

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение высшего образования  
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий  
(наименование института полностью)

Кафедра «Прикладная математика и информатика»  
(наименование)

09.04.03 Прикладная информатика  
(код и наименование направления подготовки)

Управление корпоративными информационными процессами  
(направленность (профиль))

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)**

на тему «Анализ алгоритмов и моделирование системы обнаружения вторжений и предотвращения информационных атак»

Обучающийся

А.А. Пиксаева

(Инициалы Фамилия)

(личная подпись)

Научный руководитель

к.п.н., доцент, О.Ю. Копша

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

## Оглавление

Введение.....	3
Глава 1. Анализ алгоритмов систем обнаружения вторжений.....	8
1.1 Роль IDS в технической защите информации.....	8
1.2 Эффективность IDS в противодействии распространённым сетевым атакам.....	14
1.3 Анализ алгоритмов, используемых в IDS .....	18
1.4 Размещение IDS в периметре сети .....	22
1.5 Выбор наиболее оптимального стека алгоритмов для системы обнаружения вторжений .....	27
1.6 Проведение промежуточного вычислительного эксперимента.....	31
1.7 Сравнительный анализ систем обнаружения вторжений для применения выбранных алгоритмов.....	34
Глава 2. Моделирование системы обнаружения вторжений.....	42
2.1 Описание процессов системы обнаружения вторжений .....	42
2.2 Логическое взаимодействие с компонентами системы .....	47
2.3 Физическая модель системы обнаружения вторжений .....	51
Глава 3. Тестирование системы обнаружения вторжений.....	55
3.1 Настройка IDS Suricata для работы со сторонними алгоритмами. ....	55
3.2 Зеркалирование трафика на IDS.....	61
3.3 Динамическая генерация правил алгоритмами и тестирование полученной системы.....	64
Заключение .....	68
Список используемой литературы .....	71
Приложение А Скрипт автоматического запуска Suricata в случае сбоев.....	76

## Введение

В вопросе защиты информации очень большое значение для предотвращения несанкционированного доступа имеют системы обнаружения и предотвращения атак (IDS/IPS-системы или СОВ).

Такие системы в реальном времени отслеживают аномальную активность на основании потоков данных, получаемых из информационных систем, сетевого оборудования, антивирусных приложений, систем предотвращения утечек данных и многих других источников. Системы обнаружения вторжений могут мониторить весь трафик сети, что позволяет им обнаруживать подозрительную активность, даже если она происходит внутри защищенной сети.

СОВ могут быть масштабированы для работы в больших сетях и системах, что делает их более универсальными и гибкими по сравнению с другими методами защиты.

Получив информацию, IDS/IPS-система проводит анализ и выполняет определённые действия по результатам анализа, например, информирует оператора системы, блокирует порты брандмауэра, изолирует участок сети и многое другое. Это позволяет предотвратить возможный ущерб до наступления серьёзных последствий.

Чаще всего анализ информации на предмет аномального поведения представляет собой комбинацию статистических и сигнатурных методик обнаружения. IDS-система сравнивает анализируемые данные с имеющимся набором заранее определённых правил и сигнатур, при необходимости также опираясь на данные того же источника за более длительный период. Такой подход позволяет защищать информационные и сетевые ресурсы от угроз, поведение которых заранее известно или легко прогнозируемо.

Однако, способы и методики сетевых вторжений постоянно изменяются и модернизируются злоумышленниками. Растягивание атаки во времени, либо одновременная работа нескольких злоумышленников затрудняют

обнаружение вторжения. В таких динамичных условиях необходим пересмотр используемых в работе СОВ алгоритмов для надёжной работы системы.

Новые алгоритмы работы должны опираться не только на сигнатуры известных инструментов и методов, но и адаптироваться к новым угрозам.

Аномальное поведение в контексте компьютерных атак – это явления, несвойственные определённому сетевому узлу или информационной системе. К таким явлениям можно отнести: резко возросший объём исходящего или входящего трафика узла, злоупотребление узлом отправки широковещательных пакетов, многократные неудачные попытки авторизации учётной записи, использование привилегированных системных учётных записей для создания новых учётных записей и многое другое.

В решении задачи по совершенствованию систем обнаружения и предотвращения атак исследователи выделяют несколько основных направлений:

- совершенствование сигнатурного и статистического анализа данных;
- обработка нечетких онтологий на основании предварительно утвержденной политики безопасности;
- использование нейросетей для постоянного обучения IDS-системы и противодействия сложнопрогнозируемым атакам.

Вариант совершенствования уже имеющихся сигнатур представляет собой дополнение существующих правил и алгоритмов поведения СОВ. По аналогичной схеме происходит обновление баз антивирусных средств и многих других систем информационной безопасности.

Минусы такого метода заключаются в том, что IDS-система не может эффективно противодействовать т.н. «уязвимостям нулевого дня» – неустраненным уязвимостям аппаратного или программного обеспечения, о которых может быть известно злоумышленникам. В остальном же такой метод совершенствования СОВ считается наиболее традиционным и стабильным.

Использование предварительно утвержденной политики безопасности – метод, который совмещает в себе плюсы использования статистического и сигнатурного анализа, и самостоятельного обучения СОВ.

Несомненно, такая IDS-система также требует присутствия оператора для реагирования на ложные срабатывания и дополнительной оценки рисков, однако в целом использование нечетких онтологий позволяет добиться большего охвата угроз, чем при использовании традиционного сигнатурного анализа.

Наиболее перспективными, но и наиболее сложными в реализации являются СОВ на базе нейросетей. Такие системы сначала накапливают большие объемы данных, соответствующих «нормальной» работе ИС, а затем, основываясь на этих данных, выявляют аномальное поведение.

Несмотря на то, что на этапе запуска и обучения такая система крайне требовательна к действиям оператора, в дальнейшем она способна функционировать практически без вмешательства извне. Однако, такие IDS-системы весьма требовательны к аппаратной части комплекса, в котором они располагаются.

Проблемы и вопросы, связанные с развитием алгоритмов работы СОВ, нашли отражение в научных трудах отечественных и зарубежных ученых и практиков, которые послужили основой при выявлении критериев и разработке концептуальной модели системы обнаружения вторжений и предотвращения атак.

Структурный подход к вопросу технической защиты информации широко освещен в трудах: Герасименко В.А., Малюка А.А., Ганиева А.А., Гатчина Ю.А., Шаньгина В.Ф., Белова Е.Б., Пирогова В.Ю.

Более подробно системы IPS и IDS рассматриваются следующими авторами: Казиенко П., Аксельссон С., Робук К., Лант Т., Пьетро Р.; и нашими соотечественниками: Дородниковым М.А., Яремчуком С., Бурлаковым М.Е. и другими.

Вопросы машинного обучения и построения нейронных сетей освещаются такими авторами, как: Мовахеде М., Эскин Э., Коллиас С.; а также российскими учёными: Емельяновой, Ю.Г., Талалаевым А.А., Тищенко И.П., Фраленко В.П.

Таким образом, актуальность исследования обусловлена в первую очередь тем, что в условиях постоянно меняющегося арсенала инструментов и техник злоумышленников классические алгоритмы работы СОВ, основанные на сигнатурном и поведенческом анализе, не обеспечивают достаточную степень безопасности сети и не могут предотвратить динамическую атаку на системы.

Цель работы заключается в проведении исследования существующих алгоритмов работы IDS-систем, оптимизации их под конкретные информационные системы и сетевую топологию, и разработку модели системы обнаружения вторжений.

Объект исследования – системы обнаружения вторжений (информационная безопасность компьютерных структур).

Предмет исследования – алгоритмы работы систем обнаружения вторжений (СОВ).

Гипотеза исследования заключается в том, что разработка нового алгоритма обнаружения вторжений позволит сократить количество ложных срабатываний системы, повысит общую безопасность сетевой структуры и минимизирует ущерб, нанесённый компьютерными атаками.

Задачи исследования:

- определить оценку существующим подходам и решениям на основе анализа литературы, научных и аналитических статей в области обнаружения вторжений и предотвращения компьютерных атак;
- выявить основные принципы работы СОВ, современные концепции и подходы к анализу потоков данных;
- выбрать алгоритмы, применимые в каждой конкретной ситуации, используемые СОВ при обработке данных;

- разработать модель СОВ, использующую оптимизированные алгоритмы для решения задач по предотвращению атак.

Теоретическая основа исследования включает в себя научные труды и материалы российских и зарубежных ученых, осуществляющих работу по исследуемой проблематике.

Методологической основой исследования являются: анализ и синтез, сравнение, наблюдение, моделирование.

В рамках работы решается задача по анализу имеющихся тематических источников и литературы, с целью выявления в предметной области проблем, не изученных достаточно подробно, чтобы на их основе планировать дальнейшее направление работ.

# Глава 1. Анализ алгоритмов систем обнаружения вторжений

## 1.1 Роль IDS в технической защите информации

Под безопасностью информации подразумевается, согласно исследователю Василькову А.В., способность системы ее обработки обеспечить в заданный промежуток времени возможность выполнения заданных требований по величине вероятности наступления событий, выражающихся в утечке, модификации или утрате информации, представляющих ту или иную ценность для их владельца [7]. Кроме того, защита информации в Российской Федерации регулируется национальным стандартом ГОСТ Р 50922-2006, который определяет следующие понятия:

- защита информации; ЗИ: Деятельность, направленная на предотвращение утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию [9];

- техническая защита информации; ТЗИ: Защита информации, заключающаяся в обеспечении некриптографическими методами безопасности информации (данных), подлежащей (подлежащих) защите в соответствии с действующим законодательством, с применением технических, программных и программно-технических средств [9].

Критерии оценки защищаемой информации введены в 1997 году Герасименко В.А. и Малюком А.А. Они включают в себя следующие параметры:

**Важность информации.** Представляет собой обобщённый показатель, характеризующий значимость информации относительно ее назначения и условий ее обработки.

**Полнота информации.** Под полнотой подразумевается достаточность информации для решения определённых задач.



Адекватность информации – степень соответствия действительному состоянию тех реалий, которые эта информация отображает.

Релевантность информации. Этот показатель характеризует соответствие информации потребностям решаемой задачи.

Толерантность информации. Толерантность определяет удобство восприятия и использования информации в процессе решения задачи.

Проведя оценку защищаемой информации, необходимо выбрать технические и организационные меры по обеспечению ее безопасности. Для выбора мер используется не только оценка самой информации, но и оценка возможных способов и методик несанкционированного доступа к ней. Как правило, для анализа действий злоумышленника применяются методики государственных органов, регулирующих защиту информации на территории Российской Федерации. Основными органами в вопросах защиты информации являются ФСБ и ФСТЭК. Методические документы ФСТЭК представляют собой конкретные описания возможностей злоумышленника и способы противодействия им. Аналогичным решением является матрица техник и тактик MITRE ATT&CK, разработанная американской некоммерческой организацией Mitre [33]. Для каждой техники, используемой злоумышленником, предлагается использование определённого инструмента защиты информации в зависимости от оценки её критичности. К типовым инструментам системы безопасности организации относятся следующие программные и аппаратные инструменты:

- системы антивирусной защиты – предназначены для определения и нейтрализации вредоносного программного обеспечения («вирусов», «троянов» и т.п.);

- комплексные системы защиты информации от несанкционированного доступа – предполагают совмещение нескольких подсистем в одном программном продукте: систему управления доступом, систему регистрации и учета действий, криптографическую защиту информации, подсистему обеспечения целостности [13];

- системы межсетевого экранирования (firewall, брандмауэр) – обеспечивают контроль и фильтрацию сетевых пакетов, защищают сетевые узлы от несанкционированного доступа;

- системы резервного копирования информации – обеспечивают восстановление информации в случае нарушения её целостности злоумышленником или вследствие технических и программных сбоев;

- системы предотвращения утечек данных (DLP-системы) – комплексные системы, отслеживающие действия конкретных пользователей и формирующие аналитические отчёты по результатам их деятельности, а также предупреждающие о возможных утечках служебной информации;

- системы обнаружения (предотвращения) вторжений (IDS/ IPS) – программные комплексы, которые анализируют информацию из других источников и сигнализируют об аномальной активности, либо выполняют меры по пресечению такой активности [16].

Системы обнаружения вторжений (СОВ) в сфере информационной безопасности представляют собой программные или аппаратные средства, которые позволяют автоматически отслеживать и анализировать активность на сети или компьютере с целью выявления попыток несанкционированного доступа или использования ресурсов [1] [14].

СОВ могут работать как на уровне ОС, так и на уровне приложений, их задачей является своевременное определение и реагирование на подозрительную активность, например, сканирование портов, внедрение вредоносных программ, попытки перехвата трафика и другие угрозы информационной безопасности [12].

СОВ играют важную роль в защите информации и обеспечении безопасности компьютерных систем [7].

Принцип работы систем обнаружения вторжений основан на анализе сетевой или системной активности и поиске отклонений от нормального поведения. Для этого СОВ используют модели поведения, которые могут быть созданы на основе статистических данных о том, как должен проходить обмен

данными внутри системы. Эти модели могут быть предварительно обучены и настроены на основании знаний о типичных паттернах поведения и угрозах информационной безопасности.

СОВ могут работать в режиме реального времени, постоянно просматривая и анализируя данные, или выполняться по расписанию, сканируя систему в определенные моменты времени. В случае обнаружения потенциально опасной активности, такой как сканирование портов, перехват трафика или попытки взлома, СОВ могут сигнализировать об этом администратору системы через различные каналы связи, например, электронную почту или SMS-сообщения.

СОВ могут использовать различные методы для обнаружения угроз, включая сигнатурное (signature-based) обнаружение, которое ищет соответствия между образцом заранее известной угрозы и образцами активности на сети или компьютере, и аномалийное (anomaly-based) обнаружение, которое ищет отклонения от нормального поведения в системе. Также могут применяться гибридные методы, объединяющие различные подходы для наиболее эффективного обнаружения угроз [4].

Таким образом, СОВ решает конкретную задачу информационной безопасности – предотвращение несанкционированного доступа к тем или иным информационным ресурсам и узлам сети [8].

Системы обнаружения вторжений классифицируются по нескольким параметрам: по месту установки, по принципу действия (используемым алгоритмам), по методам получения данных.

По месту установки СОВ бывают:

- сетевыми. Такие СОВ располагаются на стратегически важных участках сети и анализируют входящий/исходящий трафик всех устройств сети, учитывая трафик как канального уровня, так и уровня приложений. Недостаток таких систем в том, что анализ всего сетевого трафика неизбежно приводит к большому потреблению аппаратных ресурсов [5]. А увеличение трафика внутри сети приводит к соизмеримо большему потреблению ресурсов

СОВ. Это может приводить к задержкам в обмене данными и снижению скорости работы всей сети. Сетевые СОВ являются хорошим универсальным решением для небольших компаний и организаций.

- хостовыми. В таком варианте размещения СОВ располагается на отдельном узле (либо группе узлов) внутри сети и защищает именно их. СОВ также анализирует все входящие и исходящие пакеты трафика, но только для указанной группы. Существуют масштабируемые хостовые IDS/ IPS системы, которые работают по принципу «клиент-сервер». На защищаемые узлы ставится клиентская часть программы («агент»), которая передаёт все данные на сервер, а непосредственно анализом и обработкой занимается именно серверная часть программного обеспечения.

По используемым алгоритмам СОВ делятся на следующие виды:

- сигнатурные. В них используется заранее обозначенный набор правил, используемых для выявления действий злоумышленника.

- основанные на аномалиях. Для СОВ такого типа характерен «период обучения», когда система накапливает шаблоны данных о «нормальной» работе сети, а затем непрерывно сравнивает потоки данных с имеющимися шаблонами.

- комбинированные. В СОВ комбинированного типа используются как известные сигнатуры типовых атак, так и принципы самообучения системы.

По методам получения данных:

- пассивные. Получают информацию о работе сети из других источников и анализируют её.

- активные. Системы активно воздействуют на сеть, например, с целью найти знакомые сигнатуры в ответном трафике.

- смешанные. Используют оба вышеуказанных класса.

Системы обнаружения вторжений (СОВ) имеют существенные преимущества перед другими системами информационной безопасности [39].

Во-первых, СОВ способна обнаруживать новые угрозы и атаки, которые до этого не были известны. Это возможно благодаря использованию

механизмов обнаружения аномалий в поведении пользователей или трафика в сети. Таким образом, СОВ позволяет оперативно реагировать на новые виды угроз, что особенно важно в условиях постоянно меняющейся угрозной среды.

Во-вторых, СОВ работает в режиме непрерывного мониторинга трафика и может оперативно реагировать на инциденты в режиме реального времени. Это позволяет своевременно замечать и реагировать на подозрительную активность в сети, что снижает риск утечки конфиденциальной информации и других негативных последствий.

В-третьих, СОВ может автоматически реагировать на угрозы без участия человека, что ускоряет процесс реакции на инциденты и позволяет сократить время простоя системы.

В-четвертых, СОВ может быть масштабирована в зависимости от размера защищаемой сети и объема трафика. Это значит, что она может быть использована как в небольших компаниях, так и в крупных корпорациях и государственных учреждениях.

В-пятых, СОВ способна минимизировать ошибки человеческого фактора, связанные с невнимательностью, неопытностью или другими факторами. Это достигается автоматическим обнаружением и реагированием на угрозы без участия человека.

И, наконец, СОВ может быть расширена для обнаружения новых типов угроз и атак. Это делает ее эффективной в защите сети при появлении новых угроз и помогает предотвращать возможные атаки в будущем.

Рассмотрим подробнее, для каких сетевых атак наиболее эффективны IDS.

## **1.2 Эффективность IDS в противодействии распространённым сетевым атакам**

В целом, системы обнаружения вторжений могут обнаруживать различные типы атак, но лучше всего они справляются со следующими категориями.

Атаки отказа в обслуживании (Denial of Service) и их распределённый вариант (Distributed Denial of Service). Это тип кибератак, которые направлены на выведение из строя системы и сервисов интернет-сайтов. Они осуществляются путем блокировки доступа к ресурсу или приводят к его низкой производительности [32].

Основная цель DDoS-атаки заключается в том, чтобы перегрузить сервер или сеть большим количеством запросов, чтобы не оставалось ресурсов для обработки легитимных запросов [36]. Атакующие используют ботнеты, что представляет собой сеть зараженных компьютеров, которые управляются удаленно. Эти компьютеры могут быть заражены вирусами или троянскими программами, которые позволяют злоумышленнику удаленно контролировать эти компьютеры.

