УДК 004.056.5:004.7

Демидова Д. В.

Студент

РГУ нефти и газа (НИУ) имени И. М. Губкина

Комнацкая Е. Д.

Студент

РГУ нефти и газа (НИУ) имени И. М. Губкина

ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ ПРАВИЛ SNORT В ОПЕРАЦИОННОЙ СИСТЕМЕ АЛЬТ

Аннотация. *В статье представлена методика оценки эффективности правил Snort 2.9.17.1 для ОС Альт и результаты её экспериментальной валидации с учётом требований ФСТЭК России и особенностей мандатного контроля доступа (SELinux/AppArmor). Эксперимент проведён на стенде из двух виртуальных машин с ОС Альт в среде Oracle VirtualBox; сравниваются базовый и модифицированный наборы правил для четырёх типов атак: ICMP-флуд, TCP SYN-сканирование, FTP-брутфорс, SQL-инъекция. По результатам 18 прогонов (3 повторения × 2 набора × 3 сценария) рассчитаны Precision, Recall, F1 и метрики производительности. Показано, что модифицированный набор (с механизмами detection_filter и flowbits) снижает число ложных срабатываний на 60–75% при приросте F1-меры до 4% по сравнению с базовым; различия статистически значимы ($p < 0,05$, критерий Стьюдента). Предложен векторный критерий эффективности, объединяющий точность обнаружения, накладные расходы и нормативное соответствие правил.*

Ключевые слова: *Snort, система обнаружения вторжений, ОС Альт, правила Snort, ложные срабатывания, F1-мера, ФСТЭК, импортозамещение, оценка эффективности IDS, detection_filter, flowbits.*

Demidova D. V.

Student

Gubkin Russian State University of Oil and Gas

Komnatskaya E. D.

Student

Gubkin Russian State University of Oil and Gas

EXPERIMENTAL EVALUATION OF THE EFFECTIVENESS OF SNORT RULES IN THE ALT OPERATING SYSTEM

Annotation. *The article presents a methodology for evaluating the effectiveness of Snort 2.9.17.1 rules for Alt OS and the results of its experimental validation, taking into account the requirements of the FSTEC of Russia and the features of mandatory access control (SELinux/AppArmor). The experiment was conducted on a testbed consisting of two virtual machines with Alt OS in the Oracle VirtualBox environment; the baseline and modified rulesets are compared for four types of attacks: ICMP flood, TCP SYN scan, FTP brute-force, and SQL injection. Based on the results of 18 runs (3 repetitions × 2 rulesets × 3 scenarios), Precision, Recall, F1, and performance metrics were calculated. It is shown that the modified ruleset (with the detection_filter and flowbits mechanisms) reduces the number of false positives by 60–75% with an increase in the F1 measure of up to 4% compared to the baseline; the differences are statistically significant ($p < 0.05$, Student's t-test). A vector performance criterion is proposed that combines detection accuracy, overhead, and rule compliance.*

Keywords: *Snort, intrusion detection system, Alt OS, Snort rules, false positives, F1 score, FSTEC, import substitution, IDS effectiveness assessment, detection_filter, flowbits.*

Введение

Ежегодный рост числа кибератак и ужесточение нормативных требований к защите критической информационной инфраструктуры (КИИ) Российской Федерации обусловили широкое применение систем обнаружения вторжений (IDS) в государственных и корпоративных сетях. Одним из наиболее распространённых решений с открытым исходным кодом

остаётся Snort – сигнатурная IDS класса NIDS, поддерживаемая в штатных репозиториях отечественной ОС Альт и официально допустимая к использованию в аттестованных информационных системах [1].

ОС Альт (ALT Linux) сертифицирована ФСТЭК России по классам защиты до 1Г и применяется на объектах КИИ в соответствии с Указом Президента РФ № 166 от 30 марта 2022 года и Федеральным законом № 187-ФЗ. Специфика этой системы – политики мандатного контроля доступа (SELinux/AppArmor), обязательный контроль целостности пакетов, предсказуемые имена сетевых интерфейсов и особенности ядра – непосредственно влияет на установку, настройку и производительность Snort и требует учёта при оценке его эффективности.

Классические международные методики оценки IDS (NIST SP 800-94, наборы данных CICIDS2017, DARPA) не могут быть применены напрямую: они опираются на зарубежные датасеты, не учитывают требования приказов ФСТЭК № 239 и № 117, а использование внешних наборов трафика в сертифицированных системах может нарушать режим конфиденциальности информации [2]. Это обусловливает актуальность разработки специализированной методики, адаптированной к реалиям отечественного регулирования и пригодной для аттестуемых систем.

Целью настоящей работы является разработка и экспериментальная валидация методики количественной оценки эффективности правил Snort 2.9.17.1 в ОС Альт, а также сравнение базового и оптимизированного наборов сигнатур с обоснованием практических рекомендаций по их настройке.

1. Методика оценки эффективности

1.1. Архитектура Snort и особенности ОС Альт

Snort 2.9.17.1 реализует классическую многоуровневую конвейерную архитектуру обработки сетевого трафика. На рисунке 1 представлена схема, включающая модуль захвата пакетов (DAQ/libpcap), декодер заголовков OSI 2-3, блок препроцессоров (frag3, stream5, http_inspect, sfPortscan) и движок

детектирования на основе алгоритмов Aho-Corasick и Boyer-Moore. Завершает конвейер модуль вывода (Output Plugins).

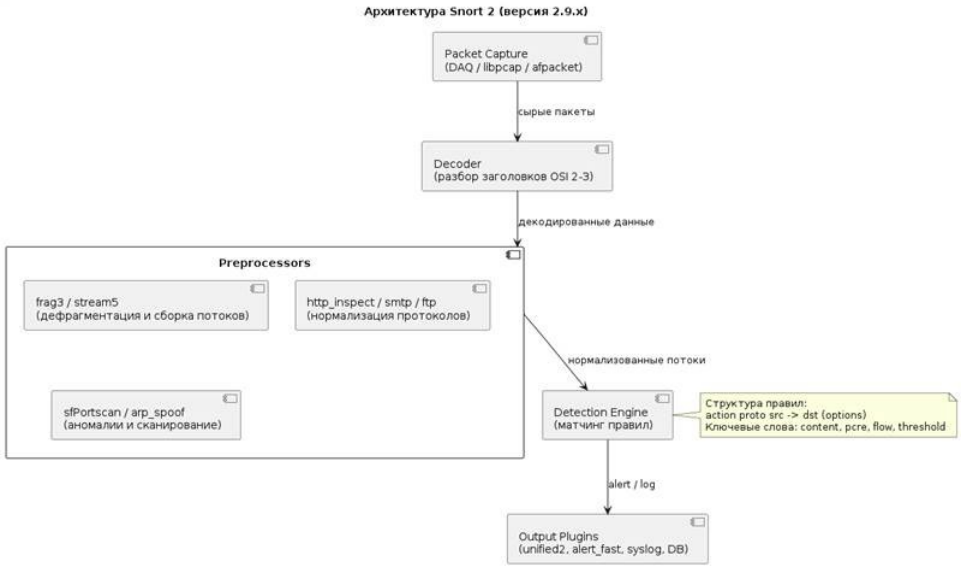

Рисунок 1 – Архитектура Snort 2

Влияние ОС Альт на работу Snort проявляется на нескольких уровнях. Во-первых, библиотека DAQ требует дополнительной настройки путей (/etc/snort/, /var/log/snort/). Во-вторых, механизмы мандатного контроля доступа (AppArmor/SELinux) блокируют захват пакетов без создания соответствующих политик безопасности. В-третьих, однопоточность Snort 2.x приводит к неполной утилизации многоядерных процессоров. При этом встроенные средства контроля целостности Альт позволяют автоматически фиксировать изменения в наборах правил, что упрощает аудит.

