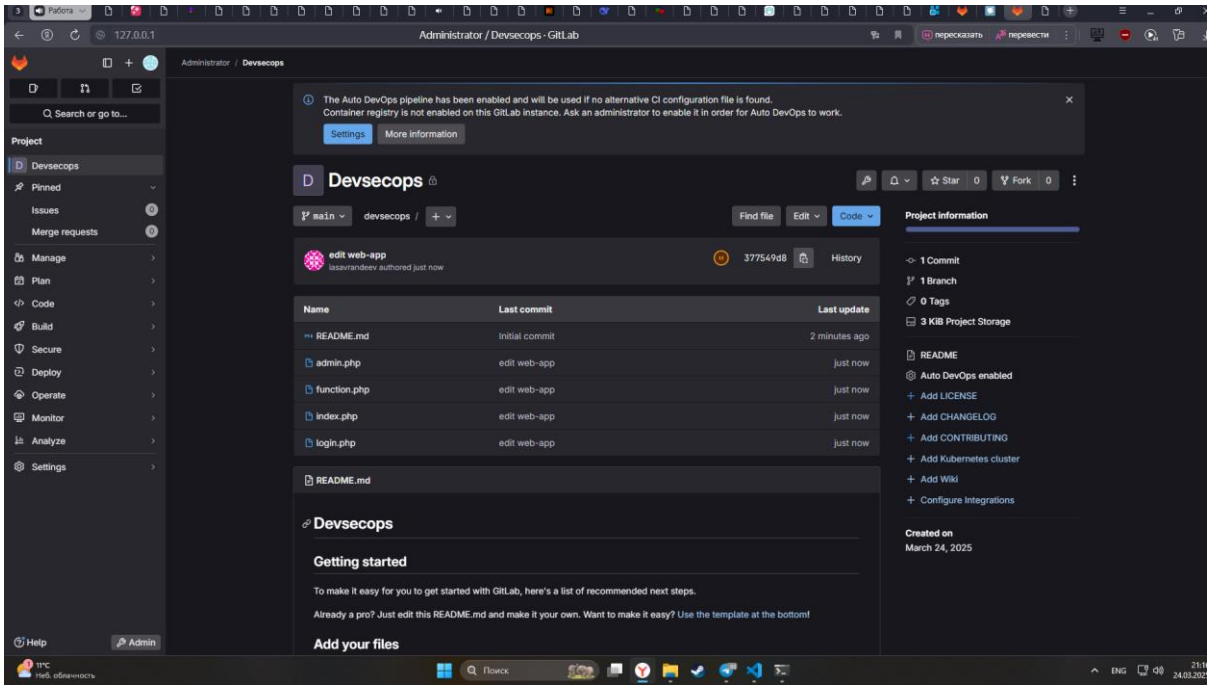
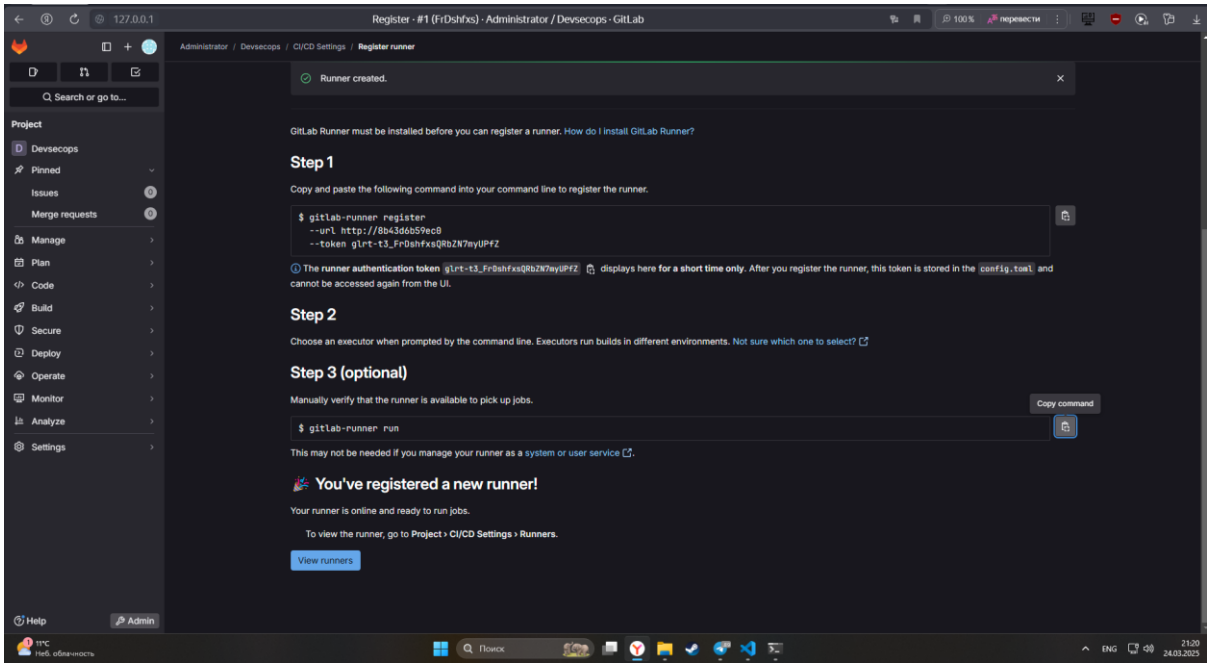
Итоговая работа

