МАТЕРИАЛЫ ЗАДАНИЙ МЕЖРЕГИОНАЛЬНОЙ ОЛИМПИАДЫ ШКОЛЬНИКОВ ИМЕНИ И.Я. ВЕРЧЕНКО ПО ПРОФИЛЮ «КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ» (2024/2025 УЧЕБНЫЙ ГОД)

ЗАКЛЮЧИТЕЛЬНЫЙ ЭТАП
8-10 КЛАССЫ

СОДЕРЖАНИЕ

Задача 1. Система защиты

Вариант 1

В компании используется система защиты от DDoS-атак, основанная на анализе активности пользователей. Система анализирует журнал активности пользователей, в котором вычисляет количество запросов от уникальных пользователей в единицу времени. Журналы содержат информацию о попытках доступа к серверу, включая IP-адреса пользователей, с которых осуществлялось обращение. Если количество одновременных подключений разных пользователей превышает пороговое значение, выдается предупреждение о возможной DDoS-атаке.

Перед запуском системы необходимо её настроить (обучить) на корректном трафике. Для этих целей был подготовлен журнал запросов от сотрудников компании за определенный период времени.

Определите, какое максимальное количество одновременных подключений разных пользователей в минуту необходимо установить в системе защиты, чтобы она корректно работала и не выдавала ложных предупреждений. Разработчики рекомендуют устанавливать пороговое значение на 10% больше максимально возможного числа подключений в единицу времени.

В ответе укажите само пороговое значение (количество подключений в единицу времени), а также обоснование выбранного значения.

К задаче прилагается:
файл «*log_v1.log*» – журнал посещений

Решение

Для решения можно подчитать количество уникальных IP-адресов для каждой минуты, однако, это очень долго. Поэтому, необходимо написать скрипт, например на Python:

```
import datetime

def analyze_logs(logs):
    """Анализирует логи и вычисляет максимальное количество одновременных
подключений в минуту."""
    connections_per_minute = {}

    for log_entry in logs:
        try:
            parts = log_entry.split()
            timestamp_str = " ".join(parts[:2])  # Собираем дату и время вместе
            IP_address = parts[-1]  # Берем последний элемент как IP-адрес

            timestamp = datetime.datetime.strptime(timestamp_str,
'%Y-%m-%d %H:%M:%S')
            minute = timestamp.replace(second=0, microsecond=0)

            if minute not in connections_per_minute:
                connections_per_minute[minute] = set()
            connections_per_minute[minute].add(IP_address)
        except ValueError as e:
            print(f"Error: {e}, Log entry: {log_entry}")
            continue
        except IndexError as e:
            print(f"IndexError: {e}, Log entry: {log_entry}")
            continue
```

```
    for minute, IPs in connections_per_minute.items():
        print(f"{minute}: {len(IPs)} одновременных подключений")

    max_connections = 0
    for IPs in connections_per_minute.values():
        max_connections = max(max_connections, len(IPs))
    return max_connections

if __name__ == "__main__":
    logs = [
        "2024-10-27 10:00:05 192.168.1.1",
        "2024-10-27 10:01:10 192.168.1.2",
        "2024-10-27 10:01:15 192.168.1.1",
        "2024-10-27 10:01:20 192.168.1.3",
        "2024-10-27 10:02:25 192.168.1.4",
        "2024-10-27 10:03:30 192.168.1.5",
        "2024-10-27 10:04:35 192.168.1.6",
        "2024-10-27 10:05:40 192.168.1.7",
        "2024-10-27 10:06:45 192.168.1.8",
        "2024-10-27 10:06:50 192.168.1.9",
        "2024-10-27 10:06:00 192.168.1.10",
        "2024-10-27 10:06:05 192.168.1.11",
        "2024-10-27 10:07:10 192.168.1.12",
        "2024-10-27 10:07:15 192.168.1.13",
        "2024-10-27 10:07:20 192.168.1.14",
        "2024-10-27 10:08:25 192.168.1.15",
        "2024-10-27 10:10:30 192.168.1.16",
        "2024-10-27 10:11:35 192.168.1.17",
        "2024-10-27 10:11:40 192.168.1.18",
        "2024-10-27 10:11:45 192.168.1.19",
        "2024-10-27 10:12:00 192.168.1.20",
        "2024-10-27 10:12:05 192.168.1.21",
        "2024-10-27 10:15:10 192.168.1.22",
        "2024-10-27 10:15:15 192.168.1.23",
        "2024-10-27 10:16:20 192.168.1.24",
        "2024-10-27 10:16:25 192.168.1.25",
        "2024-10-27 10:16:30 192.168.1.26",
        "2024-10-27 10:17:35 192.168.1.27",
        "2024-10-27 10:17:40 192.168.1.28",
        "2024-10-27 10:17:45 192.168.1.29",
        "2024-10-27 10:17:50 192.168.1.30",
        "2024-10-27 10:17:00 192.168.1.31",
        "2024-10-27 10:17:05 192.168.1.32",
        "2024-10-27 10:18:10 192.168.1.33",
        "2024-10-27 10:18:15 192.168.1.34",
        "2024-10-27 10:18:20 192.168.1.35",
        "2024-10-27 10:19:25 192.168.1.36",
        "2024-10-27 10:19:30 192.168.1.37",
        "2024-10-27 10:20:35 192.168.1.38",
        "2024-10-27 10:23:40 192.168.1.39",
        "2024-10-27 10:23:45 192.168.1.40",
        "2024-10-27 10:23:50 192.168.1.41",
        "2024-10-27 10:34:00 192.168.1.42",
        "2024-10-27 10:34:05 192.168.1.43",
        "2024-10-27 10:34:10 192.168.1.44",
        "2024-10-27 10:35:15 192.168.1.45",
        "2024-10-27 10:35:20 192.168.1.46",
        "2024-10-27 10:35:25 192.168.1.47",
        "2024-10-27 10:35:30 192.168.1.48",
        "2024-10-27 10:36:35 192.168.1.49",
        "2024-10-27 10:36:40 192.168.1.50",
        "2024-10-27 10:37:45 192.168.1.51",
```

```
        "2024-10-27 10:37:50 192.168.1.52",
        "2024-10-27 10:38:00 192.168.1.53",
        "2024-10-27 10:38:05 192.168.1.54",
        "2024-10-27 10:38:10 192.168.1.55",
        "2024-10-27 10:39:15 192.168.1.56",
        "2024-10-27 10:40:20 192.168.1.57",
        "2024-10-27 10:40:25 192.168.1.58",
        "2024-10-27 10:40:30 192.168.1.59",
        "2024-10-27 10:40:35 192.168.1.60",
        "2024-10-27 10:40:40 192.168.1.61",
        "2024-10-27 10:40:45 192.168.1.62",
        "2024-10-27 10:40:50 192.168.1.63",
        "2024-10-27 10:41:00 192.168.1.64",
        "2024-10-27 10:41:05 192.168.1.65",
        "2024-10-27 10:42:10 192.168.1.66",
        "2024-10-27 10:42:15 192.168.1.67",
        "2024-10-27 10:42:20 192.168.1.68",
        "2024-10-27 10:42:25 192.168.1.69",
        "2024-10-27 10:44:30 192.168.1.70",
        "2024-10-27 10:45:35 192.168.1.71",
        "2024-10-27 10:45:40 192.168.1.72",
        "2024-10-27 10:45:45 192.168.1.73",
        "2024-10-27 10:46:50 192.168.1.74",
        "2024-10-27 10:47:00 192.168.1.75",
        "2024-10-27 10:47:05 192.168.1.76",
        "2024-10-27 10:47:10 192.168.1.77",
    ]

    max_connections = analyze_logs(logs)
    print(f"\nМаксимальное количество одновременных подключений в минуту:
{max_connections}")
```

