Frisbee WAF

Многоуровневая защита веб-приложений

Руководство по установке и эксплуатации

Сентябрь 2026
Версия 1.8.0

Все авторские права на эксплуатационную документацию защищены.

Документ входит в комплект поставки программного обеспечения. Без специального разрешения этот документ или его часть в печатном или электронном виде не могут быть подвергнуты копированию или передаче третьим лицам, или иному использованию с коммерческой целью.

Информация, содержащаяся в этом документе, может быть изменена разработчиком без специального уведомления, что не является нарушением обязательств по отношению к пользователю.

Содержание

Сокращения и обозначения **6**
1 Общие сведения **7**
1.1 Поддерживаемые типы угроз 7
1.2 Принципы функционирования 7
1.3 Структура и составные модули 7
1.4 Ролевая модель 8
2 Установка Frisbee WAF **9**
2.1 Минимальные системные требования 9
2.1.1 Рекомендации по масштабированию 10
2.1.2 Рекомендации по контролю состояния системы 10
2.2 Поддерживаемые режимы работы 11
2.3 Порядок установки и развертывания 11
3 Настройка правил безопасности **18**
3.1 Использование правил 18
3.1.1 Структура и разработка правил 18
3.1.2 Примеры правил безопасности 20
3.1.3 Простые правила и операторы 22
3.1.4 Работа с переменными 22
3.1.5 Объединение правил в цепочки 23
3.1.6 Оператор отрицания 23
3.1.7 Подсчет переменных 24
3.1.8 Использование действий 24
3.1.9 Понятие действий по умолчанию 24
3.1.10 Действия в связанных правилах 26
3.1.11 Использование функций преобразования 26
3.1.12 Блокировка 27
3.1.13 Изменение потока правил 27
3.1.14 Умный пропуск 28
3.1.15 Условные переходы 29
3.1.16 Управление ведением логов 29
3.1.17 Захват данных 30
3.1.18 Манипулирование переменными 31
3.1.19 Расширение переменной 31
3.1.20 Запись данных в оповещениях 32
3.1.21 Добавление метаданных 33
3.2 Правила безопасности 34
3.2.1 Общие правила безопасности (по умолчанию) 34
3.2.1.1 Просмотр правила 35
3.2.2 Кастомные правила безопасности (созданные пользователем) 35
3.2.2.1 Просмотр кастомных правил безопасности 36
3.2.2.2 Создание кастомных правил безопасности 37

3.2.2.3 Редактирование кастомных правил безопасности 37
3.2.2.4 Удаление кастомных правил безопасности 38
4 Настройка фильтрации данных на канальном уровне 39
4.1 Таблицы 39
4.2 Опции 39
4.3 Команды 39
4.4 Параметры 40
4.5 Дополнительные опции 41
4.6 Примеры использования 41
5 Настройка динамической маршрутизации 43
5.1 Настройка протокола OSPF 43
5.2 Общие настройки 44
5.3 Настройки зон 46
5.4 Настройка интерфейсов 48
5.5 OSPF route-map 50
5.6 Graceful Restart 51
5.7 Отображение информации 51
5.8 Суммирование внешних маршрутов 52
5.9 TI-LFA 53
5.10 Отладка OSPF 53
5.11 Примеры настройки 54
6 Настройка ротации логов 55
7 Резервная копия и восстановление 56
7.1 Создание резервной копии 56
7.2 Восстановление 56
8 Отправка событий аудита в SIEM 57
8.1 Категории событий 57
8.2 Принцип работы 57
8.3 Настройка передачи событий на ноде 57
8.4 Резервное копирование 58
9 Руководство по правилам IDS/IPS 59
9.1 IDS/IPS-система 59
9.2 Управление правилами IDS/IPS-системы 59
9.3 Разработка правил 62
9.3.1 Формат правил 64
9.3.2 Мета ключевые слова 69
9.3.3 Ключевые слова IP 72
9.3.4 Ключевые слова TCP 76
9.3.5 Ключевые слова UDP 79
9.3.6 Ключевые слова ICMP 79
9.3.7 Ключевые слова полезной нагрузки 83
9.3.8 Трансформации 92
9.3.9 Ключевые слова для предварительной фильтрации 94