Поднимаем свой гитлаб и создаём репозиторий с приложением



Регистрируем ранер



Готовим пайплайн

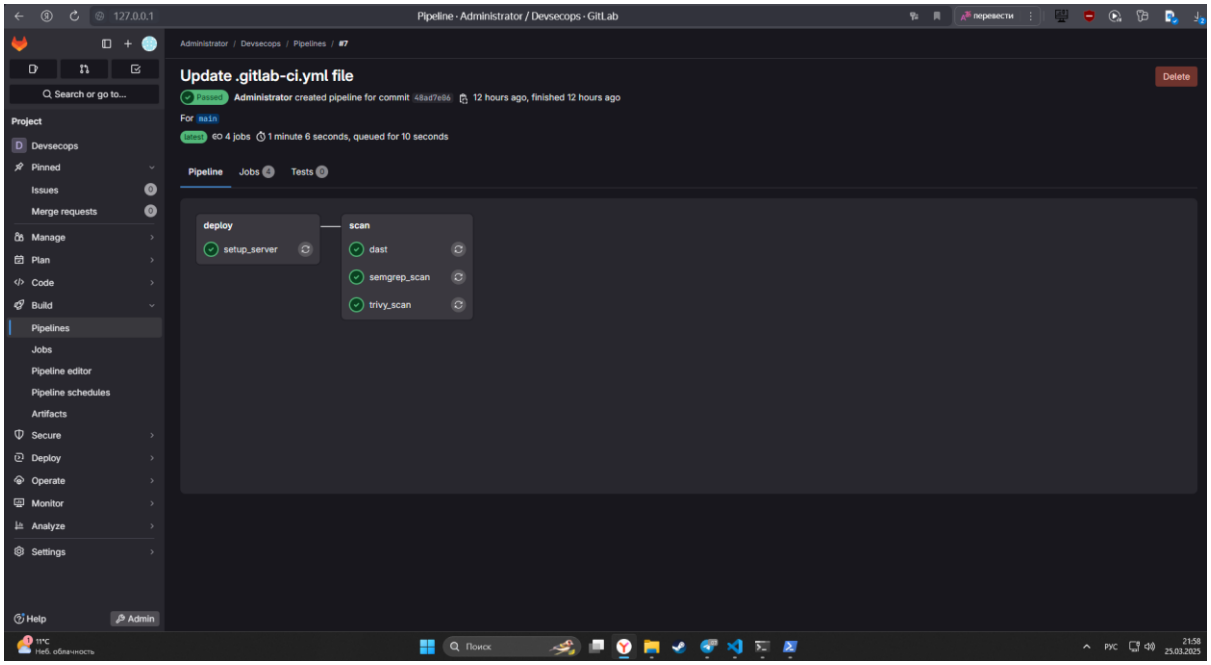
```
stages:
  - deploy
  - scan
```

```yaml
variables:
  APP_DIR: "/home/iasavrandeev/web-app"  # Путь к вашей PHP-application
  PORT: 8000

# Установка зависимостей и запуск сервера
setup_server:
  stage: deploy
  script:
    - cd $APP_DIR
    - nohup php -S localhost:$PORT > /dev/null 2>&1 &  # Запуск сервера в фоне
    - sleep 10  # Ждем инициализации сервера
    - curl -I http://localhost:$PORT || exit 1  # Проверяем доступность
  artifacts:
    paths:
      - $APP_DIR
  only:
    - main

# SAST: Semgrep
semgrep_scan:
  stage: scan
  script:
    - echo "Запуск Semgrep..."
    - semgrep --config auto --json -o semgrep-report.json $APP_DIR
  artifacts:
    paths:
      - semgrep-report.json
  only:
    - main

# SCA: Trivy
trivy_scan:
  stage: scan
  script:
    - echo "Запуск Trivy..."
    - trivy fs --severity HIGH,CRITICAL --exit-code 1 -o trivy-report.txt $APP_DIR
```

```
  artifacts:
    paths:
      - trivy-report.txt
  only:
    - main

# DAST: OWASP ZAP
dast:
  stage: scan
  script:
    - mkdir zap && chmod 777 zap
    - docker run --network="host" -v ./zap:/zap/wrk/ --rm zaproxy/zap-stable zap-baseline.py -t http://127.0.0.1:8000 -r report.html -I
  artifacts:
    paths:
      - zap/report.html
  only:
    - main
```

