Оглавление

Содержание

Предисловие

Когда-то давно решил я посмотреть на Amazon книги о написании безопасных приложений на Python. Будет из чего выбрать, подумал я. К тому времени было издано множество книг о программировании на Python на разные темы: оптимизация кода, машинное обучение, веб-разработка и т. д.

К моему удивлению, такой книги не оказалось. А ведь эта тема касается задач, с которыми мы с коллегами сталкиваемся ежедневно. Как убедиться, что весь сетевой трафик зашифрован? На каком фреймворке построить безопасный веб-сайт? Каким алгоритмом стоит подписывать данные?

Спустя годы практики мы нашли проверенные приемы для решения стандартных задач и без применения несвободного ПО. За это время мы создали с нуля несколько проектов, где в безопасности хранились личные данные миллионов пользователей. Трое наших конкурентов, кстати, были взломаны.

Начало 2020-го изменило наши жизни. Отовсюду доносились известия о COVID-19, и удаленная работа нежданно стала обыденностью. Каждый провел эту пандемию по-своему. Меня же настигла невыносимая скука.

Так что я решил одним выстрелом убить двух зайцев. Написание книги, во-первых, оказалось восхитительным способом унять тоску на протяжении целого года на карантине. Осенью 2020-го в Силиконовой долине это оказалось настоящим спасением. Из-за смога от бушующих лесных пожаров большинство местных сидели дома.

Во-вторых, что важнее, оказалось очень приятно написать книгу, которую я так и не смог тогда приобрести. Многие открывают стартапы в Силиконовой долине, чтобы звать себя основателями. Так же и множество книг создаются ради того, чтобы написавший звал себя автором. Но что стартап, что книга должны решать насущные проблемы и приносить пользу.

Надеюсь, эта книга станет вам подспорьем при написании безопасного кода.

Благодарности

Написание книги – тяжкий уединенный труд. Потому легко упустить из виду тех, кто оказывал помощь. Я хотел бы поблагодарить всех нижеуказанных людей. Перечисляю в порядке нашей встречи.

Спасибо Катрин Берковиц (Kathryn Berkowitz), вы были моей лучшей учительницей английского в старших классах. Извините,

что докучал вам. Амит Ратор (Amit Rathore), дружище, спасибо, что порекомендовал меня издательству Manning. Хочу поблагодарить Джея Филдса (Jay Fields), Брайна Гётса (Brian Goetz) и Дина Уомплера (Dean Wampler) за ваши советы и поддержку, пока я искал издателя. Кэри Кемпстон (Cary Kempston), благодарю за содействие. Без вашего опыта эта книга просто бы не состоялась. Майк Стивенс (Mike Stephens), спасибо, что взглянули на мою рукопись и увидели в ней потенциал. Тони Арритола (Toni Arritola), мой редактор-консультант, спасибо вам – вы объяснили, что к чему. Ваши советы просто бесценны, благодаря вам я столько узнал о написании технических текстов. Майкл Дженсен (Michael Jensen), мой научный редактор, спасибо вам за содержательные рекомендации в короткие сроки. Благодаря вашим замечаниям и предложениям эта книга стала отменной.

И наконец, я хотел бы поблагодарить всех рецензентов издательства Manning, уделивших время на прочтение и поделившихся впечатлением. Аарон Бартон (Aaron Barton), Адриан Байерц (Adriaan Beiertz), Бобби Лин (Bobby Lin), Дайвид Морган (Daivid Morgan), Даниэль Васкес (Daniel Vasquez), Доминго Салазар (Domingo Salazar), Гжегож Мика (Grzegorz Mika), Ховард Уолл (Håvard Wall), Игорь ван Ооствин (Igor van Oostveen), Дженс Кристиан Бредал Мадсен (Jens Christian Bredahl Madsen), Камеш Ганешан (Kamesh Ganesan), Ману Сарина (Manu Sareena), Марк-Энтони Тейлор (Marc-Anthony Taylor), Марко Симон Зуппон (Marco Simone Zuppone), Мари Анн Тюгесен (Mary Anne Thygesen), Николас Актон (Nicolas Acton), Нинослав Серкез (Ninoslav Cerkez), Патрик Реган (Patrick Regan), Ричард Воан (Richard Vaughan), Тим ван Дорсен (Tim van Deurzen), Вина Гарапати (Veena Garapaty) и Уильям Джамир Силва (William Jamir Silva), ваши отклики помогли сделать книгу лучше.