9.3.10 Ключевые слова потока 98
9.3.11 Ключевое слово bypass 102
9.3.12 Ключевые слова HTTP 102
9.3.13 Ключевые слова операций с файлами 112
9.3.14 Ключевые слова DNS 115
9.3.15 Ключевые слова SSL/TLS 115
9.3.16 Ключевые слова SSH 119
9.3.17 Ключевые слова JA3 120
9.3.18 Ключевое слово Modbus 121
9.3.19 Ключевые слова DNP3 123
9.3.20 Ключевые слова ENIP/CIP 125
9.3.21 Ключевые слова FTP/FTP-DATA 126
9.3.22 Ключевые слова Kerberos 126
9.3.23 Ключевые слова SNMP 128
9.3.24 Ключевые слова Base64 129
9.3.25 Ключевые слова SIP 130
9.3.26 Ключевые слова уровня универсальных приложений 132
9.3.27 Ключевое слово Xbits 133
9.3.28 Пороговые ключевые слова 135
9.3.29 Ключевое слово IP Reputation 137
9.3.30 Наборы данных 137
9.3.31 Lua-скрипты 140
Дополнительная документация 142
Использование стороннего программного обеспечения 143

Сокращения и обозначения

Сокращение	Обозначение
Frisbee WAF	Программный комплекс «Многоуровневая защита веб-приложений Frisbee WAF»
GUI	Graphical User Interface, графический интерфейс пользователя
ICMP	Internet Control Message Protocol, межсетевой протокол обмена контрольными сообщениями
IDS/IPS	Intrusion Detection System / Intrusion Prevention System, системы обнаружения и предотвращения вторжений и хакерских атак
RPC	Remote Procedure Call, удаленный вызов процедур
SSO	Single Sign-On, технология единого входа в систему
WAF	Web Application Firewall, межсетевой экран веб-приложений
АРМ	Автоматизированное рабочее место
ПО	Программное обеспечение

1 Общие сведения

Программный комплекс «Многоуровневая защита веб-приложений Frisbee WAF» представляет собой межсетевой экран прикладного уровня, предназначенный для защиты веб-приложений от интернет-угроз.

Frisbee WAF обладает следующими возможностями:

- анализ трафика веб-приложений и обнаружение атак (вторжений);
- блокирование попыток сетевых атак при работе с веб-приложениями;
- защита веб-приложений от основных типов угроз;
- обнаружение подозрительной активности;
- настройка защитных механизмов Frisbee WAF под конкретное защищаемое приложение;
- контроль доступа к функциям и регистрация действий пользователей защищаемых приложений;
- интеграция с внешними системами управления событиями информационной безопасности и системами управления задачами;
- настройка балансировщика;
- возможность работы в режиме Active-Standby и Active-Active (отказоустойчивый кластер);
- использование технологии машинного обучения;
- открытый API (взаимодействие осуществляется с помощью специального токена доступа).

1.1 Поддерживаемые типы угроз

Frisbee WAF поддерживает следующие типы угроз:

- Всевозможные виды инъекций (SQL injection, OS injection, RCE, XPath-injection, XXE);
- Реализация атак на обратный путь в директориях и на включение удаленных или локальных файлов (Directory traversal, Remote/Local File Inclusion);
- Межсайтовый скриптинг (XSS);
- Межсайтовая подделка запросов (CSRF);
- Реализация переборных атак (bruteforce);
- «Умный» DoS (application-level DoS);
- Автоматизированный обход приложения, массовое скачивание информации, использование сканеров уязвимостей (web scraping, automation).

1.2 Принципы функционирования

В процессе работы Frisbee WAF опирается на модели нормального поведения защищаемого приложения, выявляя аномалии. Аномалии и другие атрибуты (пользователь, сессия, действие и др.) привязываются к HTTP-транзакциям (парам запрос–ответ).

Группы транзакций объединяются в события, которые имеют время жизни и свидетельствуют о тех или иных аномалиях в работе Frisbee WAF или веб-приложения.

1.3 Структура и составные модули

Frisbee WAF включает следующие функциональные модули:

- Анализатор;

- База знаний;
- Веб-интерфейс визуализации и управления.

1.4 Ролевая модель

В Frisbee WAF реализована мультитенантность – работа нескольких изолированных друг от друга ресурсов в рамках одного сервиса с разделением глобальных и локальных групп пользователей. Полномочия по просмотру разделов и управлению платформой определены в зависимости от типа пользователя и его группы.