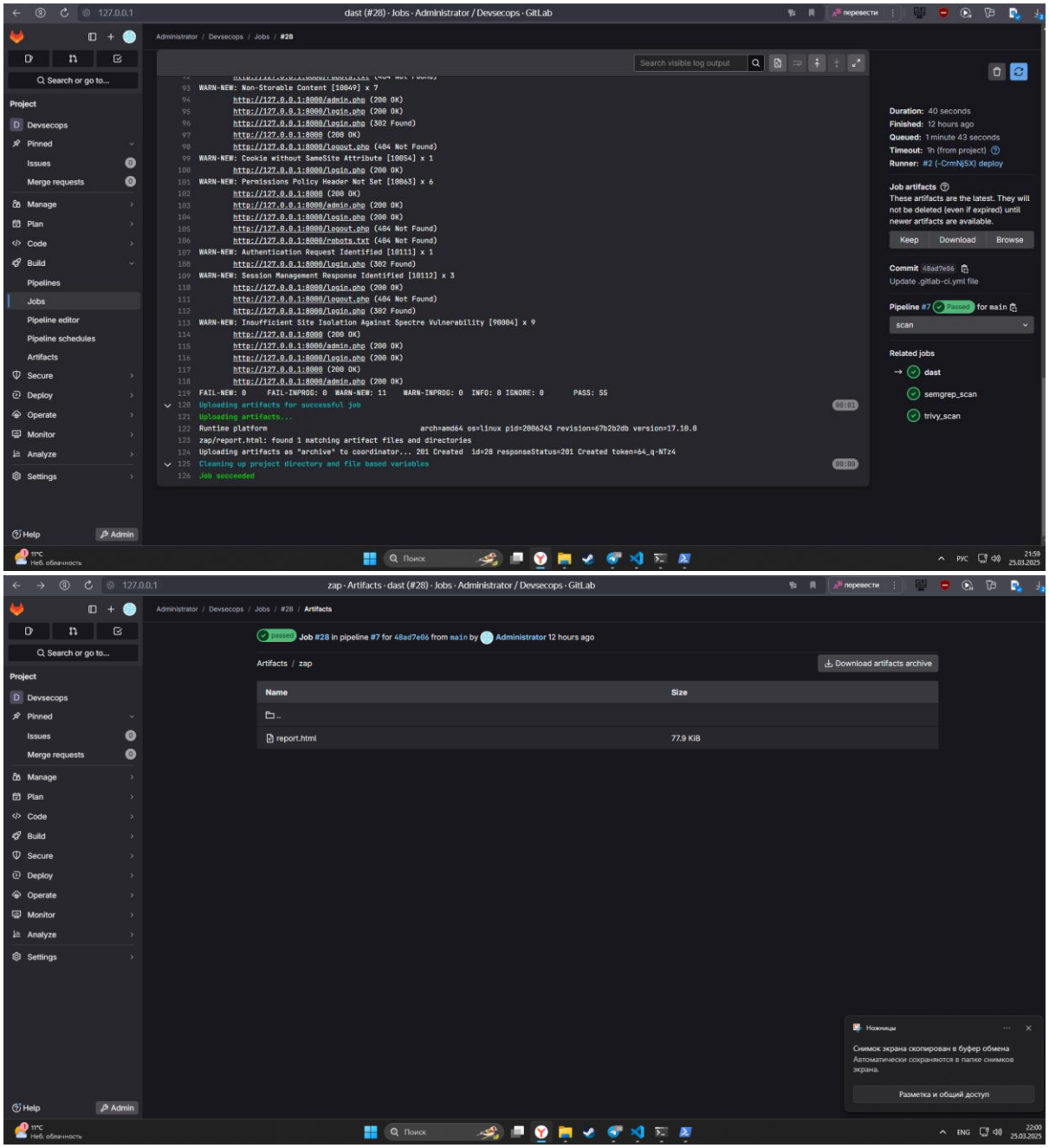
Так выглядит готовый рабочий пайплайн, теперь запускаем и проверяем работу и все отчёты

