УДК 004.056 doi: 10.15827/0236-235X.151.426-437 2025. Т. 38. № 3. С. 426–437

Обнаружение аномалий в контейнерных системах: применение частотного анализа и гибридной нейронной сети

И.В. Котенко [1]✉, М.В. Мельник [1]

[1] Санкт-Петербургский Федеральный исследовательский центр РАН, г. Санкт-Петербург, 199178, Россия

Ссылка для цитирования
Котенко И.В., Мельник М.В. Обнаружение аномалий в контейнерных системах: применение частотного анализа и гибридной нейронной сети // Программные продукты и системы. 2025. Т. 38. № 3. С. 426–437. doi: 10.15827/0236-235X.151.426-437

Информация о статье
Группа специальностей ВАК: 2.3.3
Поступила в редакцию: 27.12.2024 После доработки: 11.02.2025 Принята к публикации: 03.03.2025

Аннотация. Постоянное совершенствование атак на средства контейнеризации, оркестрации и приложения, функционирующие в контейнерных системах, ставят под угрозу успешное внедрение таких систем. Большинство современных атак сопровождаются аномальным поведением как отдельных процессов, так и всей системы в целом. В статье рассматривается проблема обнаружения аномального поведения в контейнерных системах на основе методов глубокого машинного обучения. Представлен подход к обнаружению аномальных последовательностей процессов, выполняемых внутри контейнерных систем на основе профилирования их поведения. Подход основан на частотном анализе дизассемблированных машинных инструкций, на построении гистограмм процессов фиксированного размера и на использовании их как для обучения гибридной модели нейронной сети Autoencoder – Long short-term memory, так и для последующего обнаружения. Гистограммы создаются путем подсчета количества дизассемблированных инструкций с применением частотного анализа, который основан на вычислении отношения количества определенных инструкций к общему числу инструкций. После обучения на профиле, отражающем нормальное поведение контейнера, нейронная сеть готова к обнаружению аномальных последовательностей процессов методом вычисления ошибки реконструкции. Затем тестовая последовательность, состоящая из гистограмм процессов, подается в нейронную сеть, вычисляется ошибка реконструкции входного вектора данных и сравнивается с заданным порогом. Результаты, полученные в ходе проведенного эксперимента, демонстрируют высокую точность обнаружения и низкие затраты вычислительных ресурсов на обучение моделей нейронной сети. Ложные срабатывания находятся на низком уровне, что, в свою очередь, позволяет использовать предлагаемое решения в качестве дополнительного средства обнаружения аномальной активности. С помощью предложенного подхода можно эффективно выявлять атаки, при которых злоумышленники перехватывают выполнение программного кода, изменяют его поведение и манипулируют адресами функций в бинарных файлах.

Ключевые слова: контейнерные системы, обнаружение аномалий, гистограммы процессов, нейронные сети, кибербезопасность

Благодарности. Работа выполнена при частичной финансовой поддержке бюджетной темы FFZF-2025-0016

Введение. Сложные вычислительные системы крайне требовательны к оптимизации вычислительных ресурсов и эффективному управлению компонентами инфраструктуры. В условиях растущих объемов информации это является ключевым аспектом для достижения максимальной производительности и устойчивости как отдельных компонентов, так и всей инфраструктуры в целом.

При создании таких систем, как правило, используется технология контейнеризации как альтернатива виртуализации на основе гипервизора. Данная технология позволяет добиться более высокой производительности, чем в традиционных системах, она также обеспечивает эффективное управление приложениями внутри контейнерных систем.

Динамическое распределение ресурсов и масштабируемость как основа технологии контейнеризации привели к ее широкому применению в современных центрах обработки данных и облачных вычислениях [1]. Она упрощает развертывание, миграцию и управление приложениями, повышая их отказоустойчивость, гибкость и скорость работы.

Существует множество уязвимостей и факторов риска [2], относящихся к контейнерным системам. Например, одни свойственны средствам контейнеризации и оркестрации, другие – ядру операционной системы, третьи – образу контейнера. Кроме того, дополнительные риски вносят неправильные конфигурации, которые могут возникать из-за некорректно настроенных возможностей, таких как CAP_SYS_ADMIN, CAP_NET_ADMIN, CAP_SETPCAP и др. [3].

Эти проблемы безопасности создают серьезные риски, которые ставят под угрозу успешное внедрение контейнерных систем и технологии контейнеризации в целом. Проблемы безопасности возникают в основном из-за специфики архитектуры контейнерных систем, таких как изоляция процессов, управление ресурсами и взаимодействие между контейнерами, что требует особого внимания при разработке и эксплуатации. Следует отметить, что проблемы безопасности связаны с постоянно развивающимися векторами атак, а появление новых уязвимостей в данной области оставляет открытым вопрос безопасности таких систем.

Резкий рост числа атак, их сложности и интеллектуальности способствовал использованию методов ИИ для обнаружения вредоносного воздействия на инфраструктуру. Эти методы показали высокую эффективность при реализации процессов анализа, обнаружения и классификации как известных, так и неизвестных вредоносных воздействий.

Для обнаружения атак многие исследователи основываются на так называемом превентивном обнаружении. В результате задача сводится к обнаружению аномалий, которые, в свою очередь, могут свидетельствовать о деструктивном воздействии на инфраструктуру.

Главная идея обнаружения аномального поведения в контейнерных системах основывается на технике профилирования поведения. Обнаружение осуществляется за счет того, что аномальное поведение существенно отличается от нормального благодаря правильно выбранным признакам. В качестве признаков могут выступать имена процессов и системных вызовов, их параметры, IP-адреса, размер сетевого пакета, флаги пакета, значения потребления вычислительных ресурсов или операций ввода-вывода и другие.

В [4] используется модель нейронной сети LSTM (*Long short-term memory*, долгая краткосрочная память) для обнаружения аномалий путем определения разности предсказанной последовательности системных вызовов по отношению к действительной. В [5] применяется модель автокодировщика (*Autoencoder*, AE) для классификации последовательности системных вызовов на основе вычисления ошибки реконструкции данных и на ее сравнении с заданным пороговым значением.

В работе [6] предложено использовать гибридную нейронную сеть на основе самоорганизующейся карты Кохонена (*Self-Organizing Map*, SOM) и многослойного персептрона (*Multilayer Perceptron*, MLP) для обнаружения аномалий сетевого трафика путем анализа служебной информации сетевых пакетов. В [7] предлагается платформа Deep Packet для классификации сетевого трафика с использованием глубокого AE и одномерной сверточной нейронной сети (*Convolutional Neural Network*, CNN) на основе данных полезной нагрузки и заголовков пакетов сетевого трафика.

Система обнаружения и диагностики аномалий в контейнерах на основе применения методов машинного обучения для анализа нагрузки на вычислительные ресурсы рассматривается в [8]. В [9] исследовано применение различных методов машинного обучения для обнаружения аномалий в показателях производительности.

В работе [10] предложен метод обнаружения вредоносного ПО, основанный на использовании графовых нейронных сетей для анализа данных исполняемых файлов, а в [11] – система MalpMiner (*Malware-Logic-Miner*), которая использует логическое программирование для представления поведения исполняемого двоичного файла с целью обнаружения вредоносного ПО.

Следует отметить, что технология дизассемблирования бинарных файлов, хотя и может эффективно использоваться при анализе вредоносных бинарных файлов, в исследованиях по обнаружению аномальных последовательностей процессов не является ключевым методом извлечения признаков для алгоритмов машинного обучения.

Предлагаемый в настоящей статье подход к обнаружению аномалий базируется на поведенческом анализе, используемом для создания эталонного профиля каждой контейнерной системы. Профиль строится на основе последовательностей гистограмм процессов, выполненных внутри контейнера. Как только профиль контейнера построен, он передается в модель неконтролируемой нейронной сети AE-LSTM для обучения. Обнаружение аномальных последовательностей процессов выполняется путем вычисления ошибки реконструкции входного вектора данных по отношению к заданному порогу. Ошибка реконструкции – это мера, которая используется для оценки работы AE и вычисляется как разница между оригинальными и восстановленными данными. Если ошибка реконструкции превышает заданный предел, то регистрируется аномалия.

Новизной и особенностью предлагаемого решения является построение гистограмм процессов фиксированного размера на основе подсчета инструкций дизассемблированных бинарных файлов выполненных процессов и применения к ним методики частотного анализа, а также использования неконтролируемой модели глубокого машинного обучения AE-LSTM для обнаружения.

Релевантные работы

Одним из ключевых методов анализа вредоносных файлов является дизассемблирование бинарных файлов, которое используется при статическом анализе. Статическое дизассемблирование представляет собой метод обратного инжиниринга, позволяющий исследовать бинарные файлы без их выполнения [11]. При работе с вредоносным ПО или подозрительными приложениями этот подход особенно значим, так как он исключает необходимость запуска потенциально опасного кода. В отличие от статического анализа, динамический анализ требует использования песочницы (контролируемой среды выполнения кода), что добавляет новые риски и угрозы, а также требует использования дополнительных вычислительных ресурсов.

В [12] предложено извлекать исполняемые двоичные коды операций в сочетании с использованием модели рекуррентной нейронной сети для обнаружения программ-вымогателей (Ransomware). В [13] образцы программ дизассемблируются в язык ассемблера, а затем извлекается информация о кодировании вектора признаков, которая отражает вызовы API (*Application programming interface*) и байтовый код.

В [14] для обнаружения образцов вредоносных ПО предложен подход на основе *n*-грамм. Следует отметить, что, хотя данный подход и демонстрирует высокие показатели точности, он требует значительных затрат вычислительных ресурсов и времени. В [15] для выбора признаков используется Q-learning, предоставляя признаки, управляемые форматом двоичного файла и последовательностями байтов, затем обученный агент ищет оптимальный набор признаков. В [16] предложено анализировать последовательность вызовов API, кодируя каждый API в числовую форму, а затем применять эту последовательность для обучения модели нейронной сети LSTM. Точность предложенного метода достигла 96,7 %.

В [17] представлено исследование по обнаружению вредоносных ПО с использованием разреженного AE, у которого в функцию потерь добавлен штраф за величины значений в коде. SAE обрабатывал вызовы Windows API, извлеченные из файлов *Portable Executable* (PE, переносимый исполняемый). Результаты показали, что модель с тремя скрытыми слоями и 100 нейронами в каждом слое демонстрирует наилучшие показатели точности как на этапе обучения, так и на этапе тестирования, превосходя такие методы, как ANN (*Artificial Neural Network*), SVM (*Support Vector Machine*), наивный байесовский алгоритм (*Naive Bayes*, NB) и дерево решений (*Decision Tree*, DT).

В [18] предложено применять большие языковые модели (*Large Language Model*, LLM) для анализа дизассемблированных инструкций исполняемых файлов с целью классификации их принадлежности к вредоносному ПО.

В результате проведенного анализа можно сделать следующие выводы: 1) несмотря на то, что технология дизассемблирования и применяется для сбора данных при статистическом анализе вредоносных файлов, она не используется в задачах обнаружения аномальной активности в последовательностях процессов, выполняемых внутри контейнерных систем; 2) системы, основанные на методах машинного обучения, показывают хорошие результаты по эффективности выявления атак и аномалий, но остро стоит вопрос правильности выбора признаков и выбора эффективных признаков.

Авторское решение поставленной задачи

Предлагаемый подход основан на сборе и анализе ассемблерных инструкций дизассемблированных бинарных файлов, которые используются в качестве входного вектора данных для неконтролируемой модели нейронной сети. Перед подачей входного вектора данных в нейронную сеть данные подвергаются частотному анализу, и формируются гистограммы процесса.

Для преодоления трудности явной разметки данных было решено использовать модель AE-LSTM. Модель нейронной сети была обучена на данных нормальной активности. Благодаря AE-LSTM можно устранить проблему явной разметки данных, поскольку такая разметка практически невозможна в реальных условиях. Предлагаемое решение состоит из модулей обучения и обнаружения, оно реализует этапы сбора данных, нормализации, обучения и обнаружения аномалий.

Следует отметить, что изначально для обнаружения аномалий планировалось использовать модель нейронной сети на основе AE (как в [1]), поскольку архитектура AE позволяет эффективно выявлять отклонения. Однако результаты проведенного эксперимента показали, что добавление слоя LSTM к модели значительно увеличивает точность (до 12 %).

Кроме того, в ходе анализа различных моделей нейронных сетей для задачи обнаружения аномалий AE на основе LSTM продемонстрировали преимущества по сравнению с другими архитектурами, такими, как чистые LSTM, CNN и MLP благодаря своей способности эффективно извлекать значимые признаки из последовательных данных. Используя механизм AE, AE-LSTM не только снижают размерность данных, но и повышают устойчивость к шуму, что позволяет им более точно различать нормальные и аномальные паттерны. Это особенно важно в условиях, когда данные могут содержать выбросы или шум, затрудняющие процесс обнаружения аномалий. В отличие от чистых LSTM, которые могут ограничиваться в способности к извлечению признаков, и CNN, не всегда подходящих для анализа последовательных данных, нейронная сеть AE-LSTM обеспечивает более глубокое понимание структуры данных за счет комбинирования преимуществ двух моделей нейронных сетей.

Таким образом, AE-LSTM представляют собой мощный инструмент для анализа и обнаружения аномалий, обеспечивая высокую точность при правильной настройке и обучении.

Сбор данных о файлах в контейнерных системах

Первый этап предполагает сбор информации о путях к бинарным файлам, выполненных внутри контейнерных систем с помощью приложения Falco Security. Помимо абсолютных путей, также собираются имя пользователя, идентификатор и имя контейнера, идентификатор и имя процесса, абсолютный путь к бинарному файлу процесса.

Когда данные собраны, формируются последовательности с заданным шагом, определяющим количество процессов. Шаг последовательности выбирается исходя из наилучшего результата обнаружения. Таким образом, процедуру формирования последовательностей можно представить в виде арифметической прогрессии:

$$a_n = a_1 + d(n - 1),$$

где a_n – n-й член последовательности; a_1 – первый член последовательности (начальное значение); d – шаг последовательности (в данном случае выбрано значение 3); n – номер члена последовательности.

После сбора данных выполняется процедура дизассемблирования машинных инструкций бинарных фалов с помощью утилиты objdump.

Выбор Falco Security и objdump обусловлен широким набором параметров и гибкой системой настройки правил, что позволяет эффективно проводить сбор и анализ данных.

Нормализация данных для формирования гистограмм процессов

На этапе нормализации, проходящем за четыре шага, выполняется формирование гистограмм процессов.

На первом шаге формируется словарь ассемблерных инструкций. Сформированный словарь записывается в файл instructions.txt. Поскольку особенностью модели нейронной сети является ее структура, в которой скрытый слой содержит вдвое меньше нейронов, чем входной и выходной слои, то размер входного и выходного слоев нейронной сети должен быть кратным 2 и соответствовать количеству строк в файле instructions.txt. Если количество строк в файле instructions.txt некратно 2, то в конец файла будет записана строка null. Стоит отметить, что формирование словаря задействовано исключительно в модуле обучения.

На втором шаге формируются гистограммы процессов, то есть массив элементов, размерность которого равна количеству строк в instructions.txt. Каждый элемент массива по умолчанию заполняется нулем. Процедура формирования гистограммы начинается с чтения файла с набором данных, в котором содержатся последовательности процессов. Строки в файле читаются последовательно, и, как только встречается строка формата /bin/ls: file format elf64-x86-64, начинается новый процесс. Построчное считывание останавливается, и начинается процедура заполнения гистограммы, например: пусть в рамках процесса, процедура mov выполнилась 20 раз, тогда в ячейку сформированной гистограммы, чей индекс равен индексу процедуры “mov” из файла instructions.txt, записывается значение, полученное в результате частотного анализа процедуры “mov”. В результате частотного анализа каждый счетчик инструкции делится на общее количество инструкций в процессе. Представим сформированные последовательности гистограмм процессов:

Последовательность 1
[[[0.014215, 0.144013, 0.030473, 0.023090, 0.026598, 0.028274, 0.014268, 0.005341, ...]
[0.006104, 0.145452, 0.039899, 0.018758, 0.020098, 0.030817, 0.010719, 0.003573, ...]
[0.008659, 0.165890, 0.023307, 0.016169, 0.020313, 0.053225, 0.011258, 0.002575, ...]]]
Последовательность 2
[[[0.006104, 0.145452, 0.039899, 0.018758, 0.020098, 0.030817, 0.010719, 0.003573, ...]
[0.008659, 0.165890, 0.023307, 0.016169, 0.020313, 0.053225, 0.011258, 0.002575, ...]
[0.005774, 0.145388, 0.038840, 0.016402, 0.020863, 0.027949, 0.011547, 0.003412, ...]]]
Последовательность 3
[[[0.008659, 0.165890, 0.023307, 0.016169, 0.020313, 0.053225, 0.011258, 0.002575, ...]
[0.005774, 0.145388, 0.038840, 0.016402, 0.020863, 0.027949, 0.011547, 0.003412, ...]
[0.006268, 0.168900, 0.033007, 0.026605, 0.027139, 0.039675, 0.009202, 0.005668, ...]]]

На третьем шаге после завершения формирования гистограммы значения в гистограммах могут находиться в разных масштабах, что значительно усложнит обучение или вовсе приведет к ошибкам во время обучения модели нейронной сети. Поэтому для решения этой задачи использовалась функция MinMaxScaler языка Python, что позволило привести все числовые данные в одинаковый масштаб. После этого гистограмма записывается в файл, считывание файла возобновляется и формируется гистограмма для нового процесса. Следует отметить, что каждому контейнеру будет соответствовать уникальный файл сформированных гистограмм процессов – это и будет профиль контейнера.

На четвертом шаге выполняется удаление последовательностей, которые встречаются более одного раза, то есть из двух и более тождественных последовательностей останется только одна. Таким образом, устраняется избыточность данных и существенно сокращается время обучения модели нейронной сети. Результаты записываются в файл data.txt.

Этап процесса нормализации представлен на рисунке 1.

Обучение модели нейронной сети

В основе предлагаемого решения лежит неконтролируемая модель гибридной нейронной сети AE-LSTM. Процесс ее обучения построен на данных нормальной активности, что позволят выявлять аномальную активность посредством анализа величины ошибки реконструкции тестовых данных по отношению к исходным данным. Если ошибка реконструкции входного вектора тестовых данных превышает заданный порог допустимого значения, то регистрируется аномалия, которая свидетельствует об аномальности в данных.

На этапе нормализации предобработанные данные передаются в модель нейронной сети для обучения. Модель нейронной сети AE-LSTM представлена на рисунке 2.

После загрузки данных энкодер сжимает входные данные в скрытое представление, размерность которого определяется как ROWS * LINES / 2 (скрытый слой нейронной сети, который содержит вдвое меньше нейронов, чем входной и выходной слои). Затем декодер восстанавливает данные из скрытого представления. Таким образом, модель обучается минимизировать ошибку реконструкции во время обучения.