В Frisbee WAF доступны три группы глобальных пользователей с предустановленными правами доступа:

- TW Администраторы (TW Admins) – полный доступ, управление другими пользователями и организациями, управление своим аккаунтом;
- TW Аудиторы (TW Auditors) – просмотр данных (запрет на редактирование), управление своим аккаунтом;
- TW Пользователи (TW Users) – полный доступ кроме управления пользователями и организациями, управление своим аккаунтом.

В Frisbee WAF доступны три группы локальных пользователей (организаций) с предустановленными правами доступа:

- Администраторы (Admins) – полный доступ настройки функций безопасности своей организации, управление своим аккаунтом;
- Аудиторы (Auditors) – просмотр данных (запрет на редактирование) в организации, управление своим аккаунтом;
- Пользователи (Users) – полный доступ настройки функций безопасности своей организации, управление своим аккаунтом.

В настоящем документе рассмотрены вопросы, связанные с установкой и эксплуатацией комплекса Frisbee WAF.

2 Установка Frisbee WAF

2.1 Минимальные системные требования

Минимальные системные требования для работы Frisbee WAF приведены в таблице 1.

Таблица 1 – Минимальные системные требования для работы Frisbee WAF

Компонент	Минимальные требования
Объем оперативной памяти	Не менее 16 ГБ
Процессор	8-ядерный и более с архитектурой x86-64, поддержкой инструкций SSE 4.2 и AVX, тактовая частота не ниже 2,2 ГГц
Объем жесткого диска	Не менее 500 Гбайт
Сетевые интерфейсы	Не менее 2 x 1 Гбит Ethernet

В Frisbee WAF поддерживаются два режима кластеризации:

- Active-Standby — один узел активен, второй в режиме ожидания.
- Active-Active — оба узла обрабатывают нагрузку одновременно.

Для настройки кластера Frisbee WAF необходимо два сервера с системными требованиями, приведенными в таблице 2.

Таблица 2 – Минимальные системные требования для каждого сервера Frisbee WAF

Компонент	Минимальные требования
Объем оперативной памяти	Не менее 16 ГБ
Процессор	8-ядерный и более с архитектурой x86-64, поддержкой инструкций SSE 4.2 и AVX, тактовая частота не ниже 2,2 ГГц
Объем жесткого диска	Не менее 500 Гбайт
Сетевые интерфейсы	3 x 1 Гбит Ethernet

Платформа работает на операционной системе Debian 12/Astra Linux Special Edition 1.8.

2.1.1 Рекомендации по масштабированию

Настоящий раздел содержит рекомендации по контролю загрузки системных ресурсов и планированию масштабирования сервера Frisbee WAF.

Приведенные пороговые значения являются ориентировочными и используются для оценки текущей нагрузки, своевременного выявления дефицита вычислительных ресурсов и планирования масштабирования системы.

Достижение указанных пороговых значений не требует обязательного немедленного увеличения вычислительных ресурсов, однако является основанием для анализа текущей нагрузки, оценки динамики ее изменения и определения необходимости масштабирования.

Пороги для мониторинга:

Ресурс	Нормальная загрузка	Требует внимания	Рекомендуется расширение ресурсов
Загрузка процессора (CPU)	менее 65 %	65–75 %	более 75 % при устойчивой нагрузке
Использование оперативной памяти (без учета buff/cache)	менее 70 %	70–85 %	более 85 %
Использование дискового пространства	менее 70 %	70–80 %	более 80 %

2.1.2 Рекомендации по контролю состояния системы

Для своевременного выявления роста нагрузки рекомендуется организовать автоматический сбор системных метрик с использованием средств мониторинга (например, Zabbix или аналогичных систем).

Рекомендуется:

- выполнять сбор системных метрик с интервалом не более 5 минут;
- анализировать динамику загрузки процессора, оперативной памяти и дисковой подсистемы не реже одного раза в сутки;
- в периоды повышенной нагрузки или при проведении регламентных работ осуществлять контроль состояния системы чаще.

Рекомендации по масштабированию

При превышении рекомендуемых пороговых значений рекомендуется определить ресурс, ограничивающий производительность системы, и оценить необходимость увеличения вычислительных ресурсов.