Вывод программы будет:

```
2024-10-27 10:00:00: 1 одновременных подключений
2024-10-27 10:01:00: 3 одновременных подключений
2024-10-27 10:02:00: 1 одновременных подключений
2024-10-27 10:03:00: 1 одновременных подключений
2024-10-27 10:04:00: 1 одновременных подключений
2024-10-27 10:05:00: 1 одновременных подключений
2024-10-27 10:06:00: 4 одновременных подключений
2024-10-27 10:07:00: 3 одновременных подключений
2024-10-27 10:08:00: 1 одновременных подключений
2024-10-27 10:10:00: 1 одновременных подключений
2024-10-27 10:11:00: 3 одновременных подключений
2024-10-27 10:12:00: 2 одновременных подключений
2024-10-27 10:15:00: 2 одновременных подключений
2024-10-27 10:16:00: 3 одновременных подключений
2024-10-27 10:17:00: 6 одновременных подключений
2024-10-27 10:18:00: 3 одновременных подключений
2024-10-27 10:19:00: 2 одновременных подключений
2024-10-27 10:20:00: 1 одновременных подключений
2024-10-27 10:23:00: 3 одновременных подключений
2024-10-27 10:34:00: 3 одновременных подключений
2024-10-27 10:35:00: 4 одновременных подключений
2024-10-27 10:36:00: 2 одновременных подключений
2024-10-27 10:37:00: 2 одновременных подключений
2024-10-27 10:38:00: 3 одновременных подключений
2024-10-27 10:39:00: 1 одновременных подключений
2024-10-27 10:40:00: 7 одновременных подключений
2024-10-27 10:41:00: 2 одновременных подключений
2024-10-27 10:42:00: 4 одновременных подключений
```

```
2024-10-27 10:44:00: 1 одновременных подключений
2024-10-27 10:45:00: 3 одновременных подключений
2024-10-27 10:46:00: 1 одновременных подключений
2024-10-27 10:47:00: 3 одновременных подключений
```

Максимальное количество одновременных подключений в минуту: **7**

Далее необходимо взять на 10% больше: (7 * 0,1) + 7 = 7.7, округляем это значение в большую сторону, так как количество IP-адресов у нас обязательно должно быть целым, отсюда ответ **8**.

Ответ: 8

Вариант 2

В компании используется система защиты от DDoS-атак, основанная на анализе активности пользователей. Система анализирует журнал активности пользователей, в котором вычисляет количество запросов от уникальных пользователей в единицу времени. Журналы содержат информацию о попытках доступа к серверу, включая IP-адреса пользователей, с которых осуществлялось обращение. Если количество одновременных подключений разных пользователей превышает пороговое значение, выдается предупреждение о возможной DDoS-атаке.

Перед запуском системы необходимо её настроить (обучить) на корректном трафике. Для этих целей был подготовлен журнал запросов от сотрудников компании за определенный период времени.

Определите, какое максимальное количество одновременных подключений разных пользователей в минуту необходимо установить в системе защиты, чтобы она корректно работала и не выдавала ложных предупреждений. Разработчики рекомендуют устанавливать пороговое значение на 10% больше максимально возможного числа подключений в единицу времени.

В ответе укажите само пороговое значение (количество подключений в единицу времени), а также обоснование выбранного значения.

