СТРУКТУРНЫЙ ПОДХОД К СТАТИЧЕСКОМУ АНАЛИЗУ ФАЙЛОВ ФОРМАТА ELF ДЛЯ ОБНАРУЖЕНИЯ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Матовых С. С.[1]

DOI:10.21681/3034-4050-2026-2-50-57

Ключевые слова: *структурная модель, машинное обучение, классификация вредоносного программного обеспечения, файлы формата ELF, операционная система Linux, статический анализ исполняемых файлов.*

Аннотация

Цель исследования: *разработать и экспериментально проверить интерпретируемую структурную модель статического анализа ELF-файлов для выявления вредоносного программного обеспечения без выполнения кода.*

Метод исследования: *применены методы статического анализа структуры ELF-файлов и формализация признаков в виде бинарного вектора индикаторов. Классификация выполнена методами машинного обучения с перекрёстной проверкой и сравнением нескольких алгоритмов на едином признаковом пространстве.*

Результат исследования: *сформировано пространство из 63 бинарных структурных индикаторов, охватывающих подсистемы управления памятью, процессов, сетевого взаимодействия, файловых операций, привилегий, механизмов противодействия анализу и упаковки. Проведён сравнительный эксперимент на сбалансированной выборке ELF-файлов, включающей легитимные и вредоносные файлы. Показано, что ансамблевые методы обеспечивают наилучший баланс метрик качества, для модели Random Forest получены следующие результаты Accuracy 0,874, F1-мера 0,860, что подтверждает практическую применимость предложенной модели в задачах раннего статического выявления угроз.*

Научная новизна: *предложена интерпретируемая подсистемная организация индикаторов, повышающая объяснимость решения и пригодность модели для мультиархитектурных сценариев анализа ELF-объектов.*

Введение

Обнаружение вредоносного программного обеспечения (ВПО) в исполняемых файлах формата ELF (Executable and Linkable Format) является критически важной задачей для защиты серверов, встраиваемых систем и IoT-устройств на базе Linux. Актуальность усиливается эволюцией угроз в сторону малозаметных и устойчивых к анализу образцов, например, OrBit использует перехват потока выполнения через механизмы разделяемых библиотек и ориентирован на долговременное скрытное присутствие в системе [1].

Классические сигнатурные средства обнаружения в таких условиях сталкиваются с принципиальными ограничениями. Во-первых, развитие полиморфных и метаморфных техник порождает большое число вариантов одного и того же семейства, что снижает точность обнаружения на основе сигнатур и увеличивает длительность обновления баз [2]. Во-вторых, упаковка и обфускация искажают наблюдаемые статические признаки и могут существенно ухудшать качество обнаружения ВПО, что требует учитывать фактор упаковки как отдельный источник ошибок классификации [3].

В этих условиях закономерно смещение фокуса к развитию методов анализа ELF-файлов по извлечению информативных признаков из структуры и метаданных без выполнения кода с последующей классификацией методами машинного обучения. Такой подход позволяет опираться не на один совпадающий признак, а на совокупность устойчивых структурных характеристик и их комбинаций, что делает обнаружение более пригодным для выявления ранее неизвестных вариантов при заданных ограничениях по времени анализа [4].

Подходы на основе машинного обучения уже продемонстрировали высокую эффективность при статическом обнаружении ВПО в файлах формата Portable Executable (PE-файлы). Появление открытого датасета EMBER сделало возможным воспроизводимое сравнение моделей и показало, что даже относительно простые алгоритмы на инженерных признаках могут достигать высоких показателей качества на задачах детектирования [5]. Обобщающие обзоры по обнаружению ВПО также выделяют устойчивую тенденцию к развитию различных методов анализа, опирающийся на признаки структуры файла, способные выявлять скрытые

1 Матовых Сергей Сергеевич, сотрудник Федерального государственного казённого военного образовательного учреждения высшего образования «Академия Федеральной службы охраны Российской Федерации». Орёл, Россия. E-mail: coolt88@gmail.com

закономерности и обеспечивать высокую обобщающую способность по сравнению с чисто сигнатурными механизмами, особенно в условиях роста вариативности образцов [6]. Параллельно развивается линия, в которой модели обучаются непосредственно на сыром байтовом представлении исполняемого файла, что расширяет класс используемых закономерностей и подтверждает потенциал нейросетевых подходов [7].