Пайплайн прошёл, всё выполнено

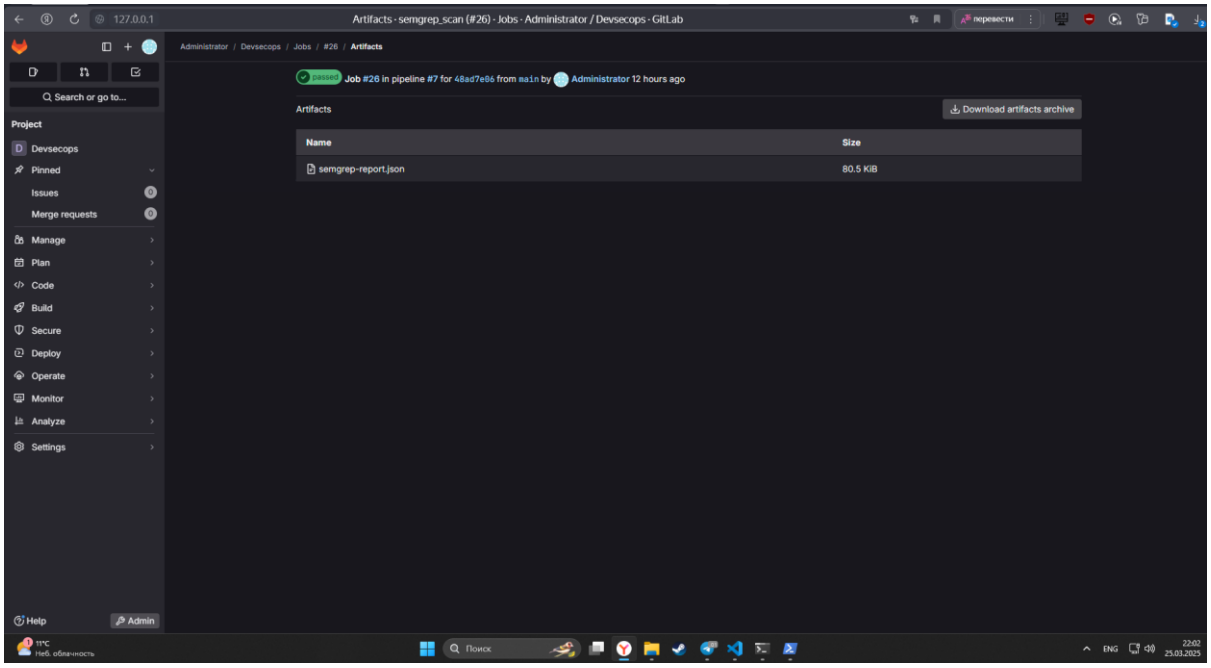


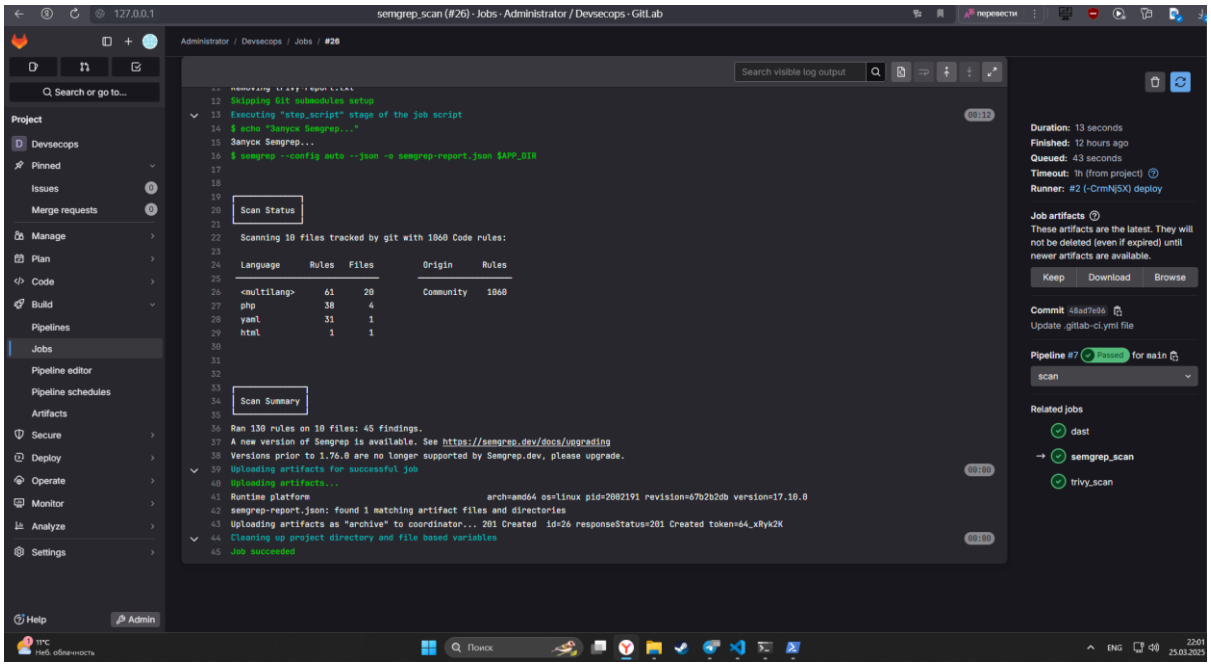
Проверяем на наличие отчётов

DAST

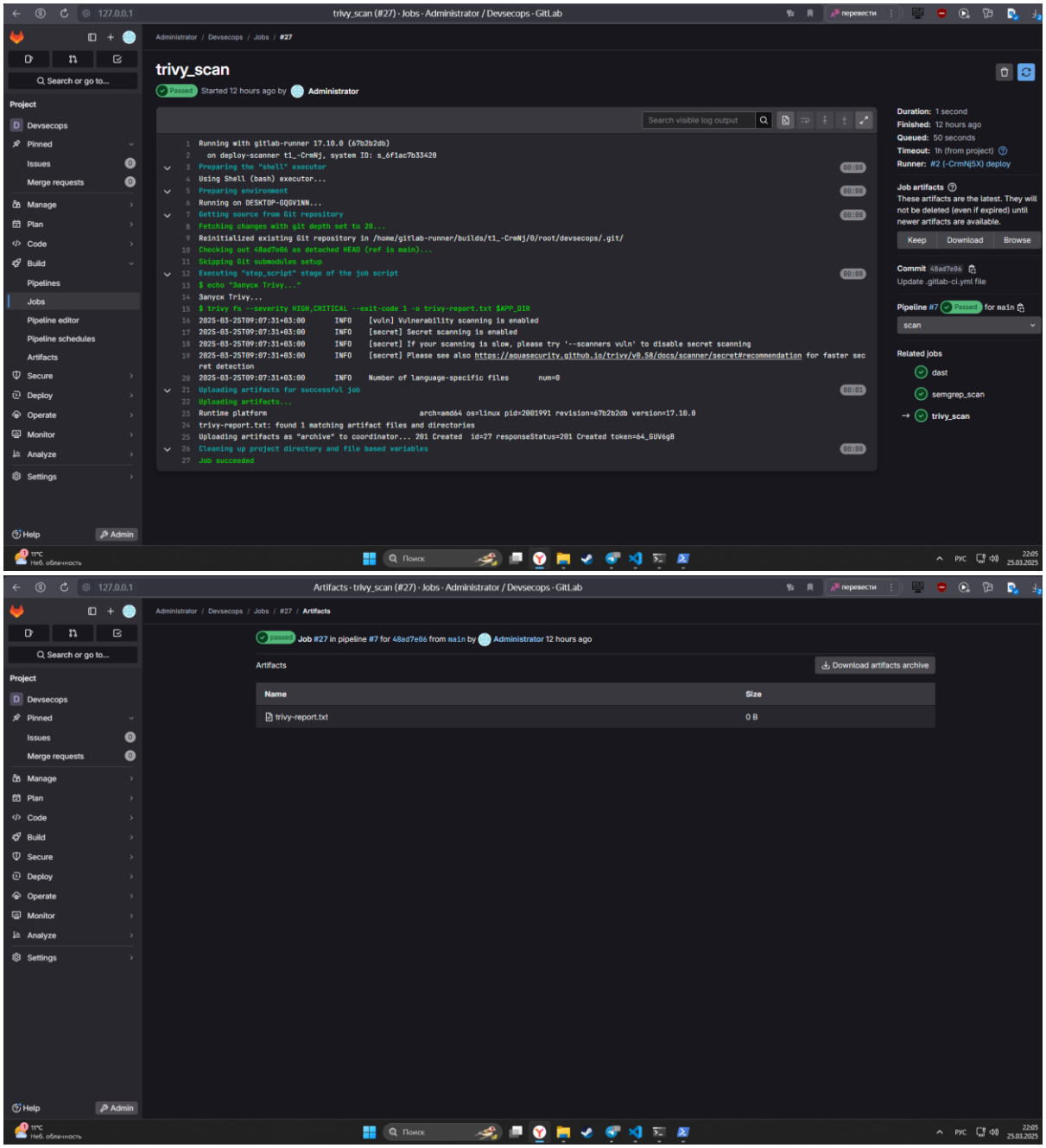


SAST – semgrep





SCA – trivy, сканирование прошло, но из-за отсутствия ”профильных” файлов в проекте, отчёт пустой, что является нормой и ничего плохого в этом нет.



Перейдём к разбору уязвимостей

Отчёт по результатам сканирования Semgrep

1. Уязвимость: Server-Side Request Forgery (SSRF)

- Файл: admin.php (строка 11)
- Описание: Использование пользовательского ввода ($_GET) для формирования имени файла в file_get_contents($file). Злоумышленник может указать произвольный путь или URL, что приведёт к несанкционированным запросам.

- Рекомендации:
    - Валидируйте и ограничивайте допустимые пути/домены.
    - Используйте белые списки разрешённых значений.
