Обзор методов выявления аномалий при аудите системных вызовов в ОС

Н.Е. Стельмах, А.В. Козачок

***Аннотация*— В статье представлен систематический обзор современных методов обнаружения аномалий в системных вызовах — критически важном компоненте систем обнаружения вторжений уровня хоста (HIDS). Анализируются подходы: статистические модели, методы машинного и глубокого обучения, гибридные архитектуры и техники анализа последовательностей. Особое внимание уделяется оценке их применимости в актуальных вычислительных средах, таких как устройства интернета вещей (IoT), контейнеры, облачные платформы и мобильные операционные системы. Результаты исследования показывают, что современные методы можно разделить по ключевым характеристикам. Глубокие архитектуры обеспечивают минимальный уровень ложноположительных срабатываний (FPR), однако достигается это ценой значительных вычислительных ресурсов. Гибридные подходы демонстрируют более сбалансированное сочетание точности и эффективности, но при этом могут оставаться уязвимыми к сложным, адаптивным атакам. Ресурсоэффективные решения находят применение в ограниченных средах IoT, однако их эффективность часто снижается из-за чувствительности к дисбалансу данных. Исследование подтверждает, что выбор оптимального метода детерминирован целевой платформой. Для ресурсоограниченных устройств IoT предпочтение отдается легковесным моделям, в то время как в облачных средах максимальную эффективность демонстрируют комплексные решения, способные обеспечить низкий FPR. Ключевыми нерешёнными проблемами остаются высокая частота ложных срабатываний и вычислительная сложность обработки потоков данных в реальном времени.**

***Ключевые слова*— системные вызовы, обнаружение аномалий, FPR, HIDS, машинное обучение, IoT-безопасность, вычислительные среды, глубокое обучение.**

I. Введение

Анализ аномалий в системных вызовах представляет собой ключевой компонент современных систем обнаружения вторжений, в особенности на уровне хоста. В условиях роста сложности кибератак и увеличения количества уязвимостей в программном обеспечении, потребность в эффективных методах детекции аномального поведения приложений приобретает повышенную значимость.

Традиционные сигнатурные методы защиты зачастую демонстрируют неспособность к своевременному реагированию на новые типы угроз, что обусловливает привлекательность подходов, основанных на обнаружении аномалий, для исследователей и практиков в сфере информационной безопасности.

Методы обнаружения аномалий в системных вызовах подразделяются на несколько категорий: статистические подходы, методы машинного обучения, техники анализа последовательностей, подходы, использующие аргументы системных вызовов [1], а также комбинированные решения. В соответствии с позицией Соколова [2], подобные классификации отражают эволюцию методологии построения систем обнаружения аномалий (ADS), где ключевым требованием выступает сочетание адаптивности и обобщающей способности моделей. Каждый из указанных подходов характеризуется собственными преимуществами и ограничениями, что обусловливает необходимость тщательного анализа для выбора оптимального решения в конкретных условиях эксплуатации. Ранние исследования, в частности работа Shen и других [3], продемонстрировали эффективность усовершенствованных моделей конечных автоматов (FSA) для анализа системных вызовов с учетом адресов возврата, что повышало точность детекции; однако дальнейшие изыскания в данном направлении не проводились.

Особое внимание уделяется применению указанных методов в контексте современных вычислительных сред, таких как контейнеризация и виртуализация, где традиционные методы обнаружения аномалий могут сталкиваться с новыми вызовами. Проблема высокой частоты ложноположительных срабатываний сохраняет статус одной из основных трудностей, с которыми сталкиваются разработчики систем обнаружения аномалий.

В рамках данной статьи представлен комплексный анализ существующих методов обнаружения и классификации аномалий в системных вызовах, рассмотрены их преимущества и недостатки, а также предложены направления для дальнейших исследований в данной области. Особый акцент сделан на сопоставлении различных подходов и оценке их применимости в разнородных вычислительных средах.

Целью настоящего исследования является систематизация современных методов обнаружения аномалий в системных вызовах.

II. Методы

Для достижения целей исследования проведен систематический обзор анализ, 39 научных работ,

Статья получена 5 мая 2025.
Стельмах Никита Евгеньевич, РТУ МИРЭА, Москва, Российская федерация (e-mail: stelmach.stels@yandex.ru).
Козачок Александр Васильевич, РТУ МИРЭА, Москва, Российская федерация (e-mail: alex.totrin@gmail.com).

опубликованных с 1999 по 2025 год. Критерии отбора источников:

– Исследования с числовыми метриками разработанных моделей;

– Исследования с анализом конкретных проблем различных методов и возможных подходов улучшения моделей.

Из выбранных источников извлечены данные о следующих аспектах:

– Методы и модели: типы алгоритмов, архитектуры нейронных сетей, используемые техники;

– Тип данных: признаки, извлекаемые из системных вызовов, а также платформы, на которых проводились эксперименты используемых моделью и информация об публичных наборах данных;

– Метрики эффективности моделей: точность, доля ложноположительных срабатываний (FPR), мера предсказательной эффективности (F1-score), кривая ошибок (AUC);

– Ограничения: специфика применения моделей и наличие точных метрик в исследовании;

– Выводы статей, об различных подходах улучшения моделей.

Для структурирования информации созданы две сводные таблицы:

1. Сводная таблица разработанных моделей и их характеристик.

2. Сводная таблица характеристик общедоступных наборов данных, использованных в исследованиях.