Однако основная масса работ исторически сосредоточена на формате PE, тогда как ELF-файлы до недавнего времени изучались заметно меньше. В последние годы ситуация меняется и появляются исследования, предлагающие извлекать признаки из ELF-структуры и статически аппроксимировать поведенческие маркеры с последующей классификацией, в том числе применительно к мультиархитектурным выборкам [8–9]. При этом сохраняется методический разрыв, где остаётся открытым вопрос переносимости и масштабируемости тех принципов статического моделирования, которые хорошо зарекомендовали себя на анализ ELF-файлов с учётом отличий в компоновке, секциях, таблицах и механизмах динамической линковки.

В настоящей работе в качестве методической основы такого переноса используется ранее разработанная и экспериментально обоснованная структурная модель PE-файлов, содержащих вредоносный код, в которой объект анализа трактуется как система взаимосвязанных компонент, отображаемая в интерпретируемое пространство признаков [10]. Цель исследования состоит в разработке и экспериментальной проверке структурной модели ELF-файлов для выявления ВПО, где под моделью понимается совокупность статических индикаторов, извлекаемых из внутренней структуры и содержимого ELF-контейнера без выполнения кода, с последующей классификацией методами машинного обучения. Предлагается набор из 63 бинарных индикаторов, охватывающих признаки, связанные с потенциально вредоносными паттернами (операции с памятью и процессами, сеть, файловая система, привилегии, закрепление, противодействие анализу и упаковка). Приоритет интерпретируемости заложен в конструкцию модели и каждый индикатор имеет однозначный семантический смысл, что согласуется с общей линией развития объяснимого машинного обучения [11]. Далее приводится описание набора признаков и алгоритма классификации, результаты экспериментов на наборе ELF-файлов и выводы о переносимости подходов статического анализа.

Модель и методология

Набор индикаторов M01–M63 задаёт бинарный профиль файла $X \in \{0,1\}^{63}$, где каждый признак фиксирует наличие наблюдаемого артефакта в контейнере ELF. Важно подчеркнуть: речь идёт не о «поведении» в динамическом смысле, а о структурно наблюдаемых следах, которые можно извлечь без исполнения кода. Источниками таких следов выступают формальные элементы ELF, такие как Program Header Table (PHT) и права сегментов PT_LOAD, Section Header Table (SHT), динамический раздел PT_DYNAMIC/.dynamic, таблицы .dynsym/.dynstr и релокации PLT/GOT, а также строковые константы и статистические профили секций/сегментов.

Таблица 1.

Распределение индикаторов по подсистемам ELF-контейнера

Подсистема	Индикаторы	Что именно фиксируется в контейнере
MEM	M01–M10	Аномалии прав/разметки исполняемых областей и следы динамической работы с памятью (RWX, mmap/mprotect, dlopen/dlsym и др.)
PROC	M11–M18, M61	Контур исполнения: порождение процессов, демонизация, запуск команд/внешних программ (exec*, fork/clone, system и др.)
NET	M19–M30, M62	Сетевой слой: достаточный набор для C2/обмена данными (socket/connect/send/recv, DNS, event-loop, TLS)
FS	M31–M42, M63	Файловые операции: развёртывание на диск, подмена/зачистка, манипуляции метаданными, низкоуровневые ioctl и др.
PRIV	M43–M46	Манипуляции привилегиями и capabilities (setuid/setgid/capset и аналоги)
PERS	M47–M48	Закрепление и обращения к чувствительным точкам ОС (/etc/passwd, cron и др.)
ANTI	M49–M56	Противодействие анализу: ptrace/TracerPid, /proc/*/status, VM/sandbox-строки, LD_PRELOAD и др.
PACK	M57–M60	Упаковка/обфускация: высокая энтропия PT_LOAD, stripped-профиль, UPX-подобные секции и маркеры

Чтобы признаки не превращались в набор разрозненных флагов, индикаторы M01–M63 заранее организованы в подсистемы. Это принципиально важно для интерпретации решений классификатора и диагностического анализа ошибок. Распределение индикаторов по подсистемам приведено в таблице 1.