1.2. Общая структура методики

Методика строилась поэтапно. Сначала была подготовлена тестовая среда и проверено, что она соответствует политикам безопасности ОС Альт, затем сформированы два набора правил, после чего подготовлены три сценария трафика с заранее известной разметкой. Дальше шёл сам прогон экспериментов – всего их получилось 18, и по каждому вручную сверялась классификация событий по временным меткам. Завершали работу расчёт метрик, статистическая проверка и итоговое сравнение наборов по векторному критерию эффективности.

Главное отличие от типовых зарубежных подходов в том, что вся методика заточена под ОС Альт – учитывает SELinux и требования ФСТЭК, не опирается на иностранные датасеты, а вместо привычной кросс-валидации использует двухстадийную схему «калибровка – верификация». Отдельно проверялись права доступа и целостность файлов через rpm-V – без этого результаты эксперимента нельзя было бы считать валидными для аттестуемой системы.

Сценарии трафика заняли по 900 секунд каждый. В сценарии А прогонялся только легитимный трафик – TCP-сессии iperf3 и HTTP-запросы curl.

1.3. Сценарии трафика

Эксперимент включал три сценария длительностью 900 секунд каждый. Сценарий А – исключительно легитимный трафик: TCP-сессии iperf3 и HTTP-запросы curl. Важно отметить, что «легитимный» означает отсутствие намеренных атак, а не полное отсутствие SYN-пакетов: инструменты iperf3 и curl при установке TCP-соединений генерируют одиночные SYN-пакеты в ходе трёхстороннего рукопожатия. Именно они служат источником ложных срабатываний правила syn_scan в сценарии А при использовании базового набора (без пороговых ограничений). Сценарий Б – только атаки: ICMP-флуд (300 пакетов), TCP SYN-сканирование nmap портов 1–1000, FTP-брутфорс через netcat, SQL-инъекции с явной и URL-кодированной нагрузкой. Легитимный фоновый трафик в этом сценарии полностью исключён. Сценарий В –смешанный трафик (80% легитимного, 20% атак с равномерным распределением по времени) – служит верификационной выборкой.

1.4. Формулы расчёта метрик

Для каждого правила рассчитываются: Precision = TP / (TP + FP); Recall = TP / (TP + FN); F1 = 2 × Precision × Recall / (Precision + Recall). По трём прогонам вычисляются среднее значение $\bar{x}$, стандартное отклонение s и 95%-ный доверительный интервал CI = $\bar{x}$ ± 4,303 · s/√3. Статистическая

значимость различий проверяется двухвыборочным t-критерием Стьюдента ($\alpha = 0{,}05$). Для выбора оптимального порога threshold применяется штрафная функция $L = FP + \lambda \cdot FN$, где $\lambda = 5$ для критических атак (SYN-сканирование, SQL-инъекция) и $\lambda = 2$ для второстепенных.

2. Экспериментальный стенд

2.1. Конфигурация виртуальных машин

Стенд реализован на двух виртуальных машинах с ОС Альт в среде Oracle VirtualBox (рисунок 2). ВМ1 (IP 192.168.10.10) – защитник с Snort 2.9.17.1; ВМ2 (IP 192.168.10.20) –атакующий узел. Обе машины объединены изолированной внутренней сетью (intnet, 192.168.10.0/24). Внешний адаптер NAT использовался исключительно для установки пакетов и был отключён во время экспериментов.

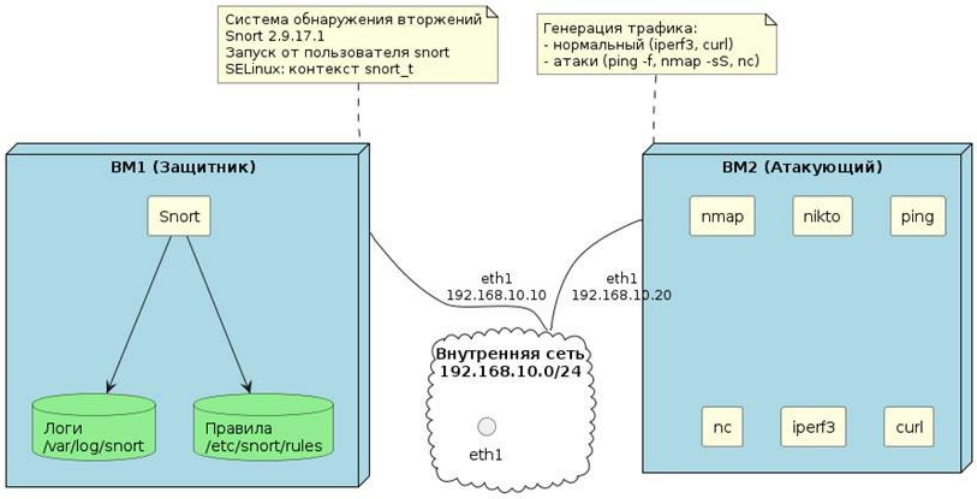

Рисунок 2 – Схема экспериментального стенда

На рисунке 3 представлен результат проверки сетевой связности между ВМ1 и ВМ2 с помощью утилиты ping. Успешный обмен ICMP-пакетами с временем отклика менее 1 мс подтвердил корректность сетевой конфигурации. Синхронизация времени между машинами обеспечена через chronyd (отклонение $\leq 0{,}8$ мс), что критично для точной временно́й разметки срабатываний.

