Общие положения об олимпиаде

Где размещать решения

На компьютере каждого участника создана папка **ispring-challenge-2026**.

Внутри папки размещаются:

- p01_название_задания...
- p02_название_задания...
- p03_название_задания...

Код решения размещать в папке вида “p01_название_задания”.

Каждое задание имеет хорошие и плохие тесты. При хороших тестах программа получает на вход валидные данные, при плохих – невалидные и должна вывести ошибку. Доступный вам набор тестов неполный и проверяет только некоторые ситуации. Вы можете написать дополнительные тесты самостоятельно.

Вы можете использовать тесты, чтобы проверить себя.

В некоторых заданиях есть **заготовки кода** для упрощения решения.

В каждой папке находится:

- Папка **good** – хорошие тесты
- Папка **bad** – плохие тесты
- **run_tests.bat** – файл для запуска тестов
- **README.txt** – инструкция по запуску тестов

Правила

Во время олимпиады запрещено пользоваться искусственным интеллектом, интернетом, телефоном и другими средствами коммуникации без разрешения наблюдателя. Если замечено использование телефона первый раз – будет сделано предупреждение.

За повторное использование – **дисквалификация и отстранение от соревнований**.

Завершение выполнения заданий

Перед тем, как покинуть рабочее место, сообщите наблюдателю, чтобы он скопировал ваше решение. Не выключайте компьютер до тех пор, пока не убедитесь, что ваше задание скопировано.

Контур защиты

SLA (Время на решение): 3 часа

Внимание, команда! Зафиксирована скоординированная атака на периметр безопасности нашей финансовой платформы. Злоумышленники используют распределенную сеть ботов для подбора учетных данных и попытки инъекций в базу данных.
Система автоматического реагирования (SOAR) перегружена объемом сырых логов и не может коррелировать события. Аналитики первой линии не справляются с потоком данных. Требуется ручное вмешательство для написания кастомных скриптов обработки, выявления паттернов атаки и генерации правил блокировки.
Ваша миссия: разработать конвейер обработки угроз из трех модулей. Каждый модуль должен быть протестирован в симуляторе. Ошибка в логике может привести к блокировке легитимных клиентов или пропуску атаки.
Действуйте быстро и точно. Время простоя системы стоит компании тысячи долларов в минуту.

Задание 1. Разработка конвейера обработки угроз

Проблема: на периметр безопасности финансовой платформы совершена скоординированная атака. Система мониторинга зафиксировала поток событий, указывающих на попытки подбора паролей и SQL-инъекций. Из-за высокой нагрузки данные поступают в несовместимых форматах, с рассинхронизированным временем и техническим «мусором».

Ваша задача: разработать модуль предобработки данных, который приведет разрозненные логи к единому виду для корректной работы аналитических систем.

1. Форматы входных данных

Логи поступают из двух различных источников. Программа должна автоматически определять тип лога по первому символу строки.

Тип 1. Web Server

- **Признак:** строка начинается с **цифры** (начало IP-адреса).
- **Структура:** IP-адрес,Имя_пользователя,Код_статуса,Сообщение
- **Пример:** `192.168.1.1,admin,Normal login`

Тип 2. DB Firewall

- **Признак:** строка начинается с **латинской буквы**.
- **Структура:** набор пар ключ=значение, разделенных одним пробелом. Обязательные ключи: src (IP), user (имя), type (сообщение).
- **Пример:** `src=192.168.1.1 user=admin type=SQL_INJECTION action=block`

2. Требования к обработке

A. Классификация угроз (Severity)

Каждому событию необходимо присвоить уровень важности на основе анализа поля сообщения (Message или type):

- **CRITICAL:** сообщение содержит подстроку "SQL Injection" или "CRITICAL".
- **HIGH:** сообщение содержит "XSS" или "Brute Force".
- **MEDIUM:** сообщение содержит "failed", "error" или "suspicious".
- **LOW:** все остальные случаи.

B. Очистка и фильтрация

1. **Пропуск битых записей:** если в логе отсутствует любое из критических полей (IP, имя пользователя, сообщение) — запись должна быть проигнорирована.
2. **Дедупликация:** если в потоке встречаются абсолютно идентичные записи, необходимо сохранить только одну.
3. **Неизвестные форматы:** пустые строки или строки, не соответствующие признакам типов 1 и 2, должны пропускаться.