Подсистема Mem отражает признаки, связанные с организацией исполняемых областей и управлением памятью. Здесь есть два структурных слоя. Первый сегментный слой по PHT проверяется наличие загружаемых сегментов PT_Load с правами записи и исполнения одновременно. Для прикладного ПО это обычно нетипичная конфигурация контейнера, вместе с тем во вредоносных цепочках подобная разметка встречается существенно чаще, поэтому она включается как риск-маркер. Второй слой механистический по .dynamic, .dynsym/.dynstr и по PLT-релокациям фиксируются обращения к механизмам mmap/mprotect, а также к dlopen/dlsym. Эти маркеры не являются доказательством вредоносности сами по себе, но в совокупности с признаками упаковки и антианализа дают хорошо интерпретируемый сигнал: файл способен динамически формировать исполняемую память или подгружать код во время работы [12].

Proc-индикаторы описывают контур исполнения, такой как порождение процессов и запуск внешних программ/команд. Статически это наблюдается через импортируемые символы и релокации, через .dynsym/.dynstr и Plt/Got. Отдельно полезны составные шаблоны, вроде комбинации fork/vfork/clone с execve или system. Такой паттерн не обнаруживает ВПО, но фиксирует характерную для загрузчиков и дропперов схему отделения процесса и передачи управления полезной нагрузке именно как структурно наблюдаемый факт наличия соответствующих механизмов.

Net-группа формируется по наличию импортов функций сетевого стека (socket, connect, send, recv и др.) и их комбинаций. Составной шаблон удобно трактовать как признак того, что исследуемый файл содержит минимально достаточный набор механизмов для исходящего соединения и обмена данными. В контуре IoT-вредоноса подобная функциональность действительно является типовой частью жизненного цикла (управление, участие в ботнете, доставочные сценарии), что хорошо показано на крупных исследованиях IoT-ботнетов и на работах по кросс-архитектурному threat hunting [13]. При этом в статье корректно оговаривать доменную специфику для легитимных сетевых утилит и сервисов такие импорты ожидаемы, поэтому диагностическая сила Net повышается именно при совместном присутствии Proc/FS/Anti/Pack.

FS-индикаторы фиксируют операции создания/изменения файлов, удаления, переименования и смены прав. Структурно они извлекаются так же, как Proc/Net через динамические символы и релокации. Наиболее информативны здесь не одиночные функции, а цепочки, описывающие типовые сценарии доставки и активации создать/открыть, записать, сделать исполняемым. Именно такой подход к статическим признакам хорошо ложится на практику анализа ВПО формата ELF.

Признаки Priv/Pers (привилегии/закрепление) намеренно сделаны интерпретируемыми по импорту функций изменения Uid/Gid (setuid, setgid, setresuid) фиксируется потенциальная возможность манипулировать привилегиями, а по строковым маркерам обращение к типовым точкам закрепления (например, crontab, /etc/passwd). Такие признаки лучше всего работают как контекстные в отдельности они могут встречаться в системном ПО, но в композиции с сетевыми и файловыми цепочками дают содержательную картину возможного сценария вредоносной активности.

Anti-группа включает как маркеры (например, импорт ptrace), так и строковые признаки окружения (TracerPid, упоминания гипервизоров). Важно, что подобные проверки встречаются не только у ВПО, однако современные исследования показывают, что анти-отладочные признаки и обнаружение исследовательского окружения устойчивый класс приёмов, особенно в Linux, где злоумышленники ориентируются на затруднение статического и инструментального анализа [14]. Поэтому Anti-признаки логично включаются как отдельная подсистема и учитываются совместно с Pack/Mem/Net.

Pack-индикаторы опираются на две вещи, статистику содержимого и правильность компоновки. Энтропия секций/сегментов – классическая эвристика для выявления сжатия и шифрования, но в литературе подчёркивается, что упаковка бывает многослойной и не всегда сводится к высокой энтропии, поэтому правильная постановка использовать энтропию как вероятностный сигнал и усиливать его структурными маркерами (strip-профиль, характерные секции/паттерны) [15]. Для практики важно, что Pack-индикаторы не «решают задачу за модель», а повышают устойчивость детектора к обфускации.

Каждый из описанных индикаторов представляет булев признак (присутствует или отсутствует).