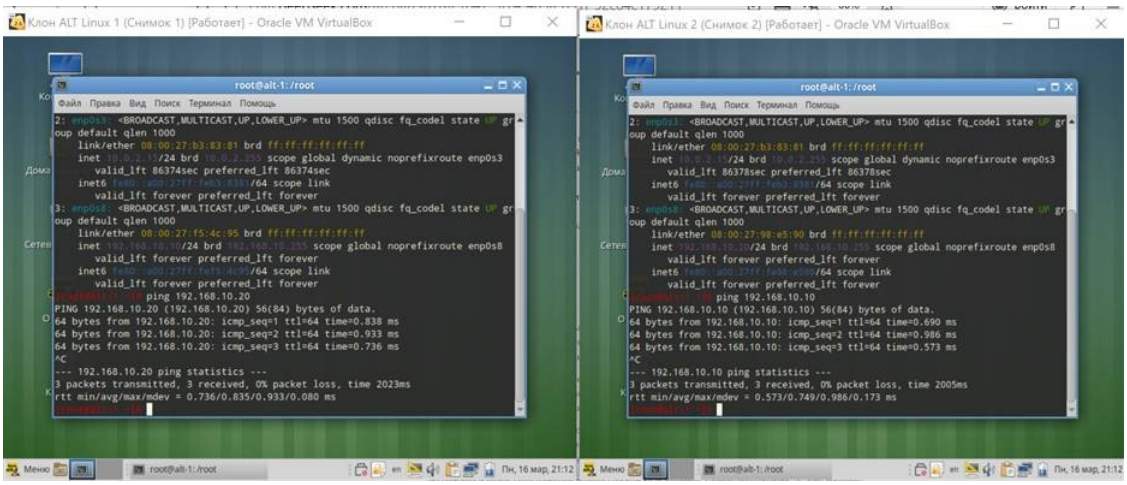

Рисунок 3 – Проверка сетевой связности между ВМ1 и ВМ2

2.2. Настройка Snort и конфигурационные файлы

На ВМ1 создан отдельный системный пользователь snort с ограниченными правами. Структура каталогов приведена к стандарту: /etc/snort/ – конфигурация и правила; /var/log/snort/ – логи. Права доступа к каталогам установлены в 5775 – это позволяет пользователю snort писать в них при сохранении полного контроля со стороны администратора. Конфигурационный файл snort.conf адаптирован под экспериментальную сеть: HOME_NET = 192.168.10.10/32, вывод – alert_fast. На рисунке 4 представлен фрагмент итогового конфигурационного файла.

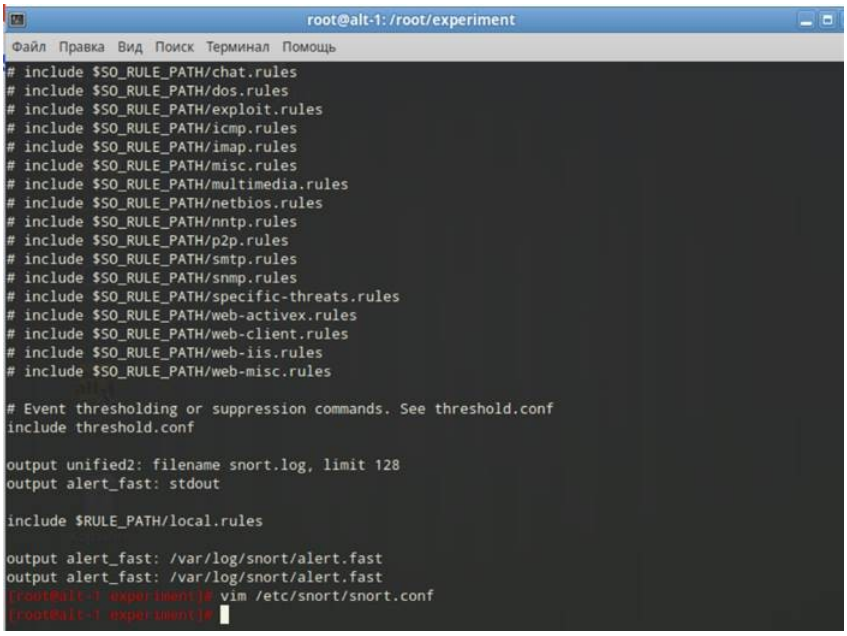

Рисунок 4 – Фрагмент /etc/snort/snort.conf с ключевыми параметрами

Чтобы не запускать каждый прогон вручную, между ВМ1 и ВМ2 настроили SSH без пароля. Была сгенерирована пару RSA-ключей, добавив публичный ключ на ВМ2 в ~/.ssh/authorized_keys (рисунок 5). Это позволит скрипту run_experiment.sh запускать генерацию трафика на атакующей машине, не дожидаясь оператора.

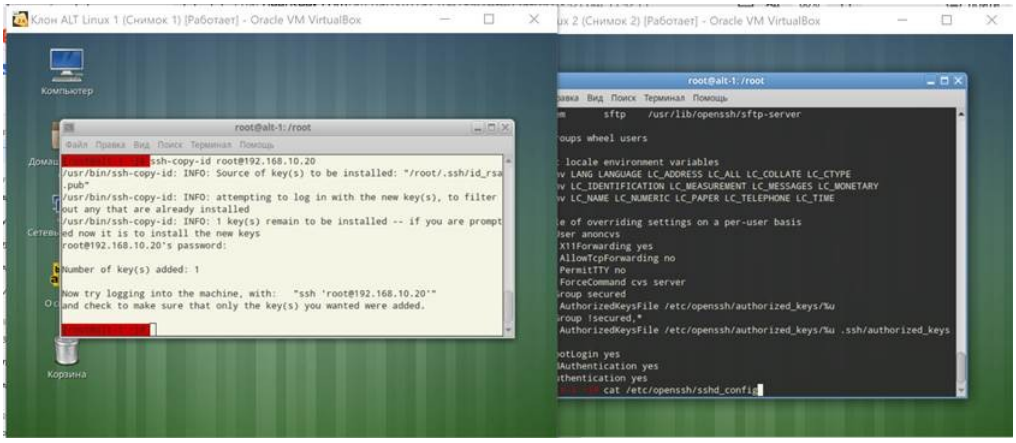

Рисунок 5 – Настройка беспарольного SSH-доступа с ВМ1 на ВМ2

3. Тестовые наборы правил snort

3.1. Базовый набор (local.rules.base)

В базовый набор вошли самые простые сигнатуры на четыре типа атак – без порогов и flowbits-меток. Каждое правило генерирует предупреждение при каждом обнаружении соответствующего паттерна. Содержимое файла показано на рисунке 6.

Рисунок 6 – Листинг базового набора правил

3.2. Модифицированный набор (local.rules.mod)

В модифицированный набор (рисунок 7) внесены целевые оптимизации: для правила TCP SYN-сканирования (sid:1000002) добавлен

порог detection_filter с параметрами track by_src, count 1, seconds 1, ограничивающий число предупреждений с одного источника; для правила FTP-брутфорса (sid:1000004) добавлен флаг flowbits:set,ftp_brute, позволяющий коррелировать события в рамках одной сессии, а также аналогичный порог; для правила SQL-инъекции (sid:1000003) добавлен порог (count 3, seconds 10) и метка flowbits:set,sqli_attempt. Правило ICMP оставлено без изменений, так как каждый пакет потенциально значим для диагностики.

Рисунок 7 – Листинг модифицированного набора правил

4. Проведение экспериментов

4.1. Структура прогонов и хранение результатов

Для каждого из двух наборов правил (base, mod) и каждого из трёх сценариев трафика (А, Б, В) выполнено по три независимых прогона – итого 18. Каждый прогон сохранял результаты в отдельный каталог, имя которого формировалось по шаблону {набор}_{сценарий}_run{N}_{дата_время}. Структура каталогов с результатами всех прогонов показана на рисунке 8.

