Труды МАИ. 2025. № 143
Trudy MAI. 2025. No. 143. (In Russ.)

Научная статья
УДК 004.056.52
URL: https://trudymai.ru/published.php?ID=185660
EDN: https://www.elibrary.ru/YVCJTY

# МОДЕЛЬ СИСТЕМЫ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВАНИИ МОНИТОРИНГА ИНЦИДЕНТОВ

**Павел Вадимович Пилькевич[1], Александр Геннадьевич Спеваков[2]✉,**

**Игорь Владимирович Калуцкий[3]**

[1,2,3]Московский политехнический университет,

Москва, Россия

[1]pavel.piksel2012@mail.ru

[2]aspev@yandex.ru✉

[3]kaluckiy_igor@yandex.ru

***Аннотация.*** В статье предложена модель доставки и развёртывания новых элементов программного обеспечения, которая заключается в предварительном анализе всех событий в исходной итерации разработки с целью принятия решения системой мониторинга о возможности внедрения разработанных элементов в инфраструктуре. Проведён анализ имеющихся технологических стеков, в рамках которого был определён наиболее подходящий для решения данной задачи. Проанализированы аспекты производительности предлагаемой модели и оценены риски и преимущества её внедрения.

***Ключевые слова:*** информационная безопасность, цикл разработки ПО, DevSecOps, мониторинг безопасной разработки, SIEM-системы, обнаружение инцидентов DevSecOps

***Для цитирования:*** Пилькевич П.В., Спеваков А.Г., Калуцкий И.В. Модель системы принятия решений на основании мониторинга инцидентов // Труды МАИ. 2025. № 143. URL:

Original article

# ARCHITECTURE OF A DECISION-MAKING SYSTEM BASED ON INCIDENT MONITORING

**Pavel V. Pilkevich[1], Alexander G. Spevakov[2]✉, Igor V. Kaluckiy[3]**

[1,2,3]Moscow Polytechnic University,

Moscow, Russia

[1]pavel.piksel2012@mail.ru

[2]aspev@yandex.ru✉

[3]kaluckiy_igor@yandex.ru

***Abstract.*** Within the framework of this article, a new approach to the delivery and deployment of new software elements was proposed, which consists in a preliminary analysis by the monitoring system of all events within the analyzed development iteration in order to make a decision by the monitoring system on whether it is possible to deploy the developed elements in the infrastructure. The analysis of the available technological stacks was carried out, within the framework of which the most suitable one for solving this problem was determined. The performance aspects of the proposed methodology are

analyzed and the risks and benefits of its implementation are assessed. The article examines the architecture of the incident monitoring system and decision-making on the deployment of a new productive software version based on the proven security of the deployed development. The scientific novelty lies in the application of a new method of managing the software delivery and development process, which prevents unsafe code elements from entering the target functioning infrastructure. The practical value lies in creating a secure infrastructure for the continuous development and analysis of unsafe development incidents with an integrated monitoring system in it. A comparative analysis of application types was carried out to determine the most relevant groups of programming languages that are prone to leaving confidential information in configuration files, and, as a result, require the creation of special analyzers to suit their specifics. Web applications were chosen as the most vulnerable type of applications for the described threat, since they are susceptible to the presence of configuration files with confidential credentials in them, as well as, in the case of unsafe development, the presence of static variables with confidential information right inside the program code, which causes critical vulnerabilities of such applications. A regression study was conducted, during which a strong correlation was established between the number of web application files and the number of files with potential critical information. The conducted statistical study allows us to conclude that there is one group of programming languages that have the largest number of dangerous files, which makes it most relevant to develop specific analyzers specifically for this group, which includes Java and Ruby languages. Based on the data obtained, a rule for detecting an incident of leaving confidential information for a group of Java and Ruby languages has been developed, taking

into account the specifics of web application configurations in these programming languages. Password entry fields, root fields, and data entry fields from administrative accounts are monitored. The developed module was applied to a labeled sample of web application code from open sources grouped by programming language to verify the correctness of detecting the threat of leaving confidential information in the software code as part of the software development cycle.

***Keywords:*** cybersecurity, software development life cycle, DevSecOps, vulnerability assessment, vulnerability management, SIEM

***For citation:*** Pilkevich P.V., Spevakov A.G., Kaluckiy I.V. Architecture of a decision-making system based on incident monitoring. *Trudy MAI*. 2025. No. 143. (In Russ.). URL:

## 1. Введение

При проектировании приложений для различных сфер критической информационной инфраструктуры одним из основных критериев является безопасность разработки и внедрения программного обеспечения [1]. Согласно Федеральному закону №187 и Федеральной системе обеспечения авиационной безопасности, определена необходимость реализации надлежащих мер защиты информации в информационных системах, используемых для целей гражданской авиации [2]. Воздушный транспорт относится к одной из сфер деятельности в области обеспечения безопасности критической информационной инфраструктуры Российской Федерации, все информационные системы, используемые в данной сфере, должны устойчиво функционировать при проведении в отношении их

компьютерных атак [3]. В информационной инфраструктуре, защите подлежат не только процессы хранения и передачи данных, но и семантическое их наполнение. Решению задач безопасной разработки и доставки программного обеспечения посвящены работы [4-8], но анализ результатов исследований, представленных в публикациях, показал, отсутствие средств мониторинга инцидентов цикла создания безопасного программного обеспечения, что усложняет разработку и повышает вероятность возникновения уязвимостей в развёртываемых сервисах [9].

