

Разработка учебного уязвимого веб-приложения на Flask для практикума по информационной безопасности

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

Статья описывает разработку воспроизводимой учебной песочницы по информационной безопасности на основе минимального уязвимого веб-приложения Flask. Рассмотрены задания по демонстрации и устранению SQL-инъекций, переходу от хранения паролей в открытом виде к хэшированию, защите сервиса комментариев и ограничению служебного доступа методом port knocking в Docker-изолированной среде. В результате было создано учебное веб-приложение, которое может быть имплементировано в образовательный процесс.

Ключевые слова: информационная безопасность, учебная песочница, уязвимое веб-приложение

Development of a training vulnerable web application on Flask for an information security workshop

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article describes the development of a reproducible training sandbox for information security based on the minimal vulnerable Flask web application. It covers tasks for demonstrating and eliminating SQL injections, transitioning from storing passwords in plain text to hashing, protecting the comment service, and restricting service access using port knocking in a Docker-isolated environment. The result is a training web application that can be implemented in the educational process.

Keywords: information security, training sandbox, vulnerable web application

1 Введение

1.1 Актуальность исследования

Подготовка начинающих специалистов по информационной безопасности требует доступных и воспроизводимых практикумов, где типовые уязвимости можно безопасно изучать и исправлять. Учебный тренажёр на Flask решает эту задачу за счёт минимального стека и сценариев с SQL-инъекцией, небезопасным хранением паролей и атаками на модуль

комментариев. Песочница позволяет отрабатывать проверку ввода, подготовленные запросы и хэширование паролей без риска для реальных систем, а port knocking демонстрирует базовую модель ограничения удалённого доступа. Такой формат закрывает разрыв между теорией и практикой на младших курсах и упрощает масштабирование заданий.

1.2 Обзор исследований

Корпус литературы задаёт методические и технологические ориентиры. Г. В. Беликов, И. Д. Крылов, В. А. Селищев систематизируют природу SQL-инъекции и способы противодействия, что напрямую отражается в заданиях по демонстрации обхода авторизации и последующему устранению уязвимости [1]. И. А. Лубкин описывает метрики оценки средств защиты от атак, основанных на возвратно-ориентированном программировании; подобные показатели уместны для формализации оценивания результатов практикума [2]. Г. А. Воробьёв, И. И. Павленко обосновывают формат ситуационного центра и интеграцию с профильными лабораториями, в рамках которого программная песочница может выступать одним из стендов [3]. И.В.Капгер, Е. Е. Журилова, А. А. Миронова конкретизируют требования к инфраструктуре и регламентам при развертывании учебного стенда, подчёркивая значимость пошаговых инструкций; эти элементы релевантны и для внедрения песочницы на Flask (Docker-хосты) [4].

1.3 Цель исследования

Цель исследования - реализовать учебную песочницу по информационной безопасности на основе минимального веб-приложения, позволяющую выявлять и устранять типовые уязвимости веб-систем.