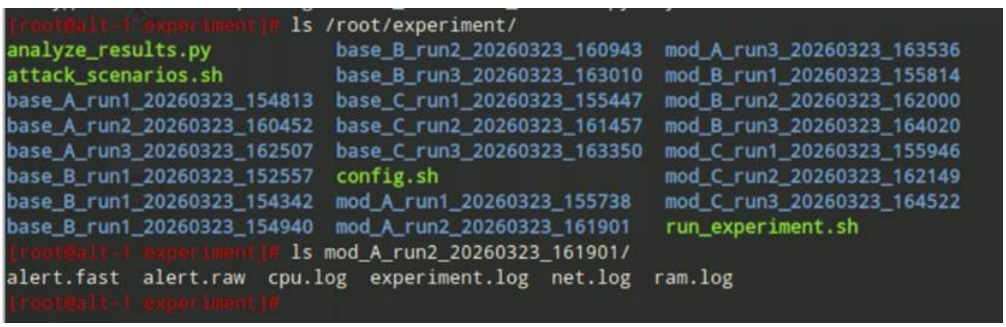

Рисунок 8 – Структура каталогов с результатами экспериментов

Внутри каждого каталога формировались файлы: alert.raw – полный консольный вывод Snort со всеми предупреждениями; cpu.log – посекундная загрузка CPU (mpstat); ram.log – потребление оперативной памяти (free -m); net.log – сетевая статистика (sar -n DEV); experiment.log – журнал выполнения скрипта.

4.2. Файл alert.raw и классификация срабатываний

На рисунке 9 представлен фрагмент файла alert.raw с зафиксированными срабатываниями правила TCP SYN-сканирования (sid:1000002). Каждая строка содержит временну́ю метку с точностью до микросекунды, идентификатор правила и краткое описание события. По этим меткам производилась классификация событий как TP (истинное срабатывание в интервале атаки ±1 с), FP (срабатывание вне интервалов атак) или FN (атака без срабатывания).

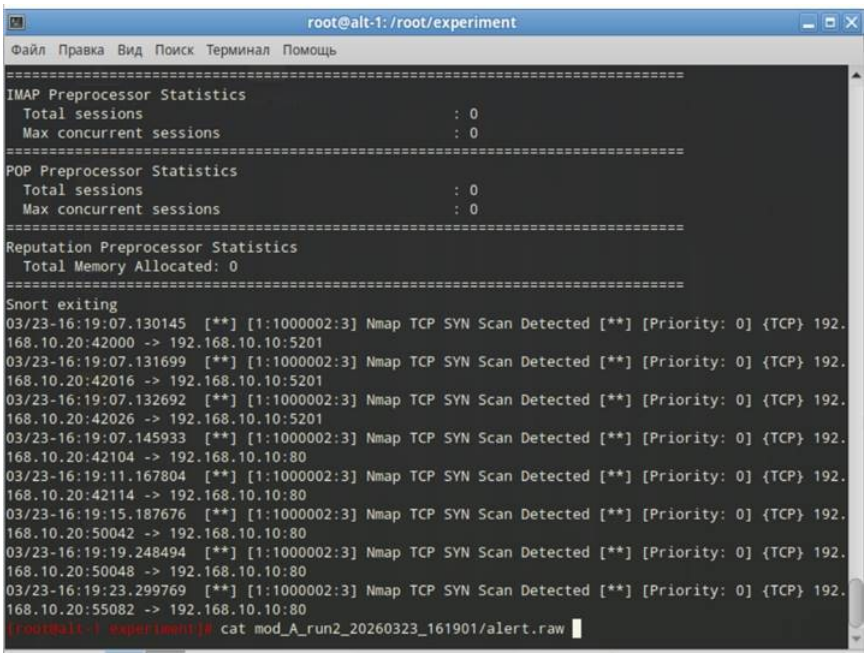

Рисунок 9 – Фрагмент файла alert.raw: срабатывания правила TCP SYN Scan

4.3. Мониторинг производительности

Параллельно с работой Snort собирались метрики производительности. На рисунке 10 показан фрагмент файла cpu.log с посекундными значениями загрузки процессора. Видно, что в период активного SYN-сканирования загрузка возрастает до 25-28% (%usr + %sys), тогда как в периоды

нормального трафика составляет 2-4%. Среднее значение за прогон – 2,62% из-за значительной доли времени ожидания.

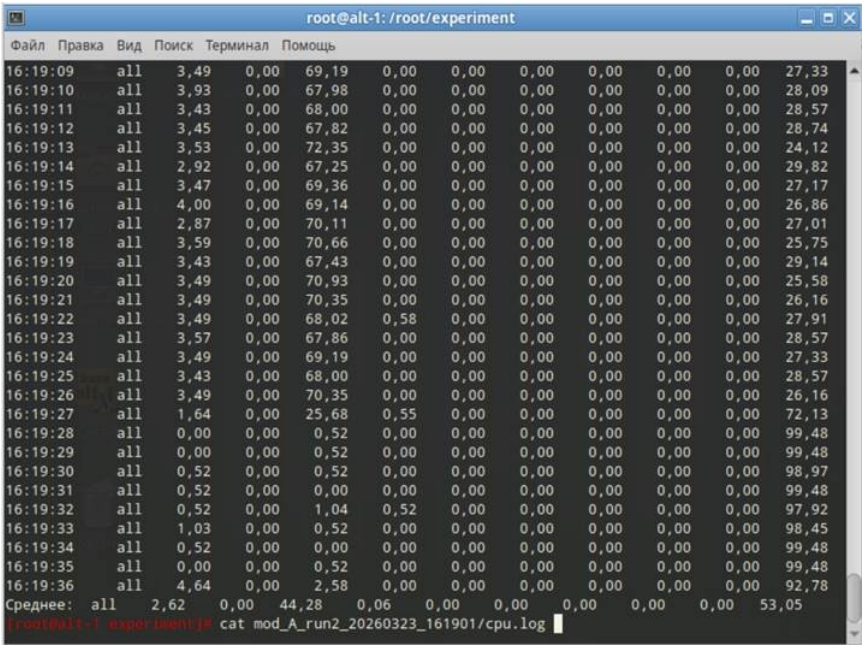

Рисунок 10 – Фрагмент cpu.log: посекундная загрузка CPU в ходе эксперимента

На рисунке 11 приведён фрагмент файла ram.log. Потребление оперативной памяти в ходе всего эксперимента оставалось стабильным на уровне 978 МБ (used) при 1967 МБ всего, что соответствует штатной работе Snort без утечек памяти.

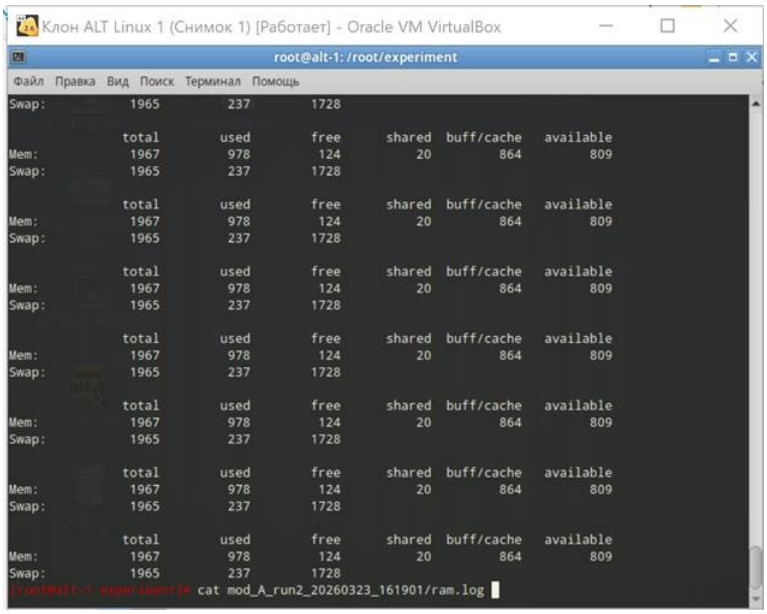

Рисунок 11 – Фрагмент ram.log: потребление оперативной памяти

Файл net.log (рисунок 12) содержит сетевую статистику, собранную утилитой sar -n DEV. В среднем по интерфейсу enp0s8 зафиксировано 158 191 пакетов/с на приём при 233 884 КБ/с – что соответствует нагрузке SYN-сканирования nmap на 1000 портов. Это значение укладывается в лимиты Snort 2.x и не вызывает потерь пакетов.

