МЕЖРЕГИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО

Профиль:
Информатика и компьютерная безопасность

ВАРИАНТ 1

Задача 1. Система обнаружения вторжений

За несколько месяцев эксплуатации была сформирована статистика работы системы обнаружения вторжений (СОВ), приведенная на рисунке 1.

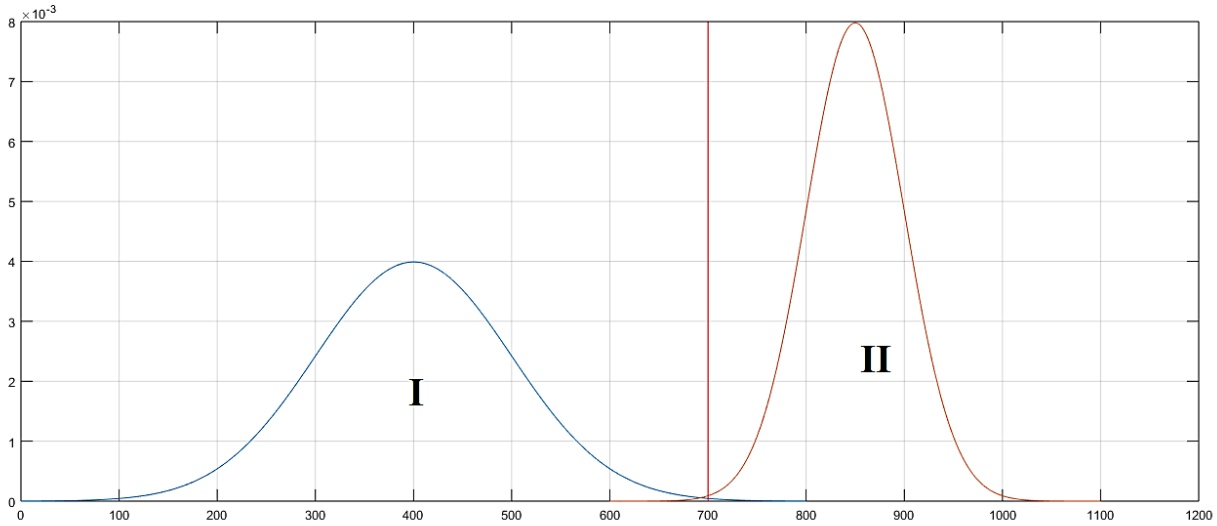

Рисунок 1 – Нормированное распределение трафика по частоте в зависимости от его интенсивности:
(I) – распределение нормального трафика,
(II) – распределение вредоносного трафика

Распределения трафиков представляют собой нормальные распределения со следующими параметрами:

– нормальный трафик (I): математическое ожидание $\mu = 400$, дисперсия $\sigma = 100$,

– вредоносный трафик (II): математическое ожидание $\mu = 850$, дисперсия $\sigma = 50$.

В настоящее время порог принятия решений для СОВ (показан красной линией на рисунке 1) задан так, что объем неправильно обнаруженного нормального трафика (O_I) и объем неправильно обнаруженного вредоносного трафика (O_{II}) равны **0,1%**.

График распределения объема выборки (площади под графиком) в зависимости от положения порога относительно математического ожидания показан на рисунке 2.

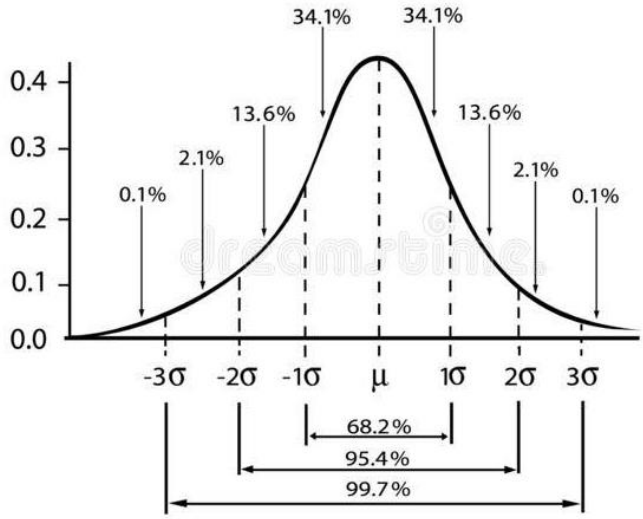

Рисунок 2 – Нормальное распределение трафика

В последнее неделю в сети появился аномальный трафик третьего вида (III), распределение которого показано на рисунке 3.