DDoS-атаки могут иметь различные формы и объём. Они могут быть запущены с одного компьютера или состоять из тысяч компьютеров, которые атакуют одновременно [40]. Кроме того, атакующие могут использовать различные методы для маскировки своих действий, например, изменение IP-адреса отправителя или использование так называемых "ботов-зомби" (заражённые узлы ботнета), которые могут быть применены для обхода системы защиты [31].

В результате DDoS-атаки доступ к сайту или приложению может быть заблокирован на несколько минут или даже часов. Если атака продолжается достаточно долго, это может привести к значительным убыткам для компании в виде потери клиентов и доходов из-за недоступности сайта [45].

IDS обнаруживают такие атаки за счёт характерного объёмного потока трафика с одного или нескольких IP или MAC-адресов. Атаки DoS внутри сети типа SYN-flooding, когда множество клиентских устройств пытаются начать с целевым хостом TCP-сессию, IDS выявляет по адресу назначения пакетов. В случаях IPS-систем, реакцией на обнаруженную DoS(DDoS)-атаку может стать как временная блокировка отдельных портов целевого хоста, так и обрыв всех новых сессий до него.

Атаки сканирования (Scanning Attacks) – один из типов сетевых атак, при которых злоумышленник сканирует сеть или систему на предмет уязвимостей. Эта атака может быть первым шагом для более серьёзной атаки на систему [38].

Сканирование происходит путем отправки запросов на различные порты и протоколы, чтобы найти уязвимости в сети. Злоумышленники могут использовать различные инструменты для сканирования, такие как Nmap, Angry IP Scanner и другие [2].

В зависимости от целей атакующего, сканирование может быть направлено на поиск конкретных уязвимостей в определенных устройствах, или же выполняться в общем виде для выявления возможных слабых мест.

Примерами атак сканирования являются:

- ping Sweep – это метод сканирования, при котором злоумышленник отправляет ICMP-запросы на все адреса в подсети, чтобы выявить активные устройства в сети.

- port Scanning – это метод сканирования, при котором злоумышленник отправляет запросы на открытые порты устройств в сети, чтобы определить, какие сервисы запущены и какие уязвимости могут быть использованы.

- vulnerability Scanning – это метод сканирования, при котором злоумышленник отправляет запросы на известные уязвимости в сети или на конкретных устройствах, чтобы выявить возможности для атаки.

Атаки сканирования выявляются системами обнаружения вторжений за счёт определения неудачных попыток подключения к портам целевого хоста.

При достижении определённого количества таких неудачных попыток система IPS может автоматически заблокировать узел, с которого ведётся атака. Кроме того, система может оповещать о незнакомых IP и MAC-адресах в сети, с которых ведётся активная рассылка пакетов.

Атаки на протоколы – это методы злоумышленников, которые используют уязвимости в процессе передачи данных между компьютерами или сетями. Цель таких атак может быть различной: от кражи личных данных до перехвата контроля над системой [46]. Существует множество видов атак на протоколы. Некоторые из них включают в себя:

- атаки на протоколы безопасности – это атаки, при которых злоумышленник пытается использовать уязвимость в защите протокола для получения доступа к системе или к информации, передаваемой между устройствами [41]. Протоколы безопасности широко используются в сетевых технологиях и приложениях, чтобы обеспечить конфиденциальность, целостность и доступность данных. Примерами таких протоколов могут быть SSL/TLS, SSH, S/MIME и другие;

- атаки на протоколы маршрутизации – это атаки, которые направлены на изменение маршрута, по которому передаются данные между устройствами. Чаще всего целью таких атак является создание канала утечки конфиденциальной информации из локальной сети [46]. В число таких атак входит DNS-redirecting, где целью злоумышленника становится DNS-сервер, через который осуществляется подмена DNS-адресов информационных ресурсов на ресурсы злоумышленника. Также популярной атакой на протоколы маршрутизации является Routing Table Poisoning, когда злоумышленник получает доступ к маршрутизатору и меняет в нём таблицы маршрутов на собственные, получая доступ ко всему внешнему трафику;

- атаки на протоколы авторизации – это атаки, при которых злоумышленник пытается получить доступ к системе, используя украденные учетные данные или атаковав процесс авторизации. Такими атаками чаще всего являются SQL или XSS-инъекции, в результате выполнения которых



злоумышленник получает опосредованный несанкционированный доступ к базе данных и возможность обойти механизм авторизации системы [42].

Атаки на протоколы выявляются IDS благодаря сигнатурному анализу пакетов, используемых в протоколах. Уязвимости протоколов чаще всего общеизвестные, поэтому способы использования этих уязвимостей шаблонны и легко определяются при помощи баз сигнатур [35]. IPS могут автоматически добавить хосты, использующие уязвимости протоколов, в список ненадёжных источников, и блокировать им доступ к внутренним ресурсам сети. Более сложные атаки на авторизацию, такие как Kerberoasting, выявляются при помощи поведенческого анализа [43].

Рассмотрим основные алгоритмы, которые используются в IDS для определения атак.

### 1.3 Анализ алгоритмов, используемых в IDS

Принцип работы сигнатурных IDS достаточно простой и во многом идентичен антивирусным системам: в базе системы имеется информация об определённых шаблонах пакетов, характерных для проведения тех или иных атак. Большинство атак не являются уникальными, и их процедуры достаточно хорошо исследованы, чтобы выделить моменты, характеризующие каждую атаку. К тому же базы сигнатур регулярно обновляются и пополняются новыми записями, увеличивая спектр применения IDS.

Такие системы обнаружения вторжений характеризуются высокой скоростью работы за счёт известных алгоритмов атаки [3]. Сигнатурные СОВ проще в первоначальной настройке, поскольку для своей работы используют фиксированный базы сигнатур, которые просто подключаются к системе [37].

Однако, из-за обилия сигнатур такие системы склонны к частым ложным срабатываниям [6]. Система генерирует сигнал подозрительной активности на все проявления подозрительной активности, хотя большая часть трафика может оказаться в итоге нормальным для данной сети, а подозрительная активность обусловлена спецификой работы внутри организации. Например, IDS будет рассматривать в качестве подозрительных действий работу стандартных систем мониторинга сети, поскольку те имеют высокую сетевую активность опроса целевых хостов. В крупной сети система мониторинга способна породить тысячи сигналов тревоги в час, оставаясь легальным регламентным средством слежения за сетевыми узлами [10]. Такую проблему можно избежать, если предварительно прописать в IDS правила, игнорирующие активность IP-адресов систем мониторинга. Другим ярким примером ложного срабатывания тревоги является высокая активность привилегированных пользователей в базах данных.

В теории, в эксплуатируемой базе данных не должно наблюдаться плотной административной активности, но в реальности зачастую доработка

баз данных происходит одновременно с их работой, поэтому система обнаружения вторжений воспринимает такую активность как аномальную.

Поэтому наиболее приоритетным этапом развертывания СОВ является именно конфигурация системы в целях минимизации ложных срабатываний и требуемых ресурсов. Большинство СОВ группируют сигналы тревоги по категориям, и если, например, в сети отсутствуют UNIX-хосты, то можно смело отключить срабатывание всех сигнатур UNIX-платформ [36].

Также сигнатурные СОВ бесполезны при обнаружении новых типов угроз, сигнатуры которых ещё не были занесены в базы. К таким угрозам относятся уязвимости «нулевого дня» в программном обеспечении и операционных системах, бэкдоры и закладки от недобросовестных разработчиков ПО и государственных спецслужб.

Для СОВ, построенных на поведенческом анализе, характерны другие особенности: работе системы обязательно предшествует сбор трафика, характерный для нормальной работы контролируемого участка сети. Затем система непрерывно проверяет трафик, проходящий через нее, с имеющимся «эталоном» и в случае проявления аномальной активности посылает соответствующий сигнал [24].

Как правило, коммерческие IDS совмещают в себе как сигнатурные методы, так и поведенческий анализ, что увеличивает общий уровень безопасности системы, но также и повышает количество ложных срабатываний. Также система может включать в себя модуль принятия решений и модуль реагирования, что обеспечивает возможность реагировать на вторжения и предотвращать атаки. Общая схема архитектуры СОВ указана на рисунке 1.



Рисунок 1 – Архитектура системы обнаружения вторжений

На представленной схеме архитектура системы обнаружения вторжений состоит из трех основных модулей и системы контроля сетевого трафика. Модуль выявления атак предназначен для анализа состояния сети и фиксирования фактов подозрительной активности или компьютерных атак. Модуль принятия решений получает информацию о совершении атаки от модуля выявления атак и на основании различных параметров (критичность инцидента, масштаб атаки в рамках сети, вероятность эскалации инцидента и т.д.) отправляет на модуль реагирования соответствующие команды [34]. В числе реакций на атаку могут быть: блокировка определённого IP или MAC адреса устройства, введение временных или постоянных правил для межсетевых экранов сетевого оборудования, блокировка или лишение привилегий определённых учётных записей систем (в т.ч. доменных), информирование оператора СОВ или системного администратора и т.д. Атаки сетевого-прикладного и канального уровней проходят фильтрацию базы знаний. Сетевой трафик контролируется подсистемой сенсоров, которые в

реальном времени фильтруют по заранее определённым правилам пакеты, характерные для наиболее распространённых атак (например, DDoS-атаки), копируют остальные пакеты и направляют копии модулю выявления атак, а также в хранилище, при необходимости позволяя проводить ретроспективный анализ трафика в ходе расследования инцидентов ИБ. Сенсоры опираются на сигнатурный метод анализа трафика, получая информацию о шаблонах проведения атак из базы знаний [38]. Также в базе знаний содержатся шаблоны реагирования на инциденты, которые помогают модулю принятия решений (и оператору СОВ, при необходимости) выбрать наиболее подходящий способ реагирования. Модули СОВ контролируются оператором через консоль управления. Всё это позволяет СОВ контролировать и диагностировать состояние сети и информационных систем внутри неё.

Современные исследователи также рассматривают возможность использования технологий нейронных сетей и интеллектуального анализа данных для работы IDS-систем [20], [22]. Научные труды В.И. Васильева и И.В. Шарабырова демонстрируют преимущество некоторых методов перед другими в определённых группах атак. Так, например, метод опорных векторов превосходит нейронные сети в обнаружении атак канального уровня, но уступает в обнаружении атак прикладного уровня методу дерева принятия решений [3], [21].

Согласно вышеупомянутому исследованию, наиболее оптимальным с точки зрения безопасности является комбинация нескольких алгоритмов обнаружения атак при участии программного арбитра, который определяет уровень модели OSI (Open Systems Interconnection) и тип сетевой активности для выбора дальнейшего алгоритма анализа трафика [22], [23].

## 1.4 Размещение IDS в периметре сети

Другим важным критерием эффективности IDS/IPS систем является место их размещения в периметре сети, если в сети также размещён межсетевой экран (фаервол, брандмауэр) [28]. От этого напрямую зависит как нагрузка на систему обнаружения вторжений, так и эффективность фильтрации трафика.

Межсетевой экран работает на уровне сетевого соединения и принимает решение о том, какие пакеты должны быть разрешены для прохода через границу сети. Принцип работы межсетевого экрана заключается в том, что он анализирует входящий и исходящий трафик и принимает решение о том, какие пакеты данных должны быть разрешены, а какие заблокированы [29]. Механизм работы межсетевого экрана основывается на использовании набора правил, которые определяют, каким образом должен быть обработан каждый пакет данных [27].

Правила межсетевого экрана могут быть заданы на основе различных параметров, таких как IP-адрес отправителя или получателя, номер порта, используемый для соединения, тип протокола и другие. Например, межсетевой экран может блокировать все пакеты, поступающие из определенной страны или от конкретного пользователя [30].

IDS же работает на уровне приложения и анализирует содержимое пакетов на предмет угроз [44]. Он может обнаруживать и блокировать пакеты, которые могут быть опасными для сети.

Чтобы обеспечить более эффективную защиту, IDS и фаервол могут быть интегрированы и работать вместе [25]. Например, межсетевой экран может блокировать доступ к определенным портам или протоколам, а IDS может анализировать разрешенный трафик на предмет угроз [26].

Кроме того, IDS может работать в режиме "внутреннего" межсетевого экрана, что позволяет ему выполнять функции, связанные с контролем и блокированием трафика на основе определенных правил. В таком режиме IDS

может быть использован для предотвращения распространения внутренних угроз (например, вредоносных программ) внутри защищенной сети.

Размещение IDS перед фаерволом (рис. 2) свойственно PIDS (protocol-based IDS), которые устанавливаются непосредственно на внешние интерфейсы веб-сервера и анализирует трафик до его непосредственной фильтрации внутри сети.

Такой подход характеризуется высокой нагрузкой на систему обнаружения вторжений, но позволяет фиксировать атаки не только на внутреннюю сеть, но и на внешние ресурсы веб-сервера, препятствуя, например, DoS и DDoS-атакам.

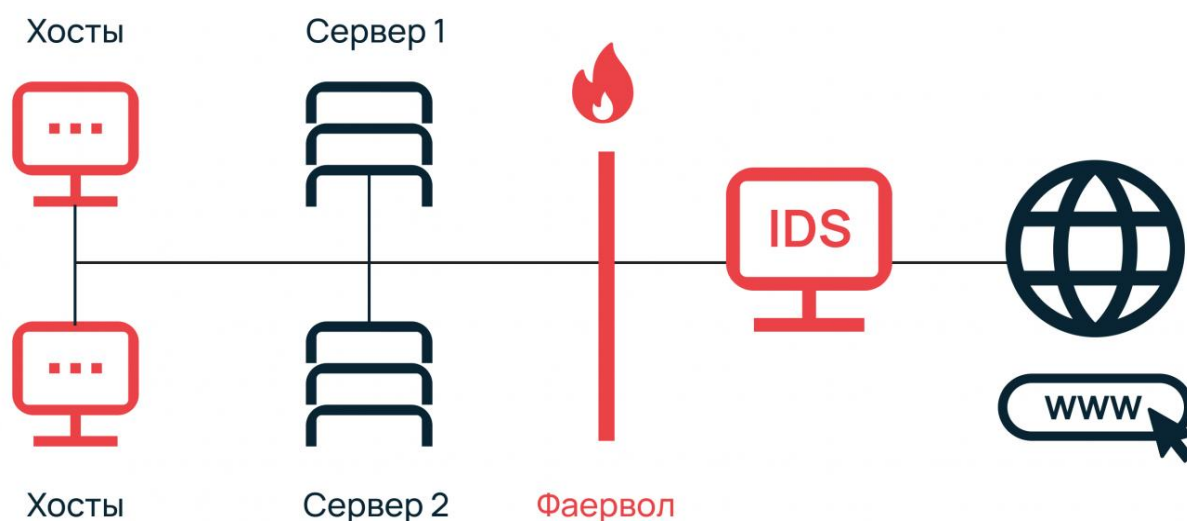


Рисунок 2 – Расположение IDS-системы вне сети

Альтернативный и наиболее частый способ расположения СОВ – непосредственно за фаерволом (рис. 3), таким образом, что весь отфильтрованный трафик проходит через сервер системы обнаружения вторжений, анализируется, при необходимости дополнительно блокируется и далее пакеты направляются непосредственно на целевой хост. Такое расположение значительно снижает нагрузку на сервер IDS, но заставляет использовать другие способы защиты веб-серверов с открытыми внешними

интерфейсами (интернет-сайты, открытые для внешних подключений веб-приложения и т.д.).

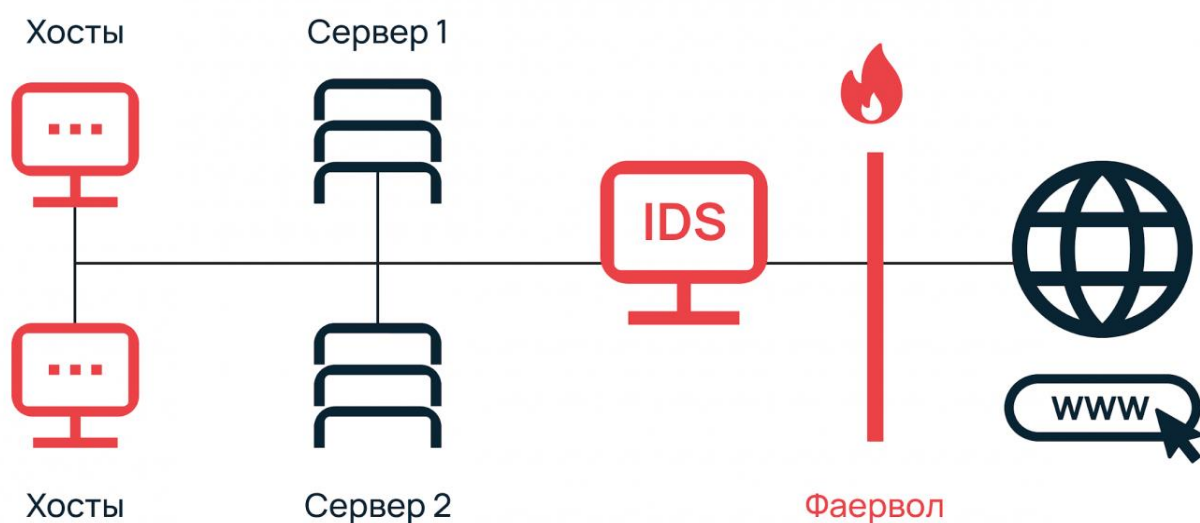


Рисунок 3 – Расположение IDS-системы в периметре внутренней сети

Третий вариант расположения предусматривает размещение IDS системы внутри сети (рис. 4), но параллельно хостам. Такой вариант больше актуален именно для IDS-систем, поскольку он не препятствует прохождению трафика до целевых хостов и не может напрямую блокировать подозрительные пакеты, но в разы снижает нагрузку на модуль анализа трафика и позволяеткратно масштабировать систему для увеличения производительности, добавляя сколько угодно отдельных серверов IDS.