Критический анализ эффективности методов проводился с учетом специфики области применения:

– Устройств IoT;

– Контейнеризация;

– Мобильные устройства (Android);

– Виртуализированные и облачные среды;

– Традиционные ОС (Windows, Unix).

III. Результаты

Систематический анализ позволил установить ключевые закономерности, определяющие эффективность современных методов обнаружения аномалий в системных вызовах. Исследование охватило широкий спектр подходов, включая статистические модели, алгоритмы машинного и глубокого обучения, а также гибридные архитектуры, применяемые в различных вычислительных средах — от IoT-устройств до облачных платформ.

Для наглядного представления результатов разработаны две сводные таблицы.

Таблица 1 содержит сравнительный анализ методов обнаружения аномалий. В ней представлены характеристики моделей, ключевые метрики эффективности (точность, FPR, F1-score, AUC) и основные ограничения.

Таблица 2 систематизирует характеристики публичных наборов данных, используемых для обучения и оценки моделей. Для каждого набора указаны: количество трассировок (трасс), типы моделируемых атак, год создания, целевая платформа и источники. Примечание: под “количеством трассировок” понимается число файлов, содержащих последовательности системных вызовов. Одна трассировка может включать от десятков тысяч до сотен тысяч отдельных вызовов; в ряде наборов данных также указывается общее количество записей системных вызовов.

Таблица 1 – Сводная таблица моделей

№	Метод/модель	Тип данных (признаки)	Точность	FPR	F1-score	AUC	Ограничения
[4]	Скрытая марковская модель (HMM)	Вероятностные переходы между состояниями, связанными с системными вызовами	0.97	0.00001	–	–	Высокая вычислительная сложность обучения и тестирования, необходимость выбора количества состояний.
[5]	Двухслойная модель (HMM + перечисление)	Последовательности системных вызовов	–	0.0022	–	–	Высокая вычислительная сложность обучения HMM, необходимость ручной настройки параметров, отсутствуют метрики кроме FPR.
[6]	Метод k-средних	Частота системных вызовов	0.993	0.013	–	–	Низкая эффективность на несбалансированных данных, отсутствуют метрики F1-score и AUC.
[7]	Байесовская сеть	Аргументы системных вызовов	0.999	0.0003	–	–	Зависит от качества обучающих данных; требуется ручная настройка CPT, модель обучалась только на простых атаках, отсутствуют метрики F1-score и AUC.
[8]	Кластеризация на основе ресурсов и значений системных вызовов	Системные вызовы (open, read, write, close, RegCreateKey и др.), ресурсы (файлы, пути), значения	–	0.002	–	–	Не учитывает меж ресурсные зависимости, не анализируются последовательности системных вызовов, модель запоминает шаблон действия приложение и начинает детектировать аномальность при любом нарушении шаблона, нет других метрик кроме FPR.
[9]	Graph Random	Графовая структура,	0.	–	0.	–	Требует предварительного построения

	State Embedding (GRSE)	системные вызовы	9419		9207		графа, ограничения на размер графа при увеличении количества системных вызовов, отсутствуют метрики FPR и AUC.
[10]	Application-Level Anomaly Detection (ALAD) на архитектуре WaveNet	Системные вызовы (n-граммы)	0.986	0.048	–	0.993	Высокие вычислительные требования.
[11]	Модифицированная архитектура длинной цепи элементов краткосрочной памяти (LSTM) – LSTM Anomaly Filter (LAF)	Системные вызовы, n-граммы	0.98	0	0.96	0.97	Зависимость от качества обучения, необходимость настройки гиперпараметров, модель обучалась только против DoS и Slow HTTP.
[12]	Нейронная сеть (MLP)	Системные вызовы (униграммы)	–	–	–	0.98	Низкая чувствительность к stealthy-атакам (APT), в качестве метрики модели использовался только AUC.
[13]	Extremely Randomized Trees (ERT)	Word2Vec + GloVe без повторений + количество системных вызовов	0.958	0.023	0.944	0.977	Высокая размерность, предложенные метод не полностью устраняет проблему дублирования образцов, модель может хорошо работать только в рамках одного набора данных, в случае новых наборов качество ухудшается.
[14]	Группировка последовательностей системных вызовов по уровню риска + метод опорных векторов (SVM)	Последовательности системных вызовов, преобразованные в уровни риска (-1, 0, 1)	0.968	0.0968	0.984	–	Упрощённая модель расчёта риска, не подходит для случаев, когда все функции вызываются примерно одинаково часто в добром и злом коде, потеря информации из-за трёхуровневой классификации.
[15]	SVM	Последовательности системных вызовов, представленные в виде деревьев процессов	0.999	0	0.999	–	Ограничен только MacOS, высокая чувствительность к изменению структуры дерева, неэффективен при наличии многослойной обфускации, модель хорошо работает только с изученными паттернами.
[16]	SCOD (SBG Cluster Outlier Detection, где SBG – Syscall Behavior Graphs)	Графы поведения системных вызовов на уровне потоков, учитывающие вершины (системные вызовы) и ребра (переходы между вызовами), взвешенные по частоте.	0.90	0.04	–	–	Необходимость задания порога чувствительности вручную, ограниченное тестирование на реальных атаках, вычислительная сложность построения и анализа графов, отсутствуют метрики F1-score и AUC.
[17]	Integrated System Calls Graph (ISCG)	Граф системных вызовов	0.988	0.09	–	0.99	Эффективен только на простых наборах.
[18]	Модифицированный HMM	Последовательности системных вызовов	0.97	0.005	–	–	Высокий уровень ложных срабатываний на длинных последовательностях, менее точен на коротких аномалиях, отсутствуют метрики F1-score и AUC.
[19]	Анализ последовательностей системных вызовов	Последовательности системных вызовов	0.97	0.03	≈ 0.9834	–	Зависит от качества сборки системных вызовов.
[20]	Мешок системных вызовов (BoSC)	Последовательности системных вызовов, частотный анализ	0.99	0.00058	–	–	Эффективен только в Linux-контейнерах, обучение проводилось в оффлайн-режиме, требуется предварительное знание нормального поведения, не поддерживается реальное время, отсутствуют метрики F1-score и AUC.
[21]	MALINE	Частота системных вызовов, зависимости между системными вызовами	0.96	0.05	–	–	Требуется значительная вычислительная мощность для обработки больших объемов данных, применимость к другим платформам не исследована, отсутствуют метрики F1-score и AUC.
[22]	Логистическая регрессия (LR) + стохастический	Упорядоченные 3-граммы, TF-IDF (от англ. TF — term frequency, IDF — inverse	–	0.00000	–	–	Зависимость от качества сбора системных вызовов, необходимость периодического обновления модели для адаптации к