Об этой книге

Python здесь является средством, чтобы обучить написанию защищенных программ. Проще говоря, эта книга больше про безопасность, чем про Python, и тому есть две причины. Во-первых, тема безопасности ПО куда обширнее, чем сам Python. Во-вторых, писать свои средства защиты – не лучшая идея. Серьезную работу стоит поручить алгоритмам, уже реализованным в Python, библиотеках кода либо сторонних утилитах.

В этой книге рассказывается о нюансах безопасного кода для начинающих программистов и профессионалов средней руки. Примеры иллюстрированы несложным кодом на Python. Для прочтения книги быть продвинутым профессионалом не требуется.

Для кого эта книга

Все примеры кода воспроизводят реальные задачи, стоящие перед разработчиками. Эта книга будет наиболее полезна тем, кто поддер-

живает уже работающие сервисы. Читателю потребуются начальные знания Python либо хорошее знакомство с другим популярным языком программирования. Эта книга будет полезна не только веб-разработчикам. Однако базовое понимание того, как работает интернет, пригодится при прочтении второй половины книги.

Возможно, вы не разработчик, а тестировщик. В таком случае вам станет ясно, что в первую очередь следует подвергать проверке. Но наша книга не рассказывает, как именно проводить тесты. Это все-таки разные навыки.

Эта книга не похожа на другие книги о безопасности. Здесь почти не будет показан процесс атаки со стороны злоумышленника, так что им здесь не будет особо что почерпнуть. Чтобы сгладить их разочарование, скажу, что временами злодеям будет дозволено одержать верх.

Структура книги

Первая глава рассказывает о том, что вам повстречается в книге. В ней кратко говорится о проверенных приемах написания безопасных программ. Остальные семнадцать глав делят книгу на три части.

Часть I «Основы криптографии» рассказывает о ее базовых понятиях. Описанное в ней вам еще повстречается во второй и третьей частях книги.

- Глава 2 начинает рассказ о криптографии с хеширования данных и проверки их целостности. Кроме того, мы впервые познакомимся с персонажами нашей книги.
- Глаза 3 вытекает из материала второй главы. В ней говорится о проверке подлинности данных с помощью генерации ключей и последующего хеширования данных секретным ключом.
- Глава 4 затрагивает две темы, без которых не обходится ни одна книга по безопасности. Это симметричное шифрование и неразглашение содержимого.
- Глава 5, опять же, вытекает из материала предыдущей. В ней рассказывается об асимметричном шифровании, цифровых подписях и неопровержимости деяния.
- Глава 6 на основе идей из предыдущих глав рассказывает об общепринятом сетевом протоколе Transport Layer Security.

Часть II «Проверка подлинности и предоставление прав на доступ» заключает в себе самую востребованную информацию. Она обязательно пригодится при поддержке коммерческих сервисов. В этой части содержатся пошаговые инструкции для воплощения типичных сценариев того, как пользователь взаимодействует с приложением.

- Глава 7 повествует о том, как установить пользовательский сеанс по протоколу HTTP, и в том числе о *cookie*. Эта информация является базой для осуществления множества атак, которые мы обсудим позже.

- В главе 8 мы говорим о том, как узнавать нашего пользователя: о регистрации и проверке подлинности.
- Глава 9 рассказывает про обращение с пользовательскими паролями. В этой главе я порядочно разошелся. Для ее понимания нужно прочтение предыдущих.
- Глава 10 переходит от проверки подлинности пользователя к разграничению его прав.
- Глава 11 завершает вторую часть рассказом об OAuth. Это широко используемый протокол предоставления прав на доступ.

Настоящее противоборство разворачивается в части III «Противостояние атакам». Она проще для понимания и в целом увлекательная.