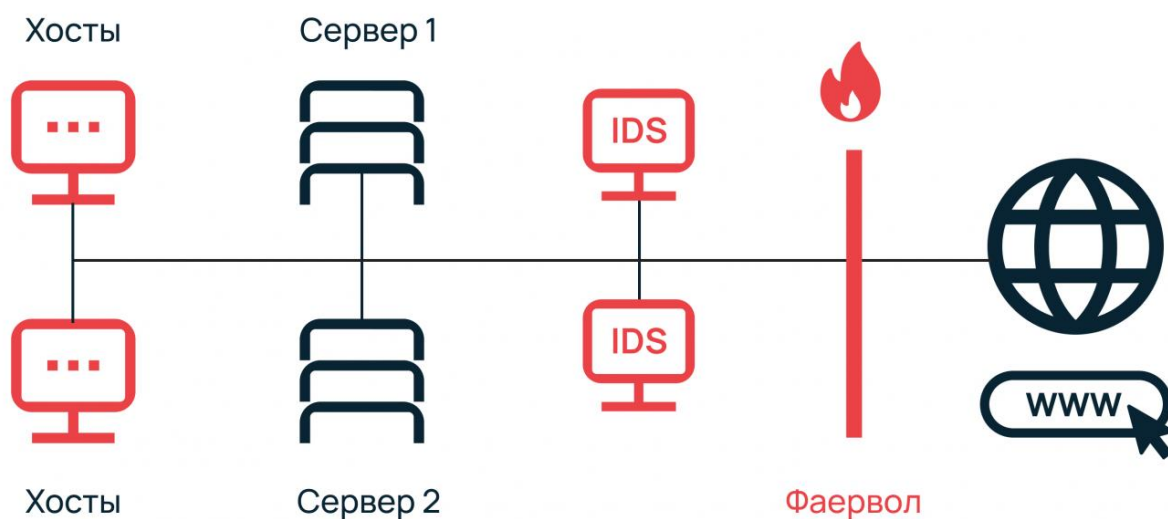


Рисунок 4 – Расположение нескольких параллельных нод IDS-системы внутри сети

Следует отметить, что такое решение возможно не со всеми используемыми в IDS алгоритмами, но отлично вписывается в модель, рассматриваемую в исследовании, когда используется комбинация нескольких алгоритмов анализа и центрального модуля принятия решений, который информирует оператора IDS о подозрительном поведении внутри сети [11].

В дальнейшем в работе будет рассматриваться именно такой способ расположения COB внутри сети, поскольку возможность масштабирования системы и распределения её на нескольких целевых серверах позволит добиться большей безопасности COB, отказоустойчивости, лучшего распределения серверных ресурсов, особенно в случае размещения COB на виртуальных машинах. При этом такой способ позволит внедрить отдельные модули и сенсоры не только на основные сетевые магистрали, но и на каждый крупный узел вплоть до размещения в каждой локальной подсети, а также упростит дальнейшее расширение системы.

На основании анализа литературы, научных и аналитических статей по проблеме совершенствования систем обнаружения вторжений и предупреждения компьютерных атак были выделены критерии оценки

защищаемой информации, определены основные инструменты её защиты, а также сформулирована задача, которую решают СОВ.

В ходе работы была проведена классификация IDS/IPS-систем, выявлены способы их размещения, алгоритмы работы и методы получения данных. Определены сильные и слабые стороны каждого упомянутого класса.

Были проанализированы алгоритмы, используемые в работе СОВ, как с использованием классических сигнатурных алгоритмов, так и более современных методов поведенческого анализа и интеллектуального анализа данных. Также была определена схема архитектуры СОВ.

По результатам анализа был сделан вывод о необходимости применения в СОВ комбинации алгоритмов анализа трафика для разных уровней сетевой модели OSI и типов сетевой активности.

## 1.5 Выбор наиболее оптимального стека алгоритмов для системы обнаружения вторжений

Рассмотрим подробнее существующие алгоритмы, используемые в системах обнаружения вторжений. Стоит отметить, что IDS выявляют атаки путём использования различных методов анализа исходных данных. В данной исследовательской работе затронем следующие методы выявления атак:

- packet header anomaly detection (PHAD) – анализ заголовков пакетов трафика;
- network traffic anomaly detection (NETAD) – анализ содержимого кадров сетевого трафика;
- application layer anomaly detection (ALAD) – анализ работы приложений;
- learning rules for anomaly detection (LERAD) – условные правила выявления аномалий в исходных данных пакетов (например, цепочки handshake, передаваемые в рамках TCP-сессии).

Каждый метод выявления атак показывает наибольшую эффективность против определённых видов атак. Например, анализ заголовков эффективен для выявления атак типа DNS-spoofing или SYN-flood, но не способен выявить внедрение SQL-инъекций, которое, в свою очередь, легко выявляется с помощью анализа работы приложений [12].

Поэтому в большинстве СОВ используются комбинации различных методов выявления атак вкупе с предварительно настроенными правилами и/или технологиями машинного обучения для обеспечения наибольшей безопасности защищаемой системы и минимизации возможного ущерба [17] [19]. Но несмотря на то, что методы выявления аномалий совершенствуются с каждым годом, первым этапом работы любой IDS является сигнатурный анализ. Он позволяет максимально быстро выявить наиболее

распространённые атаки без задействования основных ресурсов системы, что экономит время и вычислительные мощности серверов.

В общем случае работа системы обнаружения вторжений выглядит следующим образом (рис. 5).



Рисунок 5 – Общий алгоритм работы IDS

Основные угрозы отбрасываются на начальном этапе путём проверки сигнатур. Это позволяет минимизировать нагрузку на аппаратные возможности системы, поскольку в таком случае не требуется глубокая инспекция трафика [43]. Если система обнаружения вторжений выявляет угрозу по сигнатурам, дальнейшие проверки трафика нецелесообразны, поскольку пакет отбрасывается в любом случае.

Особое внимание в схеме работы стоит обратить на пункт «Альтернативный анализ пакетов». Именно на этом этапе включаются в работу вышеуказанные методы анализа (PHAD, NETAD, ALAD и LERAD).

Эффективность работы алгоритмов определяется следующим рядом параметров: скорость вычислений (время обработки исходных данных), точность определения атак (достоверность определения факта атаки и её верная классификация), частота ложных срабатываний системы. Для сравнения относительной эффективности алгоритмов мы будем опираться на исследование египетских учёных G.V. Nadiammai и M. Hemalatha от 2014 года [15].

В исследовании работа алгоритмов была основана на наиболее известной open-source IDS Snort. Эта система была создана в 1998 году, но до сих пор стабильно развивается и получает обновления. В 2021 году система получила глобальное обновление, добавившее поддержку многопоточности, разбиения ключевых компонентов на модули, поддержку широкой кроссплатформенности и многое другое.

Snort представляет интерес в первую очередь, потому что благодаря модульности и поддержке сторонних систем анализа представляется возможным сравнение работы каждого отдельного модуля анализа трафика. В коммерческих IDS такие модули являются интегрированными и неотключаемыми.

Для сравнения эффективности методов исследователями был проведён эксперимент по работе Snort с каждым модулем на фиксированных дампах трафика, в которых специалистами выявлено 180 характерных признаков атак

[15]. Результаты эксперимента в области точности определения атак представлены в таблице 1.

Таблица 1 – Эффективность алгоритмов согласно исследованию

Включённые модули	Количество обнаруженных атак (из исходных 180)	Относительная эффективность, %
SNORT	77	42,8
SNORT + PHAD	105	58,3
SNORT + PHAD + ALAD	124	68,9
SNORT + ALAD + LERAD	149	82,8

Исходя из полученных значений, можно сделать вывод, что наибольшую точность определения атак представляет сочетание модулей ALAD + LERAD поверх работы Snort. Однако, стоит иметь в виду, что исследование проводилось на устаревшей версии Snort. Для актуализации данных и определения эффективности работы алгоритмов в скорости вычислений необходимо провести новый эксперимент.

## 1.6 Проведение промежуточного вычислительного эксперимента

В качестве исходных данных мы будем использовать суточный дамп трафика узла с запущенным инструментарием Blue Team Training Toolkit (BT3), который имитирует работу злоумышленника.

Дамп включает в себя несколько итераций сканирования портов, атаки типа Kerberoasting, man-in-the-middle атаки, атаки на уязвимости ОС и многие другие. В общей сложности дамп содержит 1,93 ГБ данных о 274 атаках.

Для тестирования в качестве целевого хоста используется виртуальная машина на базе CentOS 8.2.2004. Характеристики виртуальной машины следующие: процессор AMD Ryzen 5 3350G 4.05 GHz, выделено 2 логических ядра, оперативная память DDR4 2400 MHz, выделено 4 ГБ, объём выделенного места на SSD – 60 ГБ.

На сервере поочерёдно были запущены Snort версии 3.1.18.0 и различные комбинации модулей-надстроек. Затем при помощи Wireshark была запущена имитация обмена трафика с сервером и определены время работы алгоритмов и точность выявления атак. Результаты эксперимента представлены в таблице 2.

Таблица 2 – Эффективность алгоритмов в эксперименте

Включённые модули	Количество обнаруженных атак (из исходных 274)	Время работы алгоритмов, с
SNORT	115	617,6
SNORT + PHAD	149	923,4
SNORT + PHAD + ALAD	213	2137,1
SNORT + ALAD + LERAD	246	2428,3

При подключении модуля подробного анализа пакетов заметно существенное увеличение времени работы IDS, но также значительно повышается точность определения атак.

Для оценки общей эффективности работы алгоритмов возьмём скорость работы Snort без модулей за 100%. Сравнив скорость работы алгоритмов и точность определения атак, сформируем следующий график эффективности (рис. 6).

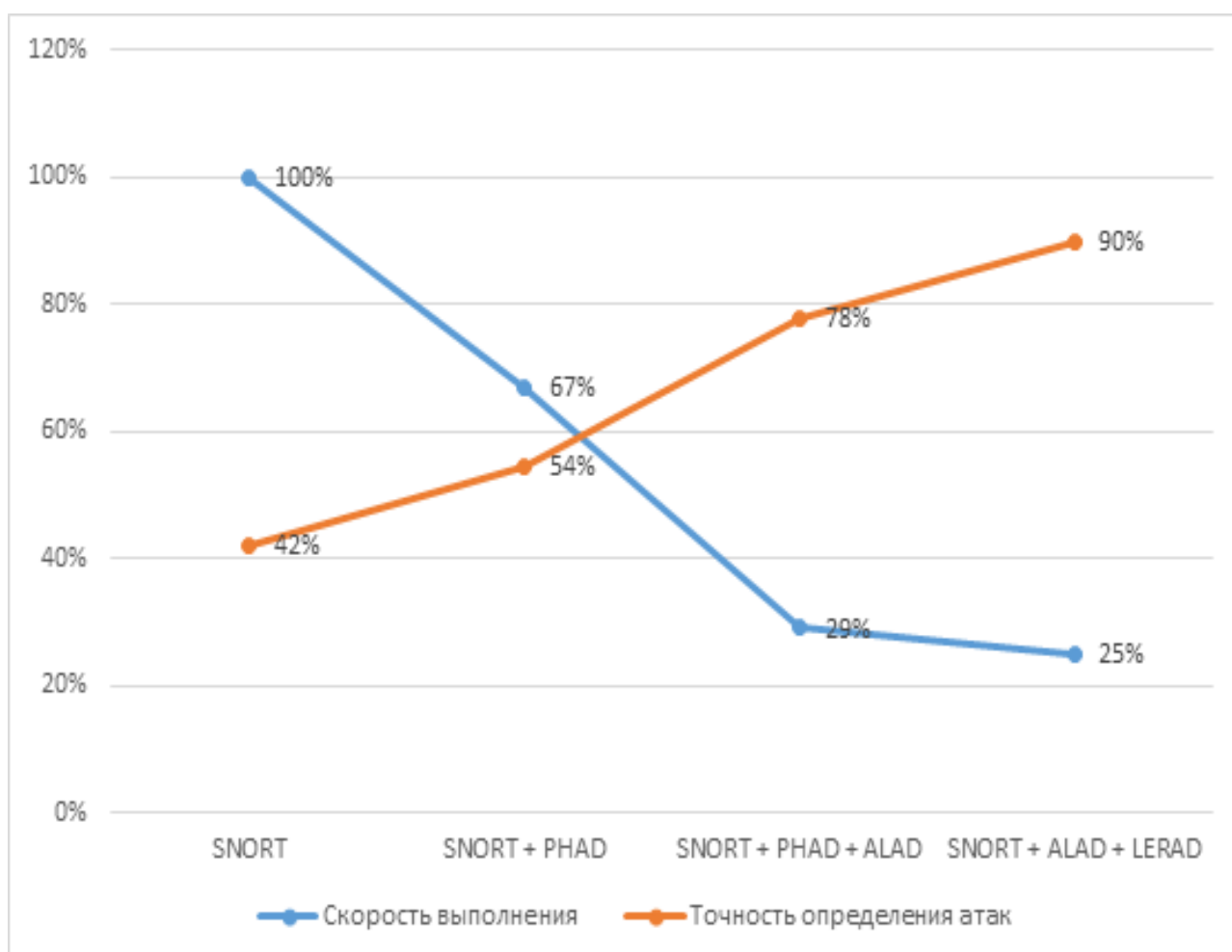


Рисунок 6 – Эффективность работы алгоритмов

Таким образом, по отношению точности определения атак к времени работы алгоритмов наибольшую эффективность представляют сигнатурный анализ Snort вкупе с анализом заголовков PHAD.



Однако, такой подход пропускает почти половину возможных атак и подходит только для первичного этапа обнаружения атак. В вопросах безопасности информационных ресурсов мы можем пренебречь четырёхкратным увеличением времени обработки, если это позволяет достичь повышения точности определения атак более чем в два раза.

Поэтому в дальнейших исследованиях мы также будем опираться на комбинацию ALAD + LERAD, поскольку она даёт наиболее полное определение атак. Однако дальнейшее использование Snort в качестве базовой IDS можно поставить под сомнение, поскольку эксперимент показал высокую нагрузку на систему.

В следующей главе будет проведён сравнительный анализ систем обнаружения вторжений на возможность использования иной IDS в качестве альтернативы Snort для применения выбранных алгоритмов.

## 1.7 Сравнительный анализ систем обнаружения вторжений для применения выбранных алгоритмов

Поскольку в исследовании изначально рассматривалась Snort, стоит изучить её архитектуру подробнее. Структурно Snort состоит из следующих элементов (рис. 7):

- сниффер пакетов – обеспечивает перехват и дублирование пакетов сетевого трафика для дальнейшей передачи в обработку;

- декодер пакетов – определяет заголовки пакетов трафика, анализирует их флаги, отсекает пакеты, не содержащие существенной информации. На этом этапе выявляются аномалии канального и сетевого уровней;

- препроцессоры. На этом этапе происходит детальный анализ каждого из протоколов, в том числе прикладного уровня, таких как HTTP, SMTP, SSH и т.д. Препроцессоры нормализуют и реконструируют потоки для дальнейшего поиска аномалий по сигнатурам или правилам;

- движок обнаружения атак – ядро системы, именно он производит анализ подготовленных данных на соответствие или несоответствие известным сигнатурам и правилам, принимает решения о дальнейшей пересылке пакетов и отображении информации в модуле вывода;

- модуль вывода – способы информирования оператора в случае обнаружения атак. Snort поддерживает все наиболее популярные форматы ведения логов и вывода – файловые отображения, syslog, PCAP, ASCII и т.д. [18].

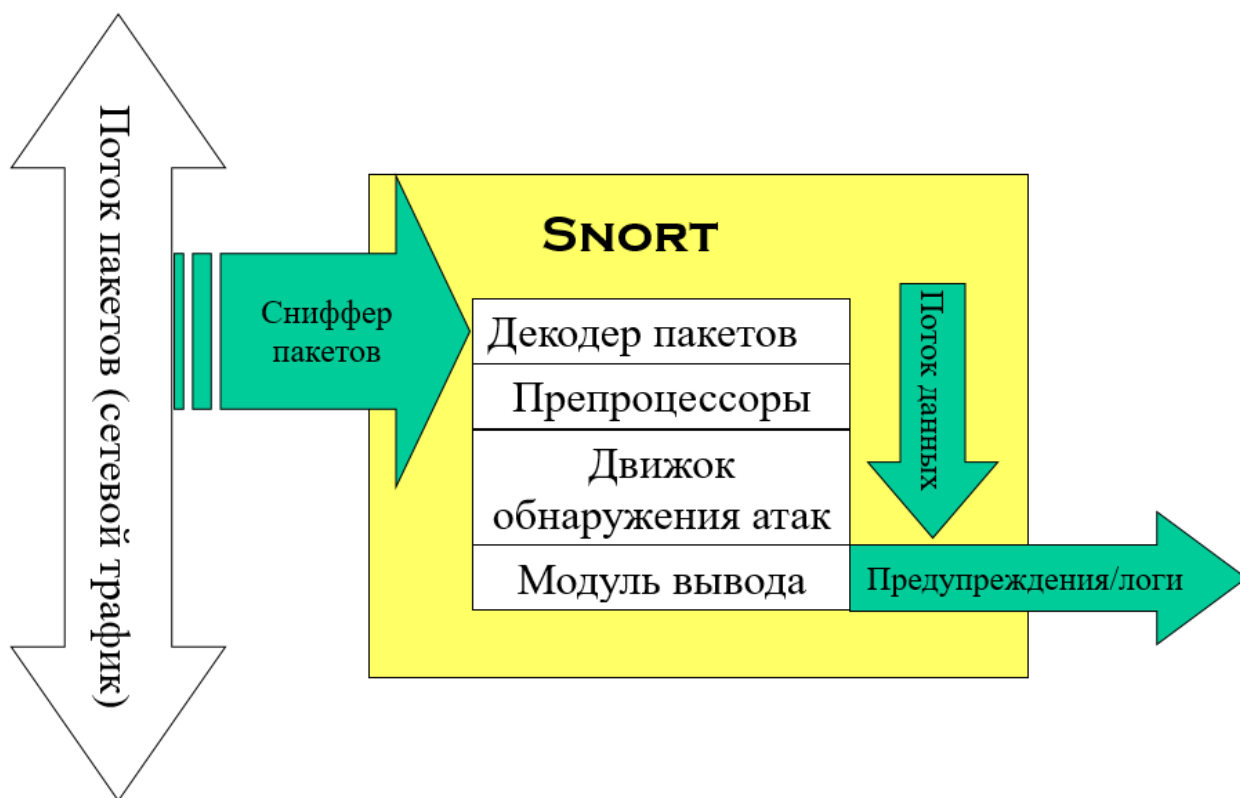


Рисунок 7 – Архитектура Snort

Наиболее близкой к Snort является свободно распространяемая IDS Suricata. Однако, несмотря на аналогичную архитектуру, она имеет несколько ключевых особенностей, среди которых:

- изначальная работа с упором на многопоточность. IDS оптимизирована под многопроцессорные системы, что позволяет добиваться стабильного прироста производительности за счёт увеличения физических ресурсов;

- аппаратное ускорение. Suricata поддерживает вычисления на графическом процессоре (за счёт использования технологий CUDA и OpenCL), что позволяет использовать в качестве эффективного сервера обычный ПК с дискретной видеокартой;

- широкие возможности конфигурации. К примеру, в Snort весь трафик через сниффер до декодера проходит одним потоком. В Suricata можно гибко настроить поведение потоков трафика сразу после их захвата;

- режим IDS штатными средствами. Suricata можно настроить не только на оповещение оператора, но и на выполнение заранее запрограммированных действий в случае обнаружения атак;

- поддержка IPv6. К сожалению, Snort поддерживает только адресацию IPv4, что накладывает ограничения по работе в современных сетях, в которых активно осуществляется переход на новую модель адресации.