При выборе способа масштабирования рекомендуется руководствоваться следующими рекомендациями:

- при устойчивой высокой загрузке процессора увеличить количество процессорных ядер и (или) использовать процессоры с более высокой производительностью;
- при использовании более 85 % оперативной памяти увеличить объем RAM. Недостаток оперативной памяти может привести к возникновению ситуации Out Of Memory (OOM) и завершению отдельных процессов операционной системой;
- при заполнении дискового пространства более 80 % увеличить объем используемого хранилища. Наиболее интенсивный рост объема хранилища, как правило, связан с накоплением журналов событий, системных метрик и данных ClickHouse;

• при одновременном превышении пороговых значений нескольких ресурсов рекомендуется выполнять комплексное увеличение вычислительных ресурсов сервера.

При планировании масштабирования рекомендуется учитывать динамику изменения нагрузки. Для этого следует проанализировать показатели за последние 2–4 недели либо за период, соответствующий типичной пиковой нагрузке.

После увеличения вычислительных ресурсов рекомендуется убедиться, что контролируемые показатели вернулись в диапазон нормальных значений и имеют достаточный резерв производительности с учетом прогнозируемого роста нагрузки.

Если определить требуемую конфигурацию сервера самостоятельно не представляется возможным, рекомендуется обратиться в службу технической поддержки Frisbee WAF.

2.2 Поддерживаемые режимы работы

Frisbee WAF поддерживает следующие режимы работы:

- Блокирующий режим – поддержка возможности мониторинга и блокировки транзакций;
- Прозрачный режим (DetectionOnly) – поддержка мониторинга транзакций без влияния на взаимодействие между клиентом и веб-сервером, без возможности блокировки транзакций.

2.3 Порядок установки и развертывания

Перед установкой необходимо ознакомиться с минимальными требованиями и порядком установки. Установку должен осуществлять специалист, обладающий необходимыми знаниями и навыками.

Настройка сетевых параметров и иных параметров конфигурации платформы Frisbee WAF должна выполняться штатными средствами управления платформой через веб-интерфейс Frisbee WAF (раздел Network) либо в соответствии с процедурами, предварительно согласованными с командой Frisbee WAF.

Самостоятельное редактирование файлов конфигурации операционной системы и компонентов платформы Frisbee WAF, включая файлы netplan и иные конфигурационные файлы, управление которыми осуществляется платформой, без предварительного согласования с командой Frisbee WAF не допускается.

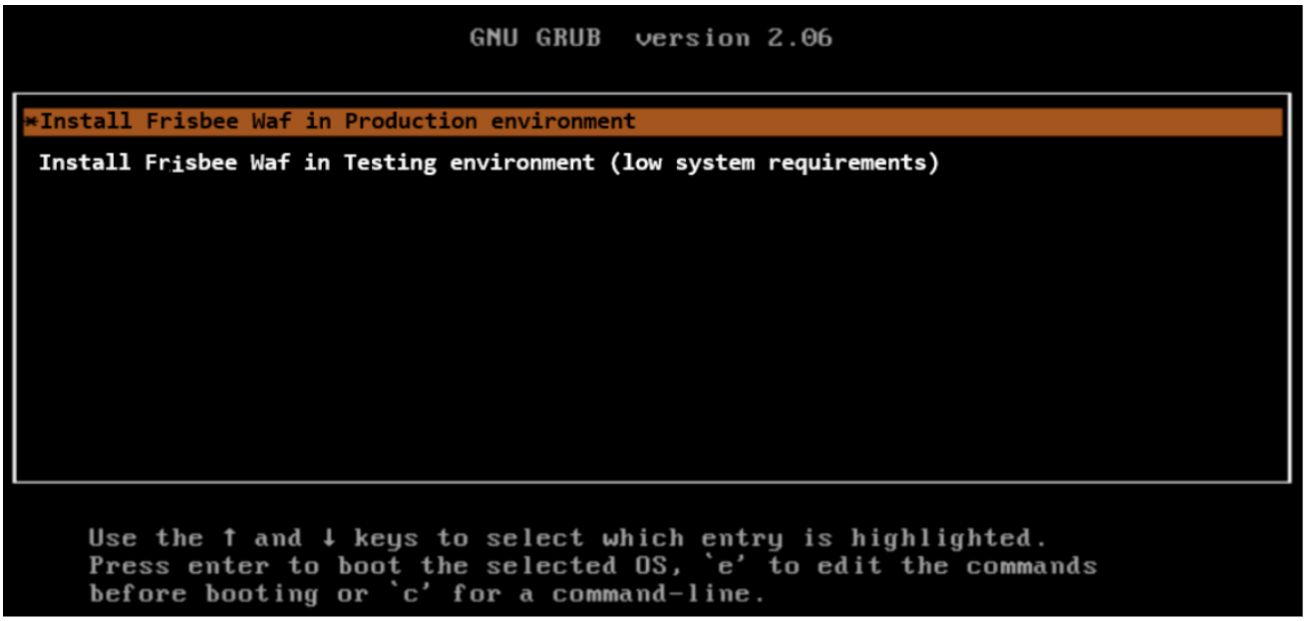
Ручное изменение конфигурационных файлов может привести к нарушению соответствия между фактическим состоянием системы и конфигурацией, управляемой платформой Frisbee WAF, а также повлечь:

- потерю сетевой доступности компонентов и сервисов;
- некорректную работу механизмов управления конфигурацией;
- возникновение ошибок при применении и сохранении настроек;
- невозможность корректного выполнения обновления платформы;
- нарушение штатного функционирования платформы.

При необходимости внесения изменений, которые невозможно выполнить средствами веб-интерфейса Frisbee WAF, необходимо предварительно согласовать порядок выполнения работ со службой технической поддержки разработчика.

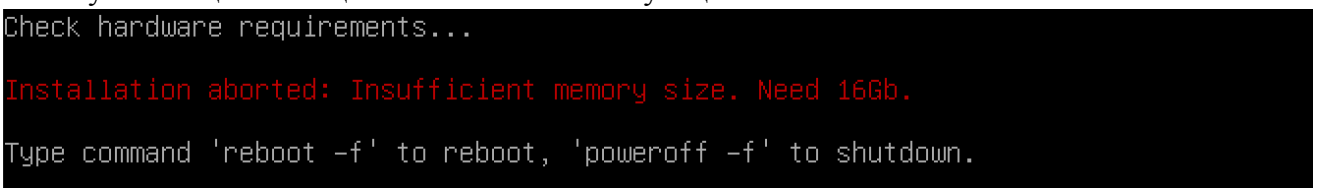
Для установки приложения необходимо:

1. Смонтировать ISO-образ в привод.
2. В открывшемся установщике выбираем версию для установки Install Frisbee WAF in Production environment.



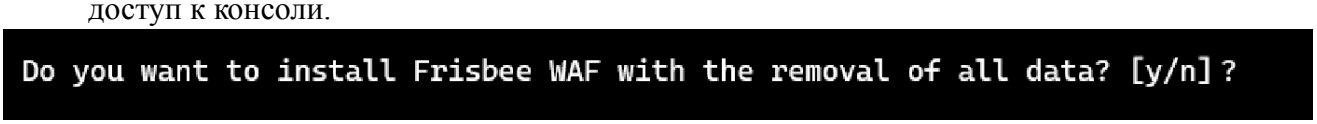
Пример выбора версии установки приложения

3. После выбора версии начинается процесс подготовки сервера к установке Frisbee WAF, в том числе проверка соответствия характеристик минимальным. При наличии ошибок установщик сообщит об этом соответствующим текстом ошибки.



Пример ошибки о наличии несоответствия минимальным характеристикам

4. Для подтверждения начала установки необходимо ввести «y». При вводе «n» открывается доступ к консоли.



Пример сообщения для подтверждения установки

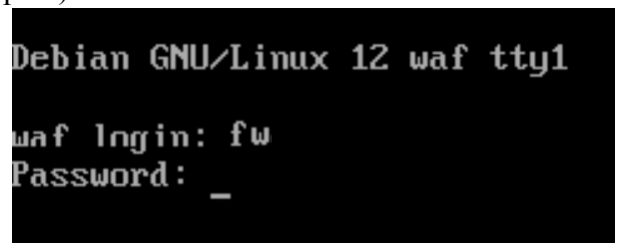
5. После ввода «y» начинается процесс копирования файлов.
6. По завершении процесса копирования появляется экран с предложением удалить установочный образ из привода и ввести «y» для перезагрузки.

```
Installation finished. No error reported.
Install installer deb to the system...
Selecting previously unselected package Frisbee Waf-installer.
(Reading database ... 46502 files and directe Waf-currently installed.)
Preparing to unpack /tmp/installer.deb ...
Unpacking Unaking Frisbee  Waf (1.0)...
Setting up Frisbee Waf-installer (1.0)
Install UEFI boot loader if applicable...
EFI variables are not supported on this system.

Frisbee  Waf initial install is finished.
Frisbee  Waf , Femove installation media and answer 'y' to reboot or 'n' to show the minimal consol
```