    - Рассмотрите замену на безопасные методы (например, загрузку файлов через контролируемые механизмы).

2. Уязвимость: Command Injection

- Файлы:
    - function.php (строка 4): Использование shell_exec($command) с неконтролируемым входом.
    - function.php (строка 10): Вызов executeCommand($userInput) с прямым использованием пользовательских данных.
- Описание: Пользовательский ввод передаётся в системные команды без санации, что позволяет выполнить произвольный код.
- Рекомендации:
    - Избегайте использования функций вроде shell_exec.
    - Если необходимо, экранируйте аргументы с помощью escapeshellarg().
    - Используйте безопасные альтернативы (например, встроенные функции PHP для работы с файлами).

3. Уязвимость: SQL Injection

- Файл: login.php (строка 9)
- Описание: Прямая подстановка переменных $username и $password в SQL-запрос, что позволяет злоумышленнику модифицировать логику запроса.
- Рекомендации:
    - Перейдите на подготовленные запросы (Prepared Statements) с использованием PDO или MySQLi.
    - Внедрите ORM-библиотеки для автоматической санации данных.

4. Уязвимость: Plaintext HTTP Links

- Файлы: Множество ссылок в report.html (например, строки 386, 417, 448 и др.).
- Описание: Использование HTTP вместо HTTPS подвергает данные риску перехвата.
- Рекомендации:
    - Замените все ссылки на HTTPS-версии.
    - Настройте сервер для принудительного использования HTTPS (редирект с HTTP).

5. Прочие замечания

- Файлы с Zone.Identifier: В списке сканируемых путей присутствуют файлы вида admin.php:Zone.Identifier. Это артефакты Windows, указывающие на происхождение файлов из ненадёжных источников (например, загрузок из интернета). Удалите их из production-окружения.