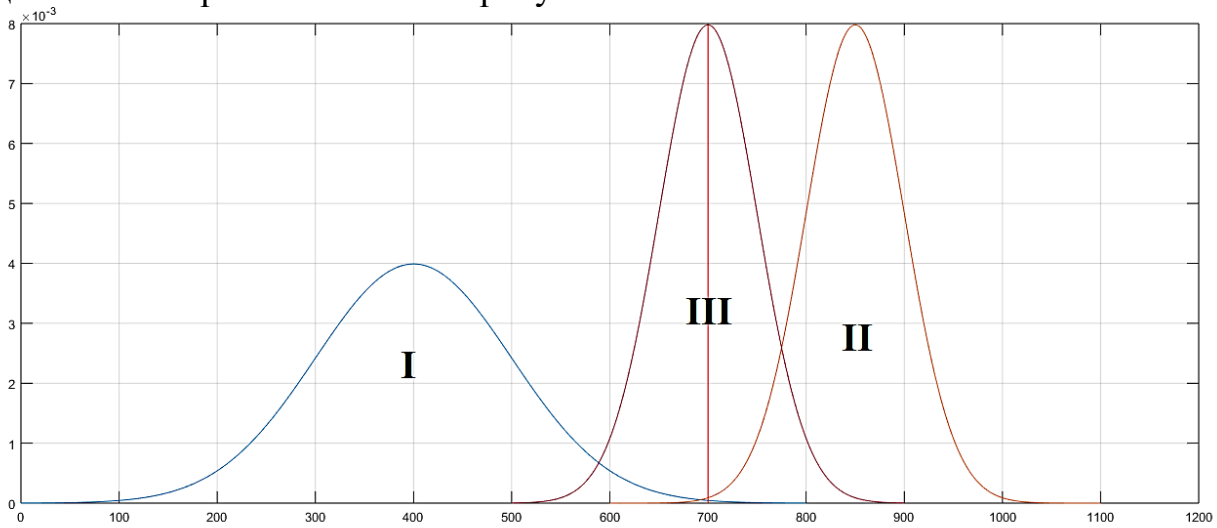

Рисунок 3 – Нормированное распределение аномального трафика по частоте в зависимости от его интенсивности (III)

Аномальный трафик представляет собой нормальное распределение с параметрами: математическое ожидание **μ = 700**, дисперсия **σ = 50**.

При текущих настройках порога СОВ объем ошибочного обнаружения трафика нового вида (O_{III}) = 50%.

В какую точку необходимо перенести порог принятия решения, чтобы суммарное значение ошибок всех видов трафика было минимальное:

$$O_{\text{сумм}} = O_I + O_{II} + O_{III} \rightarrow min?$$

Минимальный шаг изменения порога – 10.

В ответе укажите значение интенсивности, на которой должен быть установлен новый порог принятия решения, а также значения объемов ошибочного обнаружения для всех трех типов трафика.

Задача 2. Беспилотник

Ивану на Новый Год подарили беспилотник с управлением через специальное приложение на смартфоне, которое ведет журнал отправленных беспилотнику команд.

Для проверки корректности работы беспилотника в инструкции предусмотрен специальный тестовый маршрут, по которому необходимо пролететь с использованием приложения на смартфоне. Иван выполнил все команды управления, пролетел маршрут и вернул беспилотник в исходную точку (см. рисунок).

Маршрут из инструкции:

A→B→C→D→E→F→G→H→F→E→B→A

Координаты точек маршрута из инструкции (X;Y;Z):

A = (0; 0; 0) – исходная точка,
B = (0; 0; 12),
C = (12; 0; 12),
D = (12; 6; 12),
E = (0; 6; 12),
F = (0; 6; 24),
G = (0; 2; 24),
H = (-3; 2; 24).

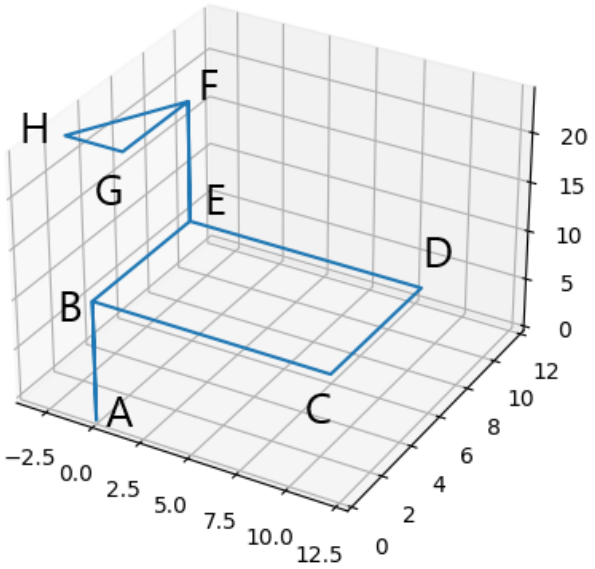

Рисунок – Тестовый маршрут беспилотника (из инструкции).
Единицы изменения шкал – метры

После этого Иван решил самостоятельно управлять беспилотником с использованием приложения. Через какое-то время беспилотник улетел так далеко, что пропал из виду.