Таким образом, профиль из 63 признаков для каждого файла фиксирует набор обнаруженных подозрительных индикаторов. Важным преимуществом такого подхода является архитектурная независимость, где многие признаки (импорты функций, строки, энтропия и т.д.) не зависят от конкретной аппаратной архитектуры CPU, что особенно ценно для анализа IoT-файлов, распространяющегося на множестве платформ. В отличие от низкоуровневых признаков (opcode, n-grams и т.д.), структурные индикаторы легко интерпретируются экспертами и непосредственно указывают на известные вредоносные техники. Практическая реализация извлечения признаков базируется на существующих инструментах статического анализа ELF (библиотеки LIEF, утилиты readELF/objdump). Это позволяет автоматизировать получение всех 63 индикаторов из каждого файла перед классификацией. Группировка индикаторов по подсистемам облегчает последующую интерпретацию решения классификатора и согласуется с линией исследований в задачах классификации ВПО формата ELF.

Экспериментальная часть

Для оценки предложенной структурной модели сформирован датасет ELF-файлов двух классов. Ввиду ограниченности публично доступных и воспроизводимо размеченных выборок ВПО файлов формата ELF для экспериментов подготовлен собственный набор объёмом порядка 3000 объектов, где 1500 легитимных ELF-файлов из стандартных дистрибутивов Linux (служебные утилиты и библиотеки) и около 1500 вредоносных ELF-файлов взятых с ресурса vx-underground.org, собранных из исследовательской коллекции Malvare Bazaar декабря 2025 года содержащего различные семейства ВПО. Выборка сбалансирована по классам, что облегчает интерпретацию метрик качества и матрицы ошибок.

Для каждого файла автоматически извлекался полный набор из 63 бинарных индикаторов M01–M63 с использованием разработанного парсера, формируя вектор признаков $X \in \{0,1\}^{63}$. Далее были обучены и сопоставлены несколько алгоритмов классификации на одном и том же признаковом пространстве XGBoost, LightGBM, Random Forest, Gradient Boosting, SVM с RBF-ядром, логистическая регрессия, Decision Tree, Naive Bayes и kNN. Датасет разделялся на обучающую и тестовую части в соотношении 80/20, подбор гиперпараметров выполнялся на обучающей части с применением перекрёстной проверки (k-fold cross-validation). По результатам сравнительного эксперимента таблица № 2 в качестве основной модели для последующего анализа выбран Random Forest – как алгоритм, обеспечивший наилучшие значения F1-меры и Accuracy в рассматриваемом наборе моделей.

Для выбранного классификатора Random Forest дополнительно рассмотрена матрица ошибок на тестовой выборке рисунок 1. Модель правильно классифицировала 523 из 598 файлов Accuracy 0,874. При этом из 301 легитимного файла 9 были ошибочно помечены как вредоносные (FP), тогда как 292 распознаны верно (TN). Одновременно из 297 вредоносных образцов детектировано 231 (TP), а 66 были ошибочно отнесены к классу легитимных (FN).

Полученные результаты подчёркивают практическую направленность детектора Precision 0,963 указывает на низкую долю ложных срабатываний, что критично для эксплуатационного применения, тогда как Recall 0,777 фиксирует, что часть ВПО всё ещё остаётся невыявленной с текущим набором статических индикаторов. Значимая часть пропусков в данном исследовании относится к различным вариантам

Таблица 2.

Сравнение алгоритмов классификации на ELF-признаках

Model	Accuracy	Precision	Recall	F1
XGBoost	0,873	0,963	0,774	0,858
LightGBM	0,872	0,964	0,772	0,857
Random Forest	0,874	0,963	0,777	0,860
Gradient Boosting	0,866	0,955	0,767	0,850
Logistic Regression	0,860	0,941	0,766	0,844
SVM	0,861	0,972	0,758	0,851
Decision Tree	0,861	0,950	0,760	0,844
Naive Bayes	0,787	0,826	0,805	0,798
KNN	0,812	0,957	0,650	0,774