К задаче прилагается:
файл «*log_v2.log*» – журнал посещений

Решение

Для решения можно подчитать количество уникальных IP-адресов для каждой минуты, однако, это очень долго. Поэтому, необходимо написать скрипт, например, на Python:

```
import datetime

def analyze_logs(logs):
    """Анализирует логи и вычисляет максимальное количество одновременных
подключений в минуту."""
    connections_per_minute = {}

    for log_entry in logs:
        try:
            parts = log_entry.split()
            timestamp_str = " ".join(parts[:2])  # Собираем дату и время вместе
            IP_address = parts[-1]  # Берем последний элемент как IP-адрес

            timestamp = datetime.datetime.strptime(timestamp_str,
'%Y-%m-%d %H:%M:%S')
```

```
            minute = timestamp.replace(second=0, microsecond=0)

            if minute not in connections_per_minute:
                connections_per_minute[minute] = set()
            connections_per_minute[minute].add(IP_address)
        except ValueError as e:
            print(f"Error: {e}, Log entry: {log_entry}")
            continue
        except IndexError as e:
            print(f"IndexError: {e}, Log entry: {log_entry}")
            continue

    for minute, IPs in connections_per_minute.items():
        print(f"{minute}: {len(IPs)} одновременных подключений")

    max_connections = 0
    for IPs in connections_per_minute.values():
        max_connections = max(max_connections, len(IPs))
    return max_connections

if __name__ == "__main__":
    logs = [
 "2024-10-27 11:15:00 192.168.1.100",
"2024-10-27 11:15:05 192.168.1.101",
"2024-10-27 11:16:10 192.168.1.102",
"2024-10-27 11:16:15 192.168.1.103",
"2024-10-27 11:17:20 192.168.1.104",
"2024-10-27 11:17:25 192.168.1.105",
"2024-10-27 11:18:30 192.168.1.106",
"2024-10-27 11:18:35 192.168.1.107",
"2024-10-27 11:18:40 192.168.1.108",
"2024-10-27 11:19:45 192.168.1.109",
"2024-10-27 11:19:50 192.168.1.110",
"2024-10-27 11:19:55 192.168.1.111",
"2024-10-27 11:26:00 192.168.1.112",
"2024-10-27 11:26:05 192.168.1.113",
"2024-10-27 11:26:10 192.168.1.114",
"2024-10-27 11:26:15 192.168.1.115",
"2024-10-27 11:27:20 192.168.1.116",
"2024-10-27 11:28:25 192.168.1.117",
"2024-10-27 11:28:30 192.168.1.118",
"2024-10-27 11:28:35 192.168.1.119",
"2024-10-27 11:29:40 192.168.1.120",
"2024-10-27 11:29:45 192.168.1.121",
"2024-10-27 11:29:50 192.168.1.122",
"2024-10-27 11:29:55 192.168.1.123",
"2024-10-27 11:37:00 192.168.1.124",
"2024-10-27 11:37:05 192.168.1.125",
"2024-10-27 11:37:10 192.168.1.126",
"2024-10-27 11:37:15 192.168.1.127",
"2024-10-27 11:37:20 192.168.1.128",
"2024-10-27 11:37:25 192.168.1.129",
"2024-10-27 11:37:30 192.168.1.130",
"2024-10-27 11:37:35 192.168.1.131",
"2024-10-27 11:38:40 192.168.1.132",
"2024-10-27 11:38:45 192.168.1.133",
"2024-10-27 11:38:50 192.168.1.134",
"2024-10-27 11:38:55 192.168.1.135",
"2024-10-27 11:39:00 192.168.1.136",
"2024-10-27 11:39:05 192.168.1.137",
"2024-10-27 11:39:10 192.168.1.138",
"2024-10-27 11:40:15 192.168.1.139",
```

```
"2024-10-27 11:40:20 192.168.1.140",
"2024-10-27 11:40:25 192.168.1.141",
"2024-10-27 11:40:30 192.168.1.142",
"2024-10-27 11:40:35 192.168.1.143",
"2024-10-27 11:41:40 192.168.1.144",
"2024-10-27 11:41:45 192.168.1.145",
"2024-10-27 11:41:50 192.168.1.146",
"2024-10-27 11:41:55 192.168.1.147",
"2024-10-27 11:42:00 192.168.1.148",
"2024-10-27 11:42:05 192.168.1.149",
"2024-10-27 11:43:10 192.168.1.150",
"2024-10-27 11:43:15 192.168.1.151",
"2024-10-27 11:48:30 192.168.1.106",
"2024-10-27 11:48:35 192.168.1.107",
"2024-10-27 11:48:40 192.168.1.108",
"2024-10-27 11:49:45 192.168.1.109",
"2024-10-27 11:49:50 192.168.1.110",
"2024-10-27 11:49:55 192.168.1.111",
"2024-10-27 11:56:00 192.168.1.112",
"2024-10-27 11:56:05 192.168.1.113",
"2024-10-27 11:56:10 192.168.1.114",
"2024-10-27 11:56:15 192.168.1.115",
"2024-10-27 11:57:20 192.168.1.116",
"2024-10-27 11:58:25 192.168.1.117",
"2024-10-27 11:58:30 192.168.1.118",
"2024-10-27 11:58:35 192.168.1.119",
"2024-10-27 11:59:40 192.168.1.120",
"2024-10-27 11:59:45 192.168.1.121",
"2024-10-27 11:59:50 192.168.1.122",
"2024-10-27 11:59:55 192.168.1.123",
    ]

    max_connections = analyze_logs(logs)
    print(f"\nМаксимальное количество одновременных подключений в минуту:
{max_connections}")
```

Вывод программы будет:

```
2024-10-27 11:15:00: 2 одновременных подключений
2024-10-27 11:16:00: 2 одновременных подключений
2024-10-27 11:17:00: 2 одновременных подключений
2024-10-27 11:18:00: 3 одновременных подключений
2024-10-27 11:19:00: 3 одновременных подключений
2024-10-27 11:26:00: 4 одновременных подключений
2024-10-27 11:27:00: 1 одновременных подключений
2024-10-27 11:28:00: 3 одновременных подключений
2024-10-27 11:29:00: 4 одновременных подключений
2024-10-27 11:37:00: 8 одновременных подключений
2024-10-27 11:38:00: 4 одновременных подключений
2024-10-27 11:39:00: 3 одновременных подключений
2024-10-27 11:40:00: 5 одновременных подключений
2024-10-27 11:41:00: 4 одновременных подключений
2024-10-27 11:42:00: 2 одновременных подключений
2024-10-27 11:43:00: 2 одновременных подключений
2024-10-27 11:48:00: 3 одновременных подключений
2024-10-27 11:49:00: 3 одновременных подключений
2024-10-27 11:56:00: 4 одновременных подключений
2024-10-27 11:57:00: 1 одновременных подключений
2024-10-27 11:58:00: 3 одновременных подключений
2024-10-27 11:59:00: 4 одновременных подключений
```

Максимальное количество одновременных подключений в минуту: **8**

Далее необходимо взять на 10% больше: (8 * 0,1) + 6 = 8.8, округляем это значение в большую сторону, так как количество IP-адресов у нас обязательно должно быть целым, отсюда ответ **9**.

Ответ: 9

Задача 2. Контрольная выборка атак

Вариант 1

На сервер зафиксировано 150 атак. Из них:
- 60 сетевых атак (20 произошли в первой половине дня, 40 во второй),
- 40 атак на приложения (15 критических и 25 некритических),
- 50 системных атак (32 произошли в первой половине дня, 18 во второй).

Для обучения системы безопасности необходимо выбрать 20 атак для анализа так, чтобы:
- среди них было ровно 10 сетевых атак, 3 в первой половине, 7 во второй половине дня;
- 5 атак на приложения обязательно должны быть критическими;
- 5 атак должны быть системными, из которых 2 атаки произошли во второй половине дня.

Сколько различных наборов таких обучающих выборок может быть получено? Ответ обоснуйте.

Решение

Нам необходимо выбрать 15 атак, а именно: 3 сетевые атаки из первой половины дня (из 20 атак), 7 сетевых атак из второй половины дня (из 40 атак), 5 критических атак на приложения (из 15 атак), 2 системные атаки из второй половины дня (из 28 атак) и 3 системных атаки из первой полвины дня (из 32 атак).

Число способов выбрать 3 сетевые атаки из первой половины дня:

$$C(20,3) = \frac{20 * 19 * 18}{3 * 2 * 1} = 1\,140$$

Число способов выбрать 7 атак из второй половины дня:

$$C(40,7) = \frac{40 * 39 * 38 * 37 * 36 * 35 * 34}{7 * 6 * 5 * 4 * 3 * 2 * 1} = 18\,643\,560$$

Общее количество способов для сетевых атак:

$$1\,140 * 18\,643\,560 = 21\,253\,658\,400$$

Число способов выбрать 5 критических атак из 15:

$$C(15,5) = \frac{15 * 14 * 13 * 12 * 11}{5 * 4 * 3 * 2 * 1} = 3\,003$$

Число способов выбрать 2 атаки из второй половины дня:

$$C(18,2) = \frac{18 * 17}{2 * 1} = 153$$

Число способов выбрать 3 атаки из первой половины дня:

$$C(32,3) = \frac{32 * 31 * 30}{3 * 2 * 1} = 4\,960$$

Общее количество способов для системных атак:

$$153 * 4960 = 758\,880$$

Общее количество способов:

$$21\,253\,658\,400 * 3\,003 * 758\,880 = 48\,435\,315\,788\,635\,776\,000$$

Ответ: 48 435 315 788 635 776 000

Вариант 2

На сервер зафиксировано 200 атак. Из них:
- 80 сетевых атак (30 произошли в первой половине дня, 50 во второй),
- 70 атак на приложения (20 критических и 50 некритических),
- 50 системных атак (30 произошли в первой половине дня, 20 во второй).

Для обучения системы безопасности необходимо выбрать 20 атак для анализа так, чтобы:

- среди них было ровно 12 сетевых атак, 4 в первой половине, 8 во второй половине дня;
- 6 атак на приложения обязательно должны быть критическими;
- 2 атаки должны быть системными, из которых 1 атака произошла во второй половине дня.

Сколько различных наборов таких обучающих выборок может быть получено? Ответ обоснуйте.

Решение

Нам необходимо выбрать 20 атак, а именно: 4 сетевые атаки из первой половины дня (из 30 атак), 8 сетевых атак из второй половины дня (из 70 атак), 6 критических атак на приложения (из 20 атак), 1 системную атаку из второй половины дня (из 30 атак) и 1 системную атаку из первой полвины дня (из 50 атак).

Число способов выбрать 4 сетевые атаки из первой половины дня:

$$C(30,4) = \frac{30 * 29 * 28 * 27}{4 * 3 * 2 * 1} = 27\,405$$

Число способов выбрать 8 атак из второй половины дня:

$$C(50,8) = \frac{50 * 49 * 48 * 47 * 46 * 45 * 44 * 43}{8 * 7 * 6 * 5 * 4 * 3 * 2 * 1} = 536\,878\,650$$

Общее количество способов для сетевых атак:

$$27\,405 * 27\,405 * 536\,878\,650 = 14\,713\,159\,403\,250$$

Число способов выбрать 6 критических атак из 20:

$$C(20,6) = \frac{20 * 19 * 18 * 16 * 15}{6 * 5 * 4 * 3 * 2 * 1} = 38\,760$$

Число способов выбрать 1 системную атаку из второй половины дня:

$$C(20,1) = 20$$

Число способов выбрать 1 системную атаку из первой половины дня:

$$C(30,1) = 30$$

Общее количество способов для системных атак:

$$20 * 30 = 600$$

Общее количество способов:

$$14\,713\,159\,403\,250 * 38\,760 * 600 = 342\,169\,235\,081\,982\,000\,000$$

Ответ: 342 169 235 081 982 000 000

Задача 3. Таблица символов

Вариант 1

В ходе анализа содержимого персонального компьютера была обнаружена таблица символов (таблица 1) и скриншот формулы (рисунок 1).