	градиентный спуск (SGD)	document frequency), с предварительной обработкой пространственных знаков		1			новым типам вредоносного ПО, авторы не указали метрик кроме FPR.
[23]	KubAnomaly	Системные вызовы: file_IO, network_IO, scheduler, memory; доступ к корневым каталогам	0.96	0.013	0.981	0.947	Не классифицирует типы атак, требуется последующий анализ специалистами, возможны ложные срабатывания при сложных нормальных сценариях.
[24]	Базируемый на обработке естественного языка (NLP) BoSC + алгоритм косинусного подобия	Последовательности системных вызовов, представленные как BoSC размерностью 450	0.99	–	–	–	Модель чувствительна к длине окна анализа, эффективность зависит от качества извлечения системных вызовов, не учитывает временные зависимости между вызовами, авторы не указали метрик кроме точности.
[25]	Рекуррентная нейронная сеть (RNN)	Системные вызовы + n-граммы	0.987	–	–	–	Бинарная классификация, малое количество эпох, авторы не указали метрик кроме точности
[26]	Комбинация из SVM и стохастического градиентного спуска (SGD)	Последовательности системных вызовов, преобразованные в числовые векторы с применением RFE (сокращение от англ. Recursive Feature Elimination) и сингулярного разложения	0.9913	0.015	–	0.993	Зависимость от качества выделенных признаков, необходимость переобучения модели при изменении характера вредоносного ПО.
[27]	Свёрточная нейронная сеть (CNN) названная Siamese-CNN	Векторы, преобразованные в изображения	0.93	0.01	0.91	–	Проблемы с классификацией некоторых типов атак, необходимость ручной переклассификации.
[28]	AdaBoost (сокращение от англ. adaptive boosting)	Фильтрованные системные вызовы	0.968	≈ 0.001636	0.935	–	Эффективен только при фильтрации безопасных вызовов, чувствителен к выбросам.
[29]	LightGBM (сокращение от англ. Light Gradient-Boosting Machine)	Последовательности системных вызовов (3-граммы)	0.75	–	0.30	–	Снижение эффективности при высокой нагрузке на систему, недостаточная способность различать классы вредоносного ПО при большом количестве легитимных системных вызовов, отсутствуют метрики FPR и AUC
[30]	Relaxed-SVM	Комбинация фиксированных и переменных подпоследовательностей, взвешенных через TF-IDF	–	0.024	0.93	0.99	Вычислительная сложность построения суффиксных деревьев, необходимость предварительной обработки длинных последовательностей, модель чувствительна к дисбалансу данных.
[31]	Метод случайного леса	Системные вызовы	0.99	0.01	0.99	–	Модель ограничена типами архитектур IoT
[32]	HMM + динамичная сегментация	Системные вызовы	0.999	0.00086	0.999	–	Обучение требует предварительной сборки данных, чувствителен к нестандартным изменениям в порядке вызовов.
[33]	Метод случайного леса	Частота системных вызовов, категории, уровень риска, энтропия, количество уникальных вызовов	0.9334	0.1948	0.9686	0.964	Не определяются косвенные системные вызовы, ошибки на нетипичных поведенческих паттернах.

Таблица 2 – сводная характеристика публичных наборов данных

Название датасета	Количество трасс/ общее количество записей	Типы атак	Год создания	Платформа	Источники
MIT Lincoln Labs 1999 Evaluation Data	412 трасс	35 типов атак	1999	Linux / Unix	[4], [1], [7], [17]
PLAID	37 033 нормальных, 1 145 вредоносных трасс	Redis-атака, PHP-FPM-атака, эскалация привилегий, Bruteforce, социальная инженерия	2013	Linux	[10]
LID-DS 2021	15 242 трасс	Bruteforce, CVE-2012-2122, CVE-2014-0160, CVE-2017-7529, CVE-2018-3760, CVE-2019-	2021	Docker-контейнер	[13]

		5418, EPS-CWE-434, PHP-CWE-434, SQL Injection, Zip Slip			
CIC-IDS2017	4180 вредоносных трасс	Trojan, miners, worms	2017	Облачные технологии (IaaS) / Linux	[26], [29]
AWSCTD	10,276 трасс	DangerousObject, Trojan, Downloader, Adware, WebToolbar	2018	Windows XP SP2	[14]
UNM	Более 10 000 нормальных, более 1 000 вредоносных трасс	Более 35 типов атак	2020	Linux / Unix	[4], [17], [30], [32]
ADFA-LD	5 206 нормальных, 746 вредоносных трасс	60 атак, 6 сложных типов	2013	Linux	[34], [9], [10], [13], [17], [30]
LID-DS 2018	10 194 трасс	Bruteforce, CVE-2012-2122, CVE-2014-0160, CVE-2017-7529, CVE-2018-3760, CVE-2019-5418, EPS-CWE-434, PHP-CWE-434, SQL Injection, Zip Slip	2018	Docker-контейнер	[27]
CERT Insider threat	Более 2.6 миллионов записей системных вызовов	Анонимные аномалии пользователей, включая злонамеренные действия	2019	Общие логи	[23]
HIDS-Docker Dataset	50 трасс	XSS через wp-admin, удалённое выполнение PHP-кода, загрузка вредоносных файлов, выполнение shell-скриптов, несанкционированный доступ	2021	Docker-контейнер + Linux	[28]

Данные Таблицы 1 демонстрируют значительный разброс в полноте отчетности метрик: многие исследования указывают лишь отдельные показатели (чаще всего точность или FPR), что затрудняет прямое сравнение моделей.

Анализ таблицы 1 позволяет выделить три группы методов с характерными особенностями. Глубокие архитектуры, такие как модифицированные LSTM (LAF) и ALAD на базе WaveNet, демонстрируют рекордно низкий уровень ложноположительных срабатываний (FPR ≤ 0.0022) и высокую AUC (до 0.993), однако требуют значительных вычислительных ресурсов, что ограничивает их применение в IoT-устройствах. Гибридные подходы, включая SVM и графовые методы (GRSE, ISCG), обеспечивают баланс между эффективностью и вычислительными затратами, но сталкиваются с проблемами классификации косвенных атак при анализе нетипичных паттернов. Ресурсоэффективные модели, такие как Random Forest и AdaBoost, достигают высокой точности (Точность $\geq$ 0.98), однако чувствительны к изменениям в поведении легитимных процессов. Легковесные методы (K-Means) показывают FPR (0.013) что позволяет их использовать в IoT, но неэффективны на несбалансированных данных.

Для мобильных устройств (Android) ключевым подходом является выявление статистических различий в паттернах системных вызовов между легитимными и вредоносными приложениями. Исследование [35] демонстрирует, что вызовы bind, brk, connect служат индикаторами атак, а метод оценки сходства на основе частотных характеристик позволяет эффективно классифицировать угрозы.

Анализ таблицы 2 подчеркивает важность унификации подходов к извлечению признаков. Использование Word2Vec и GloVe для представления системных вызовов приводит к утечке данных между обучающей и тестовой выборками, что завышает метрики. Комбинирование векторов Word2Vec и GloVe снижает проблему дублирования образцов что было исследовано в [13], но не обеспечивает универсального решения для всех наборов данных. Этот тип утечки данных представляет собой систематическую угрозу валидности сравнительного анализа. Метрики эффективности (точность, F1-score, AUC), полученные в исследованиях, использующих аналогичные методы представления признаков на основе эмбеддингов (например, [12], [24], [25]), потенциально завышены и не отражают реальную способность моделей к обобщению на неизвестных данных. Сравнение таких моделей с подходами, избегающими утечки (например, прямая работа с последовательностями или n-граммами в [11], [18], [32]), становится некорректным. Необходима переоценка результатов этих работ с применением строгих протоколов разделения данных (например, разделение на уровне трасс/приложений, а не отдельных вызовов) и кросс-валидации, исключающей утечку. Без этого сохраняется высокий риск переоценки эффективности моделей, основанных на эмбеддингах. Наборы данных, такие как ADFA-LD и LID-DS2021, отличаются по структуре и масштабу, что требует адаптации моделей под конкретные условия.

IV. Обсуждение

Одними из главных ограничений, которые встают при выявлении аномалий на уровне системных вызовов, это ограничение вычислительных мощностей и FPR.

Первое ограничение возникает из-за того, что в системе может генерироваться десятки тысяч системных вызовов в секунду, и даже мощный процессор может не успевать обрабатывать все данные. Эту проблему можно частично решить за счет фильтрации безопасных вызовов и вызовов, которые не несут полезной информации, что подтверждается в исследовании [35]. Эффективным решением для

оптимизации фильтрации может стать анализ системных вызовов на основе фаз выполнения программы, как предложено в работе De Bruin [36]. Автор демонстрирует, что комбинированный динамико-статический подход (например, с использованием инструментов Strace и Sysfilter) позволяет сократить до 60% неиспользуемых вызовов, а разделение на фазы (инициализация, рабочая фаза) дополнительно снижает их количество (например, до 21% для SSH-клиента). Это уменьшает нагрузку на системы обнаружения аномалий и снижает риск пропуска критичных событий из-за избыточного шума. Методы, основанные на визуализации, описанные в [37], также полезны для предварительного анализа и позволяют ускорить разработку HIDS.