3. Формат выходных данных

Программа должна вывести список обработанных уникальных логов в порядке их появления. Каждое событие выводится в одну строку:

```
sourceIp=<IP>, userId=<имя>, eventType=<тип>, severity=<уровень>, rawPayload="<исходная_строка>"
```

Пример 1

INPUT:

```
192.168.1.1,admin,0,failed

src=10.0.0.1 user=db_user type=CRITICAL
```

OUTPUT:

```
sourceIp=192.168.1.1, userId=admin, eventType=WEB_SERVER, severity=MEDIUM, rawPayload="192.168.1.1,admin,0,failed"

sourceIp=10.0.0.1, userId=db_user, eventType=DB_FIREWALL, severity=CRITICAL, rawPayload="src=10.0.0.1 user=db_user type=CRITICAL"
```

Пример 2

INPUT:

```
192.168.1.1,admin,0,OK

192.168.1.1,admin,0,OK
```

OUTPUT:

```
sourceIp=192.168.1.1, userId=admin, eventType=WEB_SERVER, severity=LOW, rawPayload="192.168.1.1,admin,0,OK"
```

Пример 3

INPUT:

```
#192.168.1.1,admin,0,OK
```

OUTPUT (пусто):

Задание 2. Протокол детекции аномалий

Проблема: автоматический коррелятор перегрет. Вам поручено разработать ручной алгоритм анализа телеметрии, который выделит подозрительные источники по трём жёстким критериям и подготовит детерминированный отчёт для системы перехвата. Порядок источников в отчёте критичен: протоколы маршрутизации обрабатывают адреса строго по числовому значению октетов.

Условие задачи

Вам задан лог запросов аутентификации. Каждый запрос содержит временную метку, IP-адрес источника, идентификатор пользователя и статус запроса. Необходимо проанализировать каждый IP-адрес независимо и присвоить ему один или несколько статусов угрозы, если он удовлетворяет следующим критериям:

1. **BRUTE_FORCE:** строго более **5** неудачных запросов (со статусом FAIL) с одного IP-адреса за окно времени в **60** секунд (то есть разность между метками времени последнего и первого неудачного запроса в этой группе <= **60**).
2. **RATE_ANOMALY:** строго более **15** любых запросов (независимо от статуса) с одного IP-адреса за окно времени в **60** секунд (разность между метками времени последнего и первого запроса в группе <= **60**).
3. **CREDENTIAL_STUFFING:** строго более **5** уникальных идентификаторов пользователей (userId) зафиксировано с одного IP-адреса за весь лог.

В окно времени могут попадать события с одинаковым временем. Пересечение окон допускается, но каждое превышение порога фиксируется для данного IP-адреса единожды.

Формат входных данных

В строках заданы сами записи в хронологическом порядке. Каждая запись состоит из четырех разделенных пробелом полей:

- **timestamp** — целое неотрицательное число, временная метка в секундах. События в логе отсортированы по неубыванию timestamp.
- **sourceIp** — строка, представляющая собой стандартный IPv4-адрес (четыре числа от 0 до 255, разделенные точками, без ведущих нулей).
- **userId** — строка, состоящая из строчных и/или заглавных латинских букв, цифр и символов подчеркивания (длиной от 1 до 50 символов).
- **status** — строка, принимающая одно из двух значений: **SUCCESS** или **FAIL**.

Формат выходных данных

Выведите список подозрительных IP-адресов. Для каждого адреса на отдельной строке выведите сам IP-адрес, а затем через пробел — все обнаруженные для него типы угроз в следующем порядке: **BRUTE_FORCE**, **RATE_ANOMALY**, **CREDENTIAL_STUFFING**.

В случае некорректного ввода строки(недостаточное количество параметров), выводится ошибка:

"Error: Invalid log entry in <number> line",

где <number> – это номер строки (первая строка начинается с 1, а не с 0).

Пример 1

INPUT:

```
1000 10.1.0.12 u1 FAIL
1001 10.1.0.12 u2 SUCCESS
1002 10.1.0.12 u3 FAIL
1003 10.1.0.12 u4 SUCCESS
1004 10.1.0.12 u5 FAIL
1005 10.1.0.12 u6 SUCCESS
1010 10.1.0.2 admin FAIL
1020 10.1.0.2 admin FAIL
1030 10.1.0.2 admin FAIL
1040 10.1.0.2 admin FAIL
1050 10.1.0.2 admin FAIL
1060 10.1.0.2 admin FAIL
2000 192.168.1.1 test SUCCESS
2001 192.168.1.1 test SUCCESS
2002 192.168.1.1 test SUCCESS
2003 192.168.1.1 test SUCCESS
2004 192.168.1.1 test SUCCESS
2005 192.168.1.1 test SUCCESS
2006 192.168.1.1 test SUCCESS
2007 192.168.1.1 test SUCCESS
2008 192.168.1.1 test SUCCESS
2009 192.168.1.1 test SUCCESS
2010 192.168.1.1 test SUCCESS
2011 192.168.1.1 test SUCCESS
2012 192.168.1.1 test SUCCESS
2013 192.168.1.1 test SUCCESS
2014 192.168.1.1 test SUCCESS
2015 192.168.1.1 test SUCCESS
```

OUTPUT:

```
10.1.0.12 CREDENTIAL_STUFFING
10.1.0.2 BRUTE_FORCE
192.168.1.1 RATE_ANOMALY
```

Пример 2

INPUT:

```
alice FAIL
1774437605 192.168.1.50 alice
```

OUTPUT:

```
Error: Invalid log entry in 1 line
```

Примечание (разбор примера)

1. **10.1.0.12** зафиксировал попытки входа под 6 уникальными логинами (u1...u6). Порог "строго больше 5" превышен, поэтому адрес получает статус **CREDENTIAL_STUFFING**.
2. **10.1.0.2** совершил 6 неудачных запросов (**FAIL**). Первый запрос был в секунду 1010, последний — в 1060. Разница составляет 1060 – 1010 = 50 секунд, что меньше или равно окну в 60 секунд. Присваивается статус **BRUTE_FORCE**.
3. **192.168.1.1** совершил 16 успешных запросов с секунды 2000 по 2015. Все они укладываются в 15 секунд, порог "строго больше 15" запросов нарушен. Статус — **RATE_ANOMALY**.

