Цифровой хаос в Дашборге

Elena Ubogoeva

2025-04-04

Задача:

Вечером 25 марта 2025 года система доставки пиццы в Дашборге дала сбой: заказы пропали, курьеры остались без маршрутов, а клиенты — без ужина. Это не случайность, а преднамеренное вмешательство.

Давайте выясним, что случилось, восстановим данные, найдем предполагаемого злоумышленника и распределим компенсацию 90000 рублей пострадавшим пользователям.

|  |
| --- |
| Важно |
| Решение на этой странице не обновлялось после дедлайна, можно ознакомиться с решением в pdf по [ссылке](https://drive.google.com/drive/u/1/folders/1LD2mlk5sx8K4k6ewO_CR1MYekXRNPrtc) |

## Восстановление данных

Частично восстановленные данные есть по [ссылке](https://docs.google.com/spreadsheets/d/1qhxUaTt_ICIDd24t5DlzJbS22iQ4R4_rwLfvJs2ugFM/edit?gid=0#gid=0), таблицы orders и device.

Начнем с загрузки данных и посмотрим на структуру данных.

head(orders)

order\_id order order\_price user\_id  
 <num> <char> <num> <num>  
1: 999 Мясной микс, 3 пиццы 1878 102475  
2: 680 Мясной микс 579 218841  
3: 687 Мясной микс 579 521065  
4: 685 Дьябло 560 106146  
5: 683 Сицилийская 545 854950  
6: 678 <NA> 530 263835

head(device)

ip device operator\_ip user\_id  
 <char> <char> <char> <num>  
1: 192.168.66.187 unknown 192.168.215.115 102475  
2: 192.168.100.221 mobile 192.168.203.103 218841  
3: 192.168.100.221 mobile 192.168.202.102 521065  
4: 192.168.100.221 web 192.168.212.112 106146  
5: 192.168.100.221 mobile 192.168.217.117 854950  
6: 192.168.100.221 web 192.168.209.109 263835

Подсчитаем, сколько значений было пропущено и в каких столбцах

colSums(is.na(orders))

order\_id order order\_price user\_id   
 0 269 0 0

colSums(is.na(device))

ip device operator\_ip user\_id   
 0 0 0 0

У нас есть пропущенные значения в составе заказа (поле order), которые нужно восстановить. Мы можем восстановить цену отдельных пицц, используя существующие заказы, где в составе только одна пицца и указана цена заказа.

single\_item\_orders <- orders %>%   
 filter(!str\_detect(order, ", ")) %>% # Нет запятой с пробелом - одна позиция  
 distinct(order, order\_price)  
single\_item\_orders %>%   
 arrange(order\_price)

order order\_price  
 <char> <num>  
 1: Маргарита 479  
 2: Четыре сыра 499  
 3: Вегетарианская 510  
 4: Гавайская 520  
 5: Пеперони 530  
 6: Неаполитанская 530  
 7: Сицилийская 545  
 8: Дьябло 560  
 9: Мясной микс 579  
10: 3 пиццы 1299

Таким образом, удалось восстановить цену на отдельные пиццы, и почти все имеют разную цену, только Пеперони и Неаполитанская стоят одинаково (530). Следовательно, восстановить заказы, где в составе есть пицца по такой цене мы не сможем полностью, напишем, что это Пеперони или Неаполитанская.

Восстановим пустые заказы с помощью комбинаторики, используя цены для отдельных пицц, перебирая известные цены, пока комбинация не сойдется.

find\_combination <- function(target\_price, prices,   
 items, max\_items = 4) {  
 # Создаём список комбинаций для разного числа элементов (от 1 до max\_items) и объединяем все комбинации в один список  
 all\_combos <- map(1:max\_items, ~combn(prices, .x, simplify = FALSE)) %>% flatten()  
   
 matched\_combos <- keep(all\_combos, ~sum(.x) == target\_price) %>%  
 map(sort) %>%   
 unique()  
 # Преобразуем каждую комбинацию цен в соответствующие элементы  
 matched\_items\_list <- map(matched\_combos, ~{  
 matched\_prices <- .x  
 matched\_items <- items[match(matched\_prices, prices)]  
 paste(matched\_items, collapse = ", ")  
 })  
 # Объединяем все комбинации через ИЛИ  
 return(paste(matched\_items\_list, collapse = " ИЛИ "))  
}  
  
# запускаем только для тех строк, где пропущен состав заказа  
orders[is.na(order), order\_recovered := map\_chr(order\_price, find\_combination,   
 prices = single\_item\_orders$order\_price,   
 items = single\_item\_orders$order)] %>%   
 .[, order\_recovered := coalesce(order, order\_recovered)]  
  
