1. *Проанализировать систему и понять, какие первые шаги нужно сделать для обеспечения безопасности пользовательских данных и почему. Аргументы  стройте на времени, которое уйдёт на доработку, и пользе, которую она принесёт.*

Для начала предположим, что сроки были ОООООООООЧЕНЬ непродуманно назначены и вопрос безопасности не думался от слова совсем. Так же уже не спешим. Враг уже может быть в системе, данные уже утекли, потому аккуратно снимаем бекапы критических данных и готовимся к полному краху. Так же в системе изначально были предусмотрены сервисы по доступу к БД, потому считаем, что напрямую в БД запросы никак не ходят (хоть кто-то же сделал свою работу нормально?..) и возможно получить только то и только так, как реализовано в API.

В нулевую очередь пытаемся настроить хоть какой-то мониторинг сети. Кто куда ходит и зачем, не лезет ли логистика за чужими телефонами, например. Пытаемся начать хоть что-то видеть. Даже если мы начнем что-то закрывать – мы не поймем, получилось ли, пока не найдем течь.

В первую очередь лишаем потенциальных врагов доступа к учёткам пользователей. Для это настраиваем свою систему так, чтобы юзеры могли использовать только «безопасные» пароли (более 8 символов, регистры, цифры, спецсимволы). Тут же заставляем юзеров сменить пароли (у нас же есть такой механизм? Есть же?..). Времени займет не много, закроет самую очевидную проблему «пароль 123456». Кстати, менеджерские и админские учетки, в том числе на сервисах – пересмотреть 2 раза. Особенно обратить внимание на root, admin и т. д.

Во вторую – настраиваем фаерволы. Хоть какие-то. Блокируем все, разрешаем только белые списки. Не нужен ftp доступ к серверу БД. Строго необходимо, хоть и может занять продолжительное время. И систему убить пару раз. Менять фаерволлы на проде – плохая идея. Очень пллохая.

Во третью – закрываем каналы tls`ом. Все каналы, в том числе между сервисами. Вдруг кто умудрился залезть в физический сервер или в комп пользователя и просто мониторит трафик? Чуть больше времени, но https – строго необходим.

В четвертую - меняем простые int`овые номера идентификаторов на сложные, например, Guid. Чтобы даже имея доступ к базе не набрутить всех подряд, только текущего пользователя. Времени немного, польза – колоссальна.

В пятую – настраиваем мониторинг работы менеджеров и ловим связь между утёкшими данными и конкретным менеджером. Ежели менеджер может видеть только своих Клиентов (или свои Заказы, тут как бизнес решит), можно выловить хвостато-пищащего сотрудника. Ежели нет – надо настроить, чтобы было.

В шестую - настраиваем жесткую идентификацию и авторизацию между сервисами: только тот, кому можно, только туда, куда можно и строго в определенных объемах. Складу не нужны Клиенты. Логистике не нужны Продукты, только объем и номера коробок со Склада, ровно, как и история Заказов Клиента, только часть контактных данных конкретно этого Клиента по конкретно этому Заказу. Это точно будет долго и не в первую очередь, но крайне желательно сотворить.

В седьмую – враг мог подменить код. Если есть заранее целые копии сервисов – переподнимаем их. Если нет – страдаем и вручную пересматриваем, не уходит ли траффик куда в чужую сеть. В теории, на этапе фаерволлов нежелательный траффик сам отвалится, но лучше перебдеть.

В идеале сделать мониторинг, увидеть все щели, а затем внести все изменения разом. Чтобы даже имея закладки в системе злоумышленник не сумел вернуться. Самые очевидные причины прикрыли, дальше углубленное изучение каждой из потенциальных проблем.