Задание 3. Генерация политик WAF

Проблема: угрозы идентифицированы. Требуется немедленное применение контрмер. Ручная блокировка невозможна из-за масштаба атаки. Необходимо автоматически сгенерировать конфигурацию для Web Application Firewall (WAF) и проверить её эффективность на симуляции трафика, минимизировав ложные срабатывания.

Ваша задача: разработать модуль автоматического формирования правил защиты на основе анализа рисков и провести симуляцию их работы.

1. Расчет оценки риска

Для каждого выявленного источника угрозы (Threat Actor) необходимо рассчитать балл риска по шкале от 0 до 100. Сумма баллов складывается из следующих факторов:

- **использование TOR:** +50 баллов.
- **использование VPN:** +20 баллов.
- **неудачные попытки входа:** +5 баллов за каждую попытку.
- **несоответствие локации:** +15 баллов (если текущая геопозиция не совпадает с типичной для пользователя).
- **примечание**: итоговый балл не может превышать 100 (если он превысил это значение, то записываем балл = 100).

2. Генерация политик

На основе рассчитанного риска необходимо сформировать правила блокировки. Правила генерируются по следующим критериям:

Риск (Score)	**Тип правила (Type)**	**Действие (Action)**
80 и выше	BLOCK	DROP
от 40 до 79	RATE_LIMIT	CHALLENGE
ниже 40	Правило не создается	-

3. Симуляция и критерии успеха

Для проверки конфигурации через систему пропускается поток трафика (смесь запросов и атак).

- **Mitigation Rate (Эффективность):** процент заблокированных атак от их общего числа.
- **False Positive Rate (Ложные срабатывания):** процент ошибочно заблокированных легитимных запросов.

Критерий успеха: система считается защищенной (**SECURED**), если удалось заблокировать более **90%** атак при уровне ложных срабатываний менее **1%**.

4. Формат входных данных

Блок 1. Список угроз

- Первая строка: целое число N — количество выявленных угроз.
- Следующие N строк содержат параметры угрозы, записанные через пробел:
  `IP (string), isVpn (0/1), isTor (0/1), failedAttempts (int), locationMatch (0/1), reason (string),`

  где reason – строка без пробелов.

Блок 2. Трафик для симуляции

- Строка с целым числом M — количество запросов в тестовом трафике.
- Следующие M строк:
  `IP (string), isAttack (0 /1)`

5. Формат выходных данных

Программа должна вывести отчет SOC-центра в следующем формате:

итоговые метрики:

- Mitigation Rate: <число>% (с точностью два знака после запятой).
- False Positive Rate: <число>% (с точностью два знака после запятой).
- SYSTEM STATUS: **SECURED** или **VULNERABLE**.

Пример входных данных

Пример 1

INPUT:

```
2
192.168.1.50 0 1 10 0 BruteForce
10.0.0.5 1 0 2 1 PotentialScraper
3
192.168.1.50 1
10.0.0.5 1
172.16.0.1 0
```

OUTPUT:

```
### SOC DASHBOARD METRICS ###
Mitigation Rate: 50.00%
False Positive Rate: 0.00%
SYSTEM STATUS: VULNERABLE
```

*(Пояснение: был заблокирован только первый IP, так как его Risk Score = 50(Tor) + 10*5(Attempts) + 15(Loc) = 115 -> 100. Второй IP получил 30 баллов и правило не создалось).

Пример 2

INPUT:

```
1
5.5.5.5 0 1 10 1 blocking
1
5.5.5.5 0
```

OUTPUT:

```
### SOC DASHBOARD METRICS ###
Mitigation Rate: 0.00%
False Positive Rate: 100.00%
SYSTEM STATUS: VULNERABLE
```

(Пояснение: произошло ложное срабатывание. Список угроз сообщил, что пользователь 5.5.5.5 опасен, однако его трафик не совершал атак).

Пример 3

INPUT:

```
1
6.6.6.6 0 1 10 1 blocking
2
6.6.6.6 1
7.7.7.7 0
```

OUTPUT:

```
### SOC DASHBOARD METRICS ###
Mitigation Rate: 100.00%
False Positive Rate: 0.00%
SYSTEM STATUS: SECURED
```

(Пояснение: система успешно отработала. Список угроз сообщил, что пользователь 6.6.6.1 опасен, его трафик был атакой и система его заблокировала. Пользователя 7.7.7.7 в списке угроз не было, его трафик не был атакой => система отработала успешно).