Процесс обучения модели нейронной сети основан на функции автономной настройки параметров обучения. Используются следующие параметры: размер пакета, по умолчанию равный 1, и количество эпох, которое определяется результатами обучения. Начальное количество эпох – 1. Если уровень потерь после обучения превышает 10 %, модель считается необученной, и количество эпох увеличивается. Этот процесс повторяется до тех пор, пока уровень потерь не составит менее 10 % или количество эпох не достигнет 10. Если обучение оказывается неудачным, модель выбирает наилучший показатель потерь и переобучается с параметрами, соответствующими этому уровню потерь.

Выбор такого уровня потерь при обучении модели обусловлен необходимостью достижения оптимального баланса между точностью модели и количеством ложных срабатываний. Эксперименты показали, что увеличение уровня потерь приводит к снижению точности модели, тогда как чрезмерное уменьшение уровня потерь увеличивает количество ложных срабатываний. Таким образом, получается уровень потерь, который минимизирует процент ложных срабатываний, обеспечивая адекватный уровень точности модели.

Следует отметить, в ходе эксперимента было обнаружено, что при подаче в модель нейронной сети одного тренировочного объекта между вычислениями функции потерь, модель обучается лучше, демонстрируя меньшие значения потерь и более высокую точность, чем при использовании другого количества тренировочных объектов.

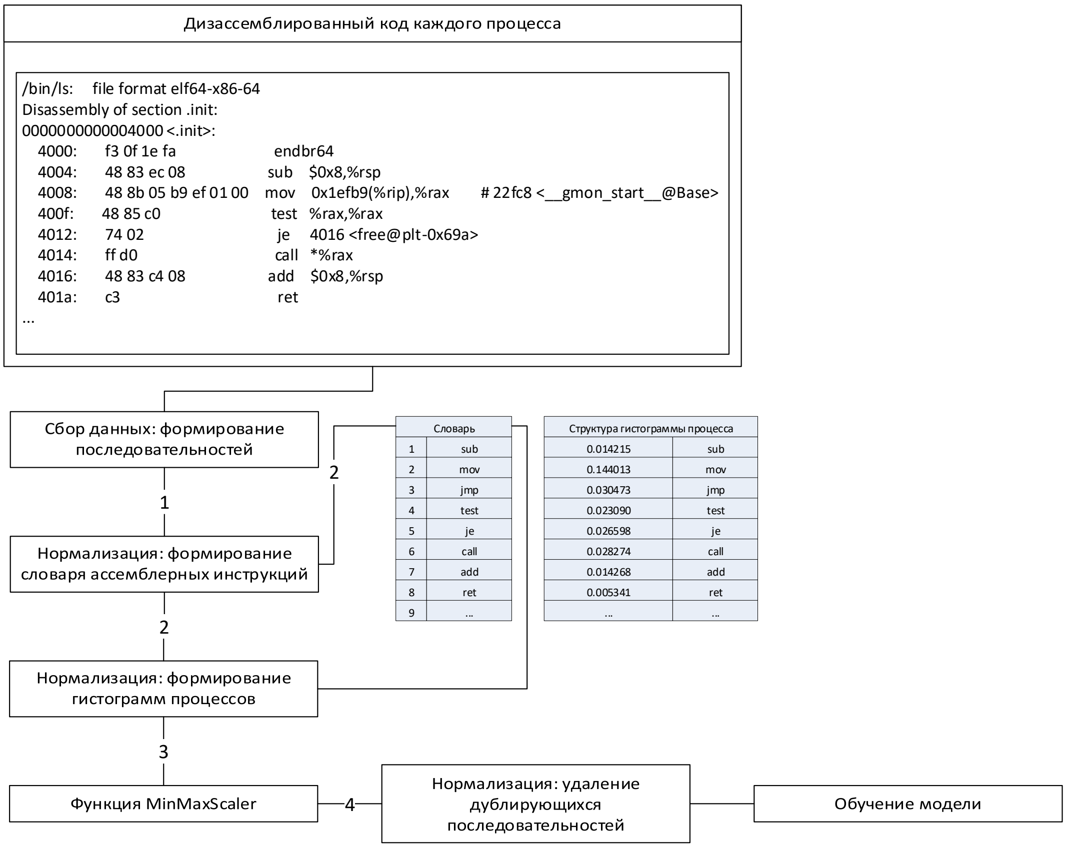

Рис. 1. Схема этапа нормализации

Fig. 1. Normalization stage diagram

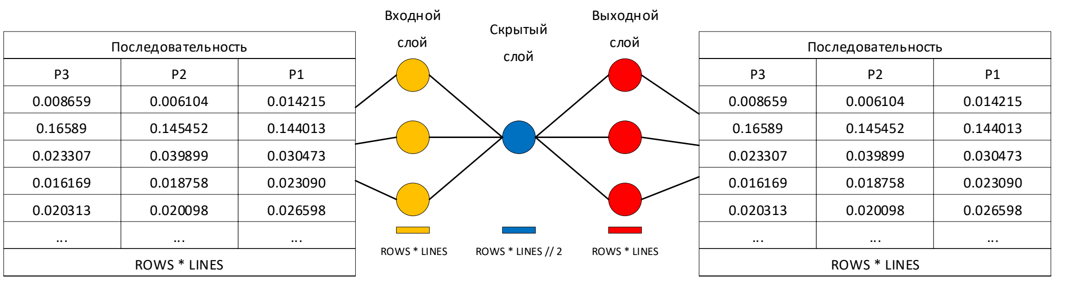

Рис. 2. Модель нейронной сети AE-LSTM

Fig. 2. AE-LSTM neural network model

В данной работе была выбрана комбинация оптимизатора Adam и функции потерь MSE для компиляции модели нейронной сети. Оптимизатор Adam обеспечивает быструю и стабильную сходимость благодаря адаптивному изменению шага обучения, что минимизирует необходимость в тонкой настройке гиперпараметров. Функция потерь MSE, будучи чувствительной к большим ошибкам и обладая хорошей интерпретируемостью, позволяет эффективно оценивать качество предсказаний. Так, синергия между Adam и MSE, обеспечивает высокую точность и устойчивость модели. В качестве функции активации была выбрана ReLU (*Rectified Linear Unit*, выпрямленный линейный блок), так как она обеспечивает быстрое выполнение операций и минимизирует затраты вычислительных ресурсов. Функция ReLU эффективно решает проблему затухания градиента, что позволяет моделям обучаться быстрее и повышает их производительность, особенно в глубоких нейронных сетях. Кроме того, функция способствует разреженности активации, активируя только значимые нейроны, что делает модели более компактными и эффективными.