Экспертами установлено, что в таблице символов скрыт пароль от рабочего стола компьютера, каждая буква которого находится на определенной позиции, вычисляемой с помощью указанной формулы.

Определите пароль, если известно, что координаты первого символа: $(x_0, y_0) = (6, 9)$, длина пароля – *13* символов, и пароль является осмысленным словом. В ответе укажите сам пароль и значения параметров формулы.

Таблица 1 – Таблица символов

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О
1	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	О	Ю
2	Я	А	Б	В	Г	Н	Е	Ё	Ж	З	И	Й	К	Л	М	Н
3	О	П	Р	С	Т	У	Ф	Ч	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
4	Ю	Я	А	Я	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М
5	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ц	Ш	Щ	Ъ	Ы	Ь
6	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л
7	М	Н	О	П	Р	С	Т	У	Н	Х	Ц	Ч	Ш	Щ	Ъ	Ы
8	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	Е
9	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
10	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	И	И	Й
11	К	Л	М	Н	С	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
12	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	О	Ё	Ж	З	И
13	Й	К	Е	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
14	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	Т	Т	Д	Е	Ё	Ж	З
15	Л	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч

$$x_{i+1} = (x_i + a)\ mod\ 16, \qquad y_{i+1} = (y_i + b)\ mod\ 16$$

Рисунок 1 – Формулы вычисления координат символов,
mod – операция остатка от деления

Решение

Для решения необходимо подобрать такие a и b, для которых будет вычислен осмысленный пароль. Вручную это делать очень долго. Для этого можем написать программу, которая рассчитает слова длинной 13 символов для всех a и b по формулам вычисления координат.

```
def solve_password_with_table(x0, y0, letter_table, target_length=13):
  for a in range(16):
    for b in range(16):
      password = ""
```

```
      x, y = x0, y0
      coords = []  # Список для хранения координат для каждого символа
      # Выводим первую координату (x0, y0)
      try:
        char = letter_table[y][x]
        password += char
        coords.append((x, y))  # Добавляем первую координату
      except IndexError:
        password = ""
        coords = []
        print(f"a={a} b={b}, Пароль: Ошибка: выход за границы таблицы")
        print("-" * 20)
        continue  # Переходим к следующей комбинации a и b
      # Рассчитываем остальные координаты
      for _ in range(target_length - 1):  # На один символ меньше, так как первый
уже учтен
        x = (x + a) % 16
        y = (y + b) % 16
        try:
          char = letter_table[y][x]
          password += char
          coords.append((x, y))  # Добавляем координаты
        except IndexError:
          password = ""
          coords = []
          break
      print(f"a={a} b={b}, Пароль: {password}")

# Таблица
letter_table = [
  ['А','Б','В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н', 'О'],
  ['П','Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'О', 'Ю'],
  ['Я','А', 'Б', 'В', 'Г', 'Н', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н'],
  ['О','П', 'Р', 'С', 'Т', 'У', 'Ф', 'Ч', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь', 'Э'],
  ['Ю','Я', 'А', 'Я', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М'],
  ['Н','О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ц', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь'],
  ['Э','Ю', 'Я', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л'],
  ['М','Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Н', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы'],
  ['Ь','Э', 'Ю', 'Я', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'Е'],
  ['Л','М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ'],
  ['Ы','Ь', 'Э', 'Ю', 'Я', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'И', 'И', 'Й'],
  ['К','Л', 'М', 'Н', 'С', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ'],
  ['Ъ','Ы', 'Ь', 'Э', 'Ю', 'Я', 'А', 'Б', 'В', 'Г', 'Д', 'О', 'Ё', 'Ж', 'З', 'И'],
  ['Й','К', 'Е', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш'],
  ['Щ','Ъ', 'Ы', 'Ь', 'Э', 'Ю', 'Я', 'А', 'Б', 'Т', 'Т', 'Д', 'Е', 'Ё', 'Ж', 'З'],
  ['Л','Й', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч']
]
# Запускаем решение с заданными параметрами и таблицей
solve_password_with_table(6, 9, letter_table)
```

Вывод программы будет:

```
…
a=5 b=3, Пароль: СОЛНЦЕСТОЯНИЕ
…
```

Видим, что нам подходят $a = 5$ и $b = 3$, так как это единственная строчка с осмысленным текстом: «**СОЛНЦЕСТОЯНИЕ**».

Ответ: СОЛНЦЕСТОЯНИЕ
a = 5, b = 3

Вариант 2

В ходе анализа содержимого персонального компьютера была обнаружена таблица символов (таблица 1) и скриншот формулы (рисунок 1).

Экспертами установлено, что в таблице символов скрыт пароль от рабочего стола компьютера, каждая буква которого находится на определенной позиции, вычисляемой с помощью указанной формулы.

Определите пароль, если известно, что координаты первого символа: $(x_0, y_0) = (4, 6)$, длина пароля – *14* символов, и пароль является осмысленным словом. В ответе укажите сам пароль и значения параметров формулы.