- поддержка надстроек Snort. Ключевая особенность Suricata – полная поддержка всех надстроек Snort, начиная от наборов правил и сигнатур, заканчивая логированием ключей и сертификатов SSL-соединений.

Одной из ключевых особенностей Suricata является ее способность работать в режиме реального времени и быстро обрабатывать большие объемы трафика. Благодаря этому она может использоваться в крупных сетях, где необходимо проверять трафик на наличие угроз без задержек и потерь производительности.

В совокупности эти факторы делают из Suricata серьезного конкурента для Snort, а за счёт гибких настроек потоков данных и оптимизации программного обеспечения для многопоточных процессоров и GPU можно обеспечить существенное ускорение работы алгоритмов обработки информации.

Zeek (до 2018 года известна как BroIDS) позиционируется как фреймворк, чем как классическая IDS. Zeek имеет свой собственный скриптовый язык и требует большей предварительной настройки, чем Snort или Suricata, однако обеспечивает сканирование большего объёма данных за счёт гибкого семантического анализа. Zeek имеет модульную архитектуру, которая состоит из следующих уровней:

- модуль захвата пакетов. В Zeek используется libpcap, аналогичная библиотека используется, например, в Wireshark для захвата трафика. Использование libpcap позволяет системе не зависеть от платформы размещения и сетевого уровня;

- ядро событий. Этот модуль формирует из отдельных пакетов цепочки событий для дальнейшего анализа. Цепочки формируются на базе известных способов обмена данными и протоколов. На данном этапе не принимаются никакие решения касательно того, является ли событие подозрительным или нет;

- интерпретатор событий. Этот модуль – главное отличие Zeek от аналогов. Обработчик для каждой цепочки событий, ожидающих реакции, выбирает наиболее подходящий скрипт и ставит события в очередь. Скриптом определяется набор действий для обнаружения подозрительной активности событий и принимается решение о дальнейшей пересылке пакетов.

Архитектура системы построена таким образом, что в случае превышения предполагаемой нагрузки на сервер часть пакетов трафика отбрасывается. Соответственно, при масштабных флуд-атаках, Zeek будет работать менее стабильно, но не будет тормозить сеть, как Snort или Suricata.

Возможностей вывода данных у Zeek меньше, чем у конкурентов – присутствует поддержка файлового вывода и поддержка Elasticsearch.

При всех своих возможностях у Zeek присутствует один серьёзный недостаток – крайне высокий порог вхождения. Чтобы построить с помощью Zeek полноценно функционирующую IDS необходимо досконально разбираться в протоколах низкого уровня, изучить скриптовый язык платформы, разработать регулярные выражения для работы скриптов и использовать кластерное размещение Zeek внутри сети для решения проблемы отброса пакетов.

OSSEC (Open Source Security) - это бесплатная система обнаружения вторжений (IDS/IPS), которая была разработана с учетом потребностей малых и средних предприятий. Она может работать на операционных системах Linux, macOS и Windows.

Основными особенностями OSSEC являются:

- централизованное управление журналами: OSSEC может собирать и хранить журналы событий с нескольких серверов в централизованном репозитории, что обеспечивает удобный доступ к информации о безопасности;

- мониторинг целостности файлов: OSSEC может отслеживать изменения в системных файлах и других критических файлах и оповещать администратора, если они были модифицированы. Это помогает предотвратить атаки на целостность данных;

- поддержка многих платформ: OSSEC поддерживает широкий спектр операционных систем, в том числе Linux, macOS, Windows, Solaris, FreeBSD и другие;

- гибкая настройка: OSSEC имеет гибкую конфигурацию, которая позволяет настраивать систему в соответствии с конкретными потребностями и требованиями безопасности;

- бесплатность и открытый исходный код: OSSEC является бесплатной и имеет открытый исходный код, что позволяет пользователям свободно использовать и изменять систему в соответствии с их потребностями;

- активное сообщество: OSSEC имеет большое сообщество пользователей и разработчиков, которые обеспечивают активную поддержку и развитие проекта.

Однако, указанная платформа не лишена недостатков. Среди основных проблем можно отметить низкую скорость обработки данных из-за постоянного контроля целостности файлов, высокое потребление аппаратных ресурсов и отсутствие графического интерфейса, что мешает интерпретации результатов работы.

OpenWIPS-ng – это бесплатная система обнаружения вторжений (IDS) для беспроводных сетей. Она предназначена для анализа трафика Wi-Fi, обнаружения угроз безопасности и предоставления защиты от них.

Основные особенности OpenWIPS-ng:

- мониторинг трафика Wi-Fi: OpenWIPS-ng может мониторить трафик Wi-Fi и анализировать его, чтобы обнаруживать потенциальные угрозы безопасности;

- гибкая конфигурация: OpenWIPS-ng имеет гибкую конфигурацию, которая позволяет настраивать систему в соответствии с конкретными потребностями и требованиями безопасности;

- поддержка различных режимов работы: OpenWIPS-ng поддерживает несколько режимов работы, включая режим "монитор", "деаутентификация" и "защита";

- поддержка различных интерфейсов: OpenWIPS-ng может работать на различных операционных системах, включая Linux, macOS и Windows, и поддерживает различные интерфейсы, такие как Ethernet, Wi-Fi и другие.

Проблема использования OpenWIPS-ng в её узкой специализации – система подходит для беспроводных сетей, но не имеет поддержки большинства протоколов L2-L3 уровней LAN.

Таким образом, для решения проблемы оптимизации работы алгоритмов наиболее подходящей IDS является Suricata, поскольку она предоставляет больше возможностей для аппаратной оптимизации, чем Snort, но при этом поддерживает алгоритмы глубокого анализа пакетов ALAD + LERAD, разработанные как надстройки для Snort. Тогда целевая система будет подвергать каждый пакет трафика двум видам анализа (рис. 8): первоначальный анализ на соответствие сигнатурам осуществляется за счёт баз сигнатур и правил Suricata, а дальнейший глубокий анализ выполняется модулями ALAD + LERAD.

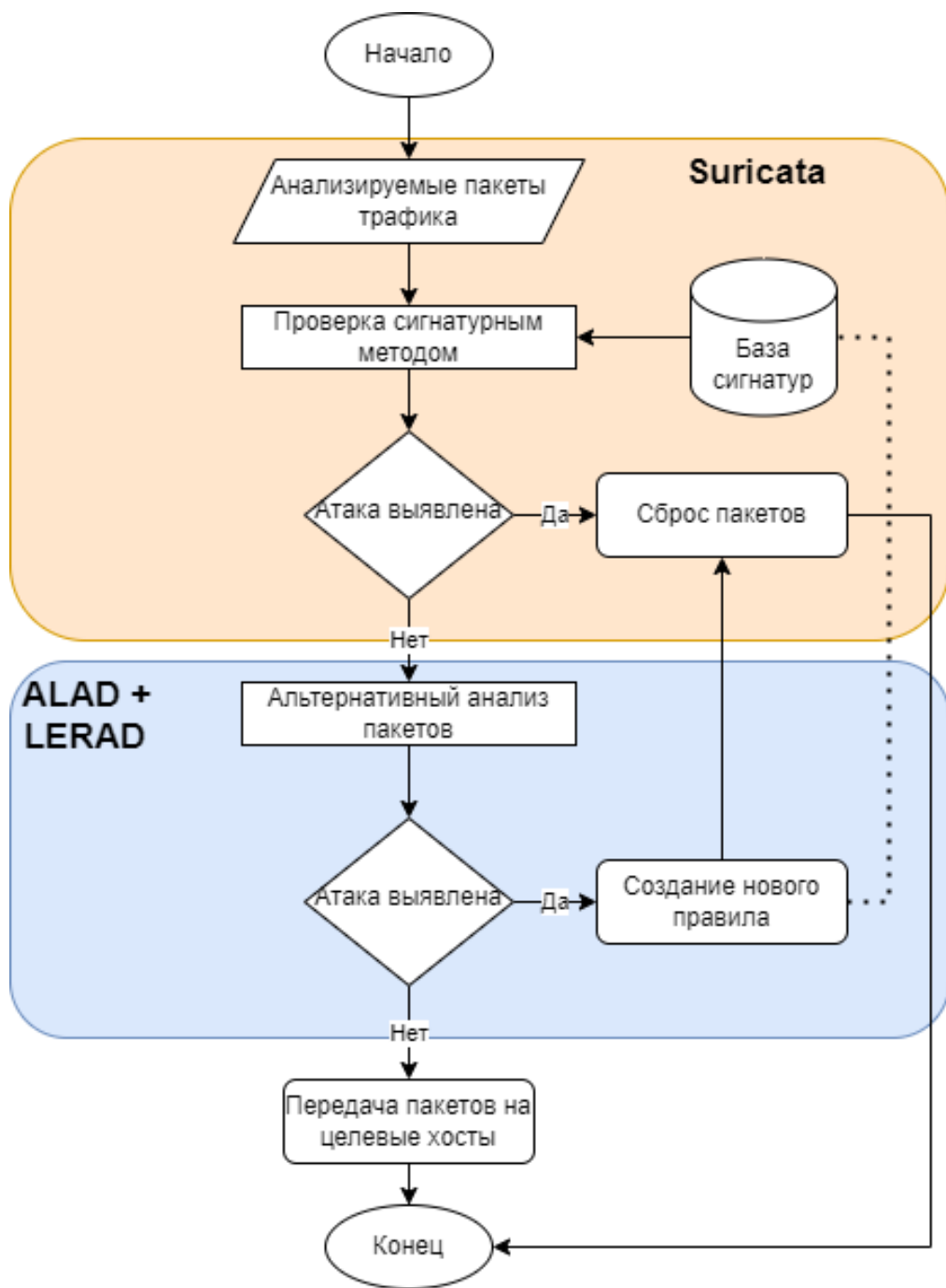


Рисунок 8 – Общая схема работы предлагаемой IDS

### Вывод по Главе 1

В ходе работы были проанализированы методы выявления атак, используемые в СОВ, смоделирован общий алгоритм работы СОВ, путём анализа иностранных источников была выявлена комбинация модулей анализа, предполагающая наибольшую эффективность в обнаружении атак.



Для подтверждения гипотезы был проведён промежуточный эксперимент, показавший, что использование комбинации алгоритмов ALAD + LERAD действительно даёт наибольшую эффективность в выявлении атак, но наиболее быстрым способом выявления массовых атак является сигнатурный анализ, а самым оптимальным по соотношению времени выполнения и эффективности является использование модуля просмотра заголовков пакетов поверх сигнатурного анализа. Для оптимизации работы системы было предложено заменить базовую IDS Snort, на которой проводились вычислительные эксперименты, на более эффективный аналог.

Также был проведён сравнительный анализ аналогов. В качестве альтернатив рассматривались IDS свободного распространения Suricata и Zeek. Наиболее подходящей альтернативой была выбрана Suricata за счёт лучшей аппаратной оптимизации и возможности работы с надстройками Snort, использовавшимися в исследовании ранее. На базе Suricata был предложен алгоритм работы оптимизированной IDS, который включает в себя предварительный сигнатурный анализ и дальнейший глубокий анализ трафика модулями ALAD и LERAD. На базе этих данных необходимо разработать модель новой системы обнаружения и протестировать её в условиях, приближенных к реальным.

## **Глава 2. Моделирование системы обнаружения вторжений**

### **2.1 Описание процессов системы обнаружения вторжений**

Чтобы сформировать логическую модель системы обнаружения вторжений, необходимо описать процесс выявления атак и реагирования на них. Для этого воспользуемся нотацией BPMN, с помощью которой можно описать как организационные, так и технические элементы бизнес-процессов.

BPMN (Business Process Model and Notation) – это стандартная графическая нотация, которая используется для моделирования бизнес-процессов и диаграмм потоков работы. Она была разработана группой экспертов из разных областей бизнеса с целью создания общей и понятной модели процессов для всех участников бизнес-процесса.

Нотация BPMN позволяет описывать бизнес-процессы в виде диаграммы, которая состоит из набора символов и обозначений. BPMN используется для моделирования бизнес-процессов, чтобы помочь компаниям лучше понимать, как работают их процессы, и оптимизировать их выполнение. Это может включать в себя:

- улучшение эффективности процессов: BPMN может помочь идентифицировать узкие места в процессах и определить, где можно сделать улучшения для повышения эффективности;

- снижение затрат: благодаря оптимизации процессов, можно снизить издержки на выполнение задач и улучшить качество работы;

- улучшение взаимодействия: BPMN позволяет различным сотрудникам и департаментам понимать, как работает бизнес-процесс, что способствует лучшему взаимодействию между ними и улучшает коммуникацию;

- создание единой системы для работы с процессами: BPMN создает стандартную нотацию, которая может быть использована всеми участниками процесса, что облегчает понимание и взаимодействие;

- автоматизация процессов: BPMN может служить основой для автоматизации бизнес-процессов, что позволяет сократить время выполнения задач и повысить точность;

- понимание сложных процессов: BPMN позволяет разбивать сложные процессы на более простые, что облегчает понимание их структуры и потоков.

В целом, BPMN является мощным инструментом для моделирования и оптимизации бизнес-процессов, который может помочь компаниям улучшить свою эффективность и конкурентоспособность. Нотация BPMN обеспечивает единую интерпретацию процессов на всех уровнях организации и снижает вероятность непонимания и ошибок в процессах. Она применяется во многих отраслях, включая производство, финансы, здравоохранение и т.д.

Процессы в нотации BPMN разбиваются на более простые компоненты для лучшего понимания и управления. Эти компоненты включают следующие элементы:

- события (Events): это события, которые начинают или заканчивают процесс или его часть. Они могут быть стартовыми (начальными), промежуточными и конечными;

- задачи (Tasks): это элементы, которые представляют выполнение определенной работы. Они могут быть автоматическими или пользовательскими;

- шлюзы (Gateways): это элементы, которые управляют потоком выполнения задач в процессе. Они могут определять, какие задачи будут выполняться, в зависимости от определенных условий;

- артефакты (Artifacts): это дополнительные элементы, которые используются для описания процесса или его частей. Они могут включать в себя комментарии, инструкции или документацию;

- соединительные объекты (Connecting Objects): это стрелки, которые связывают различные элементы процесса и определяют порядок выполнения задач.

Каждый процесс может состоять из нескольких подпроцессов, которые также могут быть разбиты на более простые компоненты. Такой подход к моделированию процессов помогает сделать их более понятными, управляемыми и оптимизированными.

Для понимания логики работы системы обнаружения вторжений, составим упрощённую модель процесса обнаружения вторжений в организации (рис. 9).

Из диаграммы можно понять, что администратор ИБ занимается не только первоначальной настройкой системы обнаружения вторжений, но и участвует в проверке угроз, поступающих по результатам анализа трафика. На основании полученной информации администратор формирует новые правила и политики, которые загружает в систему обнаружения вторжений для автоматизации реагирования на подобные инциденты в дальнейшем. Однако, число событий и атак может исчисляться десятками в минуту, и в таком случае организации придётся увеличивать нагрузку на администратора ИБ вплоть до расширения штата, что влечёт за собой дополнительные расходы. В предлагаемой модели взаимодействия (рис. 10) процесс добавления политик автоматизируется за счёт самообучения системы на базе алгоритмов интеллектуального анализа пакетов.

Таким образом, для однотипных инцидентов с разными источниками опасности (IP-адреса, URL, MAC-адреса устройств и т.д.) система будет автоматически создавать правила на основе уже известных, а при обнаружении нового подозрительного инцидента – так же оповещать администратора ИБ. Такой подход позволит точнее выявлять действительно критичные инциденты и снизить трудовые затраты на поддержание работоспособности системы.