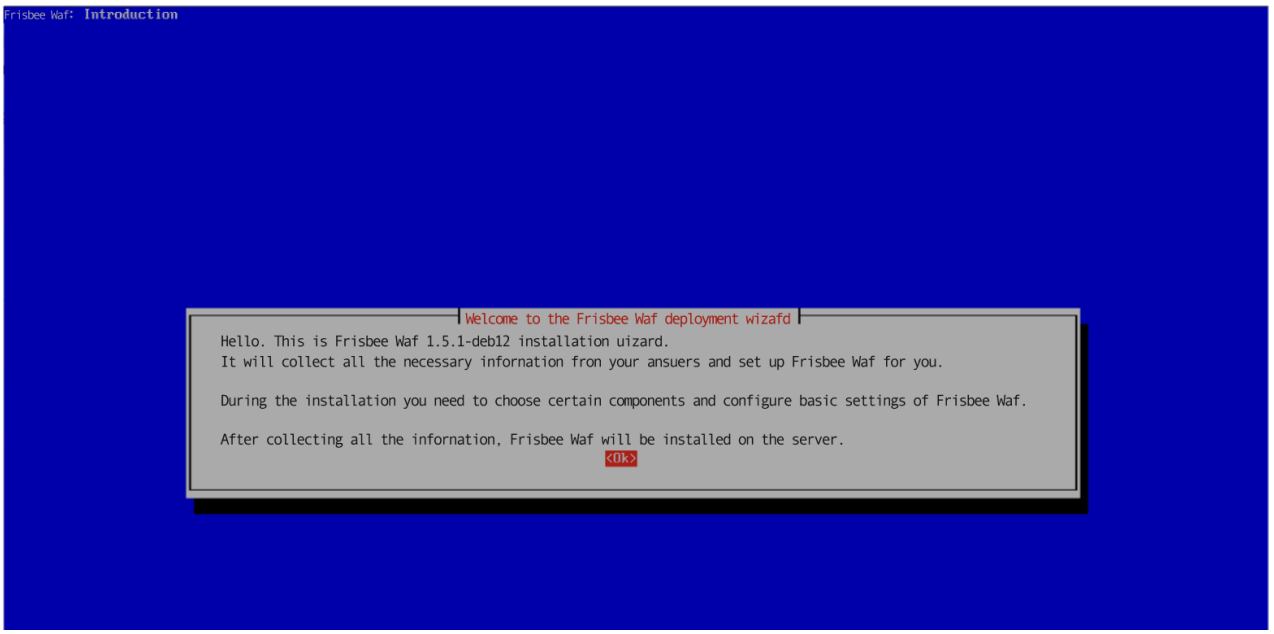
Пример сообщения об удалении образа из привода и выполнении перезагрузки

7. После выполнения перезагрузки система загружается до этапа ввода логина и пароля (по умолчанию: fw / fwpass).



Этап ввода логина и пароля после перезагрузки

8. После успешного входа в систему запускается мастер развертывания приложения (Frisbee WAF deployment wizard).