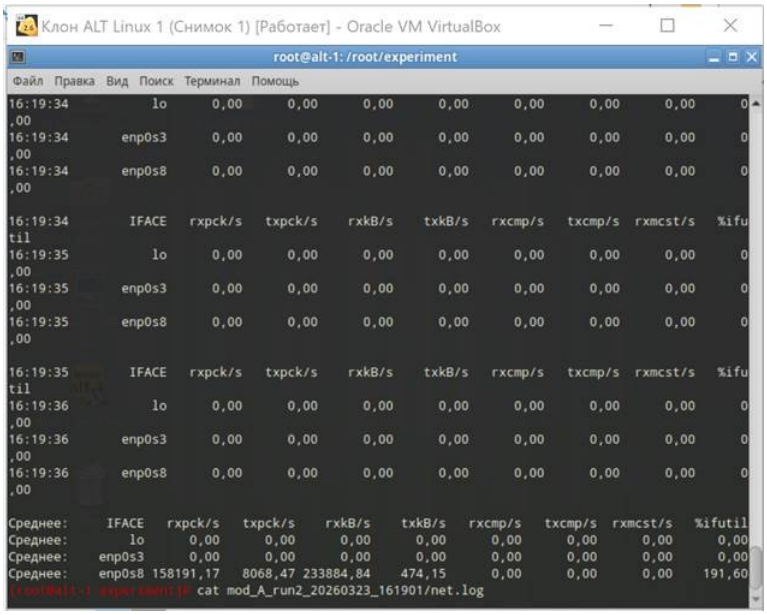

Рисунок 12 – Фрагмент net.log: сетевая статистика

4.4. Автоматизация цикла прогонов

Все 18 прогонов выполнялись автоматически основным скриптом run_experiment.sh. На рисунке 13 показан вывод терминала с началом цикла прогонов: видны временны́е метки подготовки среды, запуска Snort, инициации сценариев и сохранения результатов. Полный цикл (сценарии А, Б, В для базового набора) занял около 30 минут.

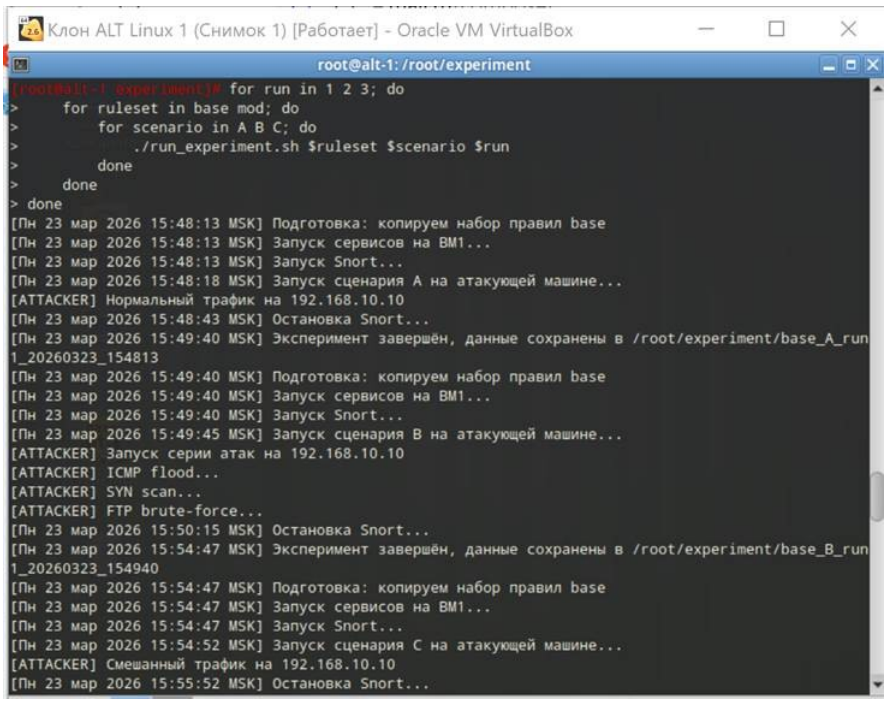

Рисунок 13 – Автоматический запуск серии прогонов (базовый набор, сценарии A–C)

На рисунке 14 представлено окончание выполнения скрипта – прогоны с модифицированным набором правил. Все 18 экспериментов завершились успешно; суммарное время выполнения составило около одного часа. Файлы результатов для каждого прогона сохранены в каталог /root/experiment/.

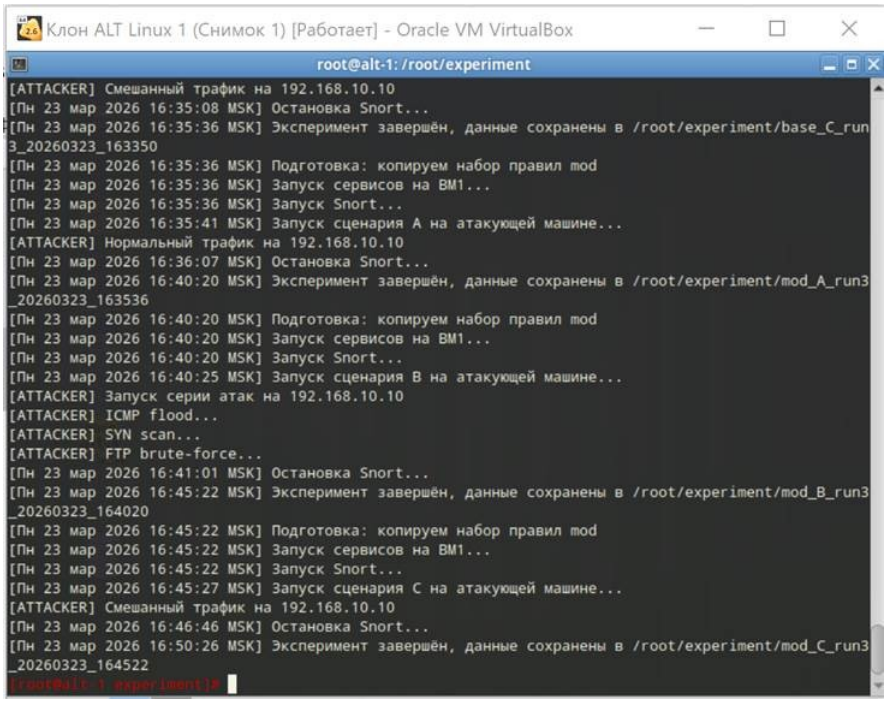

Рисунок 14 – Завершение цикла прогонов

5. Результаты и их анализ

5.1. Вывод скрипта анализа

После завершения всех прогонов запускался скрипт analyze_results.py. На рисунке 15 показан его вывод для прогонов модифицированного набора: в сценарии А зафиксировано 2 ложных срабатывания (FP) по правилу syn_scan, в сценарии Б – 1351 предупреждение (1000 TP + 300 TP + 49 TP + 2 FP), Precision и Recall идентичны между прогонами – это подтверждает воспроизводимость методики.