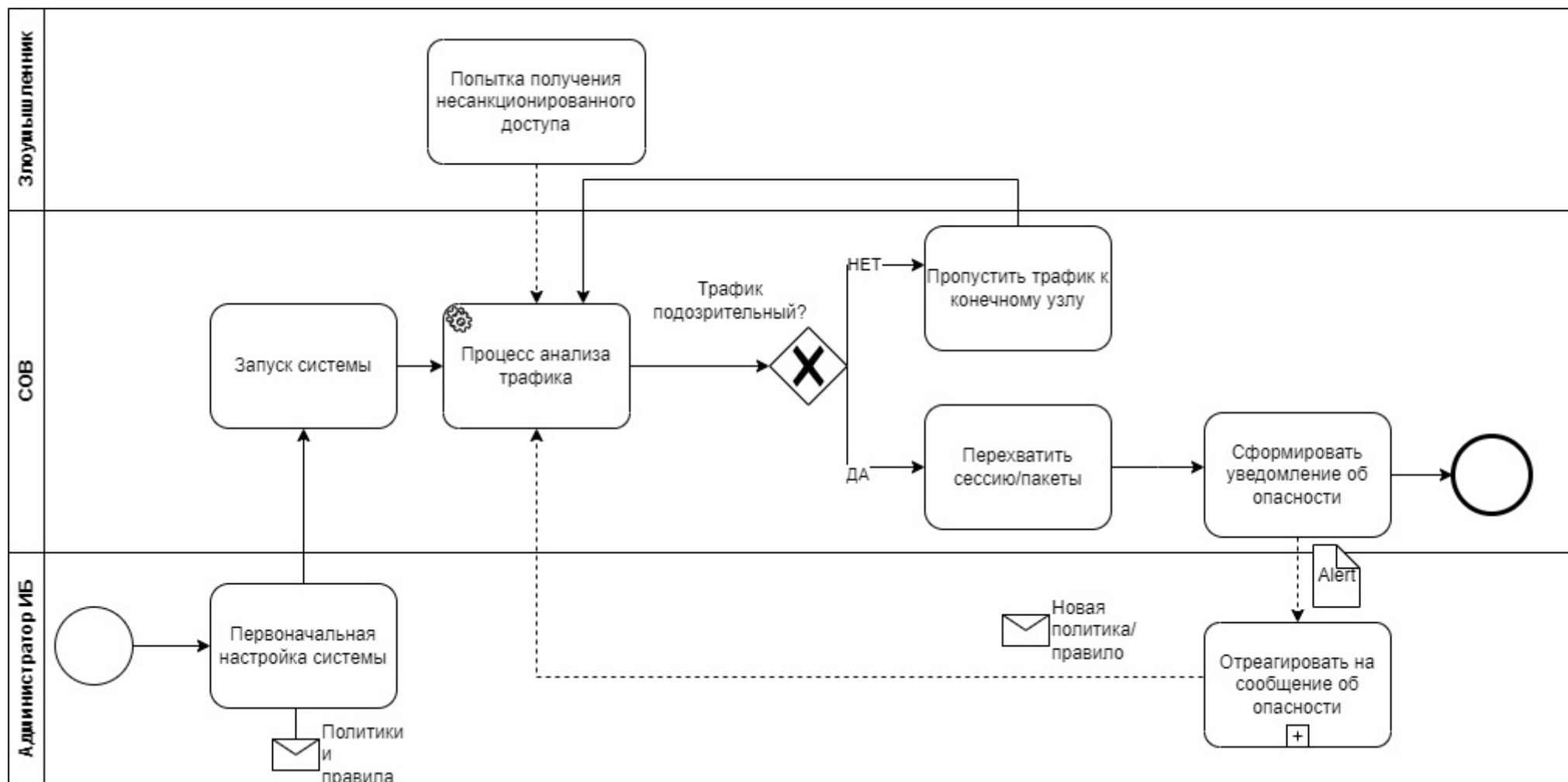


Рисунок 9 – Упрощённая модель процесса работы СОВ в нотации BPMN

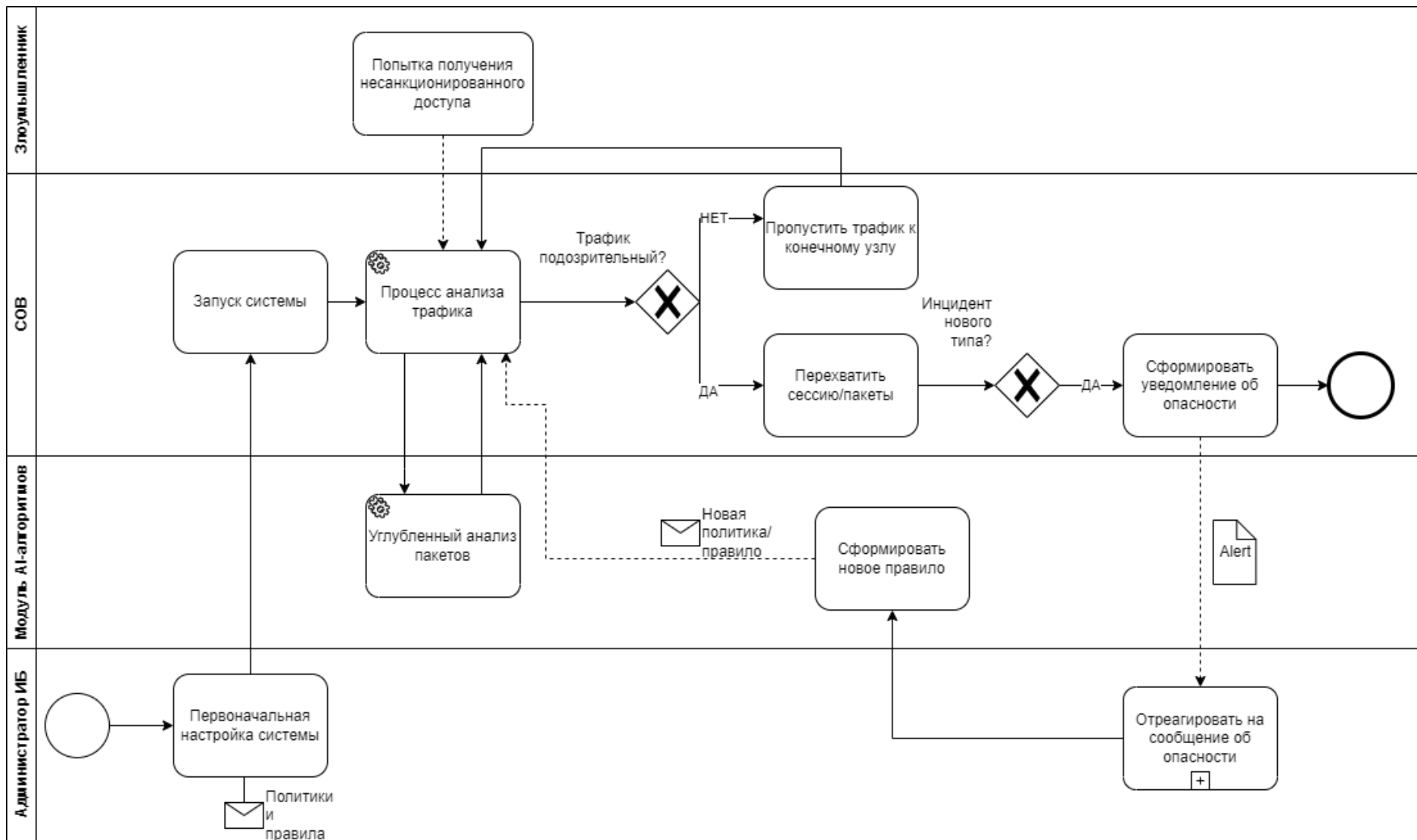


Рисунок 10 – Упрощённая модель работы разрабатываемой СОВ

## 2.2 Логическое взаимодействие с компонентами системы

Чтобы описать, как пользователь системы взаимодействует с её компонентами в части обеспечения работоспособности и настройки, обратимся к нотации UML.

UML (Unified Modeling Language) – это стандартный язык моделирования, используемый для описания проектирования программного обеспечения. UML состоит из набора графических символов и правил, которые могут использоваться для создания различных видов диаграмм, таких как диаграммы классов, диаграммы последовательностей, диаграммы состояний и т.д. UML помогает разработчикам программного обеспечения понимать требования заказчика, определять архитектуру приложения, проектировать его компоненты и проверять его соответствие спецификации. Диаграммы UML используются для визуального представления архитектуры, структуры, поведения и других характеристик программных систем. Они позволяют разработчикам лучше понимать требования к системе и проектировать ее более эффективно.

Например, диаграмма классов UML может использоваться для описания структуры объектно-ориентированной программы, включая классы, свойства и методы. Диаграмма последовательности UML может использоваться для описания взаимодействия между объектами во время выполнения задачи. Диаграмма вариантов использования (Use Case Diagram) в UML используется для описания функциональных требований к системе путем моделирования различных сценариев использования. Она показывает, как актеры (пользователи или другие системы) могут использовать систему для достижения своих целей.

Диаграмма вариантов использования состоит из следующих элементов:

- актеры (англ. «actor») – пользователи и/или другие системы, которые могут использовать систему для выполнения определенных задач;

- варианты использования – конкретные сценарии использования системы, такие как "зарегистрироваться в системе" или "совершить покупку";

- отношения между актерами и вариантами использования – показывают, какие актеры могут выполнять какие варианты использования.

Диаграмма вариантов использования помогает уточнить требования к системе путем определения ее основных функций и того, как эти функции будут использоваться пользователями. Это позволяет разработчикам лучше понимать потребности пользователей и создавать более эффективную и удобную для использования систему.

Построим диаграмму вариантов использования, исходя из наличия трёх актеров: администратора системы, отвечающего за её конфигурацию, оператора системы, который занимается мониторингом трафика для своевременного реагирования на инциденты, а также AI-алгоритмов, отвечающих за самообучение системы (рис. 11).



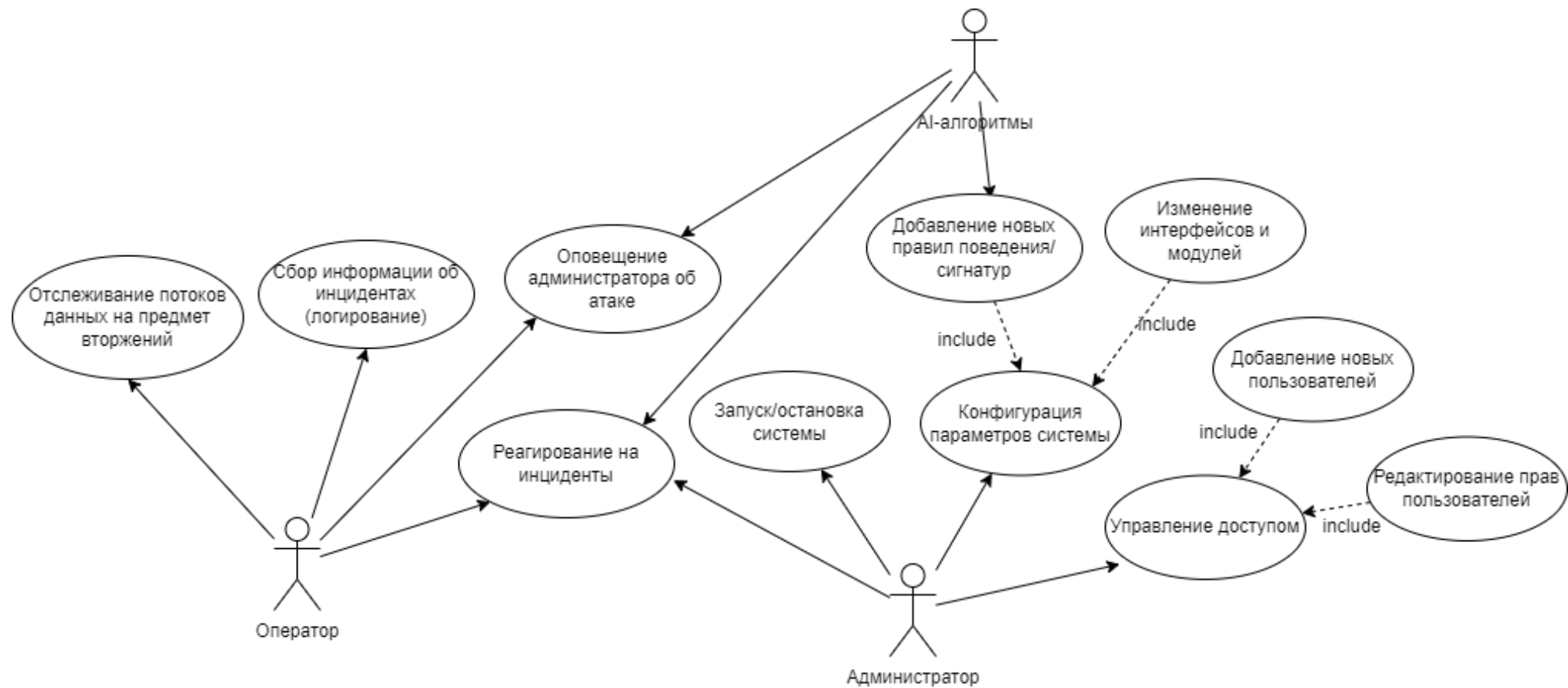


Рисунок 11 – Диаграмма вариантов использования СОВ

Таким образом, в проектируемой системе ограничиваются функции оператора по изменению конфигурации системы обнаружения вторжений, её общей работоспособности и изменению правил поведения. Указанные возможности контролирует администратор СОВ, а оператор занимается непосредственно мониторингом обработанных данных и при необходимости оповещает администратора о проблемах. Алгоритмы интеллектуального анализа облегчают работу как оператора, так и администратора, за счёт автоматизации процессов реагирования на инциденты, оповещения и добавления новых сигнатур.

Логическая модель системы и принцип обработки данных определены, теперь необходимо разработать физическую модель системы и взаимодействия её компонентов.

## 2.3 Физическая модель системы обнаружения вторжений

Физическая модель информационной системы – это описание аппаратного и программного обеспечения, используемого в системе, она помогает разработчикам и администраторам системы понять, какие компоненты необходимы для создания и поддержки системы, а также какие изменения могут быть внесены в систему в будущем. Она может включать в себя:

- системные компоненты: описание серверов, сетей, хранилищ данных и др.;
- программное обеспечение: описание используемых программных компонентов, библиотек, операционных систем и баз данных;
- структуру данных: описание данных, используемых в системе, и их хранение;
- описание узлов и соединений: указание соединений между компонентами и узлами системы;
- описание структуры интерфейсов: описание интерфейсов между различными компонентами информационной системы.

Физическая модель информационной системы является необходимым шагом в процессе ее создания, поскольку она представляет собой конкретный план физической реализации данной системы.

В рамках этого подхода разрабатываются детальные спецификации аппаратного и программного обеспечения, включая операционные системы, базы данных, сетевую инфраструктуру и другие компоненты, которые необходимы для обеспечения полноценного функционирования системы.

Физическая модель информационной системы имеет ряд преимуществ, таких как:

- создание более полной и точной визуализации системы, которая легко понятна как техническим, так и не техническим специалистам;
- возможность избежать ошибок при установке системы;

- создание более сложных систем на основе уже разработанной, что позволяет повысить эффективность процесса разработки;
- обеспечение оптимальной работы вычислительной среды;
- защита от возможных угроз безопасности.

Этот подход является важным элементом разработки информационных систем и используется в расчете на более эффективную и точную работу системы в будущем.

Система обнаружения вторжений может как пропускать трафик непосредственно через свой узел, так и сканировать зеркалированный трафик с интерфейсов других сетевых устройств. Это определяет возможность реагирования на инциденты и запрет дальнейшей передачи трафика – в случае сквозного подключения СОВ может автоматически сбрасывать соединение, но такой подход увеличит общую задержку трафика.

На этапе тестирования системы мы рассмотрим параллельное подключение системы обнаружения вторжений, с дальнейшей перспективой сквозного подключения. В схеме сети это будет выглядеть следующим образом (рис. 12).

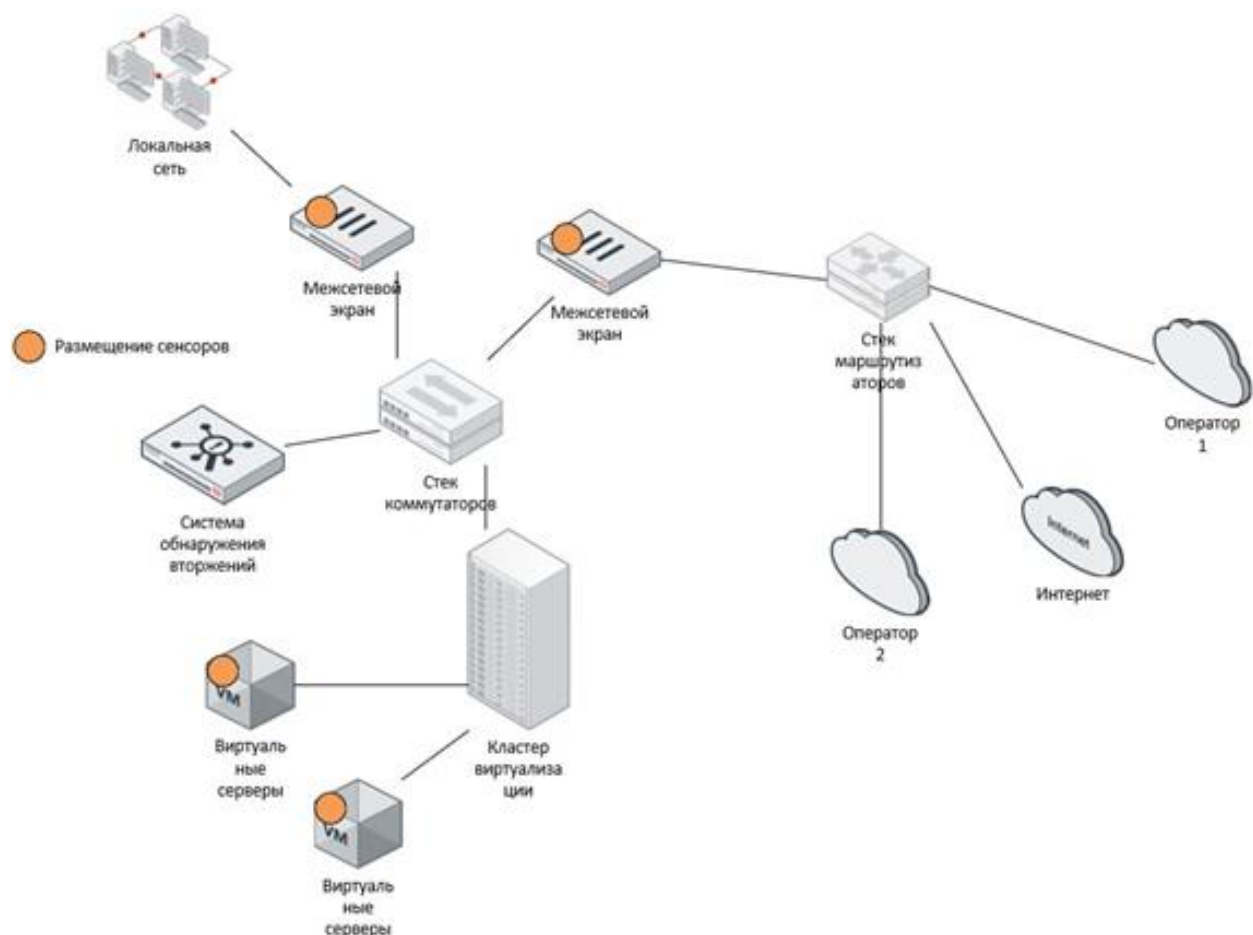


Рисунок 12 – Физическое размещение системы обнаружения вторжений в типовой сети

Основной функционал СОВ – анализ трафика – достигается за счёт зеркалирования всего внутреннего и внешнего трафика со стека коммутаторов сети. Также возможно размещение дополнительных сенсоров на сетевых узлах и передача отдельных потоков трафика с каждого сенсора. Тогда возможно настроить управление системой обнаружения вторжений отдельно для каждой группы узлов и задавать соответствующие правила реагирования. Такой функционал подходит для дальнейшего развития системы при внедрении её в имеющуюся сетевую инфраструктуру, поэтому на этапе тестирования мы используем только сканирование поступающего трафика.

Физически система обнаружения вторжений представляет собой сервер (контроллер) с установленным программным обеспечением. Ядром нашей

системы является Suricata, а плагинами, отвечающими за самообучение системы – алгоритмы ALAD + LERAD.