Процесс обнаружения аномалий в контейнерных системах

Этап обнаружения предполагает передачу тестовых данных, представленных в виде последовательности гистограмм процессов, в модуль обнаружения для анализа величины ошибки реконструкции. Если результат анализа ошибки реконструкции входных данных превышает допустимое значение, модуль обнаружения укажет на исходные последовательности процессов, в которых содержатся аномальные данные.

Следует отметить, что порог ошибки реконструкции (threshold) рассчитывается на основе показателя среднеквадратической ошибки (*Mean Squared Error*, MSE) между исходными и данными после реконструкции:

$$threshold = \mu MSE + \sigma MSE,$$

где μMSE – среднее значение MSE по всем примерам в наборе данных; σMSE – стандартное отклонение MSE.

Если значение MSE для нового примера в наборе данных превышает заданный порог, то пример считается аномальным.

Выбор порогового значения ошибки реконструкции, основанного на MSE, является наилучшим решением, поскольку MSE обладает рядом преимуществ: чувствительность к отклонениям, простота интерпретации и вычисления, хорошие статистические свойства и универсальность применения. Установка порогового значения на основе статистического анализа распределения ошибок реконструкции позволяет эффективно различать нормальные и аномальные данные, обеспечивая высокую точность и надежность модели.

Особенности проведения эксперимента. Испытательный стенд

Для экспериментальной оценки предлагаемого подхода разработан испытательный стенд, состоящий из двух Docker-контейнеров: первый контейнер представлен сервисами ssh и rdp, второй является почтовым сервером Postfix. Стенд реализует активность, приближенную к настоящей, выполняющую имитационное моделирование, которое отражает поведение системы. Такая активность подразумевает нормальную, аномальную и вредоносную. Испытательный стенд представлен на рисунке 3.

Коллектор Falco Security с помощью модуля ядра eBPF собирает данные о процессах, после данные передаются в утилиту objdump для дизассемблирования машинных инструкций бинарных файлов. Собранные данные посылаются на машину с Windows 11 для обучения или обнаружения. Легитимная, аномальная и вредоносная активность генерируется с компьютера, где установлена Kali Linux на машину с платформой CentOS 9.

Наборы данных

Для создания наборов данных задействованы два контейнера docker с имитацией нормальной активности, которая включает работу почтового сервера Postfix и сервера rdp, а также системное администрирование контейнера. Более подробная информация о сценариях активности и собранных наборах данных представлена в таблице 1.

В таблице наборы данных группы A содержат данные нормальной активности, группы B – нормальной и вредоносной активностей, C – нормальной и аномальной активностей.

Каждый набор данных (группы A, B, C) был сформирован за 2 часа. Действия, выполняемые внутри контейнерных систем, были организованы хаотично для реализации состояния, приближенного к состоянию настоящей системы. Каждый набор данных содержит не менее 700 тыс. записей последовательностей гистограмм процессов. В каждом наборе данных нормальной активности отражена активность 10 пользователей.

Обучение моделей

Процесс обучения моделей проводился на наборах данных легитимной активности norm-sr-s.txt и norm-e-s.txt (группа A). Это позволило создать эталонный профиль поведения для каждого отдельно взятого контейнера. Обучение моделей проводилось в течение одной эпохи с размером пакета 1. В качестве оптимизатора использовался Adam, а функция потерь – MSE. Полученные модели демонстрируют уровень потерь не более 5 %.

Весь процесс обучения занял не более одного часа для каждой модели с учетом времени, потраченного на переобучение. Обучение проводилось на виртуальной машине испытательного стенда со следующими характеристиками: ОС Windows 11, ОЗУ 32 Гб, ЦП

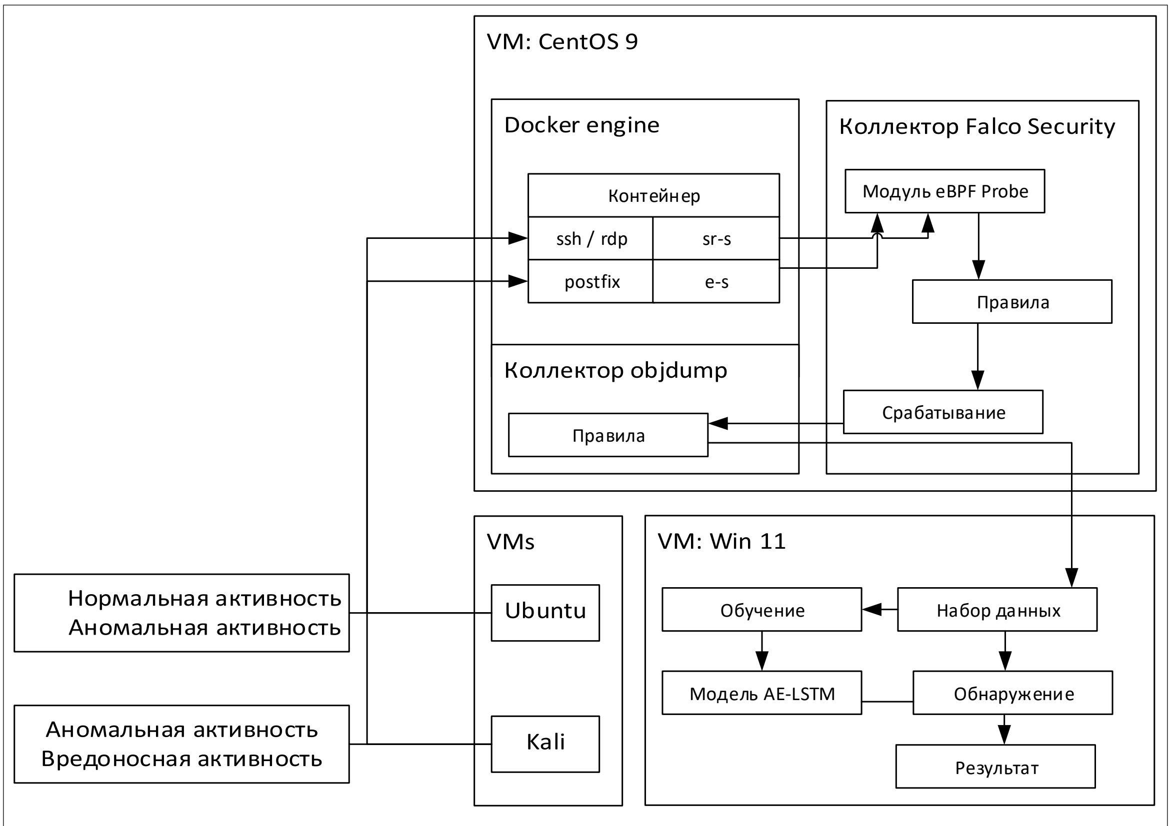

Рис. 3. Испытательный стенд

Fig. 3. Test bench

AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz.