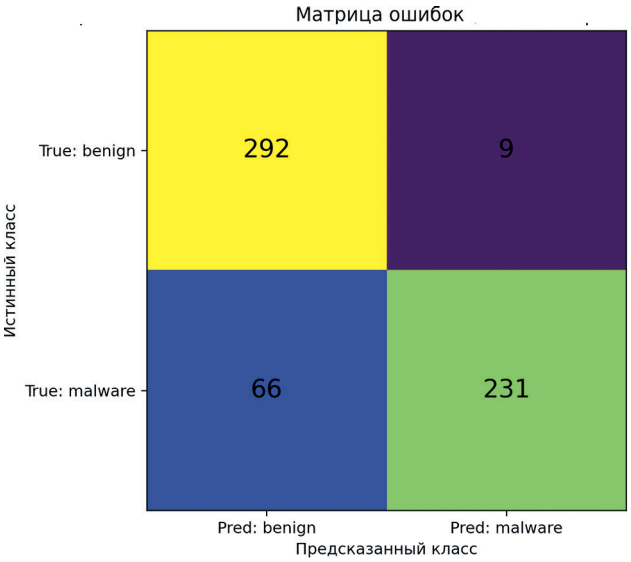

Рис. 1. Матрица ошибок для классификатора Random Forest

семейства Linux.Mirai и его производным, что отражает высокую вариативность данного ВПО. Различия сборок, архитектур и приёмов маскировки приводят к ослаблению или неуверенности в части структурных маркеров, на которых основано признаковое описание. Такая картина согласуется с общими выводами о том, что упаковка и обфускация способны снижать эффективность статического детектирования и требуют либо расширения статического признакового описания, либо введения дополнительных этапов анализа.

Для корректной интерпретации таких результатов классификатора необходимо перейти от сводных метрик к анализу механизма принятия решения какие именно структурные индикаторы оказываются наиболее значимыми

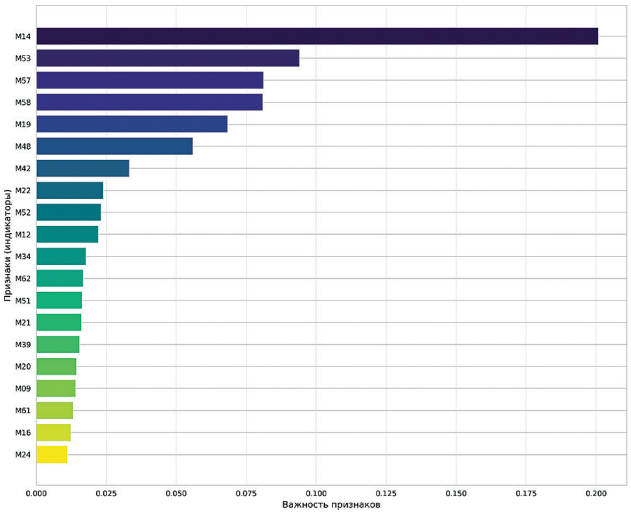

Рис. 2. Важность признаков модели Random Forest

для различения классов. В связи с этим далее приводится распределение важности признаков в модели Random Forest на рисунке 2 показана диаграмма важности, позволяющая визуально оценить вклад.

График важности признаков показывает, что вклад распределён неравномерно небольшая группа индикаторов формирует основную часть дискриминирующего сигнала, тогда как остальные признаки выполняют уточняющую функцию. Для точного представления состава ведущих индикаторов приведём десять признаков с наибольшими значениями важности в таблице 3.

Таблица 3.
Наиболее значимые индикаторы по важности

Индикатор	Подсистема	Важность
M14 (system)	PROC	0,201
M53 (CPUID present)	ANTI	0,094
M57 (High entropy PT_LOAD)	PACK	0,081
M58 (Stripped + entropy)	PACK	0,081
M19 (socket)	NET	0,068
M48 (cron persistence)	PERS	0,056
M42 (ioctl)	FS	0,033
M22 (recv/recvfrom)	NET	0,024
M52 (LD_PRELOAD/ LD_LIBRARY_PATH)	ANTI	0,023
M12 (fork/vfork)	PROC	0,022

Анализ результатов важности признаков позволяет перейти от формального ранжирования к содержательной интерпретации вклада индикаторов.

Наибольший вклад в различение классов формирует индикатор M14 (system) подсистемы PROC. Его значимость объясняется тем, что наличие механизма выполнения команд оболочки характерно для сценариев удалённого управления, запуска вспомогательных компонентов и активации полезной нагрузки. В рассматриваемой выборке данный признак выступает наиболее сильным разделяющим фактором.