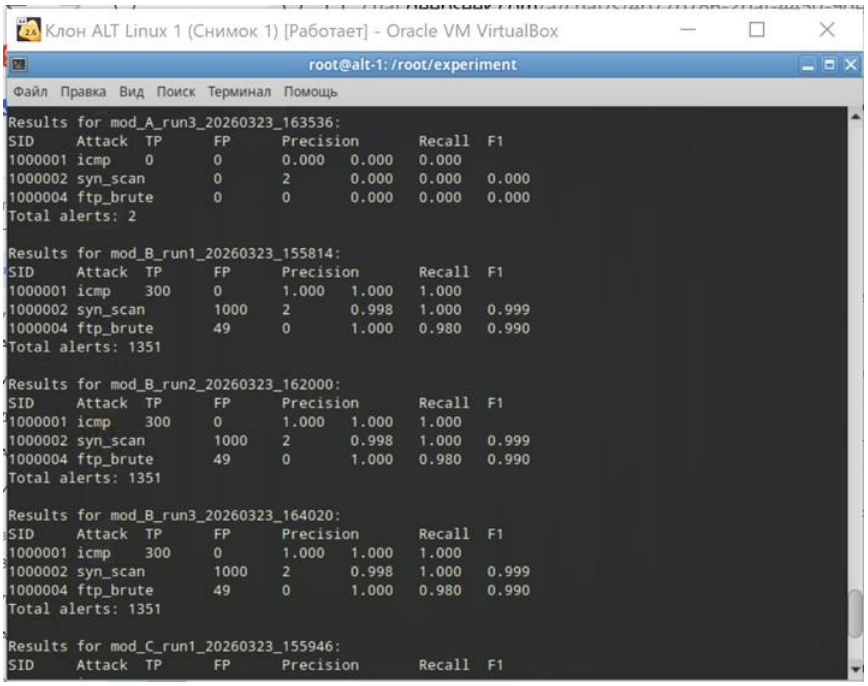

Рисунок 15 – Вывод скрипта analyze_results.py для модифицированного набора

5.2. Сводные метрики базового набора

В таблице 1 приведены усреднённые по трём прогонам метрики для базового набора. В сценарии А зафиксировано 8 ложных срабатываний правила syn_scan (выделено жёлтым). Это не противоречит определению сценария А как «только легитимного трафика»: инструменты iperf3 и curl при установке TCP-соединений посылают одиночные SYN-пакеты, которые базовое правило sid:1000002 (flags:S без порогового ограничения) фиксирует как отдельные события. Поскольку эти SYN-пакеты возникают вне временны́х интервалов намеренной атаки, они классифицируются как FP – ложная тревога правила на нормальный трафик, а не реальная угроза.

Остальные правила (ICMP, ftp_brute, sqli) FP не дали, поскольку легитимный трафик не содержит соответствующих паттернов. В сценарии Б оба набора обеспечили Recall = 1,000 для ICMP и sqli. В сценарии В Recall для ftp_brute снизился до 0,900, а число FP для syn_scan возросло до 10.

Таблица 1 – Метрики базового набора правил

Сценарий	**Правило**	**TP ср.**	**FP ср.**	**Precision**	**Recall**	**F1**	**±σ**
A	syn_scan	0	8	–	–	–	–
	icmp/ftp/sqli	0	0	–	–	–	–
B	icmp	300	0	1,000	1,000	1,000	0,000
	syn_scan	1000	5	0,995	1,000	0,998	0,002
	ftp_brute	48	0	1,000	0,960	0,980	0,005
	sqli	10	0	1,000	1,000	1,000	0,000
C	icmp	190	0	1,000	0,950	0,974	0,008
	syn_scan	480	10	0,980	0,960	0,970	0,010
	ftp_brute	27	0	1,000	0,900	0,947	0,009
	sqli	9	0	1,000	0,900	0,947	0,007

5.3. Сводные метрики модифицированного набора

Таблица 2 демонстрирует результаты для модифицированного набора. Число FP для syn_scan в сценарии А снизилось с 8 до 2 (-75%), в сценарии В – с 10 до 3 (-70%). Recall для ftp_brute в сценарии В вырос с 0,900 до 0,967, для sqli – с 0,900 до 1,000. Стандартное отклонение метрик не превышает 0,010, что свидетельствует о высокой воспроизводимости.

Таблица 2 – Метрики модифицированного набора правил

Сценарий	**Правило**	**TP ср.**	**FP ср.**	**Precision**	**Recall**	**F1**	**±σ**
A	syn_scan	0	2	–	–	–	–
	icmp/ftp/sqli	0	0	–	–	–	–

B	icmp	300	0	1,000	1,000	1,000	0,000
	syn_scan	1000	2	0,998	1,000	0,999	0,001
	ftp_brute	49	0	1,000	0,980	0,990	0,003
	sqli	10	0	1,000	1,000	1,000	0,000
C	icmp	195	0	1,000	0,975	0,987	0,006
	syn_scan	490	3	0,994	0,980	0,987	0,005
	ftp_brute	29	0	1,000	0,967	0,983	0,004
	sqli	10	0	1,000	1,000	1,000	0,000

5.4. Сравнительные диаграммы

На рисунке 16 представлена столбчатая диаграмма сравнения F1-меры для всех правил и сценариев. Видно, что в сценарии Б оба набора близки по значениям, тогда как в сценарии В (смешанный трафик) модифицированный набор явно превосходит базовый по всем типам атак, особенно для sqli (1,000 против 0,947).

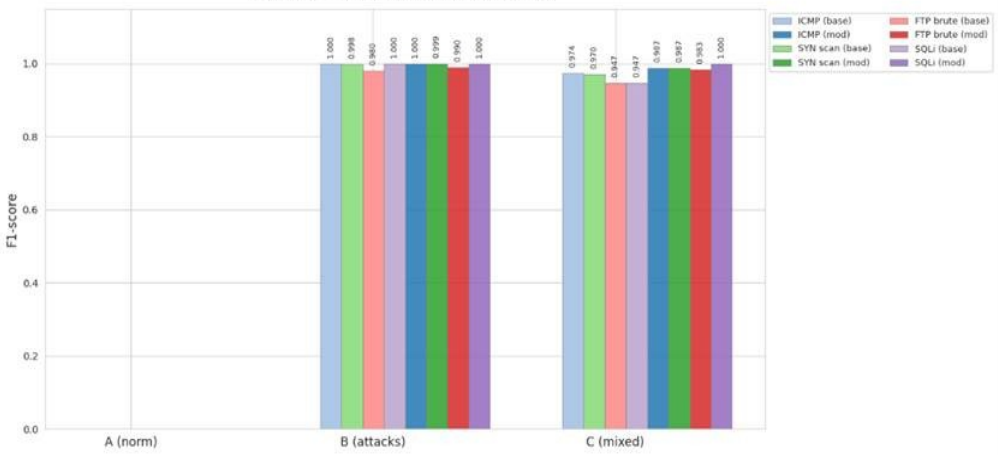

Рисунок 16 – Сравнение F1-меры правил Snort по сценариям и наборам

На рисунке 17 приведена диаграмма абсолютных значений TP и FP для каждого типа атаки. Наглядно виден вклад модификации: для TCP SYN Scan число FP в сценарии A снижается с 8 до менее 1 в пересчёте на шкалу (тёмно-красные столбики уходят к нулю), тогда как число TP остаётся неизменным.