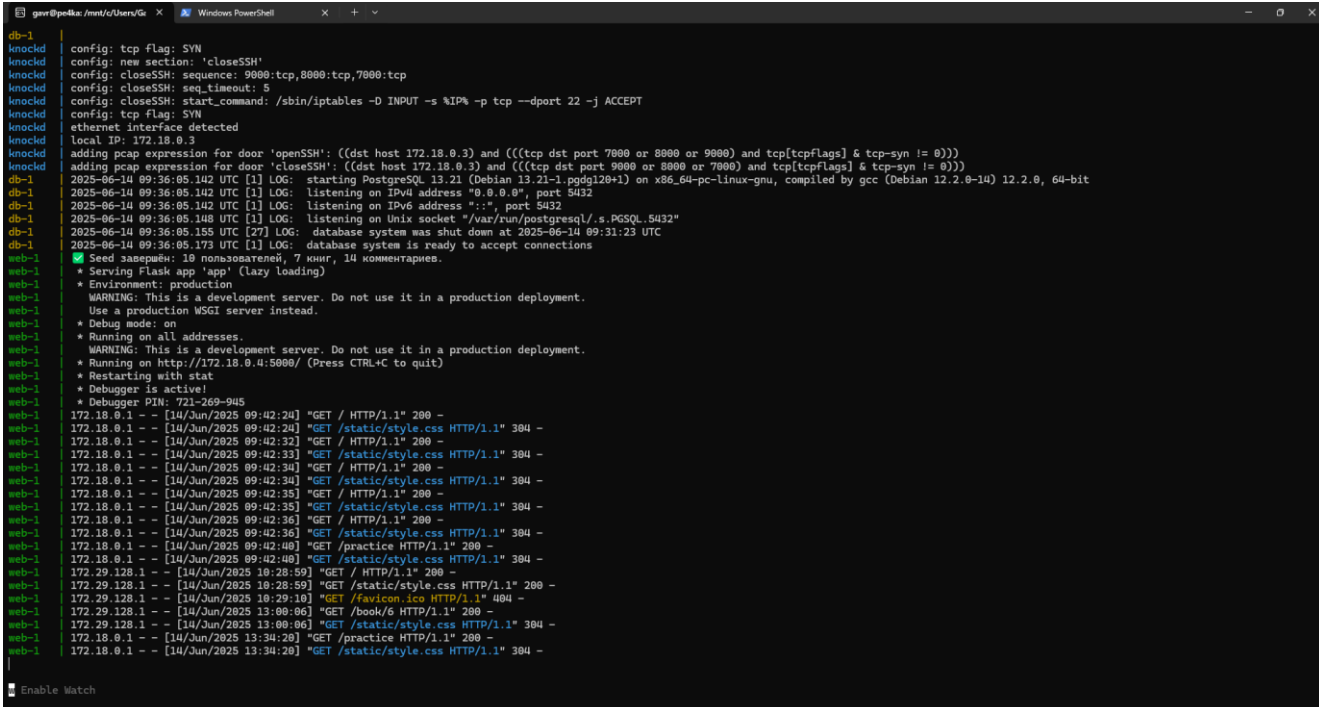
1.4 Постановка задачи

В ходе работы были сформулированы требования к учебному объекту как к контролируемой среде эксперимента, определены целевые сценарии уязвимостей и их наблюдаемые последствия, спроектированы архитектурный каркас приложения и границы тестовой среды, обеспечена воспроизводимость развёртывания и сбрасываемость состояния между попытками. Были заданы критерии наблюдаемости, включающие фиксацию проявлений уязвимостей и журналирование этапов атак и исправлений, описаны процедуры корректирующих изменений и соответствующие критерии приёмки «до/после», подготовлены методические материалы.

2 Методы исследования

В качестве серверного каркаса использован Flask, поскольку он базируется на широко изучаемом студентами языке и обеспечивает прозрачное сопоставление маршрутов с точками контроля, что облегчает выделение «зон атаки» и их изолированное исправление. Модульная организация позволила развести административные функции, основной интерфейс и комментарии, тем самым упростив таргетирование сценариев