Значимый вклад вносят признаки подсистемы PACK – M57 и M58. Они отражают повышенную энтропию загружаемых сегментов и сочетание энтропии с отсутствием таблицы символов. Это указывает на статистико-структурные признаки упаковки или обфускации. Таким образом, модель учитывает не только функциональные возможности файла, но и особенности его компоновки.

Индикаторы подсистемы ANTI (M53 и M52) также входят в число наиболее значимых. Их вклад связан с фиксацией признаков противодействия анализу и проверки окружения выполнения. Наличие подобных маркеров усиливает различение классов, особенно в сочетании с процессными и упаковочными признаками.

Сетевые индикаторы M19 (socket) и M22 (recv/recvfrom) формируют дополнительный вклад, отражая наличие коммуникационного контура. Однако сами по себе сетевые импорты встречаются и в легитимном программном обеспечении, поэтому их диагностическая сила проявляется преимущественно в комбинации с индикаторами PROC, PACK и ANTI.

Признак закрепления M48 (cron persistence) и файловый индикатор M42 (ioctl) вносят меньший, но статистически значимый вклад, уточняя решение классификатора в случаях, когда вредоносный сценарий включает элементы закрепления или низкоуровневые операции.

Таким образом, анализ важности признаков показывает, что основное качество классификации обеспечивается индикаторами подсистем PROC, PACK и ANTI, тогда как признаки подсистем NET, PERS и FS выполняют усиливающую и уточняющую функцию. Следовательно, модель различает классы не по отдельному эвристическому маркеру, а по совокупности структурных признаков, отражающих механизмы выполнения команд, маскировки, противодействия анализу и сетевого взаимодействия. Именно согласованность этих подсистем формирует устойчивый дискриминирующий профиль вредоносных ELF-объектов в рамках рассматриваемого датасета.

Обсуждение и выводы

Показано, что для задачи обнаружения вредоносного программного обеспечения, распространяемого в виде ELF-файлов, может быть построена компактная интерпретируемая модель статического анализа, опирающаяся не на единичный высокоинформативный маркер, а на согласованную совокупность структурных сигналов контейнера ELF и связанных с ним метаданных. Предложенный набор из 63 бинарных индикаторов формирует воспроизводимое признаковое представление ELF-объекта, достаточное для обучения классификаторов без выполнения кода и допускающее содержательное объяснение решения как на уровне отдельных признаков, так и на уровне подсистем признакового пространства.

Экспериментальная проверка на сформированном датасете объёмом порядка 3000 ELF-файлов показала, что на данном признаковом пространстве наилучший баланс метрик обеспечивают ансамблевые методы. Среди протестированных алгоритмов наилучшие значения качества продемонстрировал Random Forest (*Accuracy* = 0,874; *Precision* = 0,963; *Recall* = 0,777; *F1* = 0,860), что соответствует режиму высокой точности положительных решений при наличии пропусков части вредоносных объектов. Прикладная мотивация данного режима состоит в приоритете минимизации ложных срабатываний, поскольку ошибочная блокировка легитимных файлов в типовых сценариях применения является более критичной, чем пропуск части вредоносных объектов.

Анализ ошибок указывает, что основное ограничение подхода связано не с выбором конкретного классификатора, а с неполнотой текущего признакового покрытия для отдельных групп ELF-ВПО. В частности, среди пропущенных образцов присутствуют модификации семейства Linux.Mirai и близкие варианты, для которых вариативность сборок, различия целевых архитектур и приёмы маскировки приводят к ослаблению выраженности используемых статических маркеров. Следовательно, набор из 63 индикаторов достаточен для надёжного выделения значимой доли угроз при минимизации ложноположительных результатов, однако недостаточен для достижения полноты обнаружения, близкой к 1, на широком спектре современных ELF-семейств.

Дальнейшее развитие подхода целесообразно связывать с расширением структурного признакового пространства в тех подсистемах, где проявляются пропуски, а также с расширением и стратификацией датасета по семействам и архитектурам. В прикладном плане предложенная модель уже в текущем виде может рассматриваться как быстрый статический компонент раннего выявления ELF-ВПО с последующим направлением неоднозначных объектов на углублённые процедуры проверки.

Литература

1. Louis D. Advanced analysis of a Linux-dedicated malware (OrBit) [Электронный ресурс]. – URL:www.stormshield.com/news/orbit-analysis-of-a-linux-dedicated-malware. – (дата обращения: 30.01.2026).
2. Sharma A., Sahay S. Evolution and Detection of Polymorphic and Metamorphic Malware: A Survey. – 2014. – https://doi.org/10.5120/15544-4098.