Статья посвящена повышению безопасности разрабатываемых программных средств в информационных системах за счёт внедрения модели мониторинга инцидентов небезопасной разработки. В качестве объекта исследования рассматривается разрабатываемый исходный код программного обеспечения, имеющий в себе оставленные разработчиком конфиденциальные данные, в качестве предмета исследования - процесс обеспечения безопасной доставки разрабатываемого исходного кода в информационные системы.

## 2. Постановка задачи

Создание безопасного программного обеспечения отечественными компаниями на данный момент стало необходимостью. Соответствующие практики активно внедряются в существующие процессы создания программного обеспечения. Появление новых угроз информационной безопасности способствует появлению и использованию систем повышения эффективности разработки и обеспечения защищённости исходного кода создаваемых программных продуктов [10]. Происходит это в связи со следующими факторами:

- интенсивное повышение количества опасных уязвимостей программного обеспечения, способных нанести непоправимый ущерб информационной инфраструктуры предприятий [11], это выражается в регулярном обновлении международных и отечественных баз данных уязвимостей;
- появление новых зависимостей разрабатываемого кода программных продуктов от новых технологий, способных вызывать новые уязвимости или ошибки в рабочем прототипе [12];
- появление всё большего числа зависимостей разрабатываемого кода программных продуктов от проектов с открытым исходным кодом [13];
- активное увеличение количества хакерских атак на все этапы разработки и все элементы информационной инфраструктуры предприятий [14];
- развитие существующих и появление новых языков программирования [15, 16];
- актуализация вопроса защиты критической информационной инфраструктуры Российской Федерации вследствие роста востребованности обеспечения защищённости создаваемого программного обеспечения на базе отечественных компаний-разработчиков [17].

Обеспечение безопасного цикла разработки является задачей DevSecOps [18], который делится на множество этапов, каждый из которых подразумевает применение своего набора технологий для проведения работы над разработанным приложением. Так как цикл разработки полностью обеспечивается набором этапов DevSecOps [19], можно закономерно предположить, что результаты

функционирования каждого из этапов могут позволить сделать вывод о состоянии защищённости того программного обеспечения, которое обрабатывается в рамках цикла. Следовательно, становится актуальной задача отслеживания результатов функционирования технологий, применяемых на всех этапах DevSecOps.

Такая система мониторинга позволяет собирать информацию, в первую очередь, о неудачно завершённых CI/CD пайплайнах [20], что поможет формировать историю разработки безопасного программного обеспечения не с момента его успешного развёртывания, а с самого начала попыток его построения. Например, может существовать пайплайн статического анализа разрабатываемого кода на предмет оставления в нём конфиденциальной информации. Такое решение может быть полезно, когда приложение задействует конфигурационные файлы как часть исходного кода, в которых могут фиксироваться пароли или другая чувствительная информация, с помощью статических анализаторов можно предотвратить случайное попадание конфиденциальных данных в историю репозитория разрабатываемой программы. И когда разработанная программа успешно пройдёт статический анализатор и другие имеющиеся проверки/тесты в рамках остальных пайплайнов и этапов DevSecOps, система мониторинга будет иметь в себе полную историю инцидентов, произошедших от начала разработки вплоть до выпуска продуктовой версии решения.

Помимо представления и сбора информации о полной истории разработки продукта, система может предоставить возможность принятия итогового решения о том, является ли разработанное приложение безопасным и готовым для выпуска

продуктовой версии, таким образом, становится возможным внедрение системы принятия решений, которая автоматически сможет развернуть продуктовую версию в случае безопасности итогового продукта или же сформировать инцидент информационной безопасности при попытке развернуть небезопасную версию программного обеспечения в продуктовой среде. Такое поведение удается реализовать в случае внедрения логирования ключевых событий в пайплайнах на каждом из этапов DevSecOps, для формирования максимально подробной картины осуществляющегося цикла разработки, а также внедрении технических средств мониторинга и принятию решений, которые должны быть встроены в инфраструктуры разработки.

Новизна заключается в применении модели управления процессом доставки и разработки программного обеспечения, предотвращающего попадание небезопасных элементов кода в целевую функционирующую инфраструктуру.

## 3. Модель системы принятия решений на основании мониторинга инцидентов

Схема функционирования системы мониторинга и принятия решений представлена на рисунке 1, практическая ценность такой модели заключается в создании безопасной инфраструктуры непрерывной разработки и анализа инцидентов небезопасной разработки с интегрированной в нее системой мониторинга.