- Глава 12 погружается в недра операционной системы. Файловая система, запуск одной программой других, доступ к командной оболочке.
- Глава 13 учит вас противостоять различным атакам с целью внедрения вредоносного кода через пользовательский ввод.
- Глава 14 целиком посвящена самой печально известной атаке с целью внедрения кода – *межсайтовому скриптингу* (cross-site scripting – XSS). В самом деле, куда же без него.
- Глава 15 знакомит вас с *политиками защиты содержимого* (Content Security Policy – CSP). В каком-то смысле это бонусная глава про межсайтовый скриптинг.
- Глава 16 рассказывает о *межсайтовой подделке запросов* (cross-site request forgery – CSRF). В этой главе затрагиваются детали из нескольких предыдущих плюс обсуждается, как правильно использовать архитектурный стиль REST.
- Глава 17 описывает *политику одинакового источника* (same-origin policy) и объясняет, для чего нужно временами разрешать *использование ресурсов между разными источниками* (cross-origin resource sharing – CORS).
- Глава 18 завершает книгу рассказом о *кликджекинге* (clickjacking). Кроме того, в ней приводятся веб-ресурсы, которые стоит временами посматривать, чтобы ваши знания не устаревали.

О фрагментах кода

В этой книге приводится много примеров исходных кодов, как в виде отдельных пронумерованных листингов, так и прямо в тексте. В любом случае программный код для наглядности напечатан `вот таким моноширинным шрифтом`. Иногда код может быть **выделен жирным**. Таким образом выделено то, что поменялось по сравнению с предыдущим вариантом. Например, когда в существующей строке кода добавлен новый функционал.

Чаще всего исходный код был перекомпонован, чтобы вместить его на страницы книги. Были добавлены переносы строк и сокраще-

ны отступы. В редких случаях этого оказалось недостаточно. Если строка кода продолжается ниже, это обозначается символом ➥. Кроме того, если пояснение к коду дается в тексте, то из него удалены комментарии.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге, – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Manning Publications очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Деннис Бирн (Dennis Byrne) работает в команде IT-архитекторов сервиса 23andMe. В задачи команды входит защита медицинских данных более десятка миллионов клиентов. Ранее Деннис трудился разработчиком в LinkedIn. Он бодибилдер, а также кейв-дайвер под началом Global Underwater Explorers. Живет в Кремниевой долине, но вырос и отучился далеко от этих краев – на Аляске.

Об иллюстрации на обложке

Персонаж на обложке озаглавлен Homme Touralinze, или «сибиряк из Тюмени». Это иллюстрация из коллекции костюмов разных стран Жака Грассе де Сен-Совера (Jacques Grasset de Saint-Sauveur, 1751–1810). Под заголовком Costumes de Différents Pays этот сборник опубликован во Франции в 1797 году. Все изображения нарисованы и раскрашены вручную с большим вниманием к деталям. Роскошная палитра иллюстраций Сен-Совера не дает забыть, насколько же разной культурой обладал каждый уголок мира всего 200 лет назад. Расстояния значили больше, и речь в одном месте была совсем не похожа на речь в другом. Разные языки, различные диалекты. Что на городской улице, что на деревенской дороге, по одежде легко можно было сказать, где человек живет, чем занимается и какое у него положение в обществе.

Манеры одеваться со временем изменились, и некогда значительная разница от области к области сошла на нет. Теперь трудно отличить друг от друга жителей различных континентов, не то что городов, краев или стран. Похоже, что мы обменяли культурное разнообразие на разностороннюю личную жизнь и уж точно на динамичную и разномастную жизнь с современными технологиями.

Нынче едва отличишь одну техническую книгу от другой. Издательство Manning же напоминает обложками своих книг о бесконечном многообразии бытия во всех уголках мира, бывшем два века назад. Находчивый и предприимчивый мир информационных технологий заслуживает этого.

1 О защите в деталях

Темы этой главы:

- где находится пространство для атаки;
- что такое глубокая оборона;
- стандарты и проверенные приемы защиты;
- средства защиты в среде Python.

Никогда ранее нам еще не приходилось настолько доверять компаниям хранение вашей личной информации. Увы, некоторые из них уже сдали ее с потрохами взломщикам. Если вам трудно в это поверить, зайдите на https://haveibeenpwned.com. Этот сайт позволяет узнать по адресу электронной почты, есть ли ваши персональные данные среди миллиардов утекших учетных записей. Со временем эта база только растет. Если ваших учетных данных там еще нет – стоит благодарить специалистов по обеспечению безопасности сервисов, которыми вы пользуетесь.

Раз вы открыли эту книгу, вас наверняка интересует безопасность приложений не только как пользователя. Как и я, вы не только хотите быть клиентом защищенных сервисов – вы хотите их создавать. Большинство программистов признают важность написания безопасного кода, но у них не всегда есть понимание, как этого добиться. Эта книга дает крепкий базис для такого понимания.