В целом можно отметить неплохую скорость обучения на представленных вычислительных ресурсах. Время, потраченное на обучение, оказалось вдвое меньше времени сбора данных.

Результаты обнаружения

Для экспериментального подтверждения работоспособности и эффективности разработанного прототипа оценивалась каждая обученная модель на трех наборах данных с помощью Confusion Matrix. Следует отметить, что оценивались только два состояния: норма и аномалия. Поскольку в реализованном прототипе отсутствует возможность обнаружения атак, любая атака рассматривалась как аномалия. Результаты обнаруженных аномалий представлены в таблице 2.

Анализ результатов

Архитектура AE-LSTM обладает уникальными особенностями, которые делают эту нейронную сеть адекватным выбором для обнаружения аномалий в контейнерных системах. Она позволяет разработать легковесную и высокоточную модель, способную эффективно выявлять аномалии. Высокая эффективность достигается благодаря использованию сети LSTM, которая отлично справляется с обработкой последовательных данных, а также благодаря AE, сокращающему объем информации удалением избыточных данных, которые могут быть восстановлены в дальнейшем. Этот подход обеспечивает надежное и быстрое обнаружение аномалий, что особенно важно в динамичных контейнерных средах.

Научная новизна и особенность представленной работы заключаются в построении гистограмм процессов для профилирования контейнерных систем и в использовании их в качестве входных данных для модели нейронной сети AE-LSTM.

В отличие от существующих подходов, основанных на правилах [19], на анализе показателей производительности [8], на анализе системных вызовов процессов [5], предлагаемый подход позволяет получить более эффективное решение.

Согласно результатам проведенных экспериментов предложенный прототип демонстри-

Таблица 1

Сценарии легитимной, аномальной и вредоносной активности, наборы данных

Table 1

Legitimate, anomalous and malicious activity scenarios, datasets

Контейнер	Группа наборов данных	Набор данных	Описание
sr-s (ssh- и rdp-сервер)	A	norm-sr-s.txt	Выполнение команд операционной системы; работа с помощью rdp-клиента
e-s (postfix-сервер)		norm-e-s.txt	Отправка email-сообщений
sr-s (ssh- и rdp-сервер)	B	anom-sr-s.txt	Аномальная последовательность команд операционной системы; добавление задачи в task manager; создание alias; выполнение команд операционной системы; работа с помощью rdp-клиента
e-s (postfix-сервер)		anom-e-s.txt	Запуск неизвестного файла вложения почтового сообщения; остановка и запуск контейнера; отправка email-сообщений
sr-s (ssh- и rdp-сервер)	C	malic-sr-s.txt	Запуск майнингового ПО; загрузка и запуск шифровальщика; запуск Linux local Privilege Escalation; поиск возможных путей повышения привилегий (linPEAS); выполнение команд операционной системы; работа с помощью rdp-клиента
e-s (postfix-сервер)		malic-e-s.txt	Запуск майнингового ПО; загрузка и запуск шифровальщика; DOS; отправка email-сообщений

Таблица 2

Результаты проведенного эксперимента

Table 2

Experimental results

Группа наборов данных	Набор данных	True		False	
		Positive	Negative	Positive	Negative
A	norm-sr-s.txt	0	122 519	0	0
	norm-e-s.txt	0	98 891	0	0
B	anom-sr-s.txt	673	128 398	378	415
	anom-e-s.txt	714	126 439	316	426
C	malic-sr-s.txt	599	131 592	337	487
	malic-e-s.txt	613	129 874	431	365

рует высокие показатели точности, что позволяет применять данное решение не только в качестве дополнительного средства обнаружения аномального поведения в контейнерных системах, но и как основное.

К недостаткам данного подхода можно отнести невозможность обнаружения большинства сетевых атак. Например, атаки MITM, Mailbombing, IP-спуфинг и тому подобные с большой долей вероятности не будут обнаружены, а SQL и PHP определятся. Однако факт поиска уязвимостей и недостатков безопасности для реализации этих атак также не обнаружится, поскольку в большинстве случаев не будет отличаться от легитимного поведения. Следует отметить, что использование представленного подхода в качестве решения для всех задач обнаружения не является корректным.

Заключение

В статье представлен подход к обнаружению аномальной активности в контейнерных системах, основанный на применении частотного анализа к дизассемблированным инструкциям бинарных файлов выполняемых процессов. Основная идея заключается в создании легитимных профилей поведения контейнерных систем с использованием моделей нейронной сети AE-LSTM. В качестве входных данных для нейронной сети используются гистограммы процессов. Этот подход позволяет эффективно выявлять аномальные и ранее неизвестные последовательности процессов.

Особенностью представленного решения является использование алгоритма нормализации для построения гистограмм процессов фиксированного размера на основе частотного анализа. Такая нормализация позволяет существенно сократить набор данных, что положительно сказывается на скорости обучения модели нейронной сети.

Разработанный прототип, реализующий предложенный подход, был исследован с использованием ряда экспериментов. Результаты экспериментов продемонстрировали его следующие свойства:

– высокая точность обнаружения аномальных последовательностей процессов;

– низкий уровень ложных срабатываний – данный показатель обеспечивается использованием окна захвата последовательностей, размер которого составляет три процесса. Следует отметить, что увеличение размера окна захвата сетевых пакетов может повысить точность обнаружения аномальных последовательностей процессов, однако это также приводит к увеличению времени обучения модели нейронной сети;

– неспособность анализа выполнения программ, написанных на таких языках, как Java или Python. Python – интерпретируемый язык, Java компилируется в байт-код, который не является машинным.

Предлагаемый подход может служить в ряде случаев в качестве основного и являться эффективным дополнением для обнаружения аномалий в контейнерных системах. Однако на основании проведенных экспериментов рекомендуется использовать его как дополнительный инструмент, который расширяет функциональные возможности основного средства обнаружения аномалий. Кроме того, важно внедрить дополнительный инструмент, специально предназначенный для выявления атак и аномалий в пакетах сетевого трафика. Это позволит значительно повысить уровень безопасности и надежности системы.