компрометации и разграничение прав. Для хранения данных применена реляционная СУБД с начальным набором, что обеспечило воспроизводимые условия и возможность демонстрации инъекций на уязвимой версии запросов с последующей заменой на параметризованные обращения. Подсистема аутентификации изначально реализовывала хранение паролей в открытом виде для демонстрации уязвимости и была переведена на хэширование с индивидуальными солями (bcrypt) с контролируемым числом итераций, что позволило сопоставить поведение системы и следы в журналах до и после исправления. Модуль комментариев на уязвимой итерации допускал некорректную обработку пользовательского ввода, а в исправленной версии получил серверную валидацию, нормализацию и шаблонизацию с автоэкранированием, что исключило внедрение произвольного содержимого и облегчило автоматическую проверку. Изоляция учебной среды и недопущение воздействия на внешние ресурсы обеспечены контейнеризацией в Docker с фиксированными версиями зависимостей и замкнутой сетью; это позволило проводить эксперименты исключительно в тестовой среде и повторять результаты на типовом оборудовании (рис. 1).



```

db-1 config: tcp flag: SYN
knackd config: new section: 'closeSSH'
knackd config: closeSSH: sequence: 9000:tcp,8000:tcp,7000:tcp
knackd config: closeSSH: seqtimeout: 5
knackd config: closeSSH: start_command: /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
knackd config: tcp flag: SYN
knackd ethernet interface detected
knackd local IP: 172.18.0.3
knackd adding pcap expression for door 'openSSH': (((dst host 172.18.0.3) and (((tcp dst port 7000 or 8000 or 9000) and tcp[tcpflags] & tcp-syn != 0)))
knackd adding pcap expression for door 'closeSSH': (((dst host 172.18.0.3) and (((tcp dst port 9000 or 8000 or 7000) and tcp[tcpflags] & tcp-syn != 0)))
db-1 2025-06-14 09:36:05.142 UTC [1] LOG: starting PostgreSQL 13.21 (Debian 13.21-1.pgdql28+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
db-1 2025-06-14 09:36:05.142 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db-1 2025-06-14 09:36:05.142 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db-1 2025-06-14 09:36:05.148 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db-1 2025-06-14 09:36:05.155 UTC [27] LOG: database system was shut down at 2025-06-14 09:31:23 UTC
db-1 2025-06-14 09:36:05.173 UTC [1] LOG: database system is ready to accept connections
web-1 Seed завершен: 10 пользователей, 7 книг, 14 комментариев.
web-1 * Serving Flask app 'app' (lazy loading)
web-1 * Environment: production
web-1 WARNING: This is a development server. Do not use it in a production deployment.
web-1 Use a production WSGI server instead.
web-1 * Debug mode: on
web-1 * Running on all addresses.
web-1 WARNING: This is a development server. Do not use it in a production deployment.
web-1 * Running on http://172.18.0.4:5000/ (Press CTRL+C to quit)
web-1 * Restarting with stat
web-1 * Debugger is active!
web-1 * Debugger PIN: 721-269-945
web-1 172.18.0.1 - - [14/Jun/2025 09:42:24] "GET / HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:24] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:32] "GET / HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:33] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:34] "GET / HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:34] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:35] "GET / HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:35] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:36] "GET / HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:36] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:40] "GET /practice HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 09:42:40] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.128.1 - - [14/Jun/2025 18:28:59] "GET / HTTP/1.1" 200 -
web-1 172.128.1 - - [14/Jun/2025 18:28:59] "GET /static/style.css HTTP/1.1" 200 -
web-1 172.128.1 - - [14/Jun/2025 18:29:10] "GET /favicon.ico HTTP/1.1" 404 -
web-1 172.128.1 - - [14/Jun/2025 13:00:06] "GET /book/6 HTTP/1.1" 200 -
web-1 172.128.1 - - [14/Jun/2025 13:00:06] "GET /static/style.css HTTP/1.1" 304 -
web-1 172.18.0.1 - - [14/Jun/2025 13:34:20] "GET /practice HTTP/1.1" 200 -
web-1 172.18.0.1 - - [14/Jun/2025 13:34:20] "GET /static/style.css HTTP/1.1" 304 -

```

Рисунок 1 – Пример лога запуска контейнеров

Для ограничения служебного доступа к хосту использована схема port knocking на уровне фильтрации пакетов, что продемонстрировало базовую модель минимизации поверхности атаки без выдачи постоянных сетевых признаков сервиса. Наблюдаемость обеспечивалась структурированным логированием ключевых событий, маркерами успешности атак и исправлений, а также контролируемыми сценариями воспроизведения, что дало возможность количественно сравнить поведение системы на последовательных итерациях.

Веб-приложение оформлено как библиотека с книгами (рис. 2).

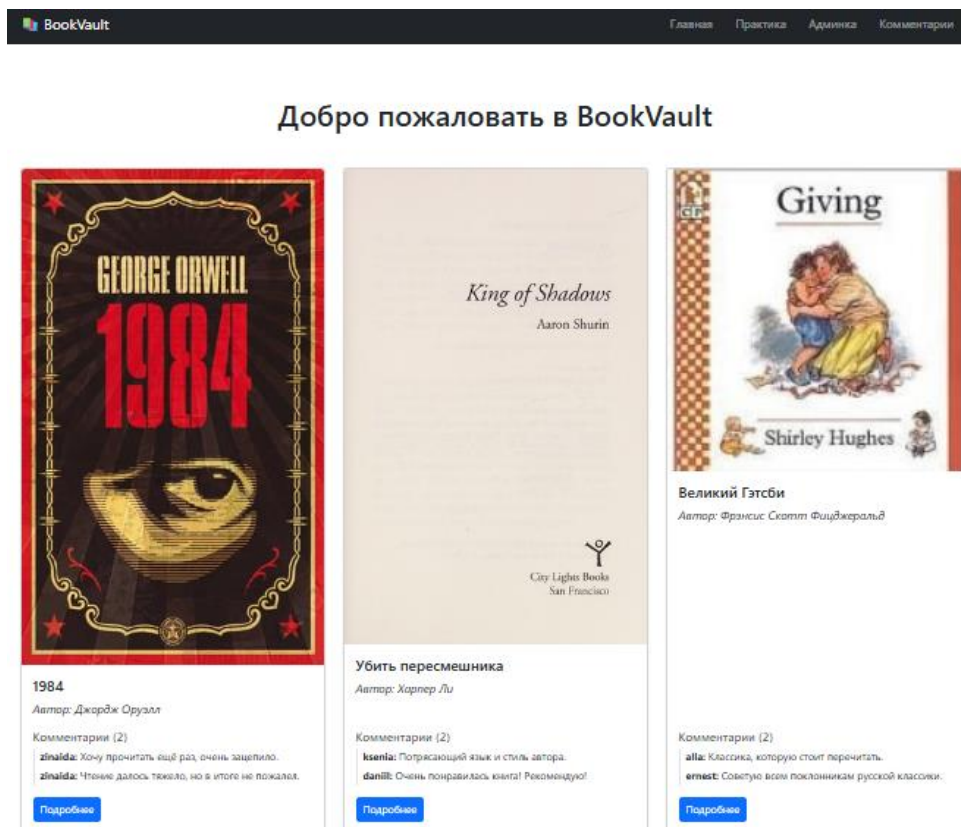


Рисунок 2 – Внешний вид главной страницы приложения

3 Результаты и обсуждения

На базовой итерации в учебном приложении реализована преднамеренно уязвимая аутентификация: параметры формы входа конкатенировались в текст запроса, что воспроизводимо позволяло обходить проверку и получать доступ к административной панели. Внутренние журналы подтверждали выполнение запросов с всегда-истинным предикатом и возврат первой записи из таблицы пользователей (рис. 3).

The screenshot shows a login form titled "Вход в админ-панель". It contains two input fields: "Имя пользователя" (Username) and "Пароль" (Password). The username field contains the text "' OR 1=1 --", which is a SQL injection payload. The password field contains three dots "...". Below the fields is a blue button labeled "Войти" (Login).

Рисунок 3 – Ввод данных на странице входа в админ-панель

После замены конкатенации на параметризованные обращения и нормализации ввода повторные попытки обхода давали отказ в доступе, а в логах исчезли характерные сигнатуры некорректного SQL. Дополнительно был введён явный сценарий «контроль до/после»: вход с тем же тестовым полезным запросом, фиксирование кодов ответа и содержания страницы, что обеспечило воспроизводимую оценку устранения уязвимости.

Вторая итерация решала проблему хранения секретов: исходно пароли присутствовали в базе в открытом виде, что демонстрировалось прямым выбором из таблицы пользователей. После перехода к хранению в виде хеша с солью и изменению процедуры проверки на сравнение хеша с предоставленным паролем, эффект атаки через компрометацию дампа перестал воспроизводиться: даже при наличии содержимого таблицы восстановление исходных паролей стало практически неосуществимым в пределах учебного времени. Отдельно зафиксировано, что значения хеша для одного и того же пароля различаются из-за соли, а процедура аутентификации больше не зависит от прямого сравнения строк.

Третья итерация касалась сервиса комментариев. В начальной версии допускалась как подмена атрибута автора, так и вмешательство в структуру запросов при сохранении, что позволяло имитировать сообщения «от лица» существующих пользователей и провоцировать ошибки базы. После внедрения серверной валидации и привязки авторства к идентификатору аутентифицированного субъекта, а также перехода на параметризованные операции при вставке и выборке, воспроизведение подмены перестало удаваться, а потенциально опасные последовательности в контенте визуализировались как текст без влияния на структуру запроса. Наблюдаемость обеспечивалась через журналирование попыток добавления и фиксацию кодов ошибок базы на уязвимой итерации с последующим исчезновением этих записей после исправления.

Четвёртая итерация демонстрировала снижение видимости служебного доступа средствами port knocking. В исходном состоянии сетевой скан показал недоступность целевого порта, после корректной последовательности «стуков» фиксировалось кратковременное появление возможности установить соединение для сеанса администрирования и последующее закрытие по обратной последовательности. Эффект наблюдался как по результатам локальных попыток подключения, так и по диагностическим сообщениям системы фильтрации пакетов, что подтвердило корректность временного изменения правил. В учебном контейнере доступ к флагу стал возможен исключительно в окне после валидной последовательности, что обеспечило простой и проверяемый критерий успешности.

Суммарно предложенная песочница продемонстрировала воспроизводимость атак и их устранений на компактном стеке, а также позволила увязать каждое вмешательство с наблюдаемым техническим эффектом: отказом в обходе аутентификации, невозможностью извлечения исходных паролей из содержимого таблицы, блокированием подмены авторства и управляемым окном доступности служебного порта. Ограничения

модели осознанно приняты: работа ведётся только в изолированной среде, покрывается ограниченный класс уязвимостей, а результаты оцениваются по поведенческим критериям системы и журналов. Этого достаточно для практикума начального уровня, где ключевая цель – формирование устойчивых навыков безопасной обработки ввода, хранения секретов и минимизации поверхности атаки в условиях, не представляющих риска для внешней инфраструктуры.

Ознакомиться с обучающим приложением можно в GitHub репозитории [5].

4 Выводы

Создана учебная песочница, на которой в контролируемой среде воспроизведены и устранены типовые угрозы безопасности веб-приложений.

Библиографический список

1. Беликов Г. В., Крылов И. Д., Селищев В. А. SQL-инъекция как способ обхода авторизации // Известия Тульского государственного университета. Технические науки. 2021. №. 12. С. 217-221.
2. Лубкин И. А. Доклады Томского государственного университета систем управления и радиоэлектроники // Доклады Томского Государственного университета систем управления и радиоэлектроники Учредители: Томский государственный университет систем управления и радиоэлектроники. 2021. Т. 24. №. 4. С. 46-51.
3. Воробьев Г. А., Павленко И. И. Ситуационный центр в обучении техническим и гуманитарным аспектам информационной безопасности // Информационное противодействие угрозам терроризма. 2015. Т. 1. №. 25. С. 117-122.
4. Капгер И. В., Журилова Е. Е., Миронова А. А. О разработке учебно-лабораторного стенда для изучения аппаратного модуля доверенной загрузки «Аккорд» // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. 2016. №. 17. С. 131-142.
5. GitHub course_materials URL: https://github.com/Gavriilbolt/course_materials/tree/main/01_basic_info_security/Prac%231software (дата обращения: 03.09.2025).