Особое внимание необходимо уделить второй проблеме — показателю доли ложноположительных срабатываний (FPR). Даже значение FPR в 1% является критически высоким в данной предметной области. Это подтверждается расчетами: при среднем потоке в 10 000 системных вызовов в секунду модель с FPR=1% генерирует до 6000 ложных аномальных событий в минуту. Такая нагрузка делает практическое применение моделей в HIDS непродуктивным, так как администраторы склонны игнорировать чрезмерное количество оповещений, что повышает риск пропуска реальных угроз. Следовательно, снижение FPR и разработка механизмов подавления одиночных ложных срабатываний становятся обязательными требованиями к любым практико-ориентированным решениям. Особенно критично это в свете того, что заявленный низкий FPR многих моделей может быть достигнут лишь на узком наборе тестовых атак или сценариев (Таблица 1), и его устойчивость к разнообразным реальным угрозам требует дальнейшей валидации.

Перспективным направлением снижения FPR является переход от реактивного обнаружения к предиктивным моделям. Например, подходы на основе sequence-to-sequence архитектур [34], прогнозирующие будущие системные вызовы, позволяют выявлять аномалии до их полной реализации, снижая нагрузку на системы мониторинга за счет упреждающего блокирования подозрительных последовательностей. Перспективным направлением является также использование не только последовательностей вызовов, но и их аргументов, как предложено в [38]. Комбинирование семантических эмбеддингов для аргументов (например, путей файлов) с кодированием числовых параметров (PID, timestamp) позволяет улучшить точность моделей глубокого обучения (LSTM, Transformer) на 11.3%, что критично для снижения FPR в реальных системах [38]. Как показано в работе Слюсаренко [39], комбинирование сегментации термов, восстановления грамматики протокола и метрики локальных аномалий (LOF) позволяет достичь FPR < 0.01, что критически важно для практического внедрения HIDS.

Таким образом, выбор оптимальной модели требует учета специфики целевой среды. Для ресурсоограниченных IoT критичны легковесные решения, несмотря на потенциально более высокий FPR, требующий дополнительных механизмов фильтрации. В облаках или на хостах с достаточными ресурсами предпочтительны гибридные или глубокие модели, способные обеспечить минимальный FPR.

V. Заключение

Проведенный анализ подтверждает критическую роль методов обнаружения аномалий в системных вызовах для HIDS. Выделены три ключевые группы: глубокие архитектуры, обеспечивающие максимальную точность (низкий FPR, высокая AUC) ценой вычислительных ресурсов; гибридные подходы, демонстрирующие оптимальный баланс точности и эффективности; и ресурсоэффективные модели, применимые в средах с ограничениями, но более чувствительные к изменениям легитимного поведения.

Основными сохраняющимися проблемами остаются высокий уровень FPR и вычислительная сложность обработки потоков данных в реальном времени. Перспективные направления их решения включают комбинирование анализа аргументов вызовов с семантическими эмбеддингами, использование предиктивных моделей (sequence-to-sequence) и оптимизацию данных через фильтрацию или фазовую сегментацию.

Выбор оптимального метода должен определяться целевой средой: для IoT предпочтительны легковесные решения (k-means, RF), в облачных инфраструктурах — гибридные или глубокие архитектуры. Дальнейшие исследования следует сфокусировать на унификации метрик оценки, разработке адаптивных моделей для динамичных сред (контейнеры, облака) и создании эффективных механизмов подавления ложных срабатываний для повышения практической применимости HIDS.

Библиография

[1] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, “On the detection of anomalous system call arguments,” Computer Security – ESORICS 2003, 2003, pp. 326–343

[2] Соколов А.М. Современные модели обнаружения аномалий в компьютерных системах // Кибернетика и вычислительная техника, С. 1–18, 1999.

[3] Y. Shen, F. Yu, L. Zhang, J. An, and M. Zhu, “An intrusion detection system based on system call,” Proc. The First IEEE and IFIP International Conference in Central Asia on, pp. 1–4, 2005.

[4] C. Warrender, S. Forrest, and B. Pearlmutter, “Detecting intrusions using system calls: alternative data models,” Proceedings of the IEEE Symposium on Security and Privacy, pp. 133–145, 1999.

[5] X. D. Hoang, J. Hu, and P. Bertok, “A multi-layer model for anomaly intrusion detection using program sequences of system calls,” Proc. of the International Conference on Computational Intelligence and Security, pp. 531–536, 2003.

[6] D.-K. Kang, D. Fuller, and V. Honavar, “Learning classifiers for misuse and anomaly detection using a bag of system calls representation,” Proc. Sixth Annual IEEE SMC Information Assurance Workshop, pp. 1–5, 2005.

[7] D. Mutz, F. Valeur, C. Kruegel, and G. Vigna, “Anomalous system call detection,” ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pp. 61–93, 2006.

[8] J. Grandhi, H. Pareek, and P. R. L. Eswari, “Detecting anomalous application behaviors using a system call clustering method over critical resources,” Proc. Communications in Computer and Information Science, pp. 53–64, 2011.

[9] Z. Hu, L. Liu, H. Yu, and X. Yu, "Using graph representation in host-based intrusion detection," Security and Communication Networks, vol. 2021, Article ID 6291276, 2021.

[10] J. H. Ring, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," Digital Threats: Research and Practice, vol. 2, no. 4, Article 26, pp. 1–29, 2021.

[11] B. Yu and J. Kim, "Using a neural network to detect anomalies given an n-gram profile," arXiv preprint arXiv:2104.05571v2, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2104.05571.

[12] J. Carter, S. Mancoridis, M. Nkomo, S. Weber, and K. R. Dandekar, "System call processing using lightweight NLP for IoT behavioral malware detection," in Lecture Notes in Computer Science, vol. 13247. Cham: Springer, pp. 103–115, 2022.