Безопасность – это способность противостоять атакам. В этой главе подробно говорится о безопасности с ее лицевой стороны: как сервисы могут быть атакованы. Следующие главы рассказывают об обеспечении защиты изнутри с помощью инструментов, доступных в Python.

Для каждой атаки требуется место проникновения. Совокупность мест проникновения некого сервиса называется *пространством для атаки* (attack surface). За пространством для атаки у защищенного приложения находятся уровни обороны. Этот архитектурный прием называется *глубокой обороной* (defence in depth). Уровни обороны строятся на основе стандартов и проверенных приемов, что исключает очевидные дыры.

1.1 Пространство для атаки

Обеспечение безопасности данных когда-то было просто небольшим перечнем того, чего стоит придерживаться и чего стоит избегать. Сейчас же это объемная область знания. Что же сделало ее таковой? Обеспечение безопасности стало нетривиальной наукой потому, что сами атаки стали нетривиальными. Какими они только не бывают. Стоит хорошо в них разбираться, прежде чем писать безопасный код.

Как говорилось выше, для каждой атаки требуется место проникновения. Совокупность возможных мест проникновения составляет пространство для атаки вашего приложения. Для каждого сервиса это пространство свое.

Атаки, как и пространство для них, изменчивы. Взломщики со временем осваивают новые приемы. Регулярно обнаруживаются доселе неизвестные уязвимости. Именно поэтому охрана вашего пространства – это непрекращающийся процесс. Компания должна быть озабочена этим постоянно.

Местом проникновения может быть пользователь, сам сервис или сеть связи между ними. Если говорить о пользователях, то взломщик в некоторых случаях может найти себе жертву через электронную почту либо чат. Средство подобных атак – обманом заставить пользователя активировать вредоносное содержимое, которое эксплуатирует уязвимость. Среди таких атак можно перечислить следующие:

- непостоянный межсайтовый скриптинг (reflected cross-site scripting);
- социальная инженерия;
- межсайтовая подделка запросов;
- непроверенная переадресация (open redirect).

Но также и сам сервис может быть местом проникновения. Подобные атаки часто основываются на недостаточной проверке данных, поступаемых приложению. Классические примеры этого:

- внедрение в запрос к базе данных (SQL injection);
- удаленное исполнение кода;
- изменение HTTP-заголовка Host (Host header attack);
- отказ в обслуживании (denial of service).

Местами проникновения могут быть одновременно и пользователь, и сервис. Среди таких атак – хранимый межсайтовый скриптинг и кликджекинг.

Наконец, злоумышленник может воспользоваться сетью связи между пользователем и сервисом, в том числе промежуточными устройствами в сети, как местом проникновения. Среди таких атак – «человек посередине» (man-in-the-middle) и «попугай» (replay attack).

Эта книга учит вас обнаруживать подобные атаки и противостоять им. Некоторым из них посвящена целая глава, а межсайтовому скриптингу – даже две. Рисунок 1.1 иллюстрирует пространство для атаки типичного сервиса. Четыре злоумышленника одновременно прощупывают пространство, как показано пунктирными линиями. Пока что не погружайтесь в детали. Это всего лишь обзор того, что вас ожидает в нашей книге. После прочтения вам будет понятно, из чего состоит та или иная атака.