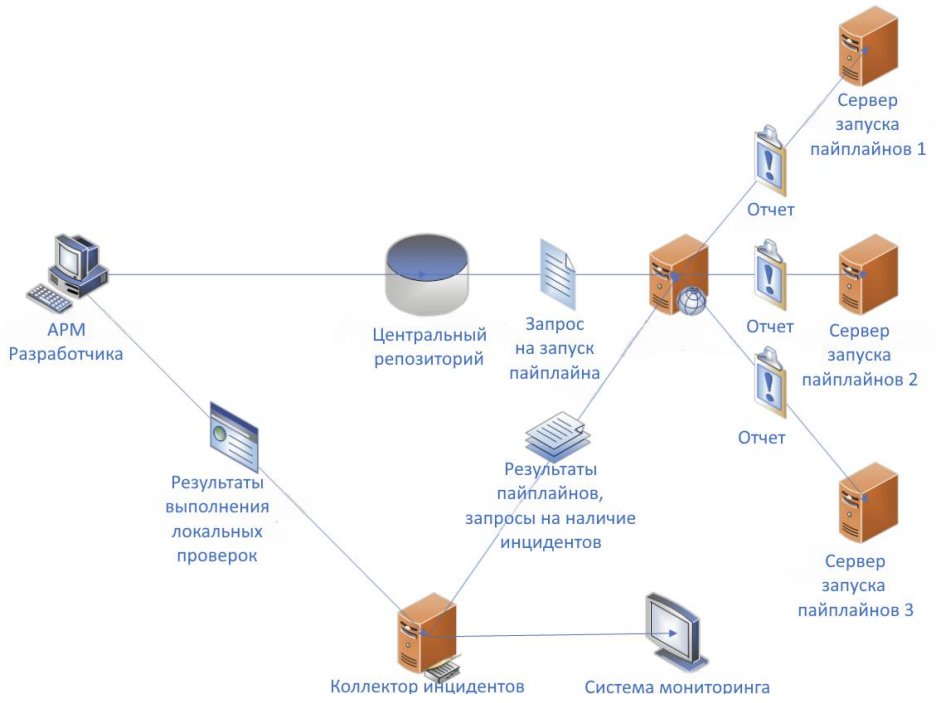


Рисунок 1 – Схема функционирования системы мониторинга и принятия решений

На схеме представлено условное взаимодействие ключевых компонентов предлагаемого решения. Пользователь, взаимодействующий с АРМ Разработчика, выполняет действия по разработке произвольного программного кода, цикл разработки которого будет анализироваться системой мониторинга. Во время попытки внести изменения в центральный репозиторий, выполняется ряд pre-commit проверок локального репозитория пользователя, результаты проверки отправляются на сервер, предназначенный для сбора мониторинговых событий. В случае неудачных проверок на стадии pre-commit, изменения не принимаются центральным репозиторием, а события о неуспешной попытке пользователя их внести фиксируются в системе. После успешной фиксации программного кода в центральном репозитории, он подлежит обработке следующих пайплайнов в рамках методологии DevSecOps. Центральный репозиторий запускает специальную машину (раннер), которая обращается к системе принятия решений, система принятия

решений, на основании имеющихся событий и инцидентов в системе мониторинга, делает вывод о допустимости запуска того или иного пайплайна, которые выполняются на других раннерах, передавая, в свою очередь, информацию о успешности или, наоборот, неуспешности отработки текущего пайплайна в систему мониторинга. Перед заключительным этапом сборки программного кода (выпуск продуктивной версии и развёртывание её в боевой среде), в предлагаемой системе мониторинга будут находиться события о полном цикле разработки текущего программного кода, который подлежит публикации в боевой среде. На основании данного пула информации, в случае непрохождения ряда ключевых проверок на безопасность итогового исходного кода, развёртывание продукта может быть отклонено системой принятия решений, что вызывает автоматическое создание инцидента информационной безопасности, подлежащего расследованию. У аналитиков, в таком случае, будет возможность исследовать причины срабатывания сигнализатора о небезопасности исходного кода с самого начала его формирования ещё на АРМ разработчика.

Предусмотреть при этом следует несколько ключевых для функционирования системы факторов:

1. Сетевую связанность пользователя с сервером для сборки событий для возможности их передачи в систему мониторинга. При этом следует предусмотреть возможность наличия внутреннего нарушителя, поэтому инцидент «намеренное отключение передачи событий с узла разработчика» необходимо также предусмотреть как один из ключевых

инцидентов информационной безопасности в предлагаемой инфраструктуре.

2. Унифицированный формат логирования при исполнении пайплайнов DevSecOps для облегчения построения правил обнаружения инцидентов и оптимизации производительности системы мониторинга в условиях разработки нескольких продуктов сразу.
3. Наличие специальных анализаторов программного кода в зависимости от типов приложений.

Был проведён сравнительный анализ типов приложений, для определения наиболее актуальных групп языков программирования, которые подвержены оставлению конфиденциальной информации в конфигурационных файлах, и, как следствие, требуют создание специальных анализаторов под свою специфику. В качестве наиболее уязвимого типа приложений под описанную угрозу выбраны web-приложения, так как они подвержены наличию в них конфигурационных файлов с конфиденциальными учётными данными, а также, в случае небезопасной разработки, наличию статических переменных с конфиденциальной информацией прямо внутри программного кода, что вызывает критические уязвимости подобных приложений.

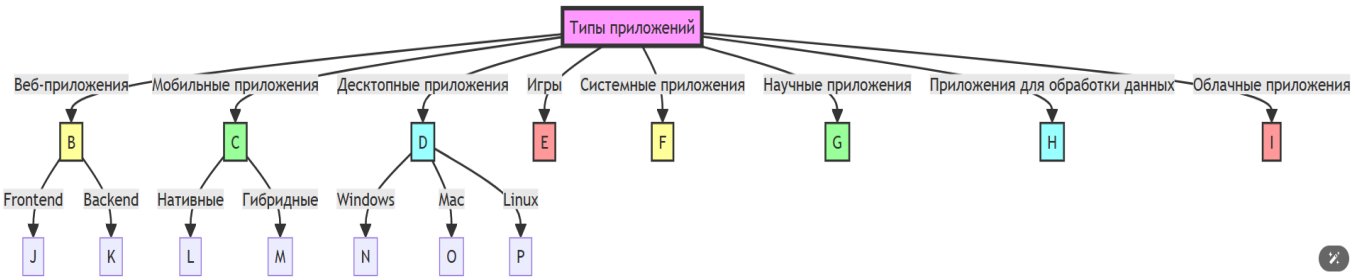


Рисунок 2 – Возможные типы разрабатываемых приложений

В рамках поставленной задачи подобрана первая выборка данных: количество файлов в анализируемом приложении на определённом языке программирования, в которых упоминается «web application». Таким образом, будет получен перечень конфигурационных файлов и файлов программного кода, которые явно отражают тип разрабатываемого приложения. Вторая выборка будет отражать количество файлов приложений на языках программирования, в которых упоминается «web application», а также в которых имеются ключевые слова (PASSWORD, ROOT, ADMINISTRATOR). Ключевые слова для второй выборки подобраны так, чтобы соответствовать поиску потенциально возможных угроз утечки конфиденциальных учётных данных. Для повышения демонстративности и получения большей информативности от датасета, также будет введён показатель доли потенциально опасных файлов для типа приложения на определённом языке программирования относительно всех файлов для данного типа и языка программирования.

Следует отметить, что выбранные языки хоть и имеют общую черту в виде возможности разработки веб-приложений на них, однако сильно отличаются друг от друга синтаксисом и иными характеристиками, что создаёт необходимость выделить среди данных языков один наиболее приоритетный для разработки специфического анализатора кода на безопасность содержимого файлов приложения. В результате проведения исследования и сбора данных, был получен датасет, представленный в таблице 1.

Таблица 1 – Датасет исследования

| № | Количество файлов приложений на языках программирования, в которых упоминается "web application" | Количество файлов приложений на языках программирования, в которых упоминается "web application", в которых имеются ключевые слова (PASSWORD, ROOT, ADMINISTRATOR) | Доля потенциально опасных файлов | Язык |
|---|---|---|---|---|
| 1 | 151000 | 64500 | 42,72% | Java |
| 2 | 89600 | 22900 | 25,56% | Python |
| 3 | 11000 | 2300 | 20,91% | Go |
| 4 | 7900 | 3100 | 39,24% | C++ |
| 5 | 124000 | 18600 | 15,00% | Js |
| 6 | 62500 | 20700 | 33,12% | php |
| 7 | 141000 | 125000 | 88,65% | ruby |
| 8 | 3100 | 952 | 30,71% | rust |
| 9 | 66000 | 10900 | 16,52% | C# |
| 10 | 832 | 213 | 25,60% | swift |
| 11 | 1500 | 438 | 29,20% | kotlin |
| 12 | 3600 | 1600 | 44,44% | perl |
| 13 | 1700 | 544 | 32,00% | scala |
| 14 | 2400 | 856 | 35,67% | C |
| 15 | 1200 | 231 | 19,25% | Haskell |
| 16 | 27000 | 10200 | 37,78% | TypeScript |

| № | Количество файлов приложений на языках программирования, в которых упоминается "web application" | Количество файлов приложений на языках программирования, в которых упоминается "web application", в которых имеются ключевые слова (PASSWORD, ROOT, ADMINISTRATOR) | Доля потенциально опасных файлов | Язык |
|---|---|---|---|---|
| 17 | 1900 | 1000 | 52,63% | Elixir |
| 18 | 2900 | 1300 | 44,83% | Dart |
| 19 | 824 | 466 | 56,55% | Lua |
| 20 | 724 | 144 | 19,89% | Clojure |

## 4. Экспериментальные исследования

На данном этапе исследования можно выявить предварительную статистическую зависимость двух выборок друг от друга, так как среднее арифметическое доли потенциально опасных файлов во всей совокупности файлов составляет 36%, что указывает на то, что, в среднем, при создании именно веб-приложения на любом языке программирования, более трети файлов разрабатываемого приложения будут потенциально опасны для утечки конфиденциальных учётных данных из них.

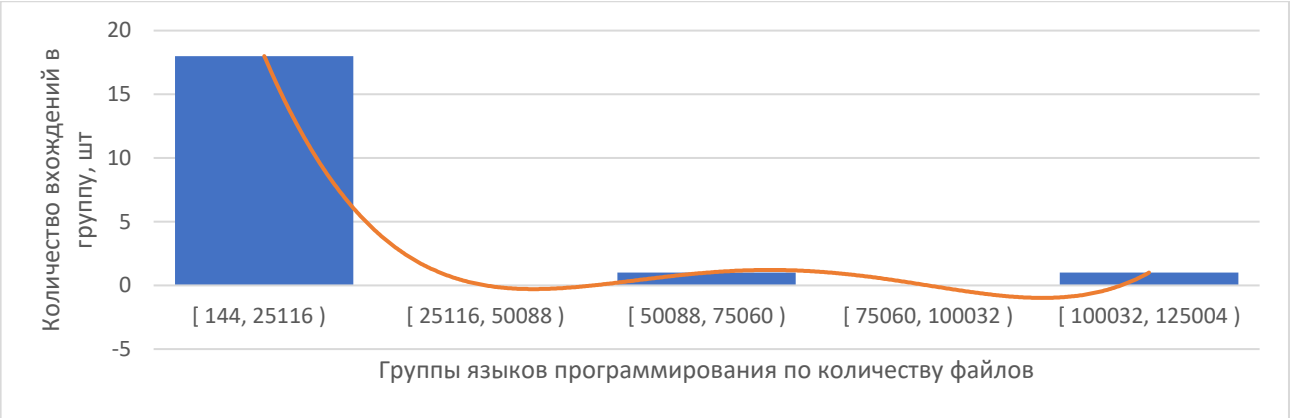


Рисунок 3 – Диаграмма частот

Проведено регрессионное исследование, для которого потребовалось рассчитать дисперсию между двумя анализируемыми выборками:

$$\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - x\text{ср}^2 \quad \sigma_y^2 = \frac{1}{n}\sum_{i=1}^{n} y_i^2 - y\text{ср}^2$$

Используя рассчитанные значения, был определён выборочный корреляционный момент:

$$\overline{\mu_{xy}} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})$$

Используя полученные значения, был вычислен выборочный коэффициент корреляции:

$$r_{xy} = \frac{\mu_{xy}}{\sigma_x \cdot \sigma_y} = 0{,}813357235$$

Взаимосвязь количества файлов веб приложений и количества файлов с потенциальной критической информацией является прямой, о чём свидетельствует регрессионное исследование.

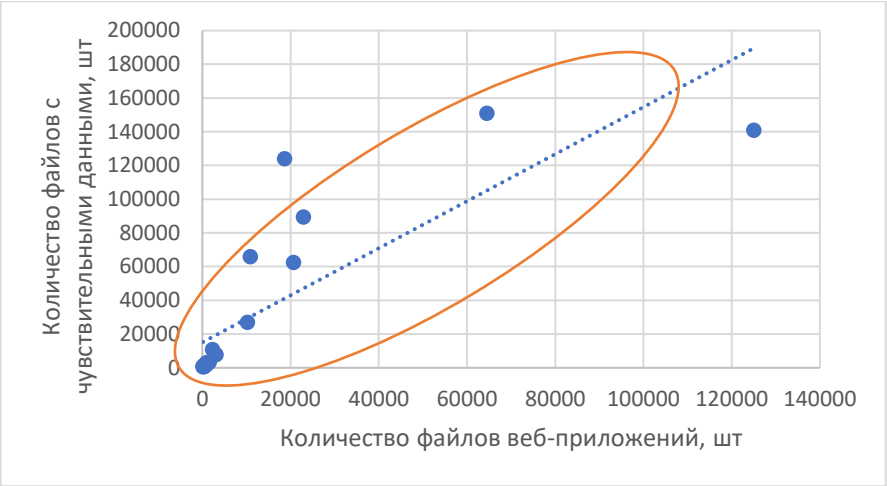


Рисунок 4 – Прямая регрессии

Статистическое исследование позволяет сделать вывод о присутствии одной группы языков программирования, которым свойственно наибольшее количество опасных файлов, что делает наиболее актуальной разработку специфических анализаторов именно для данной группы, которая включает в себя языки Java и Ruby.

Таблица 2 – Статистическое исследование

| N | Границы интервала | Частота | Накопленная частота | Частость | Накопленная частость | Среднее значение |
|---|---|---|---|---|---|---|
| i | Xhi-Xai | xi | ni | Ni | fi | Fi |
| 1 | [144, 25116) | 18 | 18 | 0,9 | 0,9 | 12630 |
| 2 | [25116, 50088) | 0 | 18 | 0 | 0,9 | 37602 |
| 3 | [50088, 75060) | 1 | 19 | 0,05 | 0,95 | 62574 |
| 4 | [75060, 100032) | 0 | 19 | 0 | 0,95 | 87546 |
| 5 | [100032, 125004) | 1 | 20 | 0,05 | 1 | 112518 |

В результате удалось разработать правило обнаружения инцидента оставления конфиденциальной информации для группы языков Java, Ruby с учётом особенности конфигураций web-приложений на данных языках программирования, рис. 5. Отслеживаются поля ввода пароля, root-поля, поля ввода данных от административных учётных записей.

```
[[rules]]
id = "custom-password"
description = "Custom password field"
regex = '''(?i)(?:administrator_login_password|password)(?:[0-9a-z\-_\t.]
{0,20})(?:[\s|']|[\s|"]){0,3}(?:=|>|:{1,3}=|\|\|:|<=|=>|:|\?=)(?:'|\"|\s|=|\x60){0,5}
("[a-z0-9=_\-]{8,20}")(?:['|\"|\n|\r|\s|\x60|;]|$)'''
keywords = [
    "administrator_login_password","password",
]
[[rules]]
id = "root"
description = "Root field"
keywords = [
    "root", "root_pass", "root_password"
]
[[rules]]
regex = "api"
max_severity = "medium"
[[rules]]
regex = "user"
max_severity = "low"
```

Рисунок 5 – Правило обнаружения инцидента оставления конфиденциальной информации

Для внедрения созданной логики на АРМ Разработчика понадобилось отредактировать модуль логирования решения gitleaks для интеграции формируемых логов с системой сбора событий, рис. 6.

```
package report
import (
"encoding/json"
"io"
)

func writeJson(findings []Finding, w io.WriteCloser) error {
if len(findings) == 0 {
    findings = []Finding{}
}
defer w.Close()
encoder := json.NewEncoder(w)

for _, finding := range findings {
    err := encoder.Encode(finding)
    if err != nil {
        return err
    }
}
return nil
}
```

Рисунок 6 – Модуль логирования решения для интеграции формируемых логов с системой сбора событий

Для автоматизации запуска во время стадии pre-commit была внедрена конфигурация git hook, рис. 7.

```
- id: gitleaks
  name: Detect hardcoded secrets
  description: Detect hardcoded secrets using Gitleaks
  entry: gitleaks detect --source . -v --report-path report-gitleaks.json
  language: golang
  pass_filenames: false
```

Рисунок 7 – Конфигурация git hook

Разработанный модуль был применён к промаркированной выборке программного кода веб-приложений из открытых источников, сгруппированных по языку программирования для проверки корректности обнаружения угрозы

оставления конфиденциальной информации в программном коде в рамках цикла разработки программного обеспечения, рис. 8.

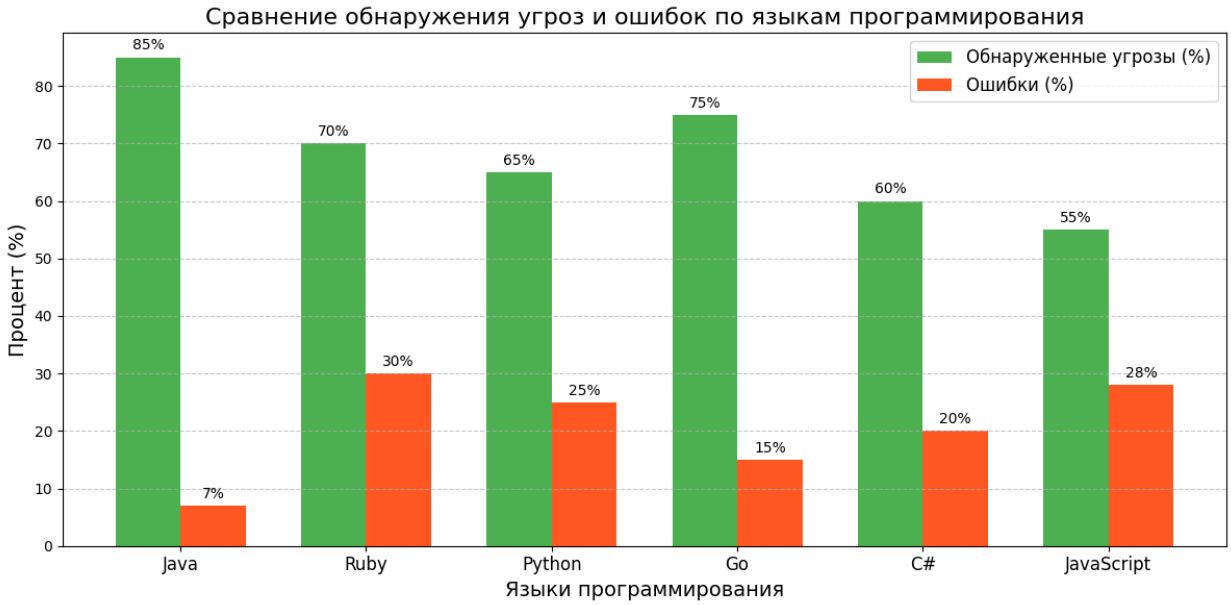


Рисунок 8 – Результат апробации модуля обнаружения угроз в коде на различных языках программирования

Так как модуль был рассчитан на специфику языка программирования Java, закономерно ожидать наиболее точных результатов обнаружения угроз именно по данной группе файлов с кодом. Однако, модуль уже сейчас демонстрирует эффективность обнаружения угроз и для приложений на языке программирования Go.

## Заключение

Предложенная модель позволяет построить полноценную систему обнаружения угроз во время разработки программных продуктов на любых языках программирования с возможностью автоматического принятия решения о недопущении распространении обнаруженных угроз последствием развёртывания

уязвимого кода. Обоснован подход к обнаружению угроз типа оставления конфиденциальных данных в программном коде перед фиксацией изменений в центральном репозитории, проведённое исследование показывает эффективность применения статических методов анализа кода и его интеграции с механизмами управления репозиториями. При этом, апробация решения показывают свою работоспособность на основных группах языков программирования, что делает возможным расширение данного подхода на выбранные к апробации группы.

**Список источников**

1. Кокорев Д.С., Сидоренко В.Г. Совершенствование процесса разработки программного обеспечения интеллектуальных транспортных систем // Материалы II Международной научно-практической конференции Интеллектуальные транспортные системы (Москва, 25 мая 2023). – Москва: Российский университет транспорта, 2023. С. 493-499.

2. Барановский А.М., Мусиенко А.С. Динамические диагностические модели и метод обеспечения устойчивости контроля технического состояния бортовых систем управления летательных аппаратов // Труды МАИ. 2024. № 139. URL: https://trudymai.ru/published.php?ID=183468

3. Девицына С.Н., Пилькевич П.В. Обеспечение совместимости технических компонентов при создании системы мониторинга инцидентов информационной безопасности // Вопросы кибербезопасности. 2024. № 4. DOI: 10.21681/2311-3456-2024-4-38-44

4. Шабуров А.С., Борисов В.И. О применении сигнатурных методов анализа информации в SIEM-системах // Вестник УрФО. Безопасность в информационной сфере. 2015. № 3 (17). С. 23-27.