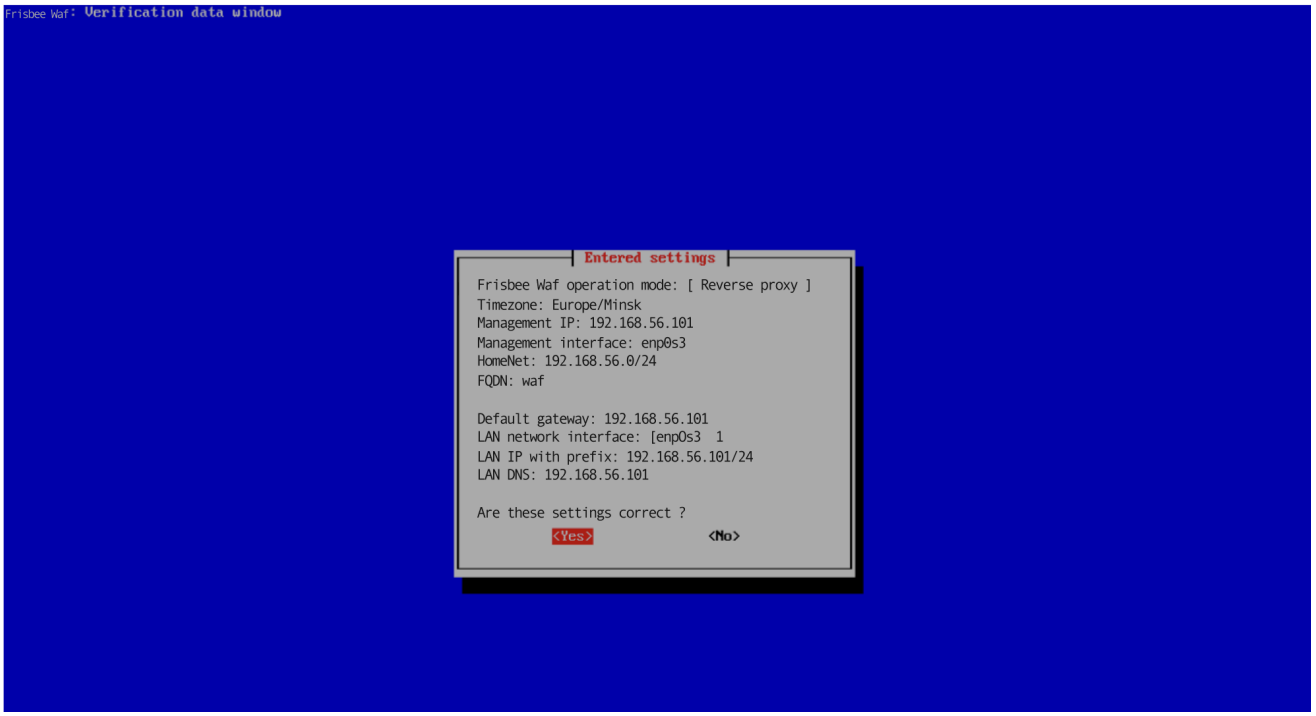
Приветственное окно мастера развертывания приложения

9. Далее с учетом подсказок мастера развертывания выбираем/вводим необходимые параметры:
   - Operation mode (Reverse proxy или Gateway);

Режим Reverse proxy — WAF перехватывает и фильтрует входящие запросы до того, как они достигают веб-сервера. Он действует как шлюз между клиентом и сервером, скрывая исходный IP-адрес и предоставляя дополнительные функции безопасности.
Режим Gateway — в этом режиме сервер Frisbee WAF устанавливается «в разрыв» сети между локальной сетью, которую требуется обезопасить, и сетью Интернет. Отличается от режима прокси-сервера «сбоку» тем, что может проверять весь проходящий между локальной сетью (LAN) и внешней сетью (WAN) сетевой трафик, а не только веб-трафик на наличие угроз. Более явно внедряется в топологию вашей корпоративной сети и требует более производительного оборудования и внимательного обслуживания. Сервер должен иметь как минимум два физических сетевых интерфейса. Один — для подключения к LAN, другой — для WAN.

- Network interface (для удобства выбора отображаются названия интерфейсов с MAC-адресами);
- Protected IP-network address;
- SIEM;
- Zabbix agent;
- Frisbee WAF Management IP;
- Domain name (FQDN);
- Time Zone.
- Примечание. Полная настройка передачи событий аудита во внешнюю SIEM выполняется вручную на каждой ноде в соответствии с разделом 8 настоящего руководства; настройка через веб-интерфейс и API не предусмотрена.

10. После выбора параметров необходимо проверить их корректность.



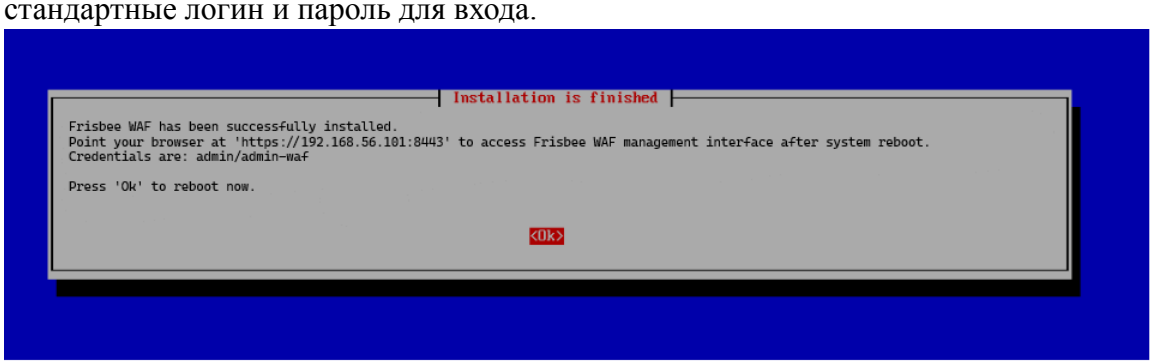
Окно проверки введенных параметров

11. После настройки необходимых параметров установка продолжается.