# теперь заменяем Пеперони или Неаполитанская везде, где мы восстановили заказ на Пеперони/Неаполитанская  
  
orders[str\_detect(order\_recovered, 'Пеперони|Неаполитанская') & is.na(order),   
 order\_recovered := str\_replace(order\_recovered, 'Пеперони|Неаполитанская', 'Пеперони/Неаполитанская')]  
  
# проверим, сколько осталось пропущенных значений:  
sum(is.na(orders$order\_recovered))

[1] 0

# запишем это в файл  
writexl::write\_xlsx(orders, 'data/orders\_recovered.xlsx')

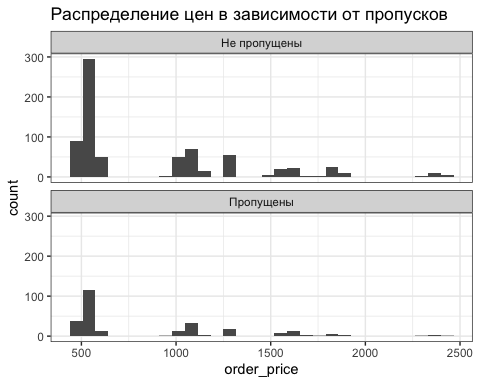
Данные восстановлены, почти всё, кроме наличия в составе Пеперони и Неаполитанской удалось восстановить точно. Кроме этого, некоторые заказы допускают несколько комбинаций, они указаны через ИЛИ в поле order\_recovered, например в заказе с ценой 1549, с такой ценой могут быть Маргарита, Вегетарианская, Дьябло ИЛИ Четыре сыра, Гавайская, Пеперони/Неаполитанская.

По [ссылке](https://docs.google.com/spreadsheets/d/10kjykmWYuKnPzQxWiIVUBZgz2fhoAseO/edit?gid=146993528#gid=146993528) можно ознакомиться с восстановленными данными (или [здесь](https://github.com/ubogoeva/R4Analytics/blob/master/posts/data/orders_recovered.xlsx), альтернативно на случай проблем с доступом).

## Определить злоумышленников

Построим распределение цен по заказам, чтобы понять, есть ли какой-то паттерн в пропущенных значениях в зависимости от цены.

orders %>%   
 mutate(is\_missed = if\_else(is.na(order), 'Пропущены', 'Не пропущены')) %>%   
 ggplot(aes(order\_price))+  
 geom\_histogram()+  
 facet\_wrap(~is\_missed, nrow = 2)+  
 ggtitle('Распределение цен в зависимости от пропусков')+  
 theme\_bw()



Определенного паттерна не наблюдается, пропущенные заказы имеют схожее распределение по цене с существующими.

Распределение по устройствам тоже не дает определенной информации, встречаются устройства mobile, unknown и web с примерно одинаковой частотой.

device %>%   
 count(device)

device n  
 <char> <int>  
1: mobile 359  
2: unknown 323  
3: web 318

Давайте посмотрим на распределение ip по таблице device.

Сначала проанализируем диапазон ip.

device %>%   
 summarise(min(ip), max(ip))

min(ip) max(ip)  
1 192.168.1.122 192.168.99.220

Диапазон ip в датасете от 192.168.1.122 до 192.168.99.220, что говорит о том, что все ip внутренние, а значит заказы обрабатываются через внутреннюю систему. Другая гипотеза, что для кейса замаскированы реальные ip. Рассмотрим вариант, в котором это будут действительно внутренние ip, через которые проходит заказ.

### Анализ частоты встречаемости ip-адресов пользователей

Найдем топ-10 самых часто встречающихся ip в таблице

device %>%   
 count(ip) %>%   
 arrange(desc(n)) %>%   
 head(10)

ip n  
 <char> <int>  
 1: 192.168.47.168 222  
 2: 192.168.27.148 182  
 3: 192.168.4.125 75  
 4: 192.168.84.205 75  
 5: 192.168.96.217 75  
 6: 192.168.13.134 24  
 7: 192.168.69.190 24  
 8: 192.168.100.221 19  
 9: 192.168.29.150 19  
10: 192.168.77.198 19

Самый часто встречающийся ip 192.168.47.168, с 222 вхождениями в таблице device, остальные айпи из топ-10 тоже подозрительны.