[13] P. K. Mvula, P. Branco, G.-V. Jourdan, and H. L. Viktor, "Evaluating word embedding feature extraction techniques for host-based intrusion detection systems," Discover Data, vol. 1, no. 2, pp. 1–27, 2023.

[14] T. Vyšniunas, D. Čeponis, N. Goranin, and A. Čenys, "Risk-based system-call sequence grouping method for malware intrusion detection," Electronics, vol. 13, no. 1, p. 206, 2024.

[15] V. Van Mieghem, "Detecting malicious behaviour using system calls," M.S. thesis, Delft University of Technology, Delft, The Netherlands, 2016.

[16] M. Pendleton, "System call anomaly detection in multi-threaded programs," Ph.D. dissertation, University of Texas at San Antonio, San Antonio, TX, USA, 2017.

[17] F. J. Mora-Gimeno, H. Mora-Mora, B. Volckaert, and A. Atrey, "Intrusion detection system based on integrated system calls graph and neural networks," IEEE Access, vol. XX, pp. 1–9, 2021.

[18] A. Frossi, F. Maggi, G. L. Rizzo, and S. Zanero, "Selecting and improving system call models for anomaly detection," in Proc. DIMVA 2009, pp. 206–223, 2009.

[19] G. Canfora, F. Mercaldo, E. Medvet, and C. A. Visaggio, "Detecting android malware using sequences of system calls," in Proc. 30th Annu. Comput. Security Appl. Conf., pp. 13–20, 2014.

[20] A. S. Abed, T. C. Clancy, and D. S. Levy, "Applying bag of system calls for anomalous behavior detection of applications in Linux containers," in Proc. IEEE Global Commun. Conf. (GLOBECOM), pp. 1–5, 2015.

[21] Dimjašević, M., Atzeni, S., Ugrina, I., & Rakamarić, Z. "Android malware detection based on system calls UUCS-15-003," School of Computing, University of Utah.

[22] R. Canzanese, S. Mancoridis, M. Kam, "System call-based detection of malicious processes," 2015 IEEE 15th International Conference on Quality Software, pp. 147-156, 2015.

[23] Tien, C.-W., Huang, T.-Y., Tien, C.-W., Huang, T.-C., & Kuo, S.-Y. "KubAnomaly: anomaly detection for the docker orchestration platform with neural network approaches," Engineering Reports, vol. 1, no. 5, p. e12080, 2019.

[24] Peddoju, S. K., H. Upadhyay, J. Soni, N. Prabakar, "Natural language processing based anomalous system call sequences detection with virtual memory introspection," International Journal of Advanced Computer Science and Applications, vol. 11, no. 5, pp. 445–460, 2020.

[25] M. Shobana and S. Poonkuzhali, "A novel approach to detect IoT malware by system calls using deep learning techniques," in Proc. 2020 Int. Conf. Innovative Trends in Information Technology (ICITIIT), pp. 436–441, 2020.

[26] Maheswari, K. U., Shobana, G., Bushra, S. N., & Subramanian, N. "Supervised malware learning in cloud through system calls analysis," in Proc. 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp. 1–8, 2021.

[27] Park, D., Kim, S., Kwon, H., Shin, D., & Shin, D. "Host-based intrusion detection model using Siamese network," IEEE Access, vol. 9, pp. 1–9, 2021.

[28] G. R. Castanhel, T. Heinrich, F. Ceschin and C. Maziero, "Taking a peek: an evaluation of anomaly detection using system calls for containers," 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, pp. 1–6, 2021, doi: 10.1109/ISCC53001.2021.9631251.

[29] P. Brown, A. Brown, M. Gupta, and M. Abdelsalam, "Online malware classification with system-wide system calls in cloud IaaS," in Proc. 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pp. 146–151, 2022.

[30] X. Liao, C. Wang, and W. Chen, "Anomaly detection of system call sequence based on dynamic features and relaxed-SVM," Security and Communication Networks, vol. 1, no. 2, pp. 1–13, 2022.

[31] I. Tahir and S. Qadir, "Machine learning-based detection of IoT malware using system call data," in Proc. 2024 4th International Conference on Digital Futures and Transformative Technologies (ICoDT2), Islamabad, Pakistan, pp. 1-8, 2024.

[32] N. Shamim, M. Asim, T. Baker, and A. I. Awad, "Efficient approach for anomaly detection in IoT using system calls," Sensors, vol. 23, no. 2, Art. no. 652, 2023.

[33] J. Ramamoorthy, K. Gupta, R. C. Kafle, N. K. Shashidhar, and C. Varol, "A novel static analysis approach using system calls for Linux IoT malware detection," Electronics, vol. 13, no. 15, p. 2906, 2024.

[34] S. Lv, J. Wang, Y. Yang and J. Liu, "Intrusion prediction with system-call sequence-to-sequence model," in IEEE Access, vol. 6, pp. 71413-71421, 2018.

[35] Y. J. Ham, D. Moon, H.-W. Lee, J. D. Lim, and J. N. Kim, "Android mobile application system call event pattern analysis for determination of malicious attack," Int. J. Secur. Appl., vol. 8, no. 1, pp. 231–246, 2014.