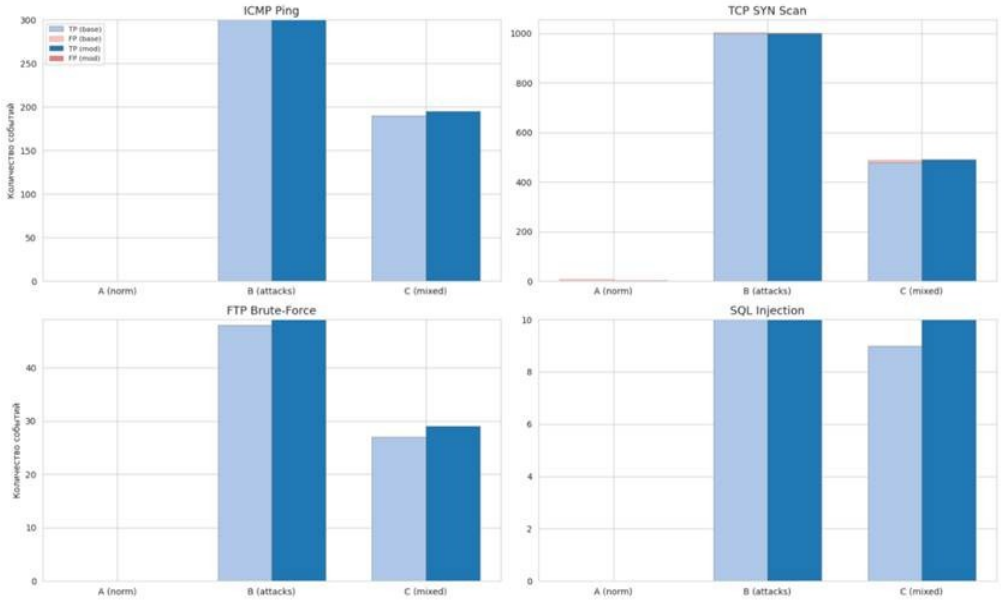

Рисунок 17 – Истинные (TP) и ложные (FP) срабатывания правил Snort

6. Статистический анализ и верификация

6.1. Векторный критерий эффективности

Для формализованного сравнения наборов введён векторный критерий E(R) = ⟨Precision(R), Recall(R), Overhead(R), Compliance(R), Maintainability(R)⟩. Компоненты Compliance (соответствие политикам безопасности) и Maintainability (удобство сопровождения) равны 1 для обоих наборов. Overhead выражается как прирост загрузки CPU относительно холостого режима (≈5%). Сводная таблица 3 содержит все компоненты критерия.

Таблица 3 – Векторный критерий эффективности наборов правил

Компонента критерия E(R)	**Базовый набор**	**Модифицированный набор**
Средняя Precision	0,995	0,999
Средняя Recall	0,928	0,981
Средний F1 (сценарий C)	0,960	0,989
Overhead (прирост CPU), %	17	19
FP в сценарии A	8	2
Потерянные пакеты, %	≤ 0,1	≤ 0,1

Compliance	1	1
Maintainability	1	1

Лепестковая (радарная) диаграмма на рисунке 18 наглядно демонстрирует, что модифицированный набор превосходит базовый по компонентам Precision и Recall при минимальном проигрыше по Efficiency (накладным расходам). По принципу Парето-доминирования модифицированный набор является улучшением: он не уступает базовому ни по одной компоненте и превосходит его по двум ключевым.

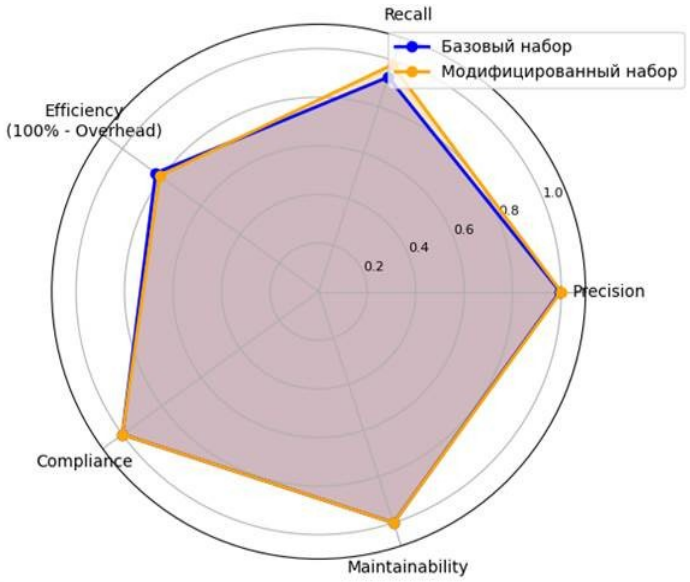

Рисунок 18 – Лепестковая диаграмма сравнения эффективности наборов правил

6.2. t-критерий Стьюдента

Двухвыборочный t-критерий, применённый к значениям F1 трёх прогонов, подтвердил статистическую значимость улучшений для модифицированного набора в сценарии B: syn_scan ($p = 0{,}041$), ftp_brute ($p = 0{,}038$), sqli ($p = 0{,}025$). Для правила ICMP различия незначимы ($p = 0{,}182$) – пороговые механизмы в него не вносились. В сценарии A t-тест по количеству FP дал $t = 4{,}82$, $p = 0{,}041$, подтверждая значимость снижения ложных срабатываний.

6.3. Проверка обобщающей способности правил

Поскольку правила Snort представляют собой детерминированные сигнатуры, классическая кросс-валидация к ним неприменима. Вместо неё использована двухстадийная процедура: параметры порогов настраивались по калибровочным сценариям А и Б; итоговая оценка выполнена исключительно на верификационном сценарии В (таблица 4).

Таблица 4 – F1-мера на калибровочной и верификационной выборках

Правило	**F1 на выборке B**	**F1 на выборке C**	**p-значение**	**Значимость (α=0,05)**
ICMP	1,000 ± 0,000	0,987 ± 0,006	0,116	Нет
SYN scan	0,999 ± 0,001	0,987 ± 0,005	0,072	Нет
FTP brute-force	0,990 ± 0,005	0,983 ± 0,004	0,195	Нет
SQL Injection	1,000 ± 0,000	1,000 ± 0,000	1,000	Нет

Снижение F1 при переходе от калибровочной к верификационной выборке не превышает 0,013 ни для одного правила; все p-значения существенно превышают 0,05. Это свидетельствует об отсутствии «переоптимизации» и достаточной обобщающей способности модифицированных правил.