## Вывод по Главе 2

В ходе работы была разработана логическая модель предлагаемой системы обнаружения вторжений, из которой стало понятно, какие именно процессы затрагиваются в результате её работы, а также какие именно аспекты улучшают внедрение алгоритмов искусственного интеллекта.

В новой модели процесс добавления политик автоматизируется за счёт самообучения системы на базе алгоритмов интеллектуального анализа пакетов.

В диаграмме вариантов использования было показано, за какие аспекты разрабатываемой информационной системы отвечают люди (администратор и оператор системы), а за какие – искусственный интеллект (AI-алгоритмы). UML диаграмма вариантов использования поможет разработчикам определить, как система должна взаимодействовать с пользователем и как пользователи будут использовать систему.

В результате можно сделать вывод, что алгоритмы глубокого анализа позволяют экономить время реагирования на инциденты, затрачиваемое оператором и администратором в процессе работы системы.

Физическая схема системы определяет расположение СОВ в периметре сети, а также как именно система взаимодействует с трафиком и уязвимыми узлами.

Такой вариант расположения системы обеспечивает более гибкий и масштабируемый подход к защите сетей и ресурсов, удобство управления, более высокую эффективность использования ресурсов, увеличение безопасности и улучшение производительности.

Таким образом, разработав модель проектируемой системы обнаружения вторжений, можно переходить к этапу практического тестирования.

## Глава 3. Тестирование системы обнаружения вторжений

### 3.1 Настройка IDS Suricata для работы со сторонними алгоритмами

Suricata – свободно распространяемое ПО, поэтому установка его возможна двумя путями – из исходников, либо из репозитория. Репозитории содержат настроенные стандартные сборки программного комплекса, поэтому для использования дополнительных функций необходима сборка ПО из исходников.

После скачивания исходного кода необходимо установить пакеты зависимостей: `libcapp`, `libnet`, `pkg-config` и другие, полный перечень приведён в документации проекта. Также важным моментом при установке Suricata является выставление правильного времени системы и часового пояса. Для синхронизации времени мы будем использовать утилиту `chrony`. Данная утилита обеспечивает своевременное обновление системного времени через протокол NTP (Network Time Protocol) и предотвратит ошибки Suricata, связанные с некорректными отметками времени пакетов трафика.

Для конфигурации Suricata в режиме IDS необходимо установить следующие параметры сборки:

- `--prefix=/usr/` – определяет место установки бинарных файлов в `/usr/bin`;
- `--sysconfdir=/etc/` – определяет расположение конфигурационных файлов в `/etc/suricata/` вместо `/usr/local/etc/` по умолчанию;
- `--localstatedir=/var/` – определяет хранение журналов событий Suricata в `/var/log/suricata/` вместо `/usr/local/var/log/suricata/` по умолчанию;
- `--enable-lua` – включает поддержку lua-скриптов для обнаружения атак и вывода данных.

Указанные установки необходимы для снижения возможных конфликтов с пользовательскими наборами правил, в число которых входят правила, генерируемые алгоритмами ALAD и LERAD.

Затем командой `make` необходимо собрать сконфигурированный пакет Suricata и установить его с помощью `make install`.

Для автоматического запуска Suricata используется `bash`-скрипт (Приложение А). Данный скрипт обеспечивает запуск IDS при старте системы, а также автоматическую перезагрузку при сбоях. Использование данного скрипта позволяет изменять особенности работы Suricata путём редактирования конфигурационного файла, без полной переустановки системы обнаружения вторжений.

Далее необходимо определить сетевой интерфейс, пакеты с которого будут подвергаться мониторингу. По умолчанию Suricata отслеживает сетевой интерфейс `eth0`, но в случае использования виртуальных машин или нестандартных дистрибутивов Linux рабочий интерфейс может отличаться. Найти необходимый сетевой интерфейс можно при помощи команды `ifconfig` с флагом `-a` (рис. 13), либо при помощи команды `ip link show`.



```
user@localhost:/home/user
File Edit View Search Terminal Help
[root@localhost user]# ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.233.131 netmask 255.255.255.0 broadcast 192.168.233.255
    inet6 fe80::9c62:46ee:d8f2:6c0e prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:5c:20:b3 txqueuelen 1000 (Ethernet)
    RX packets 83366 bytes 125306449 (119.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 36748 bytes 2219044 (2.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 68 bytes 5920 (5.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 68 bytes 5920 (5.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 192.168.122.1 netmask 255.255.255.0 broadcast 192.168.122.255
    ether 52:54:00:b0:3e:f1 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

virbr0-nic: flags=4098<BROADCAST,MULTICAST> mtu 1500
    ether 52:54:00:b0:3e:f1 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Рисунок 13 - Отображение сетевых интерфейсов системы

Как видно из скриншота, основным сетевым интерфейсом нашей виртуальной машины является ens33, поэтому именно его необходимо записать в конфигурационный файл Suricata. Выбранный интерфейс необходимо указать в двух местах: в настройках Suricata по умолчанию (/etc/default/suricata) прописывается строка IFACE=ens33, далее в пользовательских установках (/etc/suricata/suricata.yaml) прописываются интерфейсы для служб af-packet, rpsar, pfring.

После определения сетевого интерфейса для работы Suricata можно приступить к добавлению правил обработки пакетов. Правила добавляются при помощи инструмента suricata-update. Для начала выясним, какие правила

использует Suricata по умолчанию при помощи команды `suricata-update list-sources` (рис. 14).

```
Name: sslbl/ja3-fingerprints
  Vendor: Abuse.ch
  Summary: Abuse.ch Suricata JA3 Fingerprint Ruleset
  License: Non-Commercial
Name: tgreen/hunting
  Vendor: tgreen
  Summary: Threat hunting rules
  License: GPLv3
Name: sslbl/ssl-fp-blacklist
  Vendor: Abuse.ch
  Summary: Abuse.ch SSL Blacklist
  License: Non-Commercial
Name: et/open
  Vendor: Proofpoint
  Summary: Emerging Threats Open Ruleset
  License: MIT
Name: ptresearch/attackdetection
  Vendor: Positive Technologies
  Summary: Positive Technologies Attack Detection Team ruleset
  License: Custom
Name: scwx/security
  Vendor: Secureworks
  Summary: Secureworks suricata-security ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: scwx/malware
  Vendor: Secureworks
  Summary: Secureworks suricata-malware ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: et/pro
  Vendor: Proofpoint
  Summary: Emerging Threats Pro Ruleset
  License: Commercial
  Replaces: et/open
  Parameters: secret-code
  Subscription: https://www.proofpoint.com/us/threat-insight/et-pro-ruleset
Name: etnetera/aggressive
  Vendor: Etnetera a.s.
  Summary: Etnetera aggressive IP blacklist
  License: MIT
[root@localhost user]# █
```

Рисунок 14 – Списки правил Suricata по умолчанию

Для успешного добавления правил, они должны соответствовать следующему формату:

- действие, описывающее, что должно произойти при выявлении угрозы;
- заголовок, описывающий протокол, IP-адрес или порт правила, а также куда следует перенаправить пакет;
- опции правила, детализирующие его работу.

В число доступных действий входят следующие:

- alert – выводит предупреждение;
- pass – останавливает дальнейшее изучение пакета;
- drop – отбрасывает пакет и выводит предупреждение;
- reject – возвращает отправителю пакета ошибку `destination_unreachable` по протоколу RST/ICMP;
- rejectdst – отправляет вышеуказанную ошибку получателю пакета;
- rejectboth – отправляет ошибку обеим сторонам обмена данными.

Заголовки протоколов в правилах Suricata доступны в общем виде: `tcp` для tcp-трафика, `udp` для udp-трафика, `icmp` для технической информации и `ip` для всех протоколов. Кроме этого, возможна идентификация пакетов, использующих протоколы уровня приложений: `http`, `ftp`, `tls` (включая `ssl`), `smb`, `dns`, `dhcp`, `ssh`, `smtp`, `sip` и других.

Для указания IP-адресов и портов можно использовать операторы `./..` – диапазон адресов (в нотации CIDR), `!` – исключение, а также `[.., ..]` – группировка. Например, если необходимо определить правило, реагирующее на все пакеты с диапазона адресов `192.168.1.0/24`, кроме `192.168.1.5`, то запись будет выглядеть так: `[192.168.1.0/24, !192.168.1.5]`. Аналогичные операторы действуют для указания портов адресов.

Опции – это, чаще всего, самый объёмный элемент правила. Они указываются в скобках и разделяются точкой с запятой. Опциями определяется, какие именно данные в пакете необходимо учитывать для срабатывания правила.

Для адаптации правил Snort под использование в системе Suricata необходимо определить основные отличия. Во-первых, действия Snort отличаются от возможных действий Suricata за счёт задействования динамических правил Snort. Поэтому все правила Snort, содержащие ключевые слова `dynamic` и `activate` необходимо разбить на отдельные правила. Второе важное отличие – Snort требует явного указания протокола трафика и порта передачи для работы предпроцессора, а Suricata умеет автоматически определять большинство прикладных протоколов и порты. Соответственно, правила Snort,

направленные на выявление однотипных атак с различных портов можно сгруппировать, убрав привязку к исследуемым сетевым портам ключевым словом `any`. Третье принципиальное отличие в работе правил – правила Suricata распространяются на поток трафика, тогда как Snort исследует каждый пакет в отдельности. Коррекция правил на исследование потоков потребует практически полной их переработки, поэтому в указанном контексте правила Snort меняться не будут. Однако стоит учитывать, что Suricata может генерировать два оповещения на один пакет трафика в случаях, когда пакет инспектируется сам по себе и в составе потока. Остальные отличия правил Snort и Suricata относятся к использованию конкретных методов и ключевых слов, поэтому подробно разбирать их в данной работе не имеет практического смысла.

Подготовив правила работы с трафиком, необходимо обеспечить IDS сквозную передачу пакетов с других узлов, чтобы приступить к практической части эксперимента.

## 3.2 Зеркалирование трафика на IDS

Передача трафика на IDS может осуществляться как на уровне коммутатора, так и напрямую с одного узла на другой. На коммутаторе для дублирования трафика с одного порта на другой используется механизм SPAN (Switch Port Analyzer). Для его использования необходимо указать источник трафика и получателя трафика. В качестве источника могут выступать отдельные порты коммутатора, либо группы портов (VLAN). В качестве получателя указывается один из портов коммутатора. Принцип действия SPAN указан на рисунке 15.

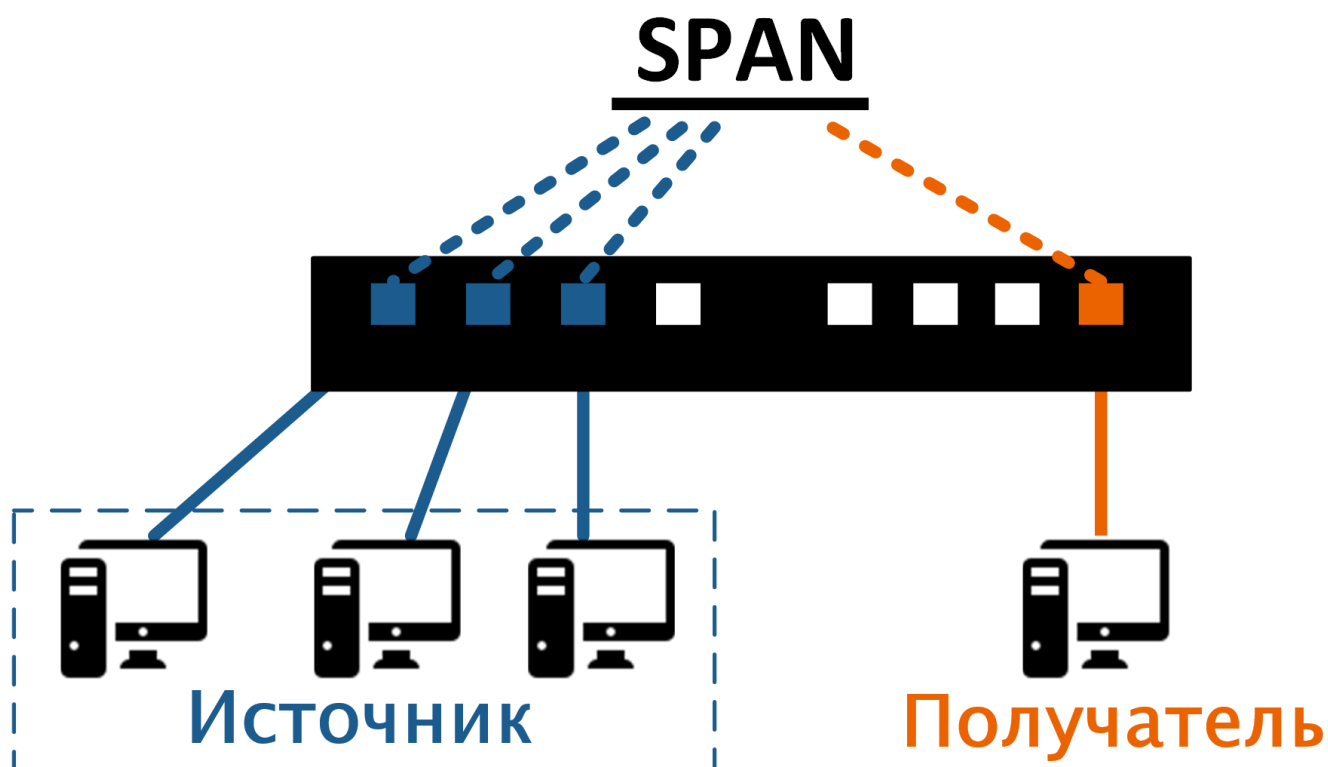


Рисунок 15 – Принцип зеркалирования SPAN

Очевидно, такой способ передачи трафика сработает только в том случае, если IDS будет находиться на том же коммутаторе, что и исследуемые узлы. В масштабных сетях такое возможно далеко не всегда, поэтому существуют альтернативные способы передачи трафика.

Для сетей коммутаторов чаще всего применяется метод RSPAN (Remote SPAN). В случае его использования создаётся специальная RSPAN-сессия, в которую входят порты-источники, trunk-порты, соединяющие коммутаторы, а также порт назначения (рис. 16).

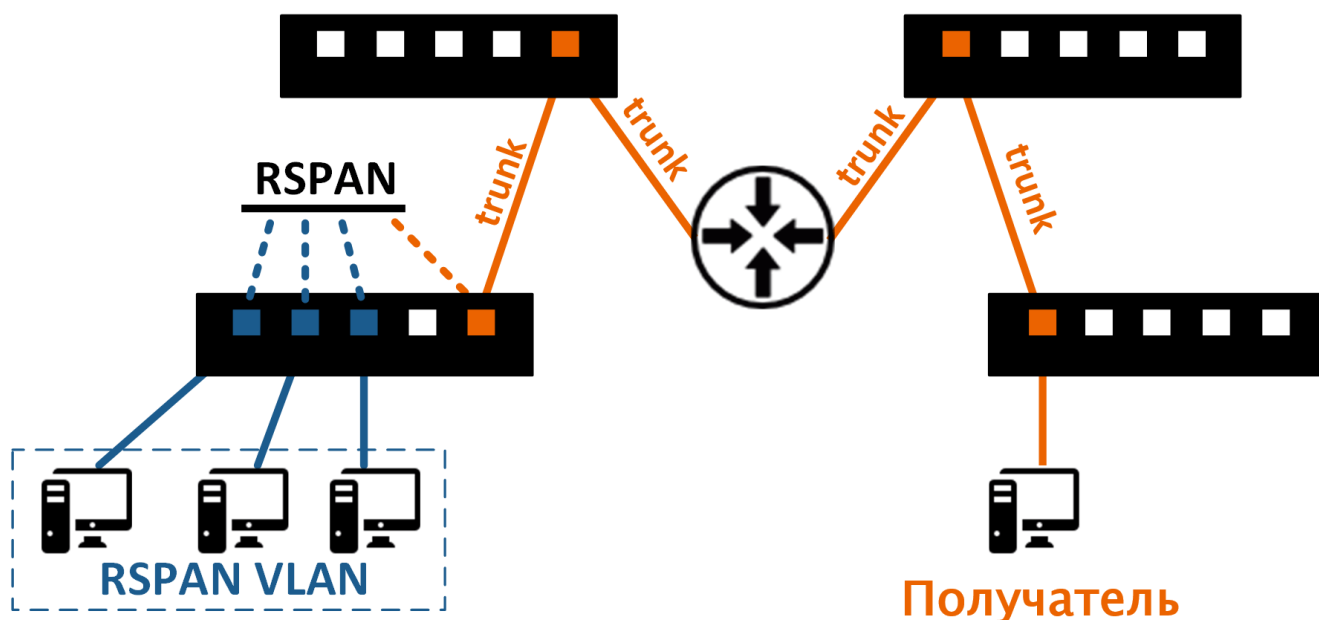


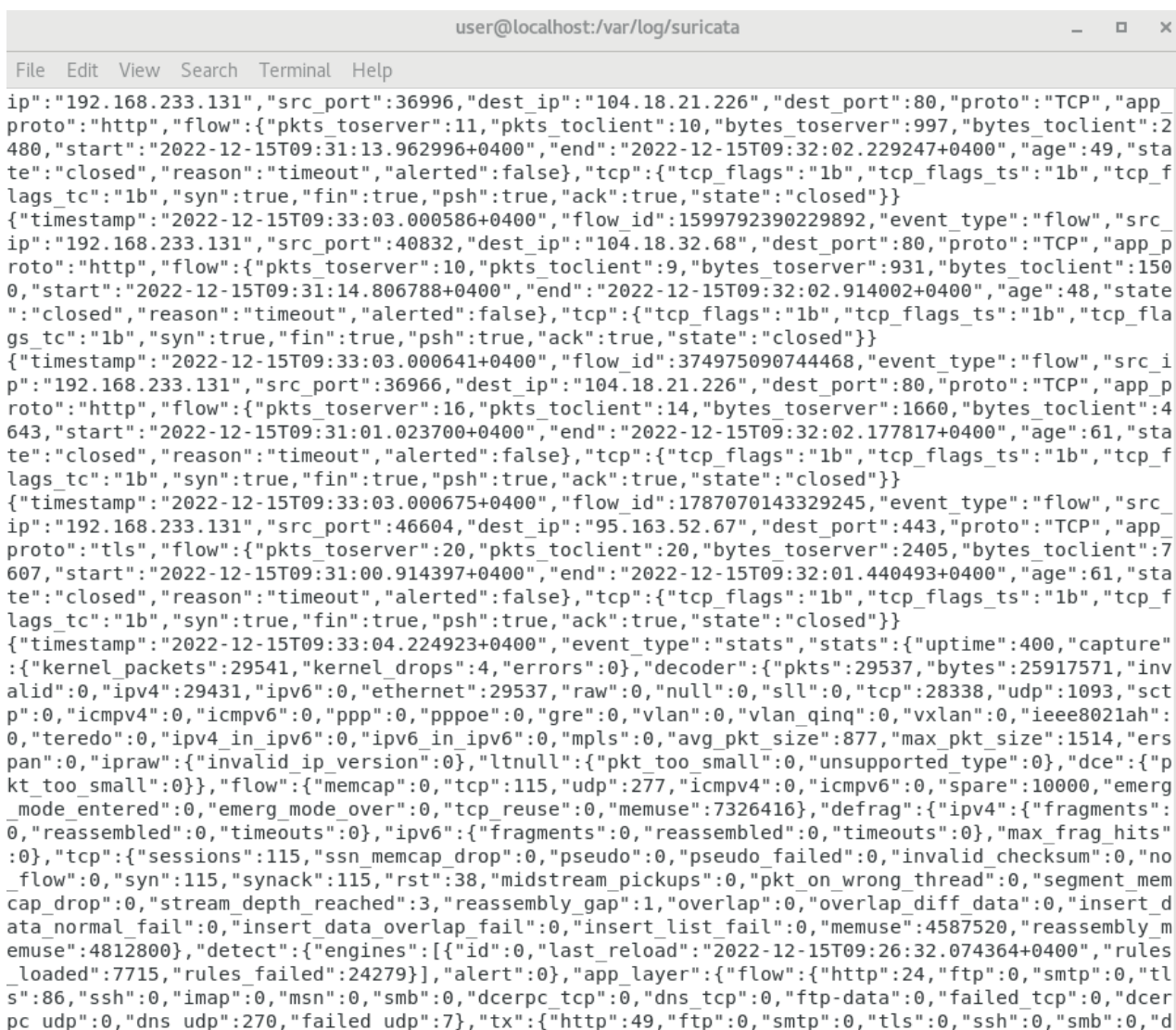
Рисунок 16 – Принцип зеркалирования RSPAN

Необходимо учесть, что при объявлении порта получателем, порт сможет только принимать дублированный трафик, остальные возможности порта блокируются.

Прямая передача трафика между двумя хостами осуществляется при помощи сторонних программных средств и целесообразна при невозможности использования коммутаторов для зеркалирования, например, в случаях значительной физической удалённости узлов и отсутствия общей сети. В нашем случае для дублирования трафика будет достаточно RSPAN.

Для проверки корректности работы Suricata и сессии RSPAN создадим сетевую активность на исходном узле. Логи и предупреждения Suricata записываются в файл `/var/log/suricata/eve.json`. Открываем его при помощи

команды tail с флагом -f и наблюдаем постоянное обновление файла при активной передаче пакетов исходного узла (рис. 17).