Таблица 1 – Таблица символов

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Я	Ю	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц	О	Ф	У	Т	С	Р
1	П	О	Н	М	Л	К	Й	И	З	Ж	Ё	Е	Д	Г	В	Б
2	А	Я	Ю	Э	Ь	Ы	Ъ	Щ	И	Ч	Ц	Х	Ф	У	Т	С
3	Р	П	О	Н	М	Л	К	Й	И	З	Ж	Ё	Е	Д	Г	О
4	Б	А	Я	Ю	Э	Ь	О	Ъ	Щ	Ш	Ч	Ц	Х	Ф	У	Т
5	С	Р	П	О	Н	М	Л	К	Й	И	З	Ж	Ё	Т	Д	Г
6	В	Б	А	Я	П	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц	Х	Ф	У
7	Т	С	Р	П	О	Н	М	Л	К	Й	И	Е	Ж	Ё	Е	Д
8	Г	В	Я	А	Я	Ю	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц	Х	Ф
9	У	Т	С	Р	П	О	Н	М	Л	С	Й	И	З	Ж	Ё	Е
10	И	Г	В	Б	А	Я	Ю	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц	Х
11	Ф	У	Т	С	Р	П	О	Р	М	Л	К	Й	И	З	Ж	Ё
12	Е	Д	Г	В	Б	А	Я	Ю	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц
13	Х	Ф	У	Т	С	Н	П	О	Н	М	Л	К	Й	И	З	Ж
14	Ё	Е	Д	Г	В	Б	А	Я	Ю	Э	Ь	Ы	Т	Щ	Ш	Ч
15	Ц	Х	Ф	В	Т	С	Р	П	О	Н	М	Л	К	Й	И	З

$$x_{i+1} = (x_i + a)\ mod\ 16, \qquad y_{i+1} = (y_i + b)\ mod\ 16$$

Рисунок 1 – Формулы вычисления координат символов,
mod – операция остатка от деления

Решение

Для решения необходимо подобрать такие a и b, для которых будет подобран осмысленный пароль. Вручную это делать очень долго. Для этого можем написать программу, которая рассчитает слова длинной 14 символов для всех a и b.

```
def solve_password_with_table(x0, y0, letter_table, target_length=14):

  for a in range(16):
    for b in range(16):
      password = ""
      x, y = x0, y0
      coords = []  # Список для хранения координат для каждого символа
```

```
      # Выводим первую координату (x0, y0)
      try:
        char = letter_table[y][x]
        password += char
        coords.append((x, y))  # Добавляем первую координату
      except IndexError:
        password = ""
        coords = []
        print(f"a={a} b={b}, Пароль: Ошибка: выход за границы таблицы")
        print("-" * 20)
        continue  # Переходим к следующей комбинации a и b
      # Рассчитываем остальные координаты
      for _ in range(target_length - 1):  # На один символ меньше, так как первый
уже учтен
        x = (x + a) % 16
        y = (y + b) % 16
        try:
          char = letter_table[y][x]
          password += char
          coords.append((x, y))  # Добавляем координаты
        except IndexError:
          password = ""
          coords = []
          break
      print(f"a={a} b={b}, Пароль: {password}")

# Таблица (убедитесь, что она заполнена корректно!)
letter_table = [
  ['Я','Ю', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц', 'О', 'Ф', 'У', 'Т', 'С', 'Р'],
  ['П','О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'З', 'Ж', 'Ё', 'Е', 'Д', 'Г', 'В', 'Б'],
  ['А','Я', 'Ю', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'И', 'Ч', 'Ц', 'Х', 'Ф', 'У', 'Т', 'С'],
  ['Р','П', 'О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'З', 'Ж', 'Ё', 'Е', 'Д', 'Г', 'О'],
  ['Б','А', 'Я', 'Ю', 'Э', 'Ь', 'О', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц', 'Х', 'Ф', 'У', 'Т'],
  ['С','Р', 'П', 'О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'З', 'Ж', 'Ё', 'Т', 'Д', 'Г'],
  ['В','Б', 'А', 'Я', 'П', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц', 'Х', 'Ф', 'У'],
  ['Т','С', 'Р', 'П', 'О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'Е', 'Ж', 'Ё', 'Е', 'Д'],
  ['Г','В', 'Я', 'А', 'Я', 'Ю', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц', 'Х', 'Ф'],
  ['У','Т', 'С', 'Р', 'П', 'О', 'Н', 'М', 'Л', 'С', 'Й', 'И', 'З', 'Ж', 'Ё', 'Е'],
  ['И','Г', 'В', 'Б', 'А', 'Я', 'Ю', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц', 'Х'],
  ['Ф','У', 'Т', 'С', 'Р', 'П', 'О', 'Р', 'М', 'Л', 'К', 'Й', 'И', 'З', 'Ж', 'Ё'],
  ['Е','Д', 'Г', 'В', 'Б', 'А', 'Я', 'Ю', 'Э', 'Ь', 'Ы', 'Ъ', 'Щ', 'Ш', 'Ч', 'Ц'],
  ['Х','Ф', 'У', 'Т', 'С', 'Н', 'П', 'О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'З', 'Ж'],
  ['Ё','Е', 'Д', 'Г', 'В', 'Б', 'А', 'Я', 'Ю', 'Э', 'Ь', 'Ы', 'Т', 'Щ', 'Ш', 'Ч'],
  ['Ц','Х', 'Ф', 'В', 'Т', 'С', 'Р', 'П', 'О', 'Н', 'М', 'Л', 'К', 'Й', 'И', 'З']
]
# Запускаем решение с заданными параметрами и таблицей
solve_password_with_table(4, 6, letter_table)
```

Вывод программы будет:

```
…
a=3 b=5, Пароль: ПРОТИВОСТОЯНИЕ
…
```

Видим, что нам подходят $a = 3$ и $b = 5$, так как это единственная строчка с осмысленным текстом: «**ПРОТИВОСТОЯНИЕ**».

Ответ: ПРОТИВОСТОЯНИЕ,
a = 3, b = 5

Задача 4. Исходный код

Вариант 1

В связи с усилением требований к защите данных, руководство компании решило разработать алгоритм шифрования для безопасной передачи сообщений, которая должна шифровать конфиденциальные данные перед отправкой по сети. Алгоритм основан на принципе блочного шифрования и работает с текстами фиксированной длины в байтах.