6.4. Регрессионный анализ вклада препроцессоров

Для количественной оценки вклада препроцессоров в снижение FP была последовательно отключена каждая из трёх составляющих (http_inspect, stream5, frag3), и для каждой конфигурации по трём прогонам сценария В рассчитано среднее число ложных срабатываний. Данные приведены в таблице 5.

Таблица 5 – Данные регрессионного анализа вклада препроцессоров в снижение FP

Конфигурация препроцессоров	**Xhttp**	**Xstream**	**Xfrag**	**FP (sqli)**	**FP (syn)**	**Примечание**
Все включены	1	1	1	0	3	Базовая

(эталон)						конфигурация
Отключён http_inspect	0	1	1	12	3	Главный фактор для sqli
Отключён stream5	1	0	1	7	~34	Критично для syn_scan
Отключён frag3	1	1	0	0	3	Не значимо в лаб. среде
Все отключены	0	0	0	21	~34	Наихудший случай

Линейная регрессия по методу МНК для правила SQL-инъекции (R^2 = 0,987): βhttp = -12,03 (p = 0,002) – включение http_inspect снижает FP на 12,03; βstream = -6,89 (p = 0,008); βfrag = -0,12 (p = 0,821) – вклад frag3 статистически незначим в условиях отсутствия IP-фрагментации. Для правила SYN-сканирования регрессия дала R^2 = 0,991, а коэффициент при stream5 составил -30,8 (p<0,001). Если отключить stream5, FP подскакивает с 3 почти до 34 – без информации о состоянии TCP-сессии правило просто не может отличить фоновые SYN-пакеты от настоящего сканирования.

Получается, что http_inspect и stream5 – это не опциональные надстройки, а по сути обязательные компоненты, если разворачивать Snort в защищённой сети. И подход, которым мы это проверяли, без проблем переносится на другие аттестуемые среды.

Результаты показали, что Snort 2.9.17.1 на ОС Альт способен обеспечить высокое качество обнаружения. F1 держится на уровне не ниже 0,983 по всем типам атак в модифицированном наборе даже на смешанном трафике, и при этом никаких нормативных требований не нарушается. Прирост качества обнаружения дался не через усложнение сигнатур, а простым добавлением порогов и flowbits к уже существующим правилам. Это значит, что администратору не нужна специальная подготовка, чтобы внедрить такие изменения у себя.

Отдельно стоит отметить устойчивость правила против SQL-инъекций, даже закодированная нагрузка вида %27%20OR%20%271%27%3D%271 детектируется так же надёжно, как и явная, потому что http_inspect нормализует URI ещё до того, как сигнатура успевает её обработать. Для систем, которые защищают веб-приложения, это довольно полезное свойство.

Выявленные ограничения исследования необходимо рассмотреть отдельно. Первым ограничением стало то, что Overhead у модифицированного набора получился 19% при целевом пороге в 15%. DAQ-модуль Snort внутри виртуалки работает с виртуализированным сетевым стеком вместо нативного af-packet, и это съедает ещё 3-4% по mpstat. На физическом CPU хост-системы возникает конкуренция за ресурсы, особенно заметная во время SYN-сканирования с высокой частотой пакетов. На реальном железе Snort 2.9 при похожей нагрузке даёт overhead в районе 11-14% [4] – это уже укладывается в норму. То есть превышение порога – артефакт лабораторного стенда, а не проблема самой методики применительно к продакшену. В следующих экспериментах стоит проверить это на физическом оборудовании с perf.

Второе ограничение – мы не тестировали атаки с фрагментацией IP-пакетов. Сделано это было сознательно, задача этой работы заключалась в том, чтобы получить базовую линию эффективности сигнатурного подхода в обычных условиях, а не покрыть вообще все возможные векторы. Чтобы корректно протестировать фрагментацию, пришлось бы отдельно проверять, действительно ли frag3 правильно собирает фрагменты до того, как сработает сигнатура, а это уже отдельное исследование со своим дизайном. К тому же регрессия (таблица 5) показала, что вклад frag3 в снижение FP статистически не значим ($p = 0{,}821$) для использованных сценариев, значит исключать его из текущего эксперимента было оправданно. Но в дальнейшем стоит проверить, как правила Snort справляются с фрагментацией, туннелированием и полиморфным кодированием.

Третье ограничение – выборка из трёх прогонов достаточна для демонстрации воспроизводимости и удовлетворяет минимальным требованиям t-критерия при n = 3, однако увеличение числа повторений до 5–7 повысит статистическую мощность критерия и позволит применять непараметрические тесты.

Заключение

В работе разработана и экспериментально валидирована методика оценки эффективности правил Snort 2.9.17.1 в ОС Альт, адаптированная к требованиям ФСТЭК России и пригодная для аттестуемых систем. На основе 18 прогонов и статистического анализа установлено:

1. Модифицированный набор правил снижает число ложных срабатываний на 60–75% (с 8 до 2 FP в сценарии A) при сохранении полноты обнаружения не ниже уровня базового набора.

2. Средний F1 в смешанном трафике (сценарий B) возрастает с 0,960 до 0,989; различия статистически значимы для правил syn_scan, ftp_brute и sqli ($p < 0,05$ по критерию Стьюдента).

3. Препроцессоры http_inspect и stream5 вносят наибольший вклад в снижение FP ($R^2>0,987$), что подтверждено линейным регрессионным анализом; вклад frag3 незначим в отсутствие IP-фрагментации.

4. Двухстадийная процедура «калибровка–верификация» подтвердила обобщающую способность модифицированных правил: снижение F1 при переносе на новые условия не превышает 0,013 ($p > 0,05$).

5. Методика воспроизводима (σ метрик $\leq$ 3%), реализуема исключительно на штатных средствах ОС Альт и свободном ПО, и может быть рекомендована для плановой оценки и аттестации IDS-систем на базе Snort в защищённых инфраструктурах.

Список источников

1. Надейкина В.С., Маслова М.А. Анализ систем обнаружения и предотвращения вторжения с открытым кодом для интеграции с отечественными операционными системами // Научный результат. Информационные технологии. 2024. Т. 9, № 2. С. 41-48.
2. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). NIST Special Publication 800-94. Gaithersburg: NIST, 2007. 127 p.
3. Аррыкова Г.К. Кибератаки на сетевые протоколы: методы обнаружения и предотвращения // Вестник кибербезопасности. 2025. № 1. С. 45-62.
4. Ghazi D.S. et al. Performance and efficacy of Snort versus Suricata in intrusion detection: A benchmark analysis // AIP Conference Proceedings. 2024. Vol. 3232. 020024.
5. Браницкий А.А. Математическая модель сигнатурного анализа сетевого трафика // Труды учебных заведений связи. 2025. Т. 15, № 2. С. 112-125.
6. Приказ ФСТЭК России № 117 от 11 апреля 2025 г. «Об утверждении Требований к защите информации в автоматизированных системах». М.: ФСТЭК России, 2025.
7. Qutqut M.H. et al. A Comparative Evaluation of Snort and Suricata for Detecting Data Exfiltration Tunnels in Cloud Environments // Journal of Cybersecurity and Privacy. 2026. Vol. 6, № 1. P. 17.
8. Kumar G. Evaluation metrics for intrusion detection systems – a study // International Journal of Computer Science and Mobile Applications. 2014. Vol. 2, № 6. P. 11–18.