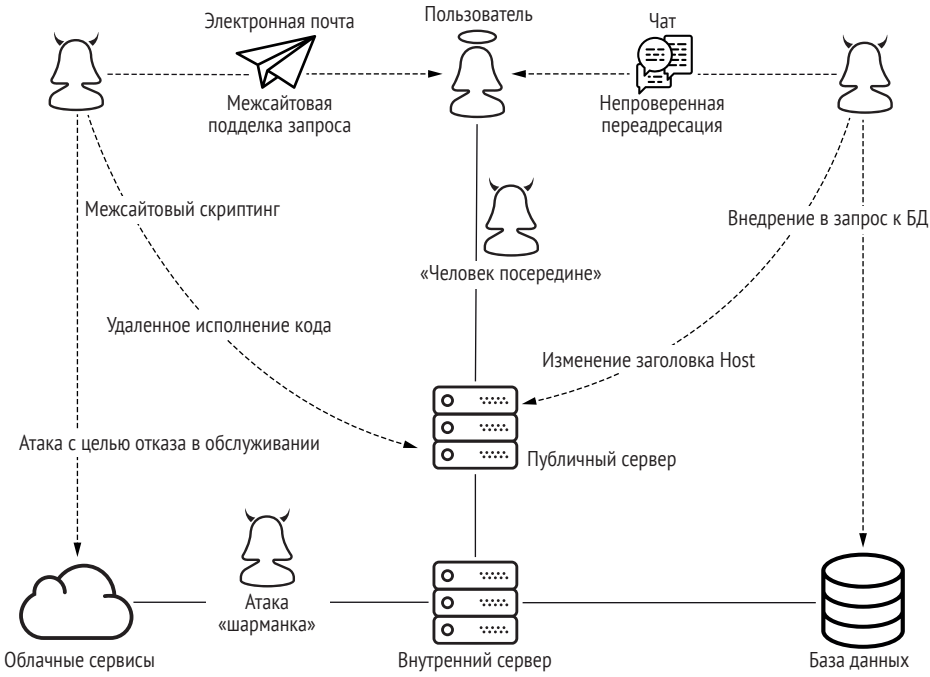

Рис. 1.1 Четыре злоумышленника используют места проникновения через пользователя, сервис и сеть связи

Под пространством для атаки у любого защищенного сервиса находятся уровни обороны. Одной защиты по периметру недостаточ-

но. Как упоминалось выше, подобный многоуровневый подход называется *глубокой обороной*.

1.2 Глубокая оборона

Этот подход зародился в Агентстве национальной безопасности США. Глубокая оборона подразумевает, что сервис должен противодействовать угрозам с помощью отдельных уровней. У каждого уровня две задачи: противостоять атакам и принимать удар на себя, если остальные уровни не справились с отражением атаки. Все потому, что не стоит класть яйца в одну корзину. Даже опытные программисты допускают ошибки, а новые уязвимости обнаруживаются постоянно.

Понятие «глубокая оборона» можно объяснить на таком примере. Представьте за́мок, у которого есть один уровень обороны – армия. Она непрестанно защищает замок от посягательств. Допустим, в 10 % случаев армия терпит поражение. Это весьма сильная армия, но королю не по себе и от 10%-ного риска. Что вас, что меня вряд ли бы устроил сервис, который пропускает 10 % атак. Наших пользователей – тоже.

У короля есть два способа снизить риск. Первый – усилить армию. Это возможно, но экономически неоправданно. Устранение оставшихся 10 % риска в разы затратнее, чем устранение первых 10 %. Вместо того чтобы усиливать армию, король решает добавить дополнительный уровень обороны: выкопать ров вокруг замка.

Насколько наличие рва снижает риски? Теперь, чтобы завоевать замок, нужно преодолеть и армию, и ров. Чтобы посчитать риски, королю понадобится обыкновенное умножение. Допустим, ров тоже не может сдержать 10 % атак. Итак, теперь захват замка возможен только в 10 % от 10 % случаев, что равняется 1 %. Представьте, насколько затратно может быть собрать армию, против которой устоит только 1 % врагов, по сравнению с тем, чтобы просто выкопать ров и залить его водой.

Как последнюю меру король возводит стену вокруг замка. Она тоже не выдержит натиск лишь 10 % атак. Теперь любая атака будет успешна только в 10 % от 10 % от 10 % случаев. Это 0,1 %.

В итоге подсчет выгоды от глубокой обороны сводится к умножению вероятностей. Добавление очередного уровня обороны всегда выгоднее шлифовки существующего. Концепция глубокой обороны признает, что стремиться к идеалу напрасно, и обращает это в достоинство.

Время показывает, какая реализация уровня обороны успешнее отражает атаки и пользуется большей популярностью. Способов выкопать тот же ров не так уж и много. Распространенная проблема рождает популярное решение. Специалисты замечают однообразную задачу, и экспериментальные приемы ее решения со временем

становятся стандартом. Стандарт в подробностях описывает шаблонную задачу и определяет ее решение.