3. Daniel G., et al.: Assessing the Impact of Packing on Machine Learning Based Malware Detection Systems. – 2024. – https://doi.org/10.1016/j.cose.2025.104495.
4. Ramamoorthy J., et al. A Novel Static Analysis Approach Using System Calls for IoT Malware Detection // Electronics. – 2024. – Vol. 13, No. 15. – Article 2906. https://doi.org/10.3390/electronics13152906.
5. Anderson H. S., Roth P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models // arXiv. – 2018. – https://doi.org/10.48550/arXiv.1804.04637.
6. Ucci D., Aniello L., Baldoni R. Survey of Machine Learning Techniques for Malware Analysis // Computers & Security. – 2019. – Vol. 81. – P. 123–147. https://doi.org/10.1016/j.cose.2018.11.001.
7. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.J.M. A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms. // ACM Comput. Surv. – 2024. – Vol. 56, Issue 6. – Article No. 145 P. 1–41. https://doi.org/10.1145/3638240.
8. Tien C.-W., Chen S.-W., Ban T., Kuo S.-Y. Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features // Digital Threats: Research and Practice. – 2020. – Vol. 1, No. 1. – Art. 5. – DOI: 10.1145/3378448.
9. Souza C. H. M. et al. On the Use of Machine Learning for Modern IoT ELF Malware Detection // IEEE.LA-CCI. – 2025 – DOI:10.1109/LA-CCI66231. 2025.11270436.
10. Козачок А. В., Матовых С. С. Структурная модель файлов формата Portable Executable, содержащих вредоносный код // Проблемы информационной безопасности. Компьютерные системы. 2025. № 2. С. 41–59. DOI:10.48612/jisp/pdu2-fvxz-g5d3.
11. Arrieta A. B. et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI // Information Fusion. – 2020. – Vol. 58. – P. 82–115. – DOI: 10.1016/j.inffus.2019.12.012.
12. Ravi A., Chaturvedi V. Static Malware Analysis using ELF features for Linux based IoT devices // Proceedings of the 35th International Conference on VLSI Design & 21st International Conference on Embedded Systems (VLSID). – IEEE, 2022. – P. 114–119. – DOI: 10.1109/VLSID2022.2022.00033.
13. Antonakakis M., April T., Bailey M., Bernhard M., Bursztein E., Cochran J., Durumeric Z., Halderman J. A., Invernizzi L., Kallitsis M., Kumar D., Lever C., Ma Z., Mason J., Menscher D., Seaman C., Sullivan N., Thomas K., Zhou Y. Understanding the Mirai Botnet // Proceedings of the 26th USENIX Security Symposium. – 2017. – P. 1093–1110.
14. Park Y. et al. A practical approach for finding anti-debugging routines in the Arm-Linux using hardware tracing // Scientific Reports. 2024. – Article number: 14728 – DOI:10.1038/s41598-024-65374-w.
15. Lyda R., Hamrock J. Using Entropy Analysis to Find Encrypted and Packed Malware // IEEE Security & Privacy. – 2007. – Vol. 5, No. 2. – P. 40–45. – DOI: 10.1109/MSP.2007.48.

A STRUCTURAL APPROACH TO STATIC ANALYSIS OF ELF FILES FOR MALWARE DETECTION

Matovykh S. S.[2]

Keywords: *structural model, machine learning, malware classification, ELF files, Linux operating system, static executable analysis.*

Abstract

Purpose of work: *the objective of this study is to develop and experimentally validate an interpretable structural model for the static analysis of ELF files aimed at detecting malicious software without code execution.*

Research method: *the proposed approach is based on static analysis of ELF file structures and formalization of features in the form of a binary indicator vector. Classification is performed using machine learning techniques with cross-validation and comparative evaluation of multiple algorithms within a unified feature space.*

Results of the study: *a feature space consisting of 63 binary structural indicators has been constructed. The indicators cover subsystems related to memory management, process control, network interaction, file system operations, privilege manipulation, anti-analysis mechanisms, and packing characteristics. A comparative experiment was conducted on a balanced dataset of ELF files containing both benign and malicious samples. The results demonstrate that ensemble methods provide the best trade-off between performance metrics. For the Random Forest model, the following values were obtained: Accuracy = 0.874 and F1-score = 0.860, confirming the practical applicability of the proposed model for early-stage static threat detection.*

