dataString;

Action = action;

}

}

**Подбор актёров**

public class ProcessingValidateActors

{

List<string> inputData;

public ProcessingValidateActors(List<string> input)

{

inputData = input;

}

public bool ValidateActor(string actorData)

{

string[] data = actorData.Trim().Split("::");

if (data.Count() == 4 &&

data[0].Length > 0 && data[0].Length <= 40 && Regex.IsMatch(data[0], "^[А-Яа-я]+$") &&

int.TryParse(data[1], out int x) && x >= 10000 && x <= 99999 &&

Convert.ToInt32(data[2]) >= 172 && Convert.ToInt32(data[2]) <= 190 && Convert.ToInt32(data[3]) >= 1960 && Convert.ToInt32(data[3]) <= 1990)

{

//Console.WriteLine($"Кандидат подходит: ФИО: {data[0]}, ID: {data[1]}, рост: {data[2]}, год рождения: {data[3]}");

return true;

}

else return false;

}

public IList<string> PrintValidActors()

{

List<string> data = new List<string>();

List<actorData> lst = new List<actorData>();

foreach (string str in inputData)

{

if (ValidateActor(str))

{

string[] actorData = str.Trim().Split("::");

lst.Add(new actorData(actorData[0], Convert.ToInt32(actorData[1]), Convert.ToInt32(actorData[2]), Convert.ToInt32(actorData[3])));

}

}

lst.Sort();

foreach (actorData actor in lst)

{

data.Add($"{actor.ActorID}->({actor.LastName}:{actor.Height}:{actor.Year})");

}

return data;

}

}

class actorData : IComparable<actorData>

{

public string LastName { get; set; }

public int ActorID { get; set; }

public int Height { get; set; }

public int Year { get; set; }

public actorData(string lastName, int actorID, int height, int year)

{

LastName = lastName;

ActorID = actorID;

Height = height;

Year = year;

}

int IComparable<actorData>.CompareTo(actorData obj)

{

int result = -Height.CompareTo(obj.Height);

if (result == 0) result = Year.CompareTo(obj.Year);

if (result == 0) result = ActorID.CompareTo(obj.ActorID);

return result;

}

}