В будущих исследованиях предполагается проанализировать влияние различных моделей нейронных сетей на процесс обучения. Основная цель в предстоящих исследованиях заключается в оптимизации вычислительных ресурсов и сокращении времени, необходимого для обучения модели при сохранении высокого уровня точности.

Также планируется усовершенствовать разработанный алгоритм нормализации, увеличив окно захвата гистограмм процессов. Кроме того, ведется работа по реализации дополнительного алгоритма, который позволит сократить объем данных. Этот алгоритм будет основываться на концепции запоминания ключевых блоков данных в гистограммах, а также на их расположении. Предполагается, что такие изменения приведут к повышению скорости анализа данных и к уменьшению общего объема обрабатываемой информации.

Список литературы

1. Kotenko I.V., Melnik M.V., Abramenko G.T. Anomaly detection in container systems: Using histograms of normal processes and an autoencoder. Proc. IEEE 25th Int. Conf. of Young Professionals in EDM, 2024, pp. 1930–1934. doi: 10.1109/EDM61683.2024.10615118.
2. Jain V., Singh B., Khenwar M. et al. Static vulnerability analysis of docker images. IOP Conf. Ser.: Mater. Sci. Eng., 2021, vol. 1131, no. 1, art. 012018. doi: 10.1088/1757-899X/1131/1/012018.
3. Rahman A., Shamim S.I., Bose D.B. et al. Security misconfigurations in open source kubernetes manifests: An empirical study. ACM TOSEM, 2023, vol. 32, no. 4, pp. 1–36. doi: 10.1145/3579639.
4. Snehi J., Bhandari A., Baggan V. et al. AIDAAS: Incident handling and remediation anomaly-based IDaaS for cloud service providers. Proc. IEEE SMART, 2021, pp. 356–360. doi: 10.1109/SMART52563.2021.9676296.
5. El Khairi A., Caselli M., Knierim C. et al. Contextualizing system calls in containers for anomaly-based intrusion detection. Proc. CCSW'22, 2022, pp. 9–21. doi: 10.1145/3560810.3564266.
6. Болодурина И.П., Парфёнов Д.И. Исследование модели нейронной сети для обеспечения безопасности и качества обслуживания мультиоблачной платформы // Информационные и математические технологии в науке и управлении. 2018. № 3. С. 18–26.
7. Lotfollahi M., Jafari S.M., Zade R.S.H. et al. Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Comput., 2020, vol. 24, no. 3, pp. 1999–2012. doi: 10.1007/s00500-019-04030-2.
8. Du Q., Xie T., He Y. Anomaly detection and diagnosis for container-based microservices with performance monitoring. Algorithms and Architectures for Parallel Proc., 2018, pp. 560–572. doi: 10.29007/43km.
9. Ramamoorthi V. Machine learning models for anomaly detection in microservices. JETI, 2020, vol. 5, no. 1, pp. 41–56.

10. Nguyen M.T., Nguyen V.H., Shone N. Using deep graph learning to improve dynamic analysis-based malware detection in PE files. J. of Comput. Virology and Hacking Techniques, 2023, vol. 20, no. 1, pp. 153–172. doi: 10.1007/s11416-023-00505-x.

11. Abdelwahed M.F., Kamal M.M., Sayed S.G. Detecting malware activities with MalpMiner: A dynamic analysis approach. IEEE Access, 2023, vol. 11, pp. 84772–84784. doi: 10.1109/ACCESS.2023.3266562.

12. Gibert D., Mateu C., Planes J. HYDRA: A multimodal deep learning framework for malware classification. Computers & Security, 2020, vol. 95, art. 101873. doi: 10.1016/j.cose.2020.101873.

13. Lim M.J., Kwon Y.M. et al. Efficient algorithm for malware classification: N-gram MCSC. IJCDS, 2020, vol. 9, no. 2, pp. 179–185. doi: 10.12785/ijcds/090204.

14. Fang Z., Wang J., Geng J. et al. Feature selection for malware detection based on reinforcement learning. IEEE Access, 2019, vol. 7, pp. 176177–176187. doi: 10.1109/ACCESS.2019.2957429.

15. Черкасов А.Н., Туркин Е.А. Разработка модели обнаружения вредоносных программ на основе анализа последовательностей API-запросов // Вестн. АГУ. Сер.: Естественно-математич. и технич. науки. 2021. Т. 2. № 281. С. 90–96. doi: 10.53598/2410-3225-2021-2-281-90-96.

16. Xiao F., Lin Z., Sun Y. et al. Malware detection based on deep learning of behavior graphs. Math. Problems in Engineering, 2019, vol. 2019, no. 1, pp. 1–10. doi: 10.1155/2019/8195395.

17. Дудкин Д.М., Кузнецов М.А., Авдосев Н.Г. и др. Применение языковых нейросетевых моделей для обнаружения вредоносного программного обеспечения // Инженерный вестн. Дона. 2024. № 7. С. 1–13.

18. Kotenko I., Saenko I., Chechulin A. et al. Detection of anomalies and attacks in container systems: An integrated approach based on black and white lists. Proc. Int. Conf. IITI’22, 2022, pp. 107–117. doi: 10.1007/978-3-031-19620-1_11.

Software & Systems doi: 10.15827/0236-235X.151.426-437 2025, 38(3), pp. 426–437

Anomaly detection in container systems: Frequency analysis and hybrid neural network

Igor V. Kotenko [1]✉, Maksim V. Melnik [1]

[1] St. Petersburg Federal Research Center of the Russian Academy of Sciences, St. Petersburg, 199178, Russian Federation

For citation

Kotenko, I.V., Melnik, M.V. (2025) ‘Anomaly detection in container systems: Frequency analysis and hybrid neural network’, *Software & Systems*, 38(3), pp. 426–437 (in Russ.). doi: 10.15827/0236-235X.151.426-437

Article info

Received: 27.12.2024 After revision: 11.02.2025 Accepted: 03.03.2025

Abstract. Continuously improving attacks on containerization tools, orchestration, and applications running in container systems threaten the successful implementation of such systems. Most contemporary attacks involve anomalous behavior in both individual processes and the whole system. The paper considers the problem of detecting anomalous behavior in container systems based on deep machine learning methods. It presents an approach for detecting anomalous process sequences within container systems using behavioral profiling techniques. This approach involves three key steps: analyzing frequencies of disassembled machine code instructions, building fixed-length process behavior histograms, and using these histograms for both training an Autoencoder – Long short-term memory hybrid model and performing detection tasks. The system generates histograms by counting disassembled instructions through frequency analysis, which calculates the ratio of specific instruction types to the total instruction count. After training on a profile reflecting normal container behavior, the neural network is ready to detect anomalous sequences of processes based on calculating the reconstruction error. The testing sequence of process histograms is fed into the neural network, which calculates the input vector's reconstruction error and compares it with a predefined threshold. The experimental results demonstrate both high detection accuracy and low computational resource requirements for neural network model training. The level of false positives is low, which allows using the proposed solution as an additional tool for detecting abnormal activity. The proposed approach allows for effective detection of attacks in which intruders intercept the execution of program code, change its behavior and manipulate function addresses in binary files.