Давайте соединим таблицы orders и device по полю user\_id и выведем заказы и юзер айди для самого часто встречающегося ip адреса

df <- orders %>% full\_join(device, by = 'user\_id')  
df %>%   
 filter(ip == '192.168.47.168') %>%   
 arrange(order\_id) %>%   
 select(order\_id, order, user\_id, ip, device, operator\_ip) %>%   
 head(5)

order\_id order user\_id ip  
 <num> <char> <num> <char>  
1: 1 3 пиццы 653310 192.168.47.168  
2: 2 Дьябло 790532 192.168.47.168  
3: 3 Вегетарианская, Неаполитанская, Дьябло 643164 192.168.47.168  
4: 4 Гавайская 262399 192.168.47.168  
5: 5 Пеперони 554538 192.168.47.168  
 device operator\_ip  
 <char> <char>  
1: mobile 192.168.214.114  
2: unknown 192.168.213.113  
3: mobile 192.168.207.107  
4: web 192.168.207.107  
5: web 192.168.214.114

Обращаю внимание, что все order\_id с этого ip-адреса идут по порядку, с 1 по 222, и затем заказ под номером 586, но при этом айди пользователя разные и попадают на разные operator\_ip.

Самый часто встречающийся user\_id с таким ip - это 796365, у которого 5 заказов. Это может указывать на возможный источник атаки. Стоит посмотреть, какие заказы были раньше у этого пользователя, так как если это новый пользователь, то возможно потенциальный кандидат на взломщика.

device %>%   
 filter(ip == '192.168.47.168') %>%   
 count(user\_id) %>%   
 arrange(desc(n)) %>%   
 head(3)

user\_id n  
 <num> <int>  
1: 796365 5  
2: 119110 1  
3: 119700 1

Допущение такое, что все заказы с одного ip - это действительно заказы с одного ip-адреса (а не условность кейса), что делает подобную частоту заказов с одного адреса подозрительной. Это похоже на намеренную перегрузку системы, что могло вызвать сбой.

Проверим следующий часто встречающийся в данных ip адрес: 192.168.27.148.

df %>%   
 filter(ip == '192.168.27.148') %>%   
 arrange(order\_id) %>%   
 select(order\_id, order, user\_id, ip, device, operator\_ip) %>%   
 head(5)

order\_id order user\_id ip device operator\_ip  
 <num> <char> <num> <char> <char> <char>  
1: 223 Дьябло 235555 192.168.27.148 unknown 192.168.210.110  
2: 224 <NA> 657248 192.168.27.148 mobile 192.168.203.103  
3: 225 3 пиццы 176961 192.168.27.148 mobile 192.168.211.111  
4: 226 Гавайская 469335 192.168.27.148 mobile 192.168.91.212  
5: 227 Пеперони 400617 192.168.27.148 mobile 192.168.218.118

Здесь тоже заказы идут по порядку, с 223 по 404, что усиливает подозрения, что это не случайность.

Найдем пользователей, которые наиболее часто заказывали с этого айпи:

device %>%   
 filter(ip == '192.168.27.148') %>%   
 count(user\_id) %>%   
 arrange(desc(n)) %>%   
 head(3)

user\_id n  
 <num> <int>  
1: 853594 5  
2: 118997 1  
3: 122358 1

Самый часто встречающийся юзер с этого айпи: 853594, 5 заказов, рекомендации такие же как и для пользователя 796365 — проверить предыдущие заказы на аномальность данных.

Подозреваем, что два ip 192.168.47.168 и 192.168.27.148 устроили атаку, если это внутренний айпи, то стоит отследить, что это за устройство и кто имеет к нему доступ, посмотреть записи с камер наблюдения.

Однако, если проанализировать остальные айпи пользователей из топ-10 встречающихся, то у них тоже заказы идут по порядку, что может указывать на нормальную работу системы. Нужно больше данных, чтобы сравнить, как это происходит в обычные дни без сбоя, чтобы понять, является ли это чем-то аномальным.

### Анализ совпадений пользовательских ip и операторских

Еще можно посмотреть на совпадающие ip-пользователей и операторов.

device[ip %in% operator\_ip, ]

ip device operator\_ip user\_id  
 <char> <char> <char> <num>  
1: 192.168.91.212 mobile 192.168.207.107 307937  
2: 192.168.91.212 web 192.168.215.115 307937  
3: 192.168.46.167 mobile 192.168.213.113 363028  
4: 192.168.46.167 web 192.168.209.109 363028  
5: 192.168.46.167 unknown 192.168.211.111 363028  
6: 192.168.85.206 web 192.168.212.112 532741  
7: 192.168.85.206 mobile 192.168.203.103 532741  
8: 192.168.85.206 web 192.168.207.107 532741

Здесь айпи адреса 192.168.91.212, 192.168.46.167, 192.168.85.206, которые встречаются и в пользовательском айпи, и в айпи оператора. Это тоже подозрительно и указывает на потенциальный взлом. Устройство одновременно клиент и оператор для разных заказов, такое может быть, например, в случае взломанного терминала. Хотя оставляем вероятность, что сотрудник пиццерии заказал пиццу с рабочего устройства :)

Соответственно, наиболее подозрительные ip это 192.168.47.168 и 192.168.27.148, по критерию наиболее частотных заказов и 192.168.91.212, 192.168.46.167, 192.168.85.206 по критерию одинаковых пользовательских ip и ip оператора. Все похоже на внутреннюю атаку, и рекомендация посмотреть также камеры видеонаблюдения и логи сотрудников, которые имеют доступ к подозрительным адресам.