На основании журнала отправленных команд помогите Ивану вернуть беспилотник с использованием минимального числа команд. В ответе укажите минимальную последовательность команд, которые необходимо отправить беспилотнику для его возвращения в исходную точку с координатами (0; 0; 0).

Считать, что беспилотник передвигается только по целочисленным координатам, то есть, если после выполнения команды беспилотник должен оказаться в точке с координатами (12,3; 7,8; 5), то он окажется в точке с координатами (12; 8; 5).

К задаче прилагается:

«drone_test_v1.log» – журнал с командами тестового маршрута из инструкции;
«drone_v1.log» – журнал с командами, которые отправлял Иван.

Задача 3. Сетевая стеганография

Агенты одной спецслужбы общаются по открытому каналу связи с помощью сетевых пакетов. Известно, что для передачи текстовых сообщений они используют значения некоторых полей заголовков пакетов.

Администратору удалось перехватить фрагмент передаваемых данных. Изучите его и извлеките передаваемое сообщение. Ответ обоснуйте.

К задаче прилагается:

«*packets_v1.cap*» – дамп сетевого трафика со скрытым текстовым сообщением.

Задача 4. Мессенджер

В компании для общения между сотрудниками используется мессенджер собственной разработки, который передает сообщения в зашифрованном виде. Шифрование производится с использованием метода «двоичного гаммирования» или путем выполнения операции «побитового исключающего ИЛИ» между байтами сообщения и ключа. Для каждого сотрудника ежедневно генерируется новый ключ по следующей формуле

$$K = (Ф_1 * И_1 + Ф_2 И_2 + \ldots + Ф_N И_N) \text{ div } C,$$

где Ф1,Ф2,…,ФN – код букв дополненной фамилии в соответствии с таблицей ASCII (регистр учитывается),

И1,И2,…,ИN – код букв дополненного имени в соответствии с таблицей ASCII (регистр учитывается),

N – максимум из длин фамилии и имени сотрудника,

C – сумма всех цифр текущей даты в формате ДД-ММ-ГГГГ,

div – операция целочисленного деления (целая часть от деления).

Если длина фамилии меньше длины имени, то фамилия дополняется путем дозаписи в конец циклического повторения букв фамилии, пока её длина не сравняется с длиной имени. Аналогично с именем, если его длина меньше длины фамилии.

Например, для сотрудника с ФИО Ivanov Petr 5 марта 2023 ключ будет вычисляться следующим образом:

1. Выравнивание длин имени и фамилии – имя дополняется двумя дополнительными символами:

```
Ivanov - 73 118 97 110 111 118
PetrPe - 80 101 116 114 80 101
```

2. Вычисление суммы цифр даты:

```
C = 0 + 5 + 0 + 3 + 2 + 0 + 2 +3 = 15
```

3. Вычисление ключа:

```
K = (73*80+118*101+97*116+110*114+111*80+118*101) div 15 =
```

$$= 4156_{10} = 103C_{16} = 0001\ 0000\ 0011\ 1100_2.$$

Далее байты текстового сообщения складывается по модулю 2 с байтами, полученными циклическим повторением последовательности байтов полученного ключа.

Руководитель отдела разработки дал поручения своим сотрудникам в течение дня 21 февраля 2023 года написать в чат название аэропорта, откуда им удобнее вылетать в командировку: *VKO* или *DME*. Проанализируйте полученный им зашифрованный поток сообщений из мессенджера и определите:

1) кто не выполнил поручение руководителя?
2) за какой аэропорт проголосовало большинство сотрудников?

К задаче прилагается:
«list_v1.txt» – список сотрудников;
«cypher_v1.txt» – зашифрованный текст мессенджера.

Задача 5. Blockchain

Существует система хранения документов, построенная на основе связного списка блоков (блокчейн). Блоки состоят из транзакций и информации о предыдущем блоке цепочки. Структура блока описана в формате JSON и содержит следующую информацию:
– номер блока (`_id`),
– дата создания блока в формате *"YYYY-MM-DD"* (`date`),
– список транзакций, входящих в состав блока (`storage`),
– хеш-значение предыдущего блока (`hash_prev`),
– контрольная строка (`nonce`).