Разработчики по ошибке опубликовали код функции шифрования в публичном репозитории (листинг 1). Зная исходный код и перехваченные зашифрованные данные определите исходное сообщение. В ответе укажите исходное сообщение и обоснуйте алгоритм дешифрования.

Зашифрованное сообщение:

```
122 72 127 124 81 93 96 122 81 74 65 117 91 85 113 94 118 170 170
```

Листинг 1. Функция шифрования на различных языках программирования

Python

```
def encrypt(text, key):
  encrypted = []
  for i, char in enumerate(text):
    encrypted.append(
        (ord(char)+i)^key )
  return encrypted
```

C

```
char* encrypt(char* text, char key)
{
  char *encrypted = new char[strlen(text)+1];
  for (int i  = 0; i < strlen(text); i++)
    encrypted[i] = (text[i] + i) ^ key;
  encrypted[strlen(text)] = '\0';
  return encrypted;
}
```

C++

```
std::string encrypt(std::string text, char key)
{
  std::string encrypted = "";
  for (int i = 0; i < text.length(); i++)
    encrypted[i] = (text[i] + i) ^ key;
  return encrypted;
}
```

Решение

Для решения задания, необходимо проанализировать функцию шифрования, так как алгоритм дешифрования – обратная функция шифрованию. Если переводить в математическую модель, то алгоритм шифрования будет иметь следующий вид:

$$C_i = (ord(P_i)+i) \oplus key,$$

где:

C_i – зашифрованный код символа на позиции i,
P_i – символ на позиции i в исходной строке,
i – индекс символа (позиция в строке),
$ord(P_i)$ – код символа P_i,
key – ключ шифрования,
$\oplus$ – операция XOR.

Тогда формула для расшифровки будет иметь следующий вид:

$$P_i = chr((C_i \oplus key) – i),$$

где:
P_i – расшифрованный символ на позиции i,
C_i – зашифрованный символ на позиции i,
chr() – преобразование числового значения обратно в символ,
ord(P_i) – код символа P_i,
key – ключ шифрования,
⊕ – операция XOR.

Поскольку ключ неизвестен, но он занимает всего 1 байт, его можно перебрать так, чтобы на выходе получалось осмысленное текстовое сообщение.

Видоизмененный код будет следующим:

```
def decrypt(encrypted, key):
    decrypted = ""
    for i, code in enumerate(encrypted):
        decrypted += chr((code ^ key) - i)
    return decrypted

encrypted_message_str = input("Введите зашифрованное сообщение (список чисел,
разделенных запятыми): ")

encrypted_message = [int(x) for x in encrypted_message_str.split(",")]

for key in range(256):
    decrypted_message = decrypt(encrypted_message, key)
    print(f"Ключ: {key},  Сообщение: {decrypted_message}")
```

При запуске подбора было определено, что ключ шифрования равен 42, а исходное сообщение: **PaSSwrDIsWaTerMeLon**

Ответ: PaSSwrDIsWaTerMeLon

Вариант 2

В связи с усилением требований к защите данных, руководство компании решило разработать алгоритм шифрования для безопасной передачи сообщений, которая должна шифровать конфиденциальные данные перед отправкой по сети. Алгоритм основан на принципе блочного шифрования и работает с текстами фиксированной длины в байтах.

Разработчики по ошибке опубликовали код функции шифрования в публичном репозитории (листинг 1). Зная исходный код и перехваченные зашифрованные данные определите исходное сообщение. В ответе укажите исходное сообщение и обоснуйте алгоритм дешифрования.

Зашифрованное сообщение:

```
99 81 102 101 72 68 89 99 72 106 64 106 66 125 77 76 111 69
```

Листинг 1. Функция шифрования на различных языках программирования

Python

```
def encrypt(text, key):
  encrypted = []
  for i, char in enumerate(text):
    encrypted.append(
        (ord(char)+i)^key )
  return encrypted
```

C

```
char* encrypt(char* text, char key)
{
  char *encrypted = new char[strlen(text)+1];
  for (int i  = 0; i < strlen(text); i++)
    encrypted[i] = (text[i] + i) ^ key;
  encrypted[strlen(text)] = '\0';
  return encrypted;
}
```

C++

```
std::string encrypt(std::string text, char key)
{
  std::string encrypted = "";
  for (int i = 0; i < text.length(); i++)
    encrypted[i] = (text[i] + i) ^ key;
  return encrypted;
}
```

Решение

Для решения задания, необходимо проанализировать функцию шифрования, так как алгоритм дешифрования – обратная функция шифрованию. Если переводить в математическую модель, то алгоритм шифрования будет иметь следующий вид:

$$C_i = (ord(P_i)+i) \oplus key,$$

где:

C_i – зашифрованный код символа на позиции i,
P_i – символ на позиции i в исходной строке,
i – индекс символа (позиция в строке),
$ord(P_i)$ – код символа P_i,
key – ключ шифрования,
$\oplus$ – операция XOR.

Тогда формула для расшифровки будет иметь следующий вид:

$$P_i = chr((C_i \oplus key) – i),$$

где:

P_i – расшифрованный символ на позиции i,
C_i – зашифрованный символ на позиции i,
chr() – преобразование числового значения обратно в символ,
$ord(P_i)$ – код символа P_i,
key – ключ шифрования,
$\oplus$ – операция XOR.

Поскольку ключ неизвестен, но он занимает всего 1 байт, его можно перебрать так, чтобы на выходе получалось осмысленное текстовое сообщение.