Итог

Инструмент: Semgrep (Community rules)

Критичность:

- ERROR: SQL Injection, Command Injection (требуют немедленного исправления).
- WARNING: SSRF, HTTP-ссылки (рекомендуется исправить в ближайшее время).

Перейдём к разбору owasp zap

Отчет о сканировании OWASP ZAP

Уязвимости уровня Medium

1. Content Security Policy (CSP) Header Not Set
    a. Описание: Отсутствует заголовок CSP, что повышает риск XSS-атак и инъекций.

    b. Затронутые URL: Главная страница, /admin.php, /login.php, /logout.php, /robots.txt.
    c. Рекомендации: Добавить заголовок Content-Security-Policy с ограничениями на источники скриптов, стилей и других ресурсов.
2. Missing Anti-clickjacking Header
    a. Описание: Отсутствуют заголовки X-Frame-Options или frame-ancestors в CSP, что позволяет встраивать страницы в <iframe>.
    b. Затронутые URL: Главная страница, /admin.php.
    c. Рекомендации: Установить X-Frame-Options: DENY или добавить frame-ancestors 'none' в CSP.

Уязвимости уровня Low

3. Cookie No HttpOnly Flag
    a. Описание: Куки PHPSESSID не помечены как HttpOnly, что делает их доступными для JavaScript.
    b. Затронутые URL: /login.php.
    c. Рекомендации: Добавить атрибут HttpOnly к куки.
4. Cookie without SameSite Attribute
    a. Описание: Куки PHPSESSID не имеют атрибута SameSite, что может привести к CSRF-атакам.
    b. Затронутые URL: /login.php.
    c. Рекомендации: Установить SameSite=Lax или SameSite=Strict.
5. Insufficient Site Isolation Against Spectre Vulnerability
    a. Описание: Отсутствуют заголовки Cross-Origin-Resource-Policy, Cross-Origin-Embedder-Policy, Cross-Origin-Opener-Policy.
    b. Затронутые URL: Главная страница, /admin.php, /login.php.
    c. Рекомендации: Добавить заголовки для изоляции ресурсов (например, Cross-Origin-Resource-Policy: same-origin).
6. Permissions Policy Header Not Set
    a. Описание: Отсутствует заголовок Permissions-Policy, ограничивающий доступ к функциям браузера (камера, микрофон и т.д.).
    b. Затронутые URL: Главная страница, /admin.php, /login.php, /logout.php, /robots.txt, /sitemap.xml.

    c. Рекомендации: Настроить заголовок с разрешением только необходимых функций.
7. Server Leaks Information via "X-Powered-By" Header
    a. Описание: Заголовок X-Powered-By: PHP/8.3.6 раскрывает информацию о сервере.
    b. Затронутые URL: Главная страница, /admin.php, /login.php.
    c. Рекомендации: Удалить заголовок в настройках сервера.
8. X-Content-Type-Options Header Missing
    a. Описание: Отсутствует заголовок X-Content-Type-Options: nosniff, что позволяет браузерам менять MIME-тип.
    b. Затронутые URL: Главная страница, /admin.php, /login.php.
    c. Рекомендации: Добавить заголовок X-Content-Type-Options: nosniff.

Информационные предупреждения

9. Authentication Request Identified
    a. Описание: Обнаружена форма аутентификации на /login.php (POST с параметрами username и password).
10. Non-Storable Content
    a. Описание: Ответы содержат Cache-Control: no-store, что предотвращает кэширование.
11. Session Management Response Identified
    a. Описание: Используется куки PHPSESSID для управления сессиями.
12. Storable and Cacheable Content
    a. Описание: Некоторые страницы кэшируются (например, главная, /logout.php).

Рекомендации по устранению

- Для CSP и заголовков безопасности:
    - Используйте генераторы CSP (например, CSP Evaluator).
    - Добавьте недостающие заголовки (X-Frame-Options, X-Content-Type-Options, Permissions-Policy).
- Для куки:
    - Установите HttpOnly, Secure и SameSite атрибуты.

- Для сервера:
    - Скрывайте версии ПО (отключите X-Powered-By).
    - Настройте политики изоляции ресурсов (Cross-Origin-*).