## Распределение компенсации пострадавшим

Предполагаем, что настоящие пользователи, которые пострадали, не с вышеуказанными айпи.

У нас есть 90000р, чтобы распределить между настоящими пострадавшими. Думаю, стоит это сделать пропорционально сумме заказов.

Отфильтруем пользователей, кто не заказывал с подозреваемых ip адресов, оставим только уникальные заказы.

df\_normal <- df %>%   
 filter(!ip %in% c('192.168.47.168', '192.168.27.148', '192.168.91.212', '192.168.46.167', '192.168.85.206' )) %>%   
 distinct(user\_id, order, .keep\_all = TRUE)

Для каждого пользователя выведем его сумму заказов:

orders\_normal\_users <- df\_normal %>%   
 group\_by(user\_id) %>%   
 summarise(total\_order\_price = sum(order\_price, na.rm = TRUE)) %>%   
 ungroup()

Посчитаем общую сумму заказов для пострадавших пользователей

order\_sum <- orders\_normal\_users$total\_order\_price %>% sum()

Рассчитаем компенсацию пропорционально общей сумме заказов, округлим в меньшую сторону, а потом до целых десятков

orders\_normal\_users <- orders\_normal\_users %>%   
 mutate(compensation = (total\_order\_price / order\_sum) \* 90000) %>%   
 mutate(compensation = floor(compensation)) %>%   
 mutate(compensation\_rounded = round(compensation / 10) \* 10)

Проверим, сколько в итоге получилась компенсация

total\_compensation\_rounded <- sum(orders\_normal\_users$compensation\_rounded)  
remained\_money <- 90000 - total\_compensation\_rounded  
remained\_money

[1] 390

Оставшуюся сумму (390) распределим так, чтобы округлить в большую сторону тех, кого на предыдущем этапе округлили в меньшую, и чтобы в сумме компенсация получилась ровно 90000.

orders\_normal\_users <- orders\_normal\_users %>%   
 mutate(diff\_to\_next = (ceiling(compensation / 10) \* 10) - compensation\_rounded) %>%   
 arrange(desc(diff\_to\_next), desc(total\_order\_price)) %>% # Сортируем по разнице и сумме заказов  
 mutate(compensation\_final = compensation\_rounded +   
 ifelse(row\_number() <= remained\_money / 10, 10, 0))

Проверяем:

orders\_normal\_users$compensation\_final %>% sum()

[1] 90000

Сумма получилась ровно 90000, пропорционально сумме заказа и с округлением до кратных 10 чисел. Полученную компенсацию можно зачислить на бонусный счет клиента или выплатить в виде промокода (но бонусный счет лучше).

Сохраним таблицу

orders\_final <- orders\_normal\_users %>%   
 select(user\_id, total\_order\_price, compensation\_final) %>%   
 full\_join(orders, by = 'user\_id')  
writexl::write\_xlsx(list(orders\_normal\_users, orders\_final), 'data/user\_compensation.xlsx')

Данные можно найти [здесь](https://docs.google.com/spreadsheets/u/1/d/18bYH_WlD1d1ymJ6jF1DUfN1MJNOwgSuv/edit?usp=drive_web&ouid=102933536119195722969&rtpof=true) или [здесь](https://github.com/ubogoeva/R4Analytics/blob/master/posts/data/user_compensation.xlsx), они одинаковые.

## Выводы:

1. Данные по заказам восстановлены, находятся по ссылке [здесь](https://docs.google.com/spreadsheets/d/10kjykmWYuKnPzQxWiIVUBZgz2fhoAseO/edit?rtpof=true&gid=1019265358#gid=1019265358)
2. Есть подозреваемые злоумышленники, с ip-адресов 192.168.47.168, 192.168.27.148, 192.168.91.212, 192.168.46.167, 192.168.85.206, но нужно больше информации, чтобы сделать более точный анализ
3. Компенсация по пользователям в размере 90000 распределена, в зависимости от цены заказа и рекомендовано зачислить на бонусный счет в приложении. Подробнее можно ознакомиться [здесь](https://docs.google.com/spreadsheets/d/18bYH_WlD1d1ymJ6jF1DUfN1MJNOwgSuv/edit?gid=429768562#gid=429768562)

Надеюсь, это решение поможет восстановить справедливость и порядок в Дашборг :)