```
user@localhost:/var/log/suricata
File Edit View Search Terminal Help
ip": "192.168.233.131", "src_port": 36996, "dest_ip": "104.18.21.226", "dest_port": 80, "proto": "TCP", "app_
proto": "http", "flow": {"pkts_toserver": 11, "pkts_toclient": 10, "bytes_toserver": 997, "bytes_toclient": 2
480, "start": "2022-12-15T09:31:13.962996+0400", "end": "2022-12-15T09:32:02.229247+0400", "age": 49, "sta
te": "closed", "reason": "timeout", "alerted": false}, "tcp": {"tcp_flags": "1b", "tcp_flags_ts": "1b", "tcp_f
lags_tc": "1b", "syn": true, "fin": true, "psh": true, "ack": true, "state": "closed"}}
{"timestamp": "2022-12-15T09:33:03.000586+0400", "flow_id": 1599792390229892, "event_type": "flow", "src_
ip": "192.168.233.131", "src_port": 40832, "dest_ip": "104.18.32.68", "dest_port": 80, "proto": "TCP", "app_p
roto": "http", "flow": {"pkts_toserver": 10, "pkts_toclient": 9, "bytes_toserver": 931, "bytes_toclient": 150
0, "start": "2022-12-15T09:31:14.806788+0400", "end": "2022-12-15T09:32:02.914002+0400", "age": 48, "sta
te": "closed", "reason": "timeout", "alerted": false}, "tcp": {"tcp_flags": "1b", "tcp_flags_ts": "1b", "tcp_fla
gs_tc": "1b", "syn": true, "fin": true, "psh": true, "ack": true, "state": "closed"}}
{"timestamp": "2022-12-15T09:33:03.000641+0400", "flow_id": 374975090744468, "event_type": "flow", "src_i
p": "192.168.233.131", "src_port": 36966, "dest_ip": "104.18.21.226", "dest_port": 80, "proto": "TCP", "app_p
roto": "http", "flow": {"pkts_toserver": 16, "pkts_toclient": 14, "bytes_toserver": 1660, "bytes_toclient": 4
643, "start": "2022-12-15T09:31:01.023700+0400", "end": "2022-12-15T09:32:02.177817+0400", "age": 61, "sta
te": "closed", "reason": "timeout", "alerted": false}, "tcp": {"tcp_flags": "1b", "tcp_flags_ts": "1b", "tcp_f
lags_tc": "1b", "syn": true, "fin": true, "psh": true, "ack": true, "state": "closed"}}
{"timestamp": "2022-12-15T09:33:03.000675+0400", "flow_id": 1787070143329245, "event_type": "flow", "src_
ip": "192.168.233.131", "src_port": 46604, "dest_ip": "95.163.52.67", "dest_port": 443, "proto": "TCP", "app_
proto": "tls", "flow": {"pkts_toserver": 20, "pkts_toclient": 20, "bytes_toserver": 2405, "bytes_toclient": 7
607, "start": "2022-12-15T09:31:00.914397+0400", "end": "2022-12-15T09:32:01.440493+0400", "age": 61, "sta
te": "closed", "reason": "timeout", "alerted": false}, "tcp": {"tcp_flags": "1b", "tcp_flags_ts": "1b", "tcp_f
lags_tc": "1b", "syn": true, "fin": true, "psh": true, "ack": true, "state": "closed"}}
{"timestamp": "2022-12-15T09:33:04.224923+0400", "event_type": "stats", "stats": {"uptime": 400, "capture"
: {"kernel_packets": 29541, "kernel_drops": 4, "errors": 0}, "decoder": {"pkts": 29537, "bytes": 25917571, "inv
alid": 0, "ipv4": 29431, "ipv6": 0, "ethernet": 29537, "raw": 0, "null": 0, "sll": 0, "tcp": 28338, "udp": 1093, "sct
p": 0, "icmpv4": 0, "icmpv6": 0, "ppp": 0, "pppoe": 0, "gre": 0, "vlan": 0, "vlan_qinq": 0, "vxlan": 0, "ieee8021ah":
0, "teredo": 0, "ipv4_in_ipv6": 0, "ipv6_in_ipv6": 0, "mpls": 0, "avg_pkt_size": 877, "max_pkt_size": 1514, "ers
pan": 0, "ipraw": {"invalid_ip_version": 0}, "ltnull": {"pkt_too_small": 0, "unsupported_type": 0}, "dce": {"p
kt_too_small": 0}, "flow": {"memcap": 0, "tcp": 115, "udp": 277, "icmpv4": 0, "icmpv6": 0, "spare": 10000, "emerg
_mode_entered": 0, "emerg_mode_over": 0, "tcp_reuse": 0, "memuse": 7326416}, "defrag": {"ipv4": {"fragments":
0, "reassembled": 0, "timeouts": 0}, "ipv6": {"fragments": 0, "reassembled": 0, "timeouts": 0}, "max_frag_hits"
: 0}, "tcp": {"sessions": 115, "ssn_memcap_drop": 0, "pseudo": 0, "pseudo_failed": 0, "invalid_checksum": 0, "no
_flow": 0, "syn": 115, "synack": 115, "rst": 38, "midstream_pickups": 0, "pkt_on_wrong_thread": 0, "segment_mem
cap_drop": 0, "stream_depth_reached": 3, "reassembly_gap": 1, "overlap": 0, "overlap_diff_data": 0, "insert_d
ata_normal_fail": 0, "insert_data_overlap_fail": 0, "insert_list_fail": 0, "memuse": 4587520, "reassembly_m
emuse": 4812800}, "detect": {"engines": [{"id": 0, "last_reload": "2022-12-15T09:26:32.074364+0400", "rules
_loaded": 7715, "rules_failed": 24279}], "alert": 0}, "app_layer": {"flow": {"http": 24, "ftp": 0, "smtp": 0, "tl
s": 86, "ssh": 0, "imap": 0, "msn": 0, "smb": 0, "dcerpc_tcp": 0, "dns_tcp": 0, "ftp_data": 0, "failed_tcp": 0, "dcer
pc_udp": 0, "dns_udp": 270, "failed_udp": 7}, "tx": {"http": 49, "ftp": 0, "smtp": 0, "tls": 0, "ssh": 0, "smb": 0, "d
```

Рисунок 17 – Состав предупреждений и событий Suricata

Выделение интересующих событий осуществляется при помощи утилит грер, как и в первом вычислительном эксперименте. Для проверки системы и динамических алгоритмов необходимо проведение эксперимента, аналогичного предыдущим.

### 3.3 Динамическая генерация правил алгоритмами и тестирование полученной системы

Для пересылки пакетов, обработанных правилами Suricata, для дальнейшей обработки алгоритмами ALAD и LERAD необходимо изменить конфигурационные файлы, указав сервис получения пакетов в соответствии со способом автозапуска (etc/default/suricata).

В случае Snort дополнение правил указанными алгоритмами расширяет стандартный список правил, а в случае Suricata рекомендуется создать отдельный список правил для исключения конфликтов в работе IDS из-за возможной несовместимости ключевых слов и методов.

Первоначальное тестирование полученной системы проведём аналогично предыдущему эксперименту. Для этого используется тот же дамп пакетов объёмом 1,93 ГБ, содержащий 274 атаки. Передача пакетов осуществляется при помощи Wireshark на сетевой интерфейс lo (локальная петля). Для тестирования используются различные конфигурации виртуальной машины с целью оценки влияния режима многопоточной обработки Suricata и алгоритмов ALAD и LERAD на скорость анализа трафика. Процессор для всех итераций используется один и тот же (AMD Ryzen 5 3350G 4.05 GHz), оперативная память DDR4 частотой 2400 MHz, объём дискового пространства не меняется. Фиксация времени выполнения осуществляется сравнением штампов времени первого и последнего пакетов в логах Suricata, а также встроенным профайлером Linux time для алгоритмов ALAD и LERAD. Результаты эксперимента приведены в таблице 3.

Таблица 3 – Эффективность работы IDS Suricata с алгоритмами ALAD+LERAD

Конфигурация виртуальной машины	Количество обнаруженных атак (из исходных 274)	Время работы алгоритмов, с
2 логических ядра, 4 ГБ оперативной памяти	297	321



### Продолжение таблицы 3

4 логических ядра, 4 ГБ оперативной памяти	297	215
4 логических ядра, 8 ГБ оперативной памяти	297	198
8 логических ядер, 8 ГБ оперативной памяти	297	170

При наличии в дампе 274 атак Suricata во всех случаях сигнализировала о 297 угрозах. Анализ логов предупреждений показал, что одни и те же пакеты в некоторых случаях обрабатывались дважды – в составе tcp-сессии и сами по себе. Таким образом, некоторые предупреждения дублируются. При этом Suricata пропустила 8 атак, которые в дальнейшем были распознаны алгоритмами динамического анализа.

Также по результатам эксперимента прослеживается зависимость скорости анализа в зависимости от характеристик системы. Наибольшее влияние оказывает добавление вычислительных ядер – Suricata гораздо эффективнее работает в многопоточном режиме, чем Snort. Скачок производительности при добавлении двух ядер к двум уже имеющимся составил 33%. Дальнейшие изменения конфигурации показывают меньшее повышение производительности, поскольку сторонние алгоритмы хуже оптимизированы для систем с более 4-х логических ядер.

В результате работы алгоритмов были динамически сгенерированы новые правила для Suricata. Добавляем правила при помощи `suricata-update` и перезапускаем сервис. Результат представлен на рисунке 18.