Для добавления блока в связный список необходимо вычислить значение контрольной строки (NONCE) такое, чтобы хеш-функция этого блока возвращала строку, начинающуюся с символов '00'. Хеш-функция блока вычисляется на основе значения хеш-функции предыдущего блока, значения хеш-функции от списка идентификаторов транзакций блока и значения контрольной строки NONCE (см. листинги 1, 2).

Листинг 1 – Функция получения хеш-строки по массиву идентификаторов транзакций

Язык C++

```
// алфавит хеш-строки
const std::string ALPHABET =
"0123456789@<=>ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";

// получение символа из алфавита хеш-строки по его индексу (по модулю длины алфавита)
char getHashChar(unsigned long long index) {
    return ALPHABET[index % ALPHABET.length()];
}


// Получение хеш-строки по массиву из ID транзакций
// STORAGE - массив (вектор) чисел-идентификаторов транзакций
// SIZE    - длина получаемой хеш-строки
// RETURN
//   вычисленную хеш-строку длины SIZE
std::string StorageHash(std::vector<int> storage, int size) {
    // строка - результат (резервируем необходимое количество символов)
    std::string res(size, 0);

    // необходимые константы
    unsigned long long intHash;
    unsigned long long sum = 42;
    unsigned long long mul = 37;

    // основной цикл
    for (int i = 0; i < size; i += 2) {
```

```
            // вычисление суммы с предыдущим вычисленным значением hash
            intHash = sum + i + (i > 0 ? res[i - 1] : 0);
            // нормирование - получение символа хеш-строки
            res[i] = getHashChar(intHash);
            // вычисление произведения на основе суммы
            intHash = (mul + i) * intHash;
            // нормирование - получение символа хеш-строки
            res[i + 1] = getHashChar(intHash);
        }

        // замешивание идентификаторов транзакций
        for (int i = 0; i < storage.size(); i++) {
            // сложение
            intHash = res[i % size] + storage[i];
            // нормирование - получение символа хеш-строки
            res[i % size] = getHashChar(intHash);
            // умножение
            intHash = res[(i + 1) % size] * storage[i];
            // нормирование - получение символа хеш-строки
            res[(i + 1) % size] = getHashChar(intHash);
        }
        // возвращение результата
        return res;
}
```

Листинг 2 – Функция получения хеш-строки по блоку в цепочке блокчейн

Язык C++

```
// Получение хеш-строки на блок
// PREVHASH           - хеш-строка предыдущего блока блокчейна
// STORAGEHASH - хеш-строка по массиву ID транзакций текущего блока
// NONCE      - контрольная строка
// SIZE       - длина получаемой хеш-строки
// RETURN
//   вычисленную хеш-строку
std::string BlockHash(std::string prevHash, std::string storageHash, std::string nonce, int size) {
        // строка - результат (резервируем необходимое количество символов)
        std::string res(size, 0);

        // необходимые константы
        unsigned long long intHash;
        unsigned long long hash = 2139062143;
        unsigned long long sum = 42;
        unsigned long long mul = 37;

        // склеивание prevHash и storageHash
        for (int i = 0; i < size; i += 2) {
            res[i] = unsigned char(prevHash[i] + storageHash[i]);
            res[i + 1] = unsigned char(prevHash[i + 1] * storageHash[i + 1]);
        }

        // основной цикл - склеивание с NONCE
        for (int i = 0; i < size; i += 2) {
            // сложение с очередным символом NONCE
            intHash = sum + i + res[i] + nonce[i];
            // нормирование - получение символа хеш-строки
```

```
            res[i] = getHashChar(intHash);
            // сложение и умножение с другим символом NONCE
            intHash = (mul + i + 1) * hash + res[i + 1] + nonce[size/2 + i/2];
            // нормирование - получение символа хеш-строки
            res[i + 1] = getHashChar(intHash);
        }
        // возвращение результата
        return res;
}
```

Один сотрудник решил добавить только что созданный документ (транзакция с идентификатором **ID=33**) в блок, сформированный в предыдущем месяце. Укажите, в какой блок необходимо добавить транзакцию и какое новое значение контрольной строки NONCE в этом блоке нужно задать, чтобы значение хеш-функции блока не изменилось, и модификация блока не была обнаружена в блокчейне?

К задаче прилагается:

«blockchain_v1.json» – содержимое блокчейна.