Видоизмененный код будет следующим:

```
def decrypt(encrypted, key):
    decrypted = ""
    for i, code in enumerate(encrypted):
        decrypted += chr((code ^ key) - i)
    return decrypted

encrypted_message_str = input("Введите зашифрованное сообщение (список чисел,
разделенных запятыми): ")

encrypted_message = [int(x) for x in encrypted_message_str.split(",")]
```

```
for key in range(256):
    decrypted_message = decrypt(encrypted_message, key)
    print(f"Ключ: {key},  Сообщение: {decrypted_message}")
```

При запуске подбора было определено, что ключ шифрования равен 51, а исходное сообщение: **PaSSwrdIsPiNeAppLe**

Ответ: PaSSwrdIsPiNeAppLe (pAssWRDiSpInEaPPlE)

Задача 5. Обфускация

Вариант 1

На компьютере хакера нашли фрагменты вредоносного кода. Автор обфусцировал весь исходный код. В ходе анализа был определен фрагмент функции аутентификации (листинг 1), которая возвращает `'True'` при успешном вводе пароля и запускает другие фрагменты кода.

Определите, какое секретное слово необходимо ввести, чтобы обнаруженный фрагмент аутентификации вернул `'True'`. Ответ обоснуйте.

Листинг 1. Фрагмент функции аутентификации

Python

```
exec('dAzD chjhQwek_sjhQweryuSWbE(sjhQweryuSWbE):\n    ryuSWbEurn
sjhQweryuSWbE[:17] ==  "Ha[3m[3mYNioskjear_" + c(487tHH0) + c(42)(48) +
c(487tHH0) +
c(487tHH3)\n[3mrinthE(chjhQwek_sjhQweryuSWbE(sjhQweryuSWbE))\n'.replace('87tHH',
'2)(5').replace('yuSWb', 'eth').replace('ioskj', 'ewY').replace('(42)',
'hr').replace('AzD', 'ef').replace('qQT', 'ut').replace('[3m',
'p').replace('jhQwe', 'ec').replace('thE', 't'), {'secret': input()})
```

Решение

Если вывести на экран строку

```
'dAzD chjhQwek_sjhQweryuSWbE(sjhQweryuSWbE):\n    ryuSWbEurn sjhQweryuSWbE[:17]
==  "Ha[3m[3mYNioskjear_" + c(487tHH0) + c(42)(48) + c(487tHH0) +
c(487tHH3)\n[3mrinthE(chjhQwek_sjhQweryuSWbE(sjhQweryuSWbE))\n'.replace('87tHH',
'2)(5').replace('yuSWb', 'eth').replace('ioskj', 'ewY').replace('(42)',
'hr').replace('AzD', 'ef').replace('qQT', 'ut').replace('[3m',
'p').replace('jhQwe', 'ec').replace('thE', 't')
```

То получится:

```
def check_secret(secret):
    return secret[:17] ==  "HappYNewYear_" + chr(50) + chr(48) + chr(50) + chr(53)
print(check_secret(secret))
```

Это и есть исходный код функции аутентификации. Отсюда извлекаем первую часть ключа: `"HappYNewYear_"`.

С помощью ASCII-таблицы определяем символы по кодам:

50 – ‘2’,

48 – ‘0’,

50 – ‘2’,

53 – ‘5’.

Таким образом, ключ равен `"HappYNewYear_2025"`

При запуске программы, если ввести указанный ключ, программа вернет ‘True’. При этом можно заметить, что можно вводить строку любой длины, главное, чтобы первые 17 символов были “`HappYNewYear_2025`”.

Ответ: HappYNewYear_2025

Вариант 2

На компьютере хакера нашли фрагменты вредоносного кода. Автор обфусцировал весь исходный код. В ходе анализа был определен фрагмент функции аутентификации (листинг 1), которая возвращает `'True'` при успешном вводе пароля и запускает другие фрагменты кода.

Определите, какое секретное слово необходимо ввести, чтобы обнаруженный фрагмент аутентификации вернул `'True'`. Ответ обоснуйте.

Листинг 1. Фрагмент функции аутентификации

Python

```
exec('djhQwe chAzDk_sAzDryuSWbE(sAzDryuSWbE):    \n    ryuSWbEurn
sAzDryuSWbE[:18] ==  "MerioskjhrqQTt[3(42)" + chr(50) + chr(48) + chr(50) +
chr(53)\nprinthE(chAzDk_sAzDryuSWbE(sAzDryuSWbE))\n'.replace('87tHH',
'2)(5').replace('yuSWb', 'eth').replace('ioskj', 'ryC').replace('(42)',
'mas').replace('AzD', 'ec').replace('qQT', 'is').replace('[3m',
'hEm').replace('jhQwe', 'ef').replace('thE', 't'), {'secret': input()})
```

Решение

Если вывести на экран строку

```
'djhQwe chAzDk_sAzDryuSWbE(sAzDryuSWbE):    \n    ryuSWbEurn sAzDryuSWbE[:18] ==
"MerioskjhrqQTt[3(42)" + chr(50) + chr(48) + chr(50) +
chr(53)\nprinthE(chAzDk_sAzDryuSWbE(sAzDryuSWbE))\n'.replace('87tHH',
'2)(5').replace('yuSWb', 'eth').replace('ioskj', 'ryC').replace('(42)',
'mas').replace('AzD', 'ec').replace('qQT', 'is').replace('[3m',
'hEm').replace('jhQwe', 'ef').replace('thE', 't')
```

То получится:

```
def check_secret(secret):
    return secret[:18] ==  "MerryChristmas" + chr(50) + chr(48) + chr(50) +
chr(53)
print(check_secret(secret))
```

Это и есть исходный код функции аутентификации. Отсюда извлекаем первую часть ключа: `"MerryChristmas"`.

С помощью ASCII-таблицы определяем символы по кодам:

50 – ‘2’,

48 – ‘0’,

50 – ‘2’,

53 – ‘5’.

Таким образом, ключ равен `"MerryChristmas2025"`

При запуске программы, если ввести указанный ключ, программа вернет ‘True’. При этом можно заметить, что можно вводить строку любой длины, главное, чтобы первые 18 символов были “`MerryChristmas2025`”.

Ответ: MerryChristmas2025