```
user@localhost:/home/user
File Edit View Search Terminal Help
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Wi
reLurkerUA' is checked but not set. Checked in 2019663 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Su
spicious.Domain.Fake.Browser' is checked but not set. Checked in 2018572 and 0 other
sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'et.Wi
nHttpRequest' is checked but not set. Checked in 2019822 and 1 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Li
nux.Ngioweb' is checked but not set. Checked in 2027508 and 3 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Ti
nba.Checkin' is checked but not set. Checked in 2019169 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Ku
luoz' is checked but not set. Checked in 2019187 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'min.g
ethhttp' is checked but not set. Checked in 2016538 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ETGam
ut' is checked but not set. Checked in 2018246 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.PR
OPFIND' is checked but not set. Checked in 2011457 and 1 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.Ch
roject' is checked but not set. Checked in 2020749 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'et.MS
.XMLHTTP.ip.request' is checked but not set. Checked in 2022050 and 1 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.we
bc2ugx' is checked but not set. Checked in 2016472 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'et.MS
.WinHttpRequest.no.exe.request' is checked but not set. Checked in 2022653 and 0 othe
r sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.GO
OTKIT' is checked but not set. Checked in 2011287 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.ms
08067_header' is checked but not set. Checked in 2008739 and 0 other sigs
15/12/2022 -- 10:43:11 - <Warning> - [ERRCODE: SC_WARN_FLOWBIT(306)] - flowbit 'ET.as
```

Рисунок 18 – Результат добавления новых правил

Помимо ошибок SC\_WARN\_FLOWBIT, также присутствуют ошибки вида SC\_ERR\_RULE\_KEYWORD\_UNKNOWN, что говорит о том, что правила, добавленные при помощи алгоритмов, также необходимо дополнительно адаптировать.

### Вывод по Главе 3

В ходе работы была произведена конфигурация программного комплекса Suricata в качестве IDS, произведена настройка системы обнаружения вторжений для работы в виртуальной среде и автоматизация запуска IDS при сбоях, а также перезагрузках.

Также в рамках работы были описаны механизмы зеркалирования трафика и реализация такого механизма в рамках исследования. Далее был проведён вычислительный эксперимент с целью выявления эффективности работы системы в условиях различных аппаратных конфигураций. В эксперименте использовалась система Suricata с использованием сторонних алгоритмов ALAD и LERAD для глубокого анализа пакетов трафика. По результатам эксперимента были сделаны следующие выводы:

- Suricata эффективнее работает с многопроцессорными системами, чем Snort;

- некоторые пакеты обрабатываются дважды – в составе потока и отдельно от него, что приводит к дубликации предупреждений;

- сторонние алгоритмы менее эффективны в режиме многопоточности, чем сама Suricata и могут тормозить систему при высокой загруженности трафиком.

Кроме этого, по результатам работы алгоритмов были созданы правила для IDS, однако генерация правил использует принципы синтаксиса Snort и требует дальнейшей модификации в соответствии с синтаксисом Suricata.

## Заключение

В ходе исследований, проведённых в настоящей работе, были решены важные задачи информационной безопасности современных информационных систем в части анализа вредоносного трафика.

Основные выводы и научные результаты диссертации заключаются в следующем.

На основании анализа литературы, научных и аналитических статей по проблеме совершенствования систем обнаружения вторжений и предупреждения компьютерных атак были выделены критерии оценки защищаемой информации, определены основные инструменты её защиты, а также сформулирована задача, которую решают СОВ.

Была проведена классификация IDS/IPS-систем, выявлены способы их размещения, алгоритмы работы и методы получения данных. Определены сильные и слабые стороны каждого упомянутого класса. Рассмотрены основные атаки на сетевую инфраструктуру, решаемые при помощи систем обнаружения вторжений.

Были проанализированы алгоритмы, используемые в работе СОВ, как с использованием классических сигнатурных алгоритмов, так и более современных методов поведенческого анализа и интеллектуального анализа данных. Также была определена схема архитектуры СОВ.

По результатам анализа был сделан вывод о необходимости применения в СОВ комбинации алгоритмов анализа трафика для разных уровней сетевой модели OSI и типов сетевой активности.

Для подтверждения гипотезы был проведён вычислительный эксперимент, доказавший превосходство комбинации алгоритмов ALAD+LERAD для обнаружения атак. Проанализированы альтернативы программному обеспечению Snort в качестве движка проектируемой СОВ.

В качестве альтернативного движка наиболее оптимальным была выбрана система обнаружения вторжений Suricata благодаря возможностям аппаратного ускорения работы и поддержке алгоритмов Snort.

На базе Suricata и выбранных алгоритмов была построена логическая модель системы обнаружения вторжений и описаны схемы взаимодействия компонентов. Также была построена диаграмма вариантов использования проектируемой системы, на которой указаны возможности, реализуемые СОВ, и взаимодействие пользователей с логическими частями системы.

Описана физическая модель системы обнаружения вторжений и взаимодействие с другими устройствами локальной сети.

Для использования сторонних алгоритмов произведена настройка IDS Suricata для работы в виртуальной среде, исправлены ошибки работы алгоритмов ALAD+LERAD в работе с Suricata. Произведена адаптация правил Snort для работы с системой обнаружения вторжений Suricata, обеспечена поддержка пакетов готовых сигнатур и описаны возможности создания собственных сигнатурных правил анализа трафика.

Обеспечено подключение СОВ к имитационной среде, в которой проведён вычислительный эксперимент с контрольными фиксированными пакетами трафика при различных конфигурациях виртуальной среды.

По результатам вычислительного эксперимента сделаны практические выводы о преимуществах Suricata в качестве движка СОВ в части аппаратного ускорения, эффективности работы со сторонними алгоритмами поведенческого анализа и скорости работы методом сигнатурного анализа.

Недостатками созданной системы обнаружения вторжений являются избыточность сигнализации об атаках (некоторые пакеты обрабатываются дважды), невозможность инспектирования зашифрованного трафика, а также отсутствие полностью автоматической генерации новых правил (генерируемые правила требуют доработки в соответствии с синтаксисом Suricata).

Для обеспечения сбора статистики системными администраторами также необходимо подключение полученной системы обнаружения вторжений к сторонним модулям статистики и анализа логов.

Таким образом, гипотеза исследования подтверждена, цель работы достигнута, задачи исследования выполнены в полном объеме. Система обнаружения вторжений полностью работоспособна, поэтому данная работа имеет практическую значимость.

Рекомендации по практическому применению результатов работы: спроектированная система требует доработок в части автоматической генерации правил, после которых будет представлять собой готовое решение для защиты локальной сети предприятия от несанкционированного доступа и компьютерных атак.

## Список используемой литературы

1. Шелухин О.И., Чернышев А.И. Исследование и моделирование нейросетевых алгоритмов обнаружения аномальных вторжений в компьютерные сети // Т-Comm. 2014. №12. URL: <https://cyberleninka.ru/article/n/issledovanie-i-modelirovanie-neyrosetevyh-algoritmov-obnaruzheniya-anomalnyh-vtorzheniy-v-kompyuternye-seti> (дата обращения: 10.10.2021).
2. Васильев В.И., Шарабыров И.В. Интеллектуальная система обнаружения атак в локальных беспроводных сетях // Вестник УГАТУ = Vestnik UGATU. 2015. №4 (70). URL: <https://cyberleninka.ru/article/n/intellektualnaya-sistema-obnaruzheniya-atak-v-lokalnyh-besprovodnyh-setyah> (дата обращения: 10.10.2021).
3. Васильев В.И., Шарабыров, И.В. Обнаружение атак в локальных беспроводных сетях на основе интеллектуального анализа данных // Известия ЮФУ. Технические науки. 2014. №2 (151). URL: <https://cyberleninka.ru/article/n/obnaruzhenie-atak-v-lokalnyh-besprovodnyh-setyah-na-osnove-intellektualnogo-analiza-dannyh> (дата обращения: 10.10.2021).
4. Браницкий А.А., Котенко, И.В. Обнаружение сетевых атак на основе комплексирования нейронных, иммунных и нейронечетких классификаторов // Информационно-управляющие системы. 2015. №4 (77). URL: <https://cyberleninka.ru/article/n/obnaruzhenie-setevyh-atak-na-osnove-kompleksirovaniya-neyronnyh-immunnyh-i-neyronechetkih-klassifikatorov> (дата обращения: 10.10.2021).
5. Емельянова Ю.Г., Талалаев А.А., Тищенко И.П., Фраленко В.П. Нейросетевая технология обнаружения сетевых атак на информационные ресурсы // Программные системы: теория и приложения. 2011. №3. URL: <https://cyberleninka.ru/article/n/neyrosetevaya-tehnologiya-obnaruzheniya-setevyh-atak-na-informatsionnye-resursy> (дата обращения: 10.10.2021).
6. Ганиев А. А. Анализ моделей и алгоритмов обнаружения компьютерных атак на основе положений политики безопасности / А. А. Ганиев, Г. И. Касимова. — Текст : непосредственный // Молодой ученый. — 2016. — № 9 (113). — С. 54-57. — URL: <https://moluch.ru/archive/113/29266/> (дата обращения: 10.10.2021).
7. Васильков А. В. Безопасность и управление доступом в информационных системах : учеб. пособие / А.В. Васильков, И.А. Васильков. — Москва : ФОРУМ : ИНФРА-М, 2019. — 368 с. — (Среднее профессиональное образование). - ISBN 978-5-91134-360-6. - Текст : электронный. - URL: <https://znanium.com/catalog/product/987224> (дата обращения: 11.10.2021).
8. Герасименко В.А., Малюк А.А. Основы защиты информации : Учеб. для студентов вузов обучающихся по спец. «Организация и технология защиты

информации» / В. А. Герасименко, А. А. Малюк; М-во общ. и проф. образования Рос. Федерации. Моск. гос. инж.-физ. ин-т (техн. ун-т). - М., 1997. - 537 с.

9. ГОСТ Р 50922-2006. Защита информации. Основные термины и определения. М.: Стандартинформ, 2008.

10. Меткалф Б. Закон Меткалфа сорок лет спустя после рождения Ethernet [Текст]// Б. Меткалф. - Москва: Открытые системы, 2014. - № 01. - 96 с.

11. Белов Е.Б. Основы информационной безопасности. Учебное пособие для вузов [Текст]// Е.Б. Белов., В.П. Лось, Р.В. Мещеряков, А.А. Шелупанов. - Москва: Горячая линия - Телеком, 2006. - С. 69.

12. Бурлаков М.Е. Исследование динамики активности публикации угроз в открытых и закрытых источниках [Текст]// М.Е. Бурлаков. - Самара, Перспективные информационные технологии (ПИТ-2017): труды Международной научно-технической конференции. - Самара: Издательство Самарского научного центра РАН, 2017 - С. 315-320.

13. Burlakov M.E. Research the dynamic of author activities in threats through to public and private sources [Текст]// М.Е. Burlakov. - Samara: Information Technology and Nanotechnology - 2017 Information Security, 2017 - P. 958-961.

14. Allen J. State of the Practice of Intrusion Detection Technologies [Текст]// J. Allen, A. Christie, W. Fithen. - Carnegie Mellon University, Software Engineering Institute, 2000. - P. 17-29.

15. Nadiammai G.V., Hemalatha M. Effective approach toward Intrusion Detection System using data mining techniques - Egyptian Informatics Journal, Volume 15, Issue 1, 2014. - P. 37-50.

16. Пиксаева, А. А. Необходимость моделирования систем обнаружения вторжений и предотвращения информационных атак в условиях постоянно меняющихся требований / А. А. Пиксаева, О. Ю. Копша // Информационные технологии в моделировании и управлении: подходы, методы, решения : V Всероссийская научная конференция с международным участием: сборник материалов, Тольятти, 20–22 апреля 2022 года / Отв. за выпуск Е.В. Панюкова. – Тольятти: Тольяттинский государственный университет, 2022. – С. 272-278. – EDN ASUGVZ.

17. Пиксаева, А. А. Применение оптимального стека алгоритмов для системы обнаружения вторжений / А. А. Пиксаева // Юность и Знания - гарантия Успеха - 2022 : сборник научных статей 9-й Международной молодежной научной конференции : в 3 т., Курск, 15–16 сентября 2022 года. Том 2. – Курск: Юго-Западный государственный университет, 2022. – С. 234-240. – EDN HRK0XS.



18. String Matching Enhancement for Snort IDS, 2010, Safaa Almamory, Ali Jaddoa, Asala Abdul-Razak, Zainab Falah. URL: [https://www.researchgate.net/publication/251990313\\_String\\_matching\\_enhancement\\_for\\_snort\\_IDS](https://www.researchgate.net/publication/251990313_String_matching_enhancement_for_snort_IDS)
19. Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation, 2021, Gauthama Raman M. R., Chuadhry Mujeeb Ahmed & Aditya Mathur. URL: <https://link.springer.com/article/10.1186/s42400-021-00095-5>
20. Feature selection for intrusion detection systems, 2020, Kamalov, F (Kamalov, Firuz), Moussa, S (Moussa, Sherif), Zgheib, R (Zgheib, Rita), Mashaal, O (Mashaal, Omar). URL: <https://www.webofscience.com/wos/woscc/full-record/WOS:000653085400058>
21. Improving Intrusion Detection Systems Using Artificial Neural Networks, 2018, Jasim, YA (Jasim, Yaser A.). URL: <https://www.webofscience.com/wos/woscc/full-record/WOS:000439499300005>
22. Intelligent intrusion detection systems using artificial neural networks, 2018, Shenfield, A (Shenfield, Alex), Day, D (Day, David) 2, Ayesha, A (Ayesha, Aladdin). URL: <https://www.webofscience.com/wos/woscc/full-record/WOS:000435857100008>
23. Network Intrusion Detection System using Deep Learning, 2021, Ashiku, L (Ashiku, Lirim), Dagli, C (Dagli, Cihan). URL: <https://www.webofscience.com/wos/woscc/full-record/WOS:000681039400028>
24. A Sequential Classifiers Combination Method to Reduce False Negative for Intrusion Detection System, 2019, Phetlasy, S (Phetlasy, Sornxayya), Ohzahata, S (Ohzahata, Satoshi), Wu, C (Wu, Celimuge), Kato, T (Kato, Toshihito). URL: <https://www.webofscience.com/wos/woscc/full-record/WOS:000466289200003>
25. Analysis of Intrusion Detection Systems in Industrial Ecosystems, 2017, Juan Enrique Rubio, Cristina Alcaraz, Rodrigo Roman, Javier Lopez. URL: <https://www.nics.uma.es/pub/papers/1662.pdf>
26. A Survey of Intrusion Detection & Prevention Techniques, 2011, Usman Asghar Sandhu, Sajjad Haider, Salman Naseer, Obaid Ullah Ateeb. URL: [https://www.researchgate.net/publication/340931654\\_A\\_Survey\\_of\\_Intrusion\\_Detection\\_Prevention\\_Techniques](https://www.researchgate.net/publication/340931654_A_Survey_of_Intrusion_Detection_Prevention_Techniques)
27. Survey of intrusion detection systems: techniques, datasets and challenges, 2019, Ansam Khraisat, Iqbal Gondal, Peter Vamplew & Joarder Kamruzzaman. URL: <https://link.springer.com/article/10.1186/s42400-019-0038-7>

28. Axelsson S. Research in Intrusion-Detection Systems: A Survey [Текст]// S. Axelsson - Department of Computer Engineering Chalmers University of Technology, 1999. - P. 12-58.
29. Axelsson S. Intrusion detection systems: A survey and taxonomy [Текст]// S. Axelsson. - Department of Computer Engineering Chalmers University of Technology, 2000. - P. 99-115.
30. Kvarnstrom H. A survey of commercial tools for intrusion detection [Текст]// H. Kvarnstrom. - Department of Computer Engineering Chalmers University of Technology, 1999. - P. 32-54. TCP Metrics Report [Электронный ресурс]: официальный сайт компании «Tenable». URL: <https://www.tenable.com/sc-report-templates/tcp-metrics-report>
31. Usefulness of DARPA Dataset for Intrusion Detection System Evaluation [Электронный ресурс]: официальный сайт организации «Indian Institute of Science». URL: <https://core.ac.uk/download/pdf/11632149.pdf?repositoryId=375>
32. Denial of Service Attacks [Электронный ресурс]: официальный сайт организации «Texas State University». URL: <https://s2.ist.psu.edu/ist451/DDoS-Chap-Gu-June-07.pdf>
33. Lunt T.F. Automated Audit Trail Analysis and Intrusion Detection: A Survey [Текст]// T.F., Lunt. - Proceedings of the 11th National Security Conference, 1988. - P.14.
34. Deconstructing the Computer: Report of a Symposium / Committee on Deconstructing the Computer, Committee on Measuring and Sustaining the New Economy, Board on Science, Technology, and Economic Policy, Policy and Global Affairs, National Research Council - Washington: National Academies Press, 2005. - P. 49-50.
35. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. — СПб.: Питер, 2013. — 816 с.
36. ГОСТ Р 53131-2008 Защита информации. Рекомендации по услугам восстановления после чрезвычайных ситуаций функций и механизмов безопасности информационных и телекоммуникационных технологий. Общие положения. - М: Стандартинформ, 2008. - С. 1-3.
37. Гатчин Ю.А., Сухостат В.В., Куракин А.С., Донецкая Ю.В. ТЕОРИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И МЕТОДОЛОГИЯ ЗАЩИТЫ ИНФОРМАЦИИ – 2-е изд., испр. и доп. – СПб: Университет ИТМО, 2018. – 100 с.

38. Adaptation Techniques for Intrusion Detection and Intrusion Response Systems [Электронный ресурс]: официальный сайт организации «Information Technology and Operations Center». URL: <http://www.secdev.org/idsbiblio/adapt.pdf>
39. Rajesh R.S. Computer Networks: Fundamentals & Applications [Текст] // R.S. Rajesh, K.S. Easwarakumar, R. Balasubramanian. - New Delhi : Vicas Pub, 2012. - P. 19-25.
40. Mann I. Hacking the Human: Social Engineering Techniques and Security Countermeasures [Текст] // I. Mann. - Gover Publishing Ltd, 2012. - 273 p.
41. Дородников Н.А. Автоматизация процессов обеспечения безопасности сетевых ресурсов в гетерогенных ЛВС [Текст] // Н.А. Дородников, Ю.Г. Филиппова, С.А. Арустамов, Ю.А. Гатчин - Йошкар-Ола: Информационные технологии в профессиональной деятельности и научной работе, 2014. -С. 265-269.
42. Васильев В.И. Интеллектуальные системы защиты информации [Текст] // В.И. Васильев. - Москва: «Машиностроение», 2012. - 171 с.
43. Implications of Deep Packet Inspection (DPI) Internet Surveillance for Society [Электронный ресурс]: официальный сайт организации «Uppsala University». URL: <http://fuchs.uti.at/wp-content/uploads/DPI.pdf>
44. Network Traffic Anomaly Detection [Электронный ресурс]: официальный сайт организации «Klipsch School of Electrical and Computer Engineering». URL: <https://arxiv.org/pdf/1402.0856.pdf>
45. Pietro R. Intrusion Detection Systems [Текст] // R. Pietro - Springer Science + Business Media LTD, 2008. - 210 p.
46. Long J. No Tech Hacking: A Guide to Social Engineering, Dumpster Diving, and Shoulder Surfing [Текст] // J. Long. - Syngress Publishing Inc, 2011. -281 p.

## Скрипт автоматического запуска Suricata в случае сбоя

```
#!/bin/sh -e
#
### BEGIN INIT INFO
# Provides:      suricata
# Required-Start:  $time $network $local_fs $remote_fs
# Required-Stop:  $remote_fs
# Default-Start:  2 3 4 5
# Default-Stop:   0 1 6
# Short-Description: Next Generation IDS/IPS
# Description:    Intrusion detection system that will
#                capture traffic from the network cards and will
#                match against a set of known attacks.
### END INIT INFO

. /lib/lsb/init-functions

# Source function library.
if test -f /etc/default/suricata; then
    . /etc/default/suricata
else
    echo "/etc/default/suricata is missing... bailing out!"
fi

# We'll add up all the options above and use them
NAME=suricata
```

```
DAEMON=/usr/bin/$NAME
```

```
if [ -z "$RUN_AS_USER" ]; then
```

```
    USER_SWITCH=
```

```
else
```

```
    USER_SWITCH=--user=$RUN_AS_USER
```

```
fi
```

```
# Use this if you want the user to explicitly set 'RUN' in
```

```
# /etc/default/
```

```
if [ "x$RUN" != "xyes" ]; then
```

```
    log_failure_msg "$NAME disabled, please adjust the configuration to your  
needs "
```

```
    log_failure_msg "and then set RUN to 'yes' in /etc/default/$NAME to enable  
it."
```

```
    exit 0
```

```
fi
```

```
check_root() {
```

```
    if [ "$(id -u)" != "0" ]; then
```

```
        log_failure_msg "You must be root to start, stop or restart $NAME."
```

```
        exit 4
```

```
    fi
```

```
}
```

```
check_nfqueue() {
```

```

    if [ ! -e /proc/net/netfilter/nf_queue ] && [ ! -e
/proc/net/netfilter/nfnetlink_queue ]; then
        log_failure_msg "NFQUEUE support not found !"
        log_failure_msg "Please ensure the nfnetlink_queue module is loaded or built
in kernel"
        exit 5
    fi
}

```

```

check_run_dir() {
    if [ ! -d /var/run/suricata ]; then
        mkdir /var/run/suricata
    fi
    if [ ! -z "$RUN_AS_USER" ]; then
        chown $RUN_AS_USER /var/run/suricata;
    fi
    chmod 0755 /var/run/suricata
    fi
}

```

```

check_root

```

```

case "$LISTENMODE" in
    nfqueue)
        IDMODE="IPS (nfqueue)"
        LISTEN_OPTIONS=" $NFQUEUE"
        check_nfqueue
        ;;

```

```

custom_nfqueue)
IDMODE="IPS (custom multi-nfqueue)"
LISTEN_OPTIONS=" $CUSTOM_NFQUEUE"
check_nfqueue
;;
pcap)
IDMODE="IDS (pcap)"
LISTEN_OPTIONS=" -i $IFACE"
;;
af-packet)
IDMODE="IDS (af-packet)"
LISTEN_OPTIONS=" --af-packet"
;;
*)
echo "Unsupported listen mode $LISTENMODE, aborting"
exit 1
;;
esac

SURICATA_OPTIONS=" -c $SURCONF --pidfile $PIDFILE
$LISTEN_OPTIONS -D -vvv $USER_SWITCH"

```

# See how we were called.

```
case "$1" in
```

```
start)
```

```
if [ -f $PIDFILE ]; then
```

```
PID1=`cat $PIDFILE`
```

```
if kill -0 "$PID1" 2>/dev/null; then
```

```
echo "$NAME is already running with PID $PID1"
```

```

        exit 0
    else
        echo "Likely stale PID `cat $PIDFILE` with $PIDFILE exists, but
process is not running!"
        echo "Removing stale PID file $PIDFILE"
        rm -f $PIDFILE
    fi
fi
check_run_dir
echo -n "Starting suricata in $IDMODE mode..."
$DAEMON $SURICATA_OPTIONS > /var/log/suricata/suricata-start.log
2>&1 &
echo " done."
;;
stop)
echo -n "Stopping suricata: "
if [ -f $PIDFILE ]; then
    PID2=`cat $PIDFILE`
else
    echo " No PID file found; not running?"
    exit 0;
fi
start-stop-daemon --oknodo --stop --quiet --pidfile=$PIDFILE --exec
$DAEMON
if [ -n "$PID2" ]; then
    kill "$PID2"
    ret=$?
    sleep 2
    if kill -0 "$PID2" 2>/dev/null; then

```



```

ret=$?
echo -n "Waiting . "
cnt=0
while kill -0 "$PID2" 2>/dev/null; do
    ret=$?
    cnt=`expr "$cnt" + 1`
    if [ "$cnt" -gt 10 ]; then
        kill -9 "$PID2"
        break
    fi
    sleep 2
    echo -n ". "
done
fi
fi
if [ -e $PIDFILE ]; then
    rm $PIDFILE > /dev/null 2>&1
fi
echo " done."
;;
status)
    # Check if running...
    if [ -s $PIDFILE ]; then
        PID3=`cat $PIDFILE`
        if kill -0 "$PID3" 2>/dev/null; then
            echo "$NAME is running with PID $PID3"
            exit 0
        else

```

```
        echo "PID file $PIDFILE exists, but process not running!"
    fi
else
    echo "$NAME not running!"
fi
;;
restart)
    $0 stop
    $0 start
;;
force-reload)
    $0 stop
    $0 start
;;
*)
    echo "Usage: $0 {start|stop|restart|status}"
    exit 1
esac

exit 0
```