[36] D. De Bruin, "System call sandboxing: enhancing security through analysis comparing dynamic and static system call analysis for Diff and SSH," Bachelor's thesis, EEMCS Faculty, Delft University of Technology, Delft, Netherlands, 2024. [Available: https://repository.tudelft.nl/file/File_c7212fb0-5e39-4135-8f7b-0555f30baaca?preview=1]

[37] A. Singh, "System call analysis and visualization," M.S. thesis, Dept. Comput. Sci., California State Univ., Sacramento, CA, USA, 2018.

[38] Q. Fournier, D. Aloise, S.V. Azhari, and F. Tetreault, "On improving deep learning trace analysis with system call arguments," in Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR), pp. 1–11, 2021.

[39] Слюсаренко И. М. Методика обнаружения и оценивания аномалий информационных систем на основе анализа системных вызовов: дис. канд. техн. наук. Санкт-Петербург: Санкт-Петербургский государственный политехнический университет, 2005. 178 с.

Review of Anomaly Detection Methods During System Call Auditing in OS

N.E. Stelmach, A.V. Kozachok

***Abstract*— This paper presents a systematic review of state-of-the-art techniques for anomaly detection in system calls—a critical component of Host-based Intrusion Detection Systems (HIDS). The analyzed approaches include: statistical models, machine learning and deep learning methods, hybrid architectures, and sequence analysis techniques. Particular attention is paid to assessing their applicability in contemporary computing environments, such as Internet of Things (IoT) devices, containers, cloud platforms, and mobile operating systems.**

The study results show that state-of-the-art techniques can be categorized based on key characteristics. Deep learning architectures achieve the lowest false positive rate (FPR), but this comes at the cost of significant computational resources. Hybrid approaches demonstrate a more balanced combination of accuracy and efficiency, yet they may remain vulnerable to sophisticated, adaptive attacks. Resource-efficient solutions find application in constrained IoT environments; however, their effectiveness is often diminished by sensitivity to data imbalance.

The research confirms that the choice of the optimal method is determined by the target platform. For resource-constrained IoT devices, lightweight models are preferred, while in cloud environments, comprehensive solutions capable of delivering low FPR demonstrate maximum efficiency. Key unresolved challenges remain the high false positive rate and the computational complexity of processing real-time data streams.

***Keywords*— System calls, Anomaly detection, FPR, HIDS, Machine learning, IoT security, Computing environments, Deep learning.**

REFERENCES

[1] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, "On the detection of anomalous system call arguments," // Computer Security – ESORICS 2003, 2003, pp. 326–343

[2] Sokolov, A.M. Modern models of anomaly detection in computer systems // Cybernetics and Computer Science, 1999 // UDC 004.056.53.001.3. - P. 1-18.

[3] Y. Shen, F. Yu, L. Zhang, J. An, and M. Zhu, "An intrusion detection system based on system call," Proc. The First IEEE and IFIP International Conference in Central Asia on, pp. 1–4, 2005.

[4] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," Proceedings of the IEEE Symposium on Security and Privacy, pp. 133–145, 1999.

[5] X. D. Hoang, J. Hu, and P. Bertok, "A multi-layer model for anomaly intrusion detection using program sequences of system calls," Proc. of the International Conference on Computational Intelligence and Security, pp. 531–536, 2003.

[6] D.-K. Kang, D. Fuller, and V. Honavar, "Learning classifiers for misuse and anomaly detection using a bag of system calls representation," Proc. Sixth Annual IEEE SMC Information Assurance Workshop, pp. 1–5, 2005.

[7] D. Mutz, F. Valeur, C. Kruegel, and G. Vigna, "Anomalous system call detection," ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pp. 61–93, 2006.

[8] J. Grandhi, H. Pareek, and P. R. L. Eswari, "Detecting anomalous application behaviors using a system call clustering method over critical resources," Proc. Communications in Computer and Information Science, pp. 53–64, 2011.

[9] Z. Hu, L. Liu, H. Yu, and X. Yu, "Using graph representation in host-based intrusion detection," Security and Communication Networks, vol. 2021, Article ID 6291276, 2021.

[10] J. H. Ring, C. M. Van Oort, S. Durst, V. White, J. P. Near, and C. Skalka, "Methods for host-based intrusion detection with deep learning," Digital Threats: Research and Practice, vol. 2, no. 4, Article 26, pp. 1–29, 2021.

[11] B. Yu and J. Kim, "Using a neural network to detect anomalies given an n-gram profile," arXiv preprint arXiv:2104.05571v2, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2104.05571.

[12] J. Carter, S. Mancoridis, M. Nkomo, S. Weber, and K. R. Dandekar, "System call processing using lightweight NLP for IoT behavioral malware detection," in Lecture Notes in Computer Science, vol. 13247. Cham: Springer, pp. 103–115, 2022.

[13] P. K. Mvula, P. Branco, G.-V. Jourdan, and H. L. Viktor, "Evaluating word embedding feature extraction techniques for host-based intrusion detection systems," Discover Data, vol. 1, no. 2, pp. 1–27, 2023.

[14] T. Vyšniunas, D. Čeponis, N. Goranin, and A. Čenys, "Risk-based system-call sequence grouping method for malware intrusion detection," Electronics, vol. 13, no. 1, p. 206, 2024.

[15] V. Van Mieghem, "Detecting malicious behaviour using system calls," M.S. thesis, Delft University of Technology, Delft, The Netherlands, 2016.

[16] M. Pendleton, "System call anomaly detection in multi-threaded programs," Ph.D. dissertation, University of Texas at San Antonio, San Antonio, TX, USA, 2017.