1. *Учесть векторы атак, описанные в OWASP TOP 10, который мы разбирали на первом уроке, и предусмотреть защиту от них.*

* Инъекции – для доступа к БД реализованы сервисы с API. Просматриваем внимательно код, проверяем, что обратиться к каждому из методов можно только после аутентификации и авторизации при наличии прав. Ограничиваем сами методы (При всем желании не понадобится кому бы то ни было получить ВСЕ Заказы или ВСЕХ Клиентов.).
* Недочеты системы аутентификации и хранения сессий – куки хранить недолго, только 1 сессию, проверять браузер, ip, иные критерии… Пусть лучше пользователь еще раз авторизуется, чем с украденной кукой кто-то зайдет с другого компьютера.
* Межсайтовый скриптинг – мониторинг наших серверов, контроль изменений сервисов. Крайний заход на сервис был с малазийским ip и размер шаблона html страницы вырос на 30 кБ – явно подозрительно. Файерволлы и контроль доступа помогут спастись. Так же, если на сайте есть что-то типа отзывов, проверять вводимые в поля значения, заменять служебные символы, не выводить в базу и из базы без проверки. Чтобы, внезапно, не выяснилось, что браузер интерпретирует комментарий пользователя как js-скрипт.
* Небезопасные прямые ссылки на объекты – замена на сложные id, а также контроль доступа. Негоже видеть чужие Заказы Клиенту.
* Небезопасная конфигурация – бритва Оккама. Отключить и заблокировать все лишние службы, порты, сервисы, функции. Настроить нужные именно нужным тебе образом, желательно необычным. Нужен ssh? Настрой его на 60524 порту! И чтоб через knock-knock`инг.
* Незащищенность критичных данных. Пароли – не хранить в открытом виде, только солёные хэши. Номера карт вообще не хранить, отдать на откуп внешним, интегрированным, сервисам оплаты. Чем меньше таких данных – тем меньше точек утери. Https по умолчанию на всё. И заперт на http.
* Отсутствие функций контроля доступа – уже не раз говорилось. Даже api сервисов должны проверять валидность спрашивающего спрашивать сии данные. Опять же. Возможно настроить так, что лишь один только сервис Сайта смотрит наружу, остальные только локальный доступ имеют. Тогда в принципе без спроса никто-ничего не получиит.
* Межсайтовая подделка запроса - CSRF встроен на уровне браузера, и, если юзер её отключил – сам дурак. С нашей стороны важно понять, что кто-то не пытается автоматизировать запросы через фишинговую страницу, транслируя данные, вводимые пользователем к нам. Тут сложно, нужно читать заголовки запросов, анализировать их. Грамотный фишинг всё равно прорвётся. Другой вопрос, что никто для сайта мебели его строить не будет.
* Использование компонентов с известными уязвимостями. Всё обновлять до последней стабильной версии. В экспериментальных могут быть свежие 0-day`ки. Использовать минимум необходимого. Перенастраивать с дефолтных настроек.
* Непроверенные переадресации и пересылки – не использовать переадресацию. Если уж необходима – явно предупреждать пользователя куда он будет послан. Если враг уже на наших серверах – молиться и оперативно это мониторить и исправлять.

1. *Проработать систему управления заказом, чтобы злоумышленники не могли похитить данные пользователей.*

Грамотный злоумышленник всё равно получит. Это очень комплексный вопрос.

* Система должна обеспечивать защиту северов от несанкционированного доступа
* Компьютер пользователя должен быть чист от вирусов и враждебного ему ПО
* Каналы связи должны быть защищенными
* Менеджеры, админы и иные имеющие права люди должны быть не серыми и не хвостатыми
* Пользователи не должны вестись на «очень выгодные предложения» на сомнительных местах. Равно как и хранить пароли в файле «пароли.txt»

С нашей стороны указанных выше мероприятий вполне достаточно, чтобы отвадить 99% злоумышленников. Ну а те, кто всё же что-то получит – явно заслужили. Нужен мониторинг, чтобы этот факт увидеть, восхититься, закрыть дырку. Нет ни одной абсолютно надёжной системы безопасности.

1. *Рассмотреть и защититься от следующих векторов атак:*
   1. *Injections (инъекции) –* сделано выше
   2. *Broken Authentication (нарушенная аутентификация)* – сделано выше
   3. *Sensitive Data Exposure (незащищённость конфиденциальных данных)* – сделано выше
   4. *XML External Entities (XXE) Insecure Deserialization (внешние сущности XML, небезопасная десериализация)* – использовать простые json для общения между сервисами. Заполнять json после многократной проверки: на сайте не позволять вводить лишние символы в форму, отправлять эти формы с заменых технических символов на безопасные, на серверах – снова проверять и только затем раскидывать по json`ам.
   5. *Broken Access Control (нарушение контроля доступа)* – сделано выше

Примерная сеть:

