ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК 004.056.5

DOI: 10.21822/2073-6185-2025-52-1-147-161 Оригинальная статья /Original article

Архитектура интегрированного java-приложения для анализа журналов с целью обнаружения компьютерных атак в информационных системах посредством реагирования на различные аномалии безопасности

П.И. Шариков, А.В. Красов, А.В. Майоров
Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича,
193232, г. Санкт-Петербург, пр. Большевиков 22, Россия

Резюме. Цель. Необходимо при интеграции стека ELK в информационную систему иметь дублирующее java-приложение в закрытом контуре для скрытой обработки аномалий. Требуется разработка архитектуры java-приложения для скрытой интеграции с информационной системой. **Метод.** В процессе исследования были использованы методы анализа информации в журналах информационных систем, методы статического анализа, программирование для разработки приложения и алгоритмы обработки данных. **Результат.** Представлен пример реализации стека Elasticsearch для обработки и хранения журналов. Предложена реализация анализа аномалий с помощью официальной библиотеки Elasticsearch. Рассмотрены варианты использования Elasticsearch для анализа аномалий, предложена реализация анализа аномалий с помощью официальной библиотеки Elasticsearch. Предложена архитектура интегрированного в информационную систему java-приложения для автоматизированного анализа журналов с целью выявления компьютерных атак или сигналов их начала посредством поиска аномалий. Рассмотрены варианты аномалий в журналах информационных систем и описаны действия по их выявлению. Продемонстрирована обобщенная карта процесса работы java-приложения. **Вывод.** Разработана архитектура java-приложения, реализующего анализ журналов информационной системы по ключевым аномалиям.

Ключевые слова: ELK stack, elasticsearch, детектор аномалий, журналирование, anomaly logs detecting, java, анализ журналов, log analysis

Для цитирования: П.И. Шариков, А.В. Красов, А.В. Майоров. Архитектура интегрированного java-приложения для анализа журналов с целью обнаружения компьютерных атак в информационных системах посредством реагирования на различные аномалии безопасности. Вестник Дагестанского государственного технического университета. Технические науки. 2025; 52(1):147-161. DOI:10.21822/2073-6185-2025-52-1-147-161

Architecture of an integrated java application for log analysis to detect computer attacks in information systems by responding to various security anomalies

P.I. Sharikov, A.V. Krasov, A.V. Mayorov
M.A. Bonch-Bruevich St. Petersburg State University of Telecommunications,
22 Bolshevikov Ave., St. Petersburg 193232, Russia

Abstract. Objective. When integrating the ELK stack into an information system, it is necessary to have a duplicate Java application in a closed circuit for hidden anomaly processing. It is necessary to develop the architecture of a Java application for hidden integration with the information system. **Method**. The research used methods of analyzing information in information system logs, static analysis methods, programming for application development, and data processing algorithms. **Result**. An example of implementing the Elasticsearch stack for processing and storing logs is presented. An implementation of anomaly analysis using the official

Elasticsearch library is proposed. Options for using Elasticsearch for anomaly analysis are considered, an implementation of anomaly analysis using the official Elasticsearch library is proposed. The architecture of a Java application integrated into an information system for automated log analysis in order to detect computer attacks or signals of their onset by searching for anomalies is proposed. Variants of anomalies in information system logs are considered and actions for their detection are described. A generalized map of the Java application workflow is demonstrated. **Conclusion**. The architecture of a Java application implementing the analysis of logs of an information system for key anomalies has been developed.

Keywords: ELK stack, elasticsearch, anomaly logs detecting, java, log analysis

For citation: P.I. Sharikov, A.V. Krasov, A.V. Mayorov. Architecture of an integrated java application for log analysis to detect computer attacks in information systems by responding to various security anomalies. Herald of Daghestan State Technical University. Technical Sciences. 2025; 52(1):147-161. (In Russ) DOI:10.21822/2073-6185-2025-52-1-147-161.

Введение. Обнаружение аномалий можно считать ценным инструментом передовой технологической информационной системы, целью которого является оперативное обнаружение любых аномалий в поведении в или с информационной системой и, таким образом, играющего центральную роль в управлении неисправностями или действиями злоумышленников. Быстрое обнаружение аномалии позволяет специалистам решить проблему или отреагировать на нелегитимные действия в системе, как можно скорее, таким образом, чтобы сократить время, в течение которого определенная служба не может использоваться или данные могут быть скомпрометированы.

Проблемный трафик может возникать по разным причинам, включая внешние атаки, передачу устаревших данных или даже обслуживание людей для сетевых предприятий. Это становится значительно более сложной задачей, когда масштаб сети становится более масштабным. Либо системы обнаружения аномалий, которые в настоящее время используются, были обучены на устаревших наборах данных, либо у них нет возможности эффективно обрабатывать большие нагрузки. Поэтому существует потребность в масштабируемом решении, которое может обеспечить безопасность сети, обнаруживая аномалии в самой сети и уведомляя быстрым ответом всякий раз, когда происходит аномалия, получая знания из предыдущего поведения сети.

Информационные системы в настоящее время генерируют большое количество журналов, в которые они вносят информацию, считающуюся важной во время их работы: такие файлы журналов, могут использоваться в качестве источника данных для обнаружения аномалий в масштабах всей информационной системы. Именно по этой причине обнаружение аномалий с помощью журналов информационной системы крайне распространено как в академической, так и в деловой сфере. Файлы журналов отдельных программ часто анализируются построчно, чтобы выявить аномалии благодаря использованию ключевых терминов или регулярных выражений. Однако к текущему моменту времени информационные системы усложняются галопирующими темпами.

Это означает, что такая простая и понятная процедура, как анализ журналов программы построчно более невозможно ввиду постоянно возрастающей сложности архитектуры информационных систем. Высокая сложность архитектуры информационной системы приводит к большему количеству создаваемых журналов и разработке, и реализации различных механизмов устойчивости системы к сбоям. Ситуация осложняется в системах с большим количеством интеграций, где происходит интеграция не программных компонентов (микросервисов), а полноценных сложно структурированных информационных систем. Управление журналами информационной системы является неотъемлемой частью кибербезопасности. Успешный механизм киберзащиты должен иметь несколько уровней и элементов [1]. Он включает в себя людей, политики безопасности и технологии, и каждый отдельный компонент должен работать вместе, чтобы сделать защиту эффективной

[1]. Управление журналами имеет решающее значение для смягчения цифровых угроз. Журналы безопасности и аутентификации могут пролить свет на необычное поведение, трафик и шаблоны доступа [2]. В девятом ежегодном исследовании стоимости киберпреступности, опубликованном в 2019 году компанией Accenture, утверждается, что «люди по-прежнему являются самым слабым звеном безопасности» [3]. Хотя постоянные кампании по повышению осведомленности в киберпространстве остаются одним из самых мощных инструментов, нельзя предполагать, что все всегда будут делать разумный выбор в области кибербезопасности. Постоянный мониторинг журналов центральная служба аутентификации информационной системы может помочь избежать любых «слепых пятен» и усилить аппарат кибербезопасности [4].

Постановка задачи. Угрозы в сетевой инфраструктуре, такие как сбои оборудования или программного обеспечения, позволяют третьим лицам воспользоваться этими уязвимостями. Требуется постоянное осознание новых проблем, которые могут возникнуть, а затем определение действий, которые будут направлены на эти проблемы.

Необходимо уделять внимание возможности узнать с помощью имеющихся данных, имеет ли состояние системы аномалии или нет, путем обнаружения аномалий с помощью анализа журналов в режиме реального времени посредством специализированного программного обеспечения. Даже при использовании Elastic stack (Logstash, ElasticSearch, Kibana) для получения, хранения, отбора и анализа журналов информационной системы может быть недостаточным. Настройки компоновки, фильтрации и анализа логов с помощью стека ELK можно без труда посмотреть при наличии прав доступа к информационной системе, что может сделать любой сотрудник. Однако, как было указано выше, в этом заключается большая часть проблемы.

Необходим дополнительный легковесный сервис, который будет дополнительно в фоновом режиме и скрыто обрабатывать журналы информационной системы из ElasticSearch на предмет действий злоумышленника или любые другие аномалии. Данная работа сосредоточена вокруг архитектуры и фрагментов реализации такого программного обеспечения, а также интеграции его со стеком ELK. Каждая информационная система с доступом к сети Интернет имеет потенциал для нарушения безопасности в этом месте. В современном мире существует множество потенциальных опасностей, наиболее распространенными из которых являются вирусы, являющиеся вредоносными программы, микропрограммами и скриптами. В дополнение к тому, что системы безопасности всегда обновляются для борьбы с этими угрозами, сами угрозы всегда обновляются, чтобы избежать идентификации в сети. В [5] упоминается, что существуют некоторые домены сетей, такие как данные обороны, безопасности и больниц, которые не должны быть нарушены ни при каких обстоятельствах; однако, даже если это так, эти домены продолжают подвергаться риску. Отсутствие адекватных средств кибербезопасности, в дополнение к быстрому развитию технологических возможностей, подчеркивает необходимость улучшения линий защиты.

В работе «Система сбора, хранения и обработки информации и событий безопасности на основе средств elastic stack» авторы провели исчерпывающий анализ на тему использования средств анализа журналов сетевого оборудования и информационных сетей, что послужило разработкой, обобщенной архитектуры системы мониторинга на основе стека ELK [6]. Результат является общим и решает общую проблему сбора и анализа журналов информационной системы, что не совсем подходит для текущего исследования в силу того, что существует необходимость наличия скрытого программного обеспечения осуществляющего дополнительную пост-обработку данных журналов из ElasticSearch (возможно, как и у коллег, реализованного в виде микросервиса с закрытым доступом к исходному коду). Вопросы анализа журналов информационных систем в режиме реально времени исследовались зарубежными коллегами в статье «A framework for social media data analytics using Elasticsearch and Kibana» [7]. Традиционно процессы, связанные

с офлайн-анализом, продуктивны и эффективны только тогда, когда сбор данных является одноразовым процессом. Авторы демонстрируют решение проблем анализа в режиме реального времени с использованием настраиваемой поисковой системы Elasticsearch. В исследовании используется архитектура распределенной базы данных, предварительно настроенная индексация и стандартизация фреймворка Elasticsearch для крупномасштабного интеллектуального анализа текста. Таким образом, авторы показывают, что использование анализа журналов в режиме реального времени оправдан и стек ELK решает их задачу, хотя и не исследованы вопросы безопасности в данном решении.

Сравнение адаптивных подходов оптимизации на основе графов и одноклассовых опорных векторных машин (SVM) было проведено с целью выявления аномальных действий, происходящих в сети [8]. Однако из-за ограничений по пространству метод использовался только с двумя наборами данных и применялся только в двух различных контекстах; в результате производительность не могла быть обобщена. Даже принимая во внимание возросшую сложность подхода, есть возможности для совершенствования.

Самая сложная проблема, с которой приходится сталкиваться системам предотвращения и обнаружения утечек данных на основе контента (DLPD) для предприятий, — это масштабируемость. Это означает, что эти системы неспособны своевременно обрабатывать большие объемы сетевых журналов [9]. Было написано большое количество статей о системах аномалий, которые способны обнаруживать аномалии в обучающих и тестовых наборах данных, которые были получены и сохранены. Однако сетевой трафик хранится в виде временного ряда, и эти решения не имеют возможности выполнять временной проект или предоставлять оповещения в реальном времени при обнаружении аномалий. Они обучались и тестировались с использованием устаревших и непредсказуемо точных данных, поэтому их производительность в реальных сценариях с фактическими данными была низкой. Кроме того, они не показали хороших результатов.

В исследовании «Система обнаружения аномалий в журналах мониторинга состояния объекта защиты» авторы разработали систему обнаружения аномалий посредством языка программирования Python [10]. В данной работе анализ журналов осуществлялся с помощью специализированной библиотеки pygrok, которая позволяет выполнять grok-выражения внутри python-файлов. Полученные данные преобразовывались в формат данных JSON и далее конвертировались в формат Pandas DataFrame. Таким образом, данная система решает задачу поиска аномалий, является более легковесной по сравнению со всеми инструментами стека ELK, однако, не затрагивает такие вопросы как безопасное хранение grok-выражений, изменение кодировки или управляющих символов журналов, что впоследствии может привести к проблемам сериализации данных журналов в формат JSON [11], а также в недостаточной мере описана архитектурная составляющая такой системы.

Шелухин О.И. и Рябинин В.С. в исследовании «Обнаружение аномалий больших данных неструктурированных системных журналов» рассматривают аномалии, как «экземпляры в наборе данных, которые не соответствуют регулярному поведению системы» [12]. В данной работе основной упор идет на несоответствие актуальных записей в журналах информационной системе ожидаемым. В работе авторам успешно удалось достигнуть выявления аномалий типа KERNMC и типа APPSEV посредством представленных классификаторов. Подходы, выведенные авторами данной работы применимы к текущему исследованию, журналирование дефектов во время работы информационной системы, стека трассировки какой-либо функции, является регулярным и даже в определенной степени ожидаемым поведением системы [13].

Таким образом существует необходимость рассматривать цепочки действий внутри информационной системы, приводящие к дефектам, ошибкам, несанкционированному доступу и другим маркерам аномальных активностей в информационной системе или сетевом оборудовании. Ставится задача разработать архитектуру программного обеспечения,

позволяющего анализировать журналы ИС и оборудования для поиска аномалий и своевременного реагирования.

Методы исследования. Elasticsearch - это открытая распределённая поисковая и аналитическая система для различных типов данных, включая текстовые, числовые, геопространственные, структурированные и неструктурированные данные. Она построена на основе библиотеки Apache Lucene и обеспечивает мощные возможности для полнотекстового поиска, анализа и хранения данных в реальном времени. Elasticsearch часто используется в составе стека Elastic Stack (ранее известного как ELK Stack).

Elasticsearch**:** Хранение и индексирование данных. Память стека представлена Elasticsearch. Она состоит из группы узлов, по которым сетевые журналы разделяются и хранятся в отформатированном виде, с репликами, настроенными по мере необходимости. Здесь журналы проходят ежедневную индексацию, при этом для каждой даты создаются новые индексы. Индексы могут быть запрошены для получения этой информации, а затем она может быть агрегирована в более крупный набор данных для обнаружения аномалий на основе интересующего временного интервала

Logstash**:** Сбор, обработка и преобразование данных из различных источников перед их отправкой в Elasticsearch. Данные собираются Logstash, который в данном случае состоит из сетевых журналов, а также прослушивает заданный путь и затем отправляет собранные данные в конечный пункт назначения, который в данном случае является кластером Elasticsearch. Logstash дополнительно форматирует данные, которые до сих пор были только значениями, преобразуя их в пары ключ-значение. Из-за этого Logstash создает конвейер для транспортировки данных из начальной точки в конечный пункт назначения, а также применяет необходимое форматирование к данным по пути.

Kibana**:** Инструмент визуализации данных, позволяющий создавать сводные таблицы, графики и проводить интерактивный анализ данных. В качестве основного средства связи между пользователем и методом обнаружения аномалий Kibana выступает в качестве интерфейса. Данные, нормальное поведение и выбросы представлены графически. Она может создавать графические изображения, такие как тепловые карты в обозревателе аномалий или иллюстративные представления в едином средстве просмотра метрик. Доступные параметры позволяют создавать и настраивать панель мониторинга, которая может отображать все соответствующие данные в удобоваримом формате.

Варианты использования Elasticsearch:

- Анализ журналов: Сбор и анализ файлов журналов от различных приложений и систем.
- Поиск по сайту: Реализация поиска на веб-сайтах и в приложениях.
- Мониторинг приложений: Отслеживание производительности и состояния приложений в реальном времени.
- Аналитика безопасности: Выявление аномалий и потенциальных угроз в безопасности.
- Бизнес-аналитика: Анализ клиентского поведения, продаж и других бизнес-показателей.
- Мошеннические транзакции: Анализ транзакций в реальном времени для выявления подозрительной активности.
- Мониторинг доступа: Отслеживание попыток несанкционированного доступа к счетам клиентов.
- Контроль привилегированных пользователей: Мониторинг действий администраторов и пользователей с повышенными правами.
- Обнаружение внутренних угроз: Выявление аномальной активности сотрудников, которая может указывать на инсайдерские угрозы.
- Защита от ботов и сканирования: Обнаружение автоматизированных попыток подбора паролей или сбора данных.

– Анализ аномальной активности: Выявление необычных паттернов поведения пользователей, которые могут свидетельствовать о взломе аккаунтов.
– Системы управления доступом и идентификацией (IAM): Анализ журналов аутентификации и авторизации для выявления попыток компрометации учетных записей [14].

Пример реализации стека Elasticsearch представлен на рис. 1.

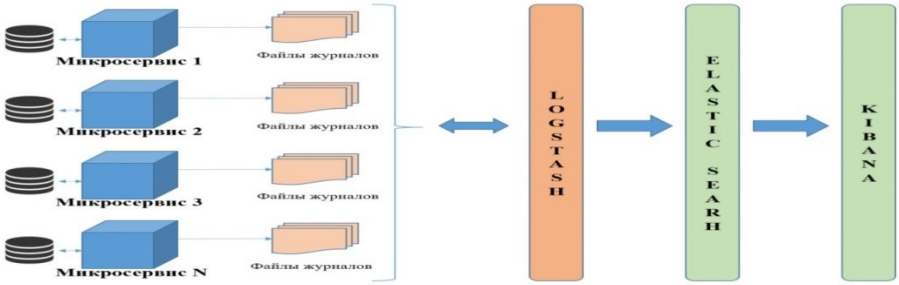

Рис. 1 - Пример реализации архитектуры Elasticsearch в информационной системе
Fig. 1 - Example of implementation of Elasticsearch architecture in an information system

Варианты использования Elasticsearch для анализа уязвимостей.

1. Сбор данных: Осуществляется сбор данных о безопасности из различных источников, таких как журналы событий, сетевые сканеры, отчеты о безопасности и другие инструменты. Эти данные могут включать информацию о попытках вторжений, уязвимостях, конфигурациях систем и т.д.
2. Индексирование данных: Собранные данные импортируются в Elasticsearch. Это осуществляется с помощью инструментов, таких как Logstash или Beats, которые помогают в обработке и передаче данных в Elasticsearch.
3. Поиск и анализ: Используются встроенные возможности поиска Elasticsearch для обнаружения аномалий, выявления уязвимостей и анализа трендов. Существует возможность создавать сложные запросы для фильтрации и агрегации данных.
4. Визуализация: Используется инструмент Kibana для визуализации данных, который интегрируется с Elasticsearch. Kibana позволяет создавать дашборды и визуализации, которые помогают лучше понять состояние безопасности вашей системы.
5. Автоматизация и оповещения: Осуществляется настройка автоматических оповещений на основе определенных условий, таких как обнаружение новых уязвимостей или аномальных действий. Это можно сделать с помощью инструмента Alerting в Elastic Stack.
6. Аналитика безопасности: используется для обнаружения и расследования угроз безопасности в режиме реального времени. Он может анализировать различные типы данных, такие как сетевой трафик, поведение пользователя и системные журналы, для выявления аномалий и угроз. Elasticsearch можно интегрировать с другими инструментами безопасности, такими как Suricata, Zeek и Snort, чтобы обеспечить комплексное решение безопасности.

Диаграмма рабочего процесса по анализу журналов и открытому выявлению аномалий представлена на рис. 2.

Рис. 2 - Диаграмма рабочего процесса анализа журналов на предмет аномалий с помощью Elasticsearch
Fig. 2 - Workflow diagram of analyzing logs for anomalies using Elasticsearch

Однако все вышеописанное не решает главную проблему. Все эти инструменты и их интеграции не являются скрытными, как для пользователя, так и для разработчиков

системы. Необходим внешний модуль, который будет защищен политиками безопасности, код которого будет скрыт от всех, кроме небольшой группы разработчиков и который будет в закрытом режиме осуществлять анализ журналов из хранилища Elasticsearch. Данное приложение должно работать по типу черного ящика для сохранения в скрытом состоянии проверок, реализованных в его логике, а базовые общеизвестные проверки должны быть реализованы с помощью официального инструментария Elastic Stack**.**

Требования к приложению, осуществляющему скрытый анализ журналов. Как было описано ранее, необходимо, чтобы в контуре информационной системы имелся такой сервис, к которому никто не имеет доступ и логика анализа журналов максимально защищена. Сервис может быть представлен программным обеспечением, отдельным модулем или системой для обнаружения компьютерных атак в корпоративных и государственных информационных системах [15-17].

Таким образом, данный сервис должен по сути дублировать обнаружение аномалий в журналах информационной системы, ее компонентов или сетевого оборудования, но по логике кардинально отличаться. Также, необходимо, чтобы данное приложение имело доступ к хранилищу данных elasticsearch, а, следовательно, возможность интеграции с ним. Действенным вариантом является использование официальной api-библиотеки elasticsearch для языка программирования Java. Язык Java выбран в силу его популярности, кроссплатформенности и легкости интеграции со стеком Elastic в силу того, что весь стек реализован на языке программирования Java. При таких условиях к java-приложению должны предъявляться определенные требования безопасности для того, чтобы минимизировать различные риски компрометации [18].

Требования к java-приложению по скрытому анализу журналов:

- Необходимость использования шифрования данных на всех уровнях (например, по ГОСТ).
- Безопасное подключение к Elasticsearch с использованием HTTPS и аутентификации.
- Ограничение доступа к микросервису для сотрудников.
- Использование ролевой модели доступа и системы управления доступом и аутентификации, такие как OAuth или JWT.
- Микросервис должен быть скрыт от общего доступа и не должен быть виден в общих списках сервисов.
- Использование нестандартных портов и имен для сокрытия микросервиса.
- Размещение микросервиса в изолированной сети или сегменте, недоступном для общего доступа.
- Разработка микросервиса должна вестись небольшой группой сотрудников с высоким уровнем квалификации в области информационной безопасности и доступами к секретности.
- Хранение кода и конфигураций для него в защищенных репозиториях с ограниченным доступом.
- Использование системы управления секретами для хранения конфиденциальной информации.
- Использование обфускации исходного кода и скрытого вложения цифрового водяного знака в байт-код class-файлов микросервиса для защиты его от компрометации, как образа Docker-контейнера или jar-файла [19-20]
- Постоянное регрессионное тестирование безопасности.

На рис. 3 приведен пример архитектурной схемы информационной системы с интеграцией java-приложения осуществляющего анализ журналов на предмет наличия аномалий.

Рис. 3 - Пример архитектуры ИС с использованием интеграции с специализированным java-приложением по анализу журналов на предмет наличия аномалий
Fig. 3 - An example of an IS architecture using integration with a specialized Java application for analyzing logs for anomalies

Архитектура java-приложения. Необходимо определить, какой функционал ожидается от приложения по анализу данных в Elasticsearch. Также, необходимо определить какие инструменты будут использоваться для реализации необходимой логики.

Данное приложение должно иметь возможность работать постоянно, не нуждаться в постоянных повторных запусках и периодически обрабатывать новые журналы, появляющиеся в Elasticsearch. Для примера, в данной работе будет приведена реализация java-приложения, разработанного с использованием Spring Boot, предназначенного для анализа журналов сетевой активности. С целью анализа журналов на наличие различных аномалий, таких как подозрительный трафик, использование небезопасных протоколов, доступ из не доверенных геолокаций и другие. Приложение использует конфигурацию из файла «application.properties» для настройки пороговых значений и списков доверенных элементов. Журналы анализируются с помощью регулярных выражений, а результаты выводятся в собственный журнал с использованием Log4j.

Таким образом, исходя из этих требований было решено реализовывать java-приложение на следующем стеке технологий:

- Java – язык программирования.
- Gradle – инструмент сборки.
- Spring Boot Starter - Основная зависимость для Spring Boot.
- Spring Boot Starter Data Elasticsearch - Зависимость для работы с Elasticsearch.
- Spring Boot Starter Test: Зависимость для тестирования с использованием Spring Boot
- Log4j - Зависимость для журналирования.
- Mockito - Зависимость для создания моков(эмуляций, заглушек) в тестах данного приложения.

Принято решение разделить приложение на 5 внутренних class-файлов:

- LogEntry.class - Класс для представления записи журнала. Он извлекает данные из сообщения журнала с помощью регулярных выражений и других функций.
- LogAnalyzerConfig.class - Класс для конфигурации java-приложения. Использует аннотации Spring для получения значений из «application.properties».
- LogAnalyzerApplication.class - Класс для запуска java-приложения. Он создает экземпляры необходимых компонентов и запускает анализ журналов, хранящихся в Elasticsearch.

– LogAnalyzerTest.class - Класс для юнит-тестов, использующий библиотеку Mockito для создания эмуляции (моков) конфигурации и тестирования логики анализа журналов, хранящихся в Elasticsearch.
– LogAnalyzer.class - Основной класс для анализа журналов. Он использует конфигурацию из LogAnalyzerConfig и анализирует каждый полученный журнал на наличие аномалий.

Обособленная схема взаимодействия частей java-приложения представлена на рис. 4.

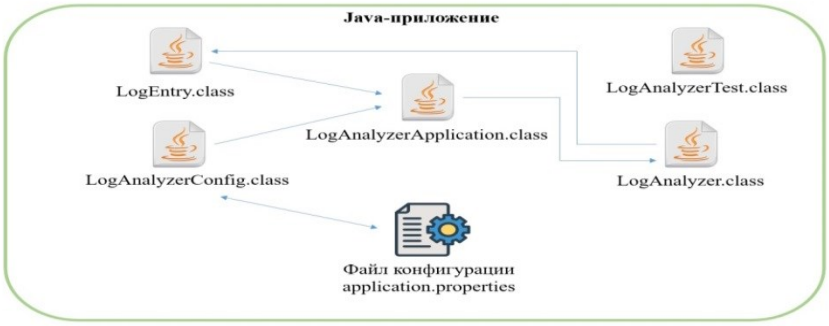

Рис. 4 - Модель схемы взаимодействия классов внутри java-приложения
Fig. 4 - Model of the interaction scheme of classes within a Java application

В описании алгоритма работы классов LogAnalyzerApplication LogAnalyzerConfig нет необходимости так как там выполняются базовые функции получения конфигурации java-приложения, настроек пользователя и параметров запуска. Остальные классы java-приложения необходимо осветить.

Возможности класса LogEntry заключаются в извлечении из данных Elasticsearch такой информации, как: количество пакетов; номер порта; протокол; IP-адрес; уровень активности пользователя; количество неудачных попыток входа; проверка наличия несанкционированных изменений; объем трафика; геолокация; временная метка файла журнала.

Класс LogAnalyzer является ключевым для java-приложения. Данный класс осуществляет проверки на наличие аномалий в полученных данных журналов из Elasticsearch. Выявление аномалий в сетевых журналах - это сложная задача, которая может включать в себя множество различных подходов и методик. Типы аномалий и методы их выявления, реализуемые классом LogAnalyzer описаны ниже.

Анализ журналов на наличие аномалий и варианты их выявления. В журналах информационных систем и сетевого оборудования могут отображаться различного рода артефакты поведения систем и пользователей в них, что выражается в аномалиях, которые необходимо отслеживать и анализировать.

Ниже представлен ряд отслеживаемых аномалий java-приложением. Также, необходимо учитывать, что информацию для удобства поиска аномалий нужно агрегировать по каким-либо параметрам, например, по пользователям, времени суток, типу события, геолокации, пакетам, портам.

Необычные объемы трафика.

Аномалия: Внезапное увеличение или уменьшение объема трафика в общем или по определенному порту.

Выявление: Необходимо использовать агрегации для вычисления среднего и стандартного отклонения объема трафика, после чего произвести сравнение текущих значений с историческими данными, чтобы выявить значительные отклонения.

В листинге 1 продемонстрирован пример фрагмента кода реализующий агрегации и проверки на наличие аномалий по объему трафика.

Листинг 1- Агрегация трафика и проверка на наличие аномалии
Listing 1- Traffic aggregation and checking for anomalies

```
// Агрегации по типу события
```

```
        TermsAggregationBuilder eventTypeAgg = AggregationBuild
ers.terms("by_event_type").field("event_type");
        searchSourceBuilder.aggregation(dateHistogramAgg);
        searchSourceBuilder.aggregation(packetStatsAgg);
        searchSourceBuilder.aggregation(portAgg);
        searchSourceBuilder.aggregation(protocolAgg);
        searchSourceBuilder.aggregation(ipAgg);
        searchSourceBuilder.aggregation(userAgg);
        searchSourceBuilder.aggregation(eventTypeAgg);
        searchRequest.source(searchSourceBuilder);
        SearchResponse searchResponse = cli-
ent.search(searchRequest,RequestOptions.DEFAULT);
        // Обработка результатов
        Stats packetStats = searchResponse.getAggregations().get("packet_stats");
        log.info("Среднее количество пакетов: " + packetStats.getAvg());
        log.info("Максимальное количество пакетов: " + packetStats.getMax());
        // Необычные объемы трафика
        if (packetStats.getMax() > 1000) {
        log.info("Аномалия: Обнаружено большое количество пакетов!");
```

Подозрительные порты.

Аномалия: Трафик на портах, которые обычно не используются в вашей сети.

Выявление: Необходимо создать список «разрешенных» портов и отслеживать трафик на портах, которые не входят в данный список доверенных портов.

Необычные IP-адреса.

Аномалия: Взаимодействие с IP-адресами, которые не являются частью обычной сетевой активности для информационной системы.

Выявление: Необходимо использовать агрегации для подсчета уникальных IP-адресов и произвести сравнения их с «белым» списком доверенных IP-адресов.

Подозрительные протоколы.

Аномалия: Использование небезопасных или необычных протоколов.

Выявление: Необходимо отслеживать использование протоколов, которые не являются стандартными для информационной системы, например, Telnet или FTP.

В листинге 2 представлен пример фрагмента кода, демонстрирующий агрегации по портам, ip-адресам, протоколам и реализацию проверок на наличие аномалий.

Листинг 2- Агрегация по портам, адресам, протоколам с проверкой на аномалии
Listing 2- Aggregation by ports, addresses, protocols with anomaly checking

```
        Terms portTerms = searchResponse.getAggregations().get("by_port");
        log.info("Агрегация по портам:");
        for (Terms.Bucket bucket : portTerms.getBuckets()) {
        log.info("Порт: " + bucket.getKeyAsString() + ", Количество: " + buck-
et.getDocCount());
        }
        Terms protocolTerms = searchResponse.getAggregations().get("by_protocol");
        System.out.println("Агрегация по протоколам:");
        for (Terms.Bucket bucket : protocolTerms.getBuckets()) {
        log.info ("Протокол: " + bucket.getKeyAsString() + ", Количество: " + buck-
et.getDocCount());
        }
        Terms ipTerms = searchResponse.getAggregations().get("by_ip");
        log.info ("Агрегация по IP-адресам:");
        for (Terms.Bucket bucket : ipTerms.getBuckets()) {
        log.info("IP: " + bucket.getKeyAsString() + ", Количество: " + buck-
et.getDocCount());
        }
        // Подозрительные порты
        for (Terms.Bucket bucket : portTerms.getBuckets()) {
           int port = Integer.parseInt(bucket.getKeyAsString());
           if (SUSPICIOUS_PORTS.contains(port)) {
             log.info("Аномалия: Обнаружен подозрительный трафик на порту " + port);
           }
        }
        // Подозрительные протоколы
        for (Terms.Bucket bucket : protocolTerms.getBuckets()) {
           String protocol = bucket.getKeyAsString();
```

```
        if (SUSPICIOUS_PROTOCOLS.contains(protocol.toLowerCase())) {
          log.info ("Аномалия: Обнаружен подозрительный протокол " + protocol);
        }
    }
    // Необычные IP-адреса
    for (Terms.Bucket bucket : ipTerms.getBuckets()) {
        String ip = bucket.getKeyAsString();
        if (!TRUSTED_IPS.contains(ip)) {
          log.info ("Аномалия: Обнаружен трафик с неизвестного IP " + ip);}}
```

Частые неудачные попытки входа.

Аномалия: Многочисленные неудачные попытки аутентификации за короткий период времени.

Выявление: Необходимо использовать агрегации по времени для подсчета неудачных попыток входа и установить порог, приемлемый для тестируемой информационной системы, для определения аномалий такого рода.

Необычные геолокации.

Аномалия: Доступ к сети из геолокаций, которые не являются доверенными для информационной системы.

Выявление: Необходимо использовать геолокационные данные для IP-адресов и произвести сравнение их с местоположениями из доверенного списка.

Временные аномалии.

Аномалия: Сетевая активность в нерабочее время.

Выявление: Необходимо использовать временные агрегации для выявления активности в нерабочие часы или дни.

Сетевые сканирования.

Аномалия: Многочисленные попытки подключения к разным портам с одного IP-адреса.

Выявление: Производится отслеживание количества уникальных портов, к которым обращается один IP-адрес, и устанавливается порог количества для определения сканирования.

Необычные шаблоны использования.

Аномалия: Выявление необычных шаблонов использования ресурсов (например, чрезмерное использование определенного сервиса).

Выявление: Необходимо использовать агрегации по ресурсам и производить сравнение с историческими данными.

Необычные последовательности событий.

Аномалия: Анализ последовательности событий для выявления необычных действий, таких как последовательные неудачные попытки входа.

Выявление: Необходимо использовать последовательные агрегации и производить детальный анализ последовательности цепочек действий пользователей в информационной системе.

Необычная частота событий.

Аномалия: Выявление аномалий на основе частоты событий, таких как слишком частые запросы от одного IP-адреса.

Выявление: Необходимо использовать агрегации по времени и IP-адресам для анализа частоты событий.

Различные сетевые атаки.

Аномалия: Выявление атак, таких как DDoS, на основе анализа объемов трафика и распределения по IP-адресам.

Выявление: Необходимо использовать агрегации по IP и времени для выявления аномальных пиков трафика.

Листинг 3 демонстрирует агрегацию по ip-адресам и простейшую проверку на реализацию сетевой атаки.

Листинг 3 - Агрегация по ip-адресам и проверка на сетевую атаку
Listing 3 - Aggregation by IP addresses and checking for network attack

```
        SearchRequest searchRequest = new SearchRequest("logs");
        SearchSourceBuilder sourceBuilder = new SearchSourceBuilder();
        sourceBuilder.query(QueryBuilders.matchAllQuery());
        searchRequest.source(sourceBuilder);
        SearchResponse searchResponse = client.search(searchRequest, RequestOp-
tions.DEFAULT);
        searchResponse.getHits().forEach(hit -> {
        String logMessage = hit.getSourceAsString();
        log.info("Анализ журнала: " + logMessage);

        private boolean isPotentialDDoSAttack(String logMessage) {
        // Пример проверки на DDoS атаку
        Pattern pattern = Pattern.compile("Объем трафика:(\\d+)");
        Matcher matcher = pattern.matcher(logMessage);
          if (matcher.find()) {
            int trafficVolume = Integer.parseInt(matcher.group(1));
            return trafficVolume > 10000; // Пороговое значение для DDoS
          }
        return false;
        }
        // Аномалия выявления атак, например, как DDoS
        if (isPotentialDDoSAttack(logMessage)) {
         log.info("Аномалия: Обнаружена потенциальная DDoS атака");
        }
```

Несанкционированные изменения в конфигурации.

Описание: Обнаружение изменений в конфигурации, которые могут указывать на несанкционированные действия пользователей или злоумышленников.

Выявление: Необходимо производить анализ журналов на предмет изменений конфигурации и постоянное сравнение с имеющимися с историческими данными.

Обобщенный пример карты процесса работы java-приложения по анализу журналов событий информационной системы из Elasticsearch продемонстрирована на рис. 5.

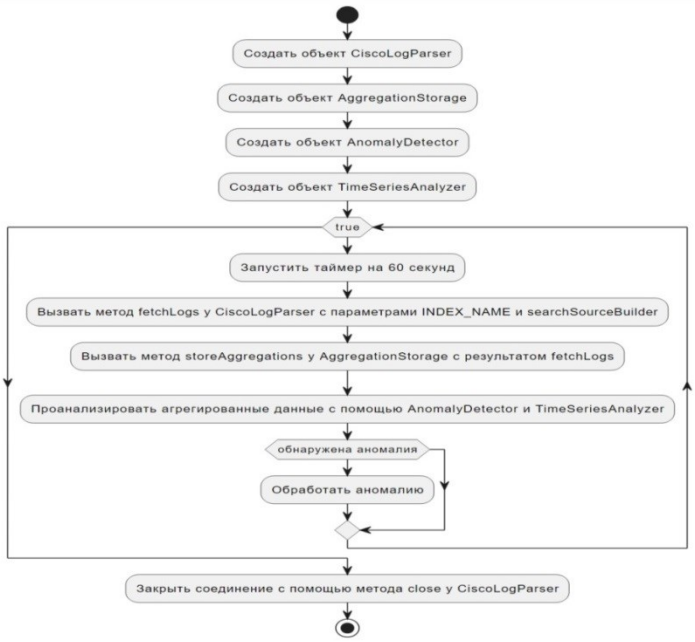

Рис. 5 - Обобщенная карты процесса работы java-приложения по анализу журналов событий информационной системы
Fig. 5- Generalized map of the process of work of the Java application for analysis of event logs of the information system

Обсуждение результатов. В данной работе исследована архитектура интеграции стека ELK в информационную систему. Определены ситуации, когда необходимо отслеживать подозрительную активность в журналах. Определены основные аномалии, кото-

рые необходимо ожидать в журналах информационной системы и оборудования, после чего реагировать.

Разработана архитектура ПО для анализа журналов в закрытом контуре и поиска аномалий. Использование разработанной архитектуры и предоставленных аномалий в журналах позволит в режиме реального времени реагировать на действия злоумышленников, сбои системы, различные атаки и внутренние действия по саботажу. В дальнейшем предстоит исследовать взаимосвязь аномалий друг с другом, а также вывести алгоритм определения цепочек вредоносных действий в рамках одной информационной системы и ее оборудования.

Вывод. Продемонстрированы возможности стека инструментов Elastic для анализа аномалий в журналах информационных систем. Представлены вариации внедрения Elastic stack в архитектуру информационной системы. Продемонстрирована возможность скрытой интеграции java-приложения для параллельного анализа журналов на предмет аномалий, что позволяет скрыть во внутреннем контуре информационной системы те проверки, по которым java-приложение сигнализирует о найденной аномалии. Представлено краткое описание возможностей для скрытой интеграции java-приложения в информационную систему, рассмотрен пример архитектуры подобной интеграции, даны рекомендации по внедрению и указаны требования к разрабатываемому приложению-анализатору.

Рассмотренные виды аномалий, на наличие которых возможно анализировать журналы информационной системы, покрывают большинство «начальных этапов» компьютерных атак, что позволяет своевременно реагировать на различные рода манипуляции с информационной системой, ее данными, ролевой моделью или системой безопасности. Для рассмотренных аномалий даны шаги их локализации в файлах журналов информационной системы, а также продемонстрированы фрагменты исходного кода, позволяющего производить проверки с помощью java-приложения в автоматизированном режиме.

Продемонстрирована обобщённая карта процесса работы интегрированного java-приложения, которая демонстрирует общий принцип работы данного приложения внутри информационной системы.

Библиографический список:

1. Zamani M., Movahedi M. Machine learning techniques for intrusion detection // arXiv preprint arXiv:1312.2177. – 2013.
2. Kononenko O. et al. Mining modern repositories with elasticsearch // Proceedings of the 11th working conference on mining software repositories. – 2014. – С. 328-331.
3. Salo F. et al. Data mining techniques in intrusion detection systems: A systematic literature review // IEEE Access. – 2018. – Т. 6. – С. 56046-56058.
4. Son S. J., Kwon Y. Performance of ELK stack and commercial system in security log analysis // 2017 IEEE 13th Malaysia international conference on communications (MICC). – IEEE, 2017. – С. 187-190.
5. Орлов, Г.А. Применение Big Data при анализе больших данных в компьютерных сетях / Г.А. Орлов, А.В. Красов, А.М. Гельфанд // Наукоемкие технологии в космических исследованиях Земли. – 2020. – Т. 12, № 4. – С. 76-84. – DOI 10.36724/2409-5419-2020-12-4-76-84. – EDN RQQTOQ.
6. Котенко, И.В. Система сбора, хранения и обработки информации и событий безопасности на основе средств Elastic Stack / И.В. Котенко, А.А. Кулешов, И.А. Ушаков // Труды СПИИРАН. – 2017. – № 5(54). – С. 5-34. – DOI 10.15622/sp.54.1. – EDN ZMREVZ.
7. Shah N., Willick D., Mago V. A framework for social media data analytics using Elasticsearch and Kibana // Wireless networks. – 2022. – Т. 28. – №. 3. – С. 1179-1187.
8. Bhattacharjee S. D. et al. Context-aware graph-based analysis for detecting anomalous activities // 2017 IEEE International Conference on Multimedia and Expo (ICME). – IEEE, 2017. – С. 1021-1026.
9. Cheng L., Liu F., Yao D. Enterprise data breach: causes, challenges, prevention, and future directions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2017;7:5: e1211.
10. Башмаков, Н. М. Система обнаружения аномалий в журналах мониторинга состояния объекта защиты / Н. М. Башмаков, В. В. Уразаев, А. М. Вульфин // Сборник избранных статей научной сессии ТУСУР. – 2023. – № 1-3. – С. 36-41. – EDN LUQUGC.

11. Шариков П.И., Красов А.В. Исследование уязвимости сериализации и десериализации данных в java//Региональная информатика и информационная безопасность:сборник научных трудов, Санкт-Петербург, 01–03 ноября 2017 г. Санкт-Петербургское Общество информатики, вычислительной техники, систем связи и управления. Выпуск 3. Санкт-Петербург:Санкт-Петербургское общество информатики, вычислительной техники, систем связи и управления, 2017. – С. 333-336. – EDN YNAETH.
12. Шелухин, О.И. Обнаружение аномалий больших данных неструктурированных системных журналов / О.И. Шелухин, В.С. Рябинин // Вопросы кибербезопасности. – 2019. – № 2(30). – С. 36-41. – DOI 10.21681/2311-3456-2019-2-36-41. – EDN IGKKGG.
13. Красов А.В., Сахаров Д.В., Тасюк А.А. Проектирование системы обнаружения вторжений для информационной сети с использованием больших данных//Наукоемкие технологии в космических исследованиях Земли. – 2020.– Т.12, № 1.– С.70-76. DOI 10.36724/2409-5419-2020-12-1-70-76. - EDN UJEKZY.
14. Миняев А.А., Красов А.В., Сахаров Д.В. Метод оценки эффективности системы защиты информации территориально-распределенных информационных систем персональных данных//Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1:Естественные и технические науки. – 2020. – № 1. – С. 29-33. – DOI 10.46418/2079-8199_2020_1_5. EDN ULHTJK.
15. Миняев, А.А. Методика оценки эффективности системы защиты информации территориально-распределенных информационных систем / А.А. Миняев, А.В. Красов // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. – 2020. – № 3. – С. 26-32. – DOI 10.46418/2079-8199_2020_3_4. – EDN YNHOEI.
16. Технические аспекты управления с использованием сети Интернет : Монография / А.А. Алейников, К.З. Билятдинов, А.В. Красов [и др.]. – Санкт-Петербург : Центр научно-информационных технологий «Астерион», 2016. – 305 с. – ISBN 978-5-00045-408-4. – EDN XGTJLL.
17. Майоров, А.В. Архитектура и программная реализация системы обнаружения компьютерных атак в корпоративных и государственных информационных системах на основе методов интеллектуального анализа / А.В. Майоров // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. – 2023. – № 2. – С. 40-46. – DOI 10.46418/2079-8199_2023_2_8. – EDN HEPDFF
18. Шариков П.И., Цветков А.Ю., Сигачева В.В., Сиротина Л.К. Исследование и алгоритм предотвращения эксплуатации уязвимостей библиотеки журналирования Log4j в информационных системах java-приложений // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. – 2023. – № 4. – С. 100-106. – DOI 10.46418/2079-8199_2023_4_19. – EDN BULSOH
19. Шариков П.И. Методика обфускации байт-кода java-приложения с целью его защиты от атак декомпиляцией // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия1:Естественные и технические науки. – 2022. – № 1. – С. 64-72. – DOI 10.46418/2079-8199_2022_1_10. – EDN AUOFNA
20. Шариков, П.И. Методика создания и скрытого вложения цифрового водяного знака в байт-код class-файла на основе не декларированных возможностей виртуальной машины java // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. – 2023. – № 7-2. – С. 165-174. – DOI 10.37882/2223-2982.2023.7-2.37. – EDN YBEWYQ

References:

1. Zamani M., Movahedi M. Machine learning techniques for intrusion detection. arXiv preprint arXiv:1312.2177. – 2013.
2. Kononenko O. et al. Mining modern repositories with elasticsearch. *Proceedings of the 11th working conference on mining software repositories*. 2014:328-331.
3. Salo F. et al. Data mining techniques in intrusion detection systems: A systematic literature review. *IEEE Access*. 2018; 6: 56046-56058.
4. Son S. J., Kwon Y. Performance of ELK stack and commercial system in security log analysis. 2017 *IEEE 13th Malaysia international conference on communications (MICC)*. IEEE, 2017;187-190.
5. Orlov, G.A. Application of Big Data in the Analysis of Big Data in Computer Networks / G.A. Orlov, A.V. Krasov, A M. Gelfand. *High-tech in space exploration of the Earth*. 2020;12(4):76-84. - DOI 10.36724/2409-5419-2020-12-4-76-84. - EDN RQQTOQ. (In Russ)
6. Kotenko I.V., Kuleshov A.A., Ushakov I.A. System for collecting, storing and processing information and security events based on Elastic Stack., *Proceedings of SPIIRAS*. 2017;5 (54):5-34. - DOI 10.15622/sp.54.1. - EDN ZMREVZ. (In Russ)
7. Shah N., Willick D., Mago V. A framework for social media data analytics using Elasticsearch and Kibana. *Wireless networks*. 2022; 28:3:1179-1187.
8. Bhattacharjee S. D. et al. Context-aware graph-based analysis for detecting anomalous activities. 2017 *IEEE International Conference on Multimedia and Expo (ICME*). IEEE, 2017:1021-1026.

9. Cheng L., Liu F., Yao D. Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. 2017;7:5:1211.
10. Bashmakov, N.M. System for detecting anomalies in logs of monitoring the state of a protected object / N.M. Bashmakov, V.V. Urazaev, A.M. Wulfin. *Collection of selected articles of the scientific session of TUSUR*. 2023;1-3:36-41. - EDN LUQUGC. (In Russ)
11. Sharikov, P.I., Krasov A.V. Study of the vulnerability of serialization and deserialization of data in Java // Regional informatics and information security: collection of scientific papers, St. Petersburg, November 01-03, 2017 / St. Petersburg Society for Informatics, Computer Engineering, Communications and Control Systems. Volume Issue 3. - St. Petersburg: St. Petersburg Society for Informatics, Computer Engineering, Communications and Control Systems, 2017: 333-336. - EDN YNAETH. (In Russ)
12. Sheloukhin O.I., Ryabinin V.S. Detection of anomalies in big data of unstructured system logs . *Cybersecurity Issues*. 2019; 2 (30):36-41. - DOI 10.21681/2311-3456-2019-2-36-41. - EDN IGKKGG. (In Russ)
13. Krasov, A.V. Design of an Intrusion Detection System for an Information Network Using Big Data / A.V. Krasov, D.V. Sakharov, A.A. Tasyuk. *High Technologies in Space Research of the Earth*. 2020;12(1):. 70-76. - DOI 10.36724/2409-5419-2020-12-1-70-76. - EDN UJEKZY. (In Russ)
14. Minyaev A.A., Krasov A.V., Sakharov D.V. Method for Assessing the Effectiveness of an Information Security System for Geographically Distributed Personal Data Information Systems. *Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and Technical Sciences*. 2020;1:29-33. – DOI 10.46418/2079-8199_2020_1_5. – EDN ULHTJK. (In Russ)
15. Minyaev, A.A. Methodology for assessing the effectiveness of the information security system of geographically distributed information systems / A.A. Minyaev, A.V. Krasov. *Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and technical sciences*. 2020; 3: 26-32. - DOI 10.46418/2079-8199_2020_3_4. - EDN YNHOEI. (In Russ)
16. Technical aspects of management using the Internet: Monograph / A.A. Aleinikov, K.Z. Bilyatdinov, A.V. Krasov [et al.]. – Saint Petersburg: Center for Scientific and Information Technologies "Asterion", 2016: 305 ISBN 978-5-00045-408-4. – EDN XGTJLL. (In Russ)
17. Mayorov, A.V. Architecture and software implementation of a system for detecting computer attacks in corporate and government information systems based on intelligent analysis methods / A.V. Mayorov. *Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and technical sciences*. 2023; 2: 40-46. – DOI 10.46418/2079-8199_2023_2_8. – EDN HEPDFF. (In Russ)
18. Sharikov P.I., Tsvetkov A.Yu., Sigacheva V.V., Sirotina L.K. Research and algorithm for preventing exploitation of vulnerabilities of the Log4j logging library in Java application information systems. *Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and Technical Sciences*. 2023; 4:100-106. – DOI 10.46418/2079-8199_2023_4_19. – EDN BULSOH. (In Russ)
19. Sharikov P.I. Methodology for obfuscation of Java application bytecode in order to protect it from decompilation attacks. *Bulletin of the St. Petersburg State University of Technology and Design. Series 1: Natural and Technical Sciences*. 2022;1:64-72. DOI 10.46418/2079-8199_2022_1_10. – EDN AUOFNA. (In Russ)
20. Sharikov P.I. Methodology for creating and hidden embedding a digital watermark in the bytecode of a class file based on undeclared capabilities of the Java virtual machine. *Modern science: current problems of theory and practice. Series: Natural and technical sciences*. 2023;7-2:165-174. – DOI 10.37882/2223-2982.2023.7-2.37. – EDN YBEWYQ. (In Russ)

Сведения об авторах:

Шариков Павел Иванович, кандидат технических наук, старший преподаватель, кафедра защищенных систем связи; sharikov.pavel@ro.ru; ORCID: 0000-0003-3996-9217

Красов Андрей Владимирович, кандидат технических наук, доцент, кафедра защищенных систем связи; krasov@inbox.ru; ORCID: 0000-0002-9076-6055

Майоров Александр Владимирович, аспирант, кафедра защищенных систем связи; avmayorov@bk.ru

Information about authors:

Pavel I. Sharikov, Cand. Sci. (Eng), Senior Lecturer, Department of Secure Communication Systems; sharikov.pavel@ro.ru; ORCID: 0000-0003-3996-9217

Andrey V. Krasov, Cand. Sci. (Eng), Assoc. Prof., Department of Secure Communication Systems; krasov@inbox.ru; ORCID: 0000-0002-9076-6055

Alexander V. Mayorov, Graduate Student, Department of Secure Communication Systems; avmayorov@bk.ru

Конфликт интересов/Conflict of interest.

Авторы заявляют об отсутствии конфликта интересов/The authors declare no conflict of interest.

Поступила в редакцию/Received 17.11.2024

Одобрена после/рецензирования Reviced 20.12.2024.

Принята в печать/Accepted for publication 12.01.2025.