5. Федорченко А.В. и др. Анализ методов корреляции событий безопасности в SIEM-системах // Труды СПИИРАН. 2016. Т. 4, № 47. С. 5-27. DOI: https://doi.org/10.15622/sp.47.1

6. Красильникова Е.В., Майорова Е.В., Соколовская С.А. Об обеспечении информационной безопасности цифрового сервиса разработки программного обеспечения в рамках AGILE-методологии // Всероссийская научно-практическая конференция «Инновационные технологии и вопросы обеспечения безопасности реальной экономики» (Санкт-Петербург, 27 марта 2020). - Санкт-Петербург: Санкт-Петербургский государственный экономический университет, 2020. С. 287-296.

7. Тулеубаева А.А., Норкина А.Н. Современные проблемы информационной безопасности в разработке программного обеспечения // Материалы VII Международной научно-практической конференции «Угрозы и риски финансовой безопасности в контексте цифровой трансформации» (Москва, 24 ноября 2021). - Москва: МИФИ, 2021. С. 670-676.

8. Шишков С.А., Путято М.М., Макарян А.С. Разработка методов обнаружения вредоносного воздействия на основе корреляционного анализа событий информационной безопасности в SIEM системах // Прикаспийский журнал: управление и высокие технологии. 2022. № 3 (59). С. 103-111.

9. Майорова Е.В., Соколовская С.А., Черток А.В. Обеспечение информационной безопасности при гибком подходе разработки программного продукта // Цифровые технологии обработки и защиты информации: сборник статей. - Санкт-Петербург: Санкт-Петербургский государственный экономический университет, 2020. С. 83-92.

10. Дубинский С.В., Стрелков В.В. Перспективные направления исследований в интересах построения комплексной системы управления безопасностью полетов // Труды МАИ. 2023. № 133. URL: https://trudymai.ru/published.php?ID=177674

11.Касатиков Н.Н., Брехов О.М., Николаева Е.О. Интеграция технологий искусственного интеллекта и интернета вещей для расширенного мониторинга и оптимизации энергетических объектов в умных городах // Труды МАИ. 2023. № 131. URL: https://trudymai.ru/published.php?ID=175929

12.Копейка Е.А., Вербин А.В. Методический подход оценивания вероятности безотказной работы сложных технических систем с учетом характеристик системы контроля на основе байесовской сети доверия // Труды МАИ. 2023. № 128. URL: https://trudymai.ru/published.php?ID=171411. DOI: 10.34759/trd-2023-128-22

13. Жиров П.В. Анализ методов сбора событий информационной безопасности при использовании систем мониторинга событий информационной безопасности с применением технологий SIEM // Десятая международная научно-техническая конференция «Безопасные информационные технологии» (Москва, 03–04 декабря 2019). – Москва: МГТУ имени Н.Э. Баумана, 2019. С. 135-139.

14.Мишурин А.О. Перспективные направления развития технологий для центров мониторинга и реагирования на инциденты информационной безопасности // I

Межвузовская научно-практическая конференция «Информационная безопасность: современная теория и практика» (Омск, 13 сентября 2019). – Омск: Сибирский государственный автомобильно-дорожный университет, 2019. С. 89-93.

15. Сас С., Вареница В.В., Марков А.С. Методические и реализационные аспекты внедрения процессов разработки безопасного программного обеспечения // Безопасность информационных технологий. 2023. Т. 30, № 2. С. 23-37.

16. Bahaa A. et al. Monitoring real time security attacks for IoT systems using DevSecOps: a systematic literature review // Information. 2021. Vol. 12, No. 4. P. 154. DOI: 10.3390/info12040154

17. Sandu A.K. DevSecOps: Integrating Security into the DevOps Lifecycle for Enhanced Resilience // Technology & Management Review. 2021. Vol. 6, No. 1. P. 1-19.

18. Cankar M. et al. Security in devsecops: Applying tools and machine learning to verification and monitoring steps // Companion of the 2023 ACM/SPEC International Conference on Performance Engineering. Coimbra, Portugal. 2023. P. 201-205. DOI: 10.1145/3578245.3584943

19. Prates L. et al. Devsecops metrics. Information Systems: Research, Development, Applications, Education // 12th SIGSAND/PLAIS EuroSymposium 2019, Gdansk, Poland, September 19, 2019. Springer International Publishing, 2019. P. 77-90.

20. Diaz J. et al. Self-service cybersecurity monitoring as enabler for DevSecOps // Ieee Access. 2019. Vol. 7, P. 100283-100295. DOI: 10.1109/ACCESS.2019.2930000

**References**

1. Kokorev D.S., Sidorenko V.G. Improving the process of developing software for intelligent transport systems. *Materialy II Mezhdunarodnoi nauchno-prakticheskoi konferentsii Intellektual'nye transportnye sistemy*. Moscow: Rossiiskii universitet transporta Publ., 2023. P. 493-499.