[17] F. J. Mora-Gimeno, H. Mora-Mora, B. Volckaert, and A. Atrey, "Intrusion detection system based on integrated system calls graph and neural networks," IEEE Access, vol. XX, pp. 1–9, 2021.

[18] A. Frossi, F. Maggi, G. L. Rizzo, and S. Zanero, "Selecting and improving system call models for anomaly detection," in Proc. DIMVA 2009, pp. 206–223, 2009.

[19] G. Canfora, F. Mercaldo, E. Medvet, and C. A. Visaggio, "Detecting android malware using sequences of system calls," in Proc. 30th Annu. Comput. Security Appl. Conf., pp. 13–20, 2014.

[20] A. S. Abed, T. C. Clancy, and D. S. Levy, "Applying bag of system calls for anomalous behavior detection of applications in Linux containers," in Proc. IEEE Global Commun. Conf. (GLOBECOM), pp. 1–5, 2015.

[21] Dimjašević, M., Atzeni, S., Ugrina, I., & Rakamarić, Z. "Android malware detection based on system calls UUCS-15-003," School of Computing, University of Utah.

[22] R. Canzanese, S. Mancoridis, M. Kam, "System call-based detection of malicious processes," 2015 IEEE 15th International Conference on Quality Software, pp. 147-156, 2015.

[23] Tien, C.-W., Huang, T.-Y., Tien, C.-W., Huang, T.-C., & Kuo, S.-Y. "KubAnomaly: anomaly detection for the docker orchestration platform with neural network approaches," Engineering Reports, vol. 1, no. 5, p. e12080, 2019.

[24] Peddoju, S. K., H. Upadhyay, J. Soni, N. Prabakar, "Natural language processing based anomalous system call sequences detection with virtual memory introspection," International Journal of Advanced Computer Science and Applications, vol. 11, no. 5, pp. 445–460, 2020.

[25] M. Shobana and S. Poonkuzhali, "A novel approach to detect IoT malware by system calls using deep learning techniques," in Proc. 2020 Int. Conf. Innovative Trends in Information Technology (ICITIIT), pp. 436–441, 2020.

[26] Maheswari, K. U., Shobana, G., Bushra, S. N., & Subramanian, N. "Supervised malware learning in cloud through system calls analysis," in Proc. 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), pp. 1–8, 2021.

[27] Park, D., Kim, S., Kwon, H., Shin, D., & Shin, D. "Host-based intrusion detection model using Siamese network," IEEE Access, vol. 9, pp. 1–9, 2021.

[28] G. R. Castanhel, T. Heinrich, F. Ceschin and C. Maziero, "Taking a peek: an evaluation of anomaly detection using system calls for containers," 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, pp. 1–6, 2021, doi: 10.1109/ISCC53001.2021.9631251.

[29] P. Brown, A. Brown, M. Gupta, and M. Abdelsalam, "Online malware classification with system-wide system calls in cloud IaaS," in Proc. 2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI), pp. 146–151, 2022.

[30] X. Liao, C. Wang, and W. Chen, "Anomaly detection of system call sequence based on dynamic features and relaxed-SVM," Security and Communication Networks, vol. 1, no. 2, pp. 1–13, 2022.

[31] I. Tahir and S. Qadir, "Machine learning-based detection of IoT malware using system call data," in Proc. 2024 4th International Conference on Digital Futures and Transformative Technologies (ICoDT2), Islamabad, Pakistan, pp. 1-8, 2024.

[32] N. Shamim, M. Asim, T. Baker, and A. I. Awad, "Efficient approach for anomaly detection in IoT using system calls," Sensors, vol. 23, no. 2, Art. no. 652, 2023.

[33] J. Ramamoorthy, K. Gupta, R. C. Kafle, N. K. Shashidhar, and C. Varol, "A novel static analysis approach using system calls for Linux IoT malware detection," Electronics, vol. 13, no. 15, p. 2906, 2024.

[34] S. Lv, J. Wang, Y. Yang and J. Liu, "Intrusion prediction with system-call sequence-to-sequence model," in IEEE Access, vol. 6, pp. 71413-71421, 2018.

[35] Y. J. Ham, D. Moon, H.-W. Lee, J. D. Lim, and J. N. Kim, "Android mobile application system call event pattern analysis for determination of malicious attack," Int. J. Secur. Appl., vol. 8, no. 1, pp. 231–246, 2014.

[36] D. De Bruin, "System call sandboxing: enhancing security through analysis comparing dynamic and static system call analysis for Diff and SSH," Bachelor's thesis, EEMCS Faculty, Delft University of Technology, Delft, Netherlands, 2024. [Available: https://repository.tudelft.nl/file/File_c7212fb0-5e39-4135-8f7b-0555f30baaca?preview=1]

[37] A. Singh, "System call analysis and visualization," M.S. thesis, Dept. Comput. Sci., California State Univ., Sacramento, CA, USA, 2018.

[38] Q. Fournier, D. Aloise, S.V. Azhari, and F. Tetreault, "On improving deep learning trace analysis with system call arguments," in Proc. IEEE/ACM Int. Conf. Mining Software Repositories (MSR), pp. 1–11, 2021.

[39] Slyusarenko I. M. Methods for detection and evaluation of information system anomalies based on system call analysis: PhD dissertation (Candidate of Technical Sciences). Saint Petersburg State Polytechnic University, St. Petersburg, Russia, 2005. 178 p.