1.2.1 Стандарты обеспечения защиты

Многие успешные стандарты обеспечения защиты были определены этими организациями: Национальный институт стандартов и технологий (National Institute of Standards and Technology – NIST), Инженерный совет Интернета (Internet Engineering Task Force – IETF), Консорциум Всемирной паутины (World Wide Web Consortium – W3C). Эта книга научит вас защищать сервисы, используя следующие стандарты:

- алгоритм симметричного шифрования Advanced Encryption Standard (AES);
- семейство криптографических хеш-функций Secure Hash Algorithm 2 (SHA-2);
- защищенный сетевой протокол Transport Layer Security (TLS);
- протокол предоставления прав доступа к разделенным ресурсам OAuth 2.0;
- протокол для использования ресурсов между разными источниками (CORS), применяющийся в веб-браузерах;
- стандарт политики защиты содержимого (CSP), который предотвращает выполнение некоторых атак в веб-браузере.

Зачем нужно создавать стандарты? Для того, чтобы программа, написанная в одной компании, могла взаимодействовать с программами от других разработчиков. Например, веб-сервер высылает любому браузеру один и тот же TLS-сертификат. У браузера же не возникнет проблем с обработкой TLS-сертификата от какого бы то ни было веб-сервера.

Кроме того, стандартизация позволяет переиспользовать код. Например, `oauthlib` – это стандартная реализация протокола OAuth. На этой библиотеке построены как Django OAuth Toolkit, так и `flask-oauthlib`. В итоге один и тот же код пригодился и сервисам на Django, и приложениям на Flask.

Буду откровенен, стандартизация – это не таблетка от всех недугов. Иногда уязвимость обнаруживается в стандарте, который используется десятки лет. В 2017 году исследователи объявили о взломе SHA-1 (https://shattered.io/). Это криптографическая хеш-функция, которую повсеместно использовали более 20 лет.

Иногда разработчики запаздывают с реализацией стандартов в своих продуктах. Реализация политик защиты содержимого в популярных веб-браузерах растянулась на годы. Но практически всегда установление стандартов идет во благо, и потому нельзя ими пренебрегать.

В дополнение к стандартам безопасности возникают также и проверенные приемы. Глубокая оборона – сама по себе проверенный

прием. Как и стандарты, проверенные приемы используются в разработке защищенных сервисов. Отличие проверенных приемов от стандартов – в том, что для приемов не существует нормативной документации.

1.2.2 Проверенные приемы

Проверенные приемы берутся не из технических документов. Они, как байки, передаются из уст в уста, в том числе через подобные книги. Это то, чего обязательно стоит придерживаться, и никто, кроме вас, этого не проконтролирует. Эта книга поможет как замечать использование, так и придерживаться этих проверенных приемов, а именно:

- шифрования хранимых и передаваемых данных;
- «не изобретай свое шифрование»;
- принципа наименьших привилегий.

Данные могут либо передаваться, либо обрабатываться, либо храниться. Когда специалист говорит о шифровании хранимых и передаваемых данных, имеется в виду необходимость шифровать данные всегда, когда они передаются между компьютерами либо записываются для хранения.

Когда речь идет о том, чтобы не изобретать свое собственное шифрование, эта речь – в целом о велосипедостроении в сфере безопасности. Смысл в том, чтобы использовать проверенное решение от умудренных опытом экспертов, а не пытаться сделать самому. Программисты полагаются на сторонние инструменты вовсе не из-за горящих сроков и не из желания писать меньше кода. Сторонний код испытан на прочность. Увы, большинство программистов приходят к пониманию этого лишь на своем горьком опыте. Вам же будет достаточно прочесть нашу книгу.

Принцип наименьших привилегий (principle of least priviledge – PLP) подразумевает, что пользователю либо системе разрешен доступ к достаточному, но минимально возможному набору полномочий. Этот принцип встретится в книге при обсуждении различных тем: разграничения прав пользователей, протоколов OAuth и CORS, и не только.

Рисунок 1.2 показывает, как стандарты безопасности и проверенные приемы сочетаются в типичном веб-сервисе.

Ни один уровень обороны не панацея. Ни один стандарт либо прием никогда не предотвратит инцидентов сам по себе. Поэтому книга и содержит в себе, как и типичная программа на Python, множество как стандартов, так и проверенных приемов. Каждая из глав предлагает наметки для очередного уровня обороны в вашем сервисе.

Стандарты и приемы могут описываться по-разному, но по сути своей каждый из них говорит об одних и тех же азах. Эти азы и есть тот самый базис, на котором выстраивается защита.