Keywords: container systems, anomaly detection, process histograms, neural networks, cybersecurity

Acknowledgements. The research received partial funding through the government-funded program FFZF-2025-0016

References

1. Kotenko, I.V., Melnik, M.V., Abramenko, G.T. (2024) ‘Anomaly detection in container systems: Using histograms of normal processes and an autoencoder’, *Proc. IEEE 25th Int. Conf. of Young Professionals in EDM*, pp. 1930–1934. doi: 10.1109/EDM61683.2024.10615118.

2. Jain, V., Singh, B., Khenwar, M. et al. (2021) ‘Static vulnerability analysis of docker images’, *IOP Conf. Ser.: Mater. Sci. Eng.*, 1131(1), art. 012018. doi: 10.1088/1757-899X/1131/1/012018.

3. Rahman, A., Shamim, S.I., Bose, D.B. et al. (2023) 'Security misconfigurations in open source kubernetes manifests: An empirical study', *ACM TOSEM*, 32(4), pp. 1–36. doi: 10.1145/3579639.

4. Snehi, J., Bhandari, A., Baggan, V. et al. (2021) 'AIDAAS: Incident handling and remediation anomaly-based IDaaS for cloud service providers', *Proc. IEEE SMART*, pp. 356–360. doi: 10.1109/SMART52563.2021.9676296.

5. El Khairi, A., Caselli, M., Knierim, C. et al. (2022) 'Contextualizing system calls in containers for anomaly-based intrusion detection', *Proc. CCSW'22*, pp. 9–21. doi: 10.1145/3560810.3564266.

6. Bolodurina, I.P., Parfenov, D.I. (2018) 'Investigation the neural network model for security and quality of service of a multi-cloud platform', *Computational and Math. Tech. for Sci. and Management*, (3), pp. 18–26 (in Russ.).

7. Lotfollahi, M., Jafari, S.M., Zade, R.S.H. et al. (2020) 'Deep packet: A novel approach for encrypted traffic classification using deep learning', *Soft Comput*., 24(3), pp. 1999–2012. doi: 10.1007/s00500-019-04030-2.

8. Du, Q., Xie, T., He, Y. (2018) 'Anomaly detection and diagnosis for container-based microservices with performance monitoring', *Algorithms and Architectures for Parallel Proc.*, pp. 560–572. doi: 10.29007/43km.

9. Ramamoorthi, V. (2020) 'Machine learning models for anomaly detection in microservices', *JETI*, 5(1), pp. 41–56.

10. Nguyen, M.T., Nguyen, V.H., Shone, N. (2023) 'Using deep graph learning to improve dynamic analysis-based malware detection in PE files', *J. of Comput. Virology and Hacking Techniques*, 20(1), pp. 153–172. doi: 10.1007/s11416-023-00505-x.

11. Abdelwahed, M.F., Kamal, M.M., Sayed, S.G. (2023) 'Detecting malware activities with MalpMiner: A dynamic analysis approach', *IEEE Access*, 11, pp. 84772–84784. doi: 10.1109/ACCESS.2023.3266562.

12. Gibert, D., Mateu, C., Planes, J. (2020) 'HYDRA: A multimodal deep learning framework for malware classification', *Computers & Security*, 95, art. 101873. doi: 10.1016/j.cose.2020.101873.

13. Lim, M.J., Kwon, Y.M. et al. (2020) 'Efficient algorithm for malware classification: N-gram MCSC', *IJCDS*, 9(2), pp. 179–185. doi: 10.12785/ijcds/090204.

14. Fang, Z., Wang, J., Geng, J. et al. (2019) 'Feature selection for malware detection based on reinforcement learning', *IEEE Access*, 7, pp. 176177–176187. doi: 10.1109/ACCESS.2019.2957429.

15. Cherkasov, A.N., Turkin, E.A. (2021) 'Development of a malware detection model based on an analysis of API request sequences', *Vestn. of Adyghe State University. Ser.: Natural-math. and Tech. Sci*., 2(281), pp. 90–96 (in Russ.). doi: 10.53598/2410-3225-2021-2-281-90-96.

16. Xiao, F., Lin, Z., Sun, Y. et al. (2019) 'Malware detection based on deep learning of behavior graphs', *Math. Problems in Engineering*, 2019(1), pp. 1–10. doi: 10.1155/2019/8195395.

17. Dudkin, D.M., Kuznetsov, M.A., Avdosev, N.G. et al. (2024) 'Application of language neural network models for detecting malware', *Engineering J. of Don*, (7), pp. 1–13 (in Russ.).

18. Kotenko, I., Saenko, I., Chechulin, A. et al. (2022) 'Detection of anomalies and attacks in container systems: An integrated approach based on black and white lists', *Proc. Int. Conf. IITI'22*, pp. 107–117. doi: 10.1007/978-3-031-19620-1_11.

Авторы

Котенко Игорь Витальевич [1], д.т.н., профессор, главный научный сотрудник, руководитель лаборатории, ivkote@comsec.spb.ru

Мельник Максим Владимирович [1], аспирант, mkmxvh@gmail.com

[1] Санкт-Петербургский Федеральный исследовательский центр РАН, г. Санкт-Петербург, 199178, Россия

Authors

Igor V. Kotenko [1], Dr.Sci. (Engineering), Professor, Chief Researcher, Head of Laboratory, ivkote@comsec.spb.ru

Maksim V. Melnik [1], Postgraduate Student, mkmxvh@gmail.com

[1] St. Petersburg Federal Research Center of the Russian Academy of Sciences, St. Petersburg, 199178, Russian Federation