2. Baranovskii A.M., Musienko A.S. Dynamic diagnostic models and method for ensuring the stability of monitoring the technical condition of on-board control systems of aircraft. *Trudy MAI*. 2024. No. 139. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=183468

3. Devitsyna S.N., Pil'kevich P.V. Ensuring the compatibility of technical components when creating an information security incident monitoring system. *Voprosy kiberbezopasnosti*. 2024. No. 4. (In Russ.). DOI: 10.21681/2311-3456-2024-4-38-44

4. Shaburov A.S., Borisov V.I. On the application of signature methods of information analysis in SIEM systems. *Vestnik UrFO. Bezopasnost' v informatsionnoi sfere*. 2015. No. 3 (17). P. 23-27. (In Russ.)

5. Fedorchenko A.V. et al. Analysis of methods of correlation of security events in SIEM systems. *Trudy SPIIRAN*. 2016. Vol. 4, No. 47. P. 5-27. (In Russ.). DOI: https://doi.org/10.15622/sp.47.1

6. Krasil'nikova E.V., Maiorova E.V., Sokolovskaya S.A. On ensuring information security of a digital software development service within the framework of AGILE methodology. *Vserossiiskaya nauchno-prakticheskaya konferentsiya «Innovatsionnye tekhnologii i*

*voprosy obespecheniya bezopasnosti real'noi ekonomiki»*. Saint-Petersburg: Sankt-Peterburgskii gosudarstvennyi ekonomicheskii universitet Publ., 2020. P. 287-296.

7. Tuleubaeva A.A., Norkina A.N. Modern problems of information security in software development. *Materialy VII Mezhdunarodnoi nauchno-prakticheskoi konferentsii «Ugrozy i riski finansovoi bezopasnosti v kontekste tsifrovoi transformatsii»*. Moscow: MIFI Publ., 2021. P. 670-676.

8. Shishkov S.A., Putyato M.M., Makaryan A.S. Development of methods for detecting harmful effects based on correlation analysis of information security events in SIEM systems. *Prikaspiiskii zhurnal: upravlenie i vysokie tekhnologii*. 2022. No. 3 (59). P. 103-111. (In Russ.)

9. Maiorova E.V., Sokolovskaya S.A., Chertok A.V. Ensuring information security with a flexible approach to software product development. *Tsifrovye tekhnologii obrabotki i zashchity informatsii*: sbornik statei. Saint-Petersburg: Sankt-Peterburgskii gosudarstvennyi ekonomicheskii universitet Publ., 2020. P. 83-92.

10. Dubinskii S.V., Strelkov V.V. Promising research areas aimed at building an integrated flight safety management system. *Trudy MAI*. 2023. No. 133. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=177674

11. Kasatikov N.N., Brekhov O.M., Nikolaeva E.O. Integration of artificial intelligence and the Internet of things for advanced monitoring and optimization of energy facilities in smart cities. *Trudy MAI*. 2023. No. 131. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=175929

12. Kopeika E.A., Verbin A.V. Methodological approach to estimating the probability of failure-free operation of complex technical systems taking into account the characteristics of the control system based on the bayesian belief network. *Trudy MAI*. 2023. No. 128. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=171411. DOI: 10.34759/trd-2023-128-22

13. Zhirov P.V. Analysis of methods for collecting information security events when using information security event monitoring systems using SIEM technologies. *Desyataya mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya «Bezopasnye informatsionnye tekhnologii»*. Moscow: MGTU imeni N.E. Baumana Publ., 2019. P. 135-139.

14. Mishurin A.O. Promising directions of technology development for information security incident monitoring and response centers. *I Mezhvuzovskaya nauchno-prakticheskaya konferentsiya «Informatsionnaya bezopasnost': sovremennaya teoriya i praktika»*. Omsk: Sibirskii gosudarstvennyi avtomobil'no-dorozhnyi universitet Publ., 2019. P. 89-93.

15. Sas S., Varenitsa V.V., Markov A.S. Methodological and implementation aspects of the implementation of secure software development processes. *Bezopasnost' informatsionnykh tekhnologii*. 2023. Vol. 30, No. 2. P. 23-37. (In Russ.)

16. Bahaa A. et al. Monitoring real time security attacks for IoT systems using DevSecOps: a systematic literature review. *Information*. 2021. Vol. 12, No. 4. P. 154. DOI: 10.3390/info12040154

17. Sandu A.K. DevSecOps: Integrating Security into the DevOps Lifecycle for Enhanced Resilience. *Technology & Management Review*. 2021. Vol. 6, No. 1. P. 1-19.

18. Cankar M. et al. Security in devsecops: Applying tools and machine learning to verification and monitoring steps. *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering*. Coimbra, Portugal. 2023. P. 201-205. DOI: 10.1145/3578245.3584943

19. Prates L. et al. Devsecops metrics. Information Systems: Research, Development, Applications, Education. *12th SIGSAND/PLAIS EuroSymposium 2019*, Gdansk, Poland, September 19, 2019. Springer International Publishing, 2019. P. 77-90.

20. Diaz J. et al. Self-service cybersecurity monitoring as enabler for DevSecOps. *Ieee Access*. 2019. Vol. 7, P. 100283-100295. DOI: 10.1109/ACCESS.2019.2930000

Статья поступила в редакцию 25.04.2025
Одобрена после рецензирования 28.04.2025
Принята к публикации 25.08.2025
The article was submitted on 25.04.2025; approved after reviewing on 28.04.2025; accepted for publication on 25.08.2025