Scientific novelty: *the study introduces an interpretable subsystem-based organization of structural indicators that enhances model explainability and ensures applicability in multi-architecture ELF analysis scenarios.*

References

1. Louis D. Advanced analysis of a Linux-dedicated malware (OrBit) [E`lektronny`j resurs]. – URL:www.stormshield.com/news/orbit-analysis-of-a-linux-dedicated-malware. – (data obrashheniya: 30.01.2026).
2. Sharma A., Sahay S. Evolution and Detection of Polymorphic and Metamorphic Malware: A Survey. – 2014. – https://doi.org/10.5120/15544-4098.

2 Sergey S. Matovykh, employee of the Russian Federation Security Guard Service Federal Academy. Orel, Russia. E-mail: coolt88@gmail.com

3. Daniel G., et al.: Assessing the Impact of Packing on Machine Learning Based Malware Detection Systems .– 2024. – https://doi.org/10.1016/j.cose.2025.104495.
4. Ramamoorthy J., et al. A Novel Static Analysis Approach Using System Calls for IoT Malware Detection // Electronics. – 2024. – Vol. 13, No. 15. – Article 2906. https://doi.org/10.3390/electronics13152906.
5. Anderson H. S., Roth P. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models // arXiv. – 2018. – https://doi.org/10.48550/arXiv.1804.04637.
6. Ucci D., Aniello L., Baldoni R. Survey of Machine Learning Techniques for Malware Analysis // Computers & Security. – 2019. – Vol. 81. – P. 123–147. https://doi.org/10.1016/j.cose.2018.11.001.
7. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.J.M. A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms. // ACM Comput. Surv. – 2024. – Vol. 56, Issue 6. – Article No. 145 P. 1–41. https://doi.org/10.1145/3638240.
8. Tien C.-W., Chen S.-W., Ban T., Kuo S.-Y. Machine Learning Framework to Analyze IoT Malware Using ELF and Opcode Features // Digital Threats: Research and Practice. – 2020. – Vol. 1, No. 1. – Art. 5. – DOI: 10.1145/3378448.
9. Souza C. H. M. et al. On the Use of Machine Learning for Modern IoT ELF Malware Detection // IEEE.LA-CCI. – 2025 – DOI:10.1109/LA-CCI66231. 2025.11270436.
10. Kozachok A. V., Matovy'x S. S. Strukturnaya model` fajlov formata Portable Executable, soderzhashhix vredonosny'j kod // Problemy' informacionnoj bezopasnosti. Komp'yuterny'e sistemy'. 2025. № 2. S. 41–59. DOI:10.48612/jisp/pdu2-fvxz-g5d3.
11. Arrieta A. B. et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI // Information Fusion. – 2020. – Vol. 58. – P. 82–115. – DOI: 10.1016/j.inffus.2019.12.012.
12. Ravi A., Chaturvedi V. Static Malware Analysis using ELF features for Linux based IoT devices // Proceedings of the 35th International Conference on VLSI Design & 21st International Conference on Embedded Systems (VLSID). – IEEE, 2022. – P. 114–119. – DOI: 10.1109/VLSID2022.2022.00033.
13. Antonakakis M., April T., Bailey M., Bernhard M., Bursztein E., Cochran J., Durumeric Z., Halderman J. A., Invernizzi L., Kallitsis M., Kumar D., Lever C., Ma Z., Mason J., Menscher D., Seaman C., Sullivan N., Thomas K., Zhou Y. Understanding the Mirai Botnet // Proceedings of the 26th USENIX Security Symposium. – 2017. – P. 1093–1110.
14. Park Y. et al. A practical approach for finding anti-debugging routines in the Arm-Linux using hardware tracing // Scientific Reports. 2024. – Article number: 14728 – DOI:10.1038/s41598-024-65374-w.
15. Lyda R., Hamrock J. Using Entropy Analysis to Find Encrypted and Packed Malware // IEEE Security & Privacy. – 2007. – Vol. 5, No. 2. – P. 40–45. – DOI: 10.1109/MSP.2007.48.