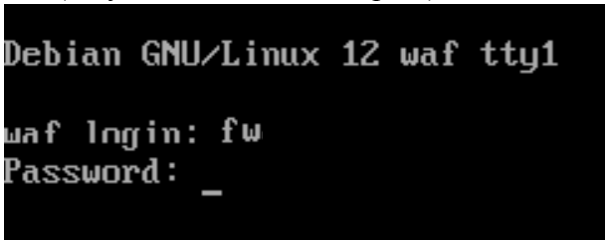
Продолжение установки приложения после выбора параметров

12. После завершения всех процессов установки появляется окно завершения процесса установки, в котором указан адрес, по которому будет доступен интерфейс Frisbee WAF и стандартные логин и пароль для входа.



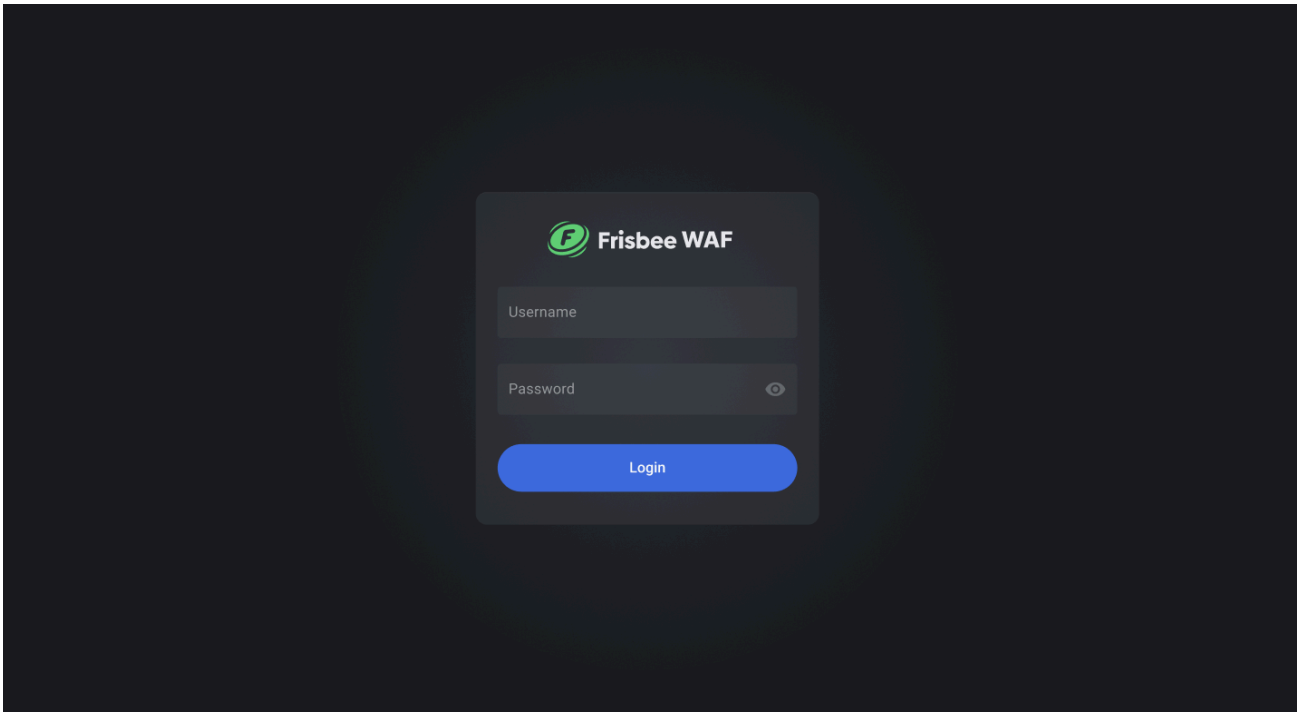
Окно завершения процесса установки приложения

13. После выбора OK система перезапустится и загрузится до этапа ввода логина и пароля операционной системы (по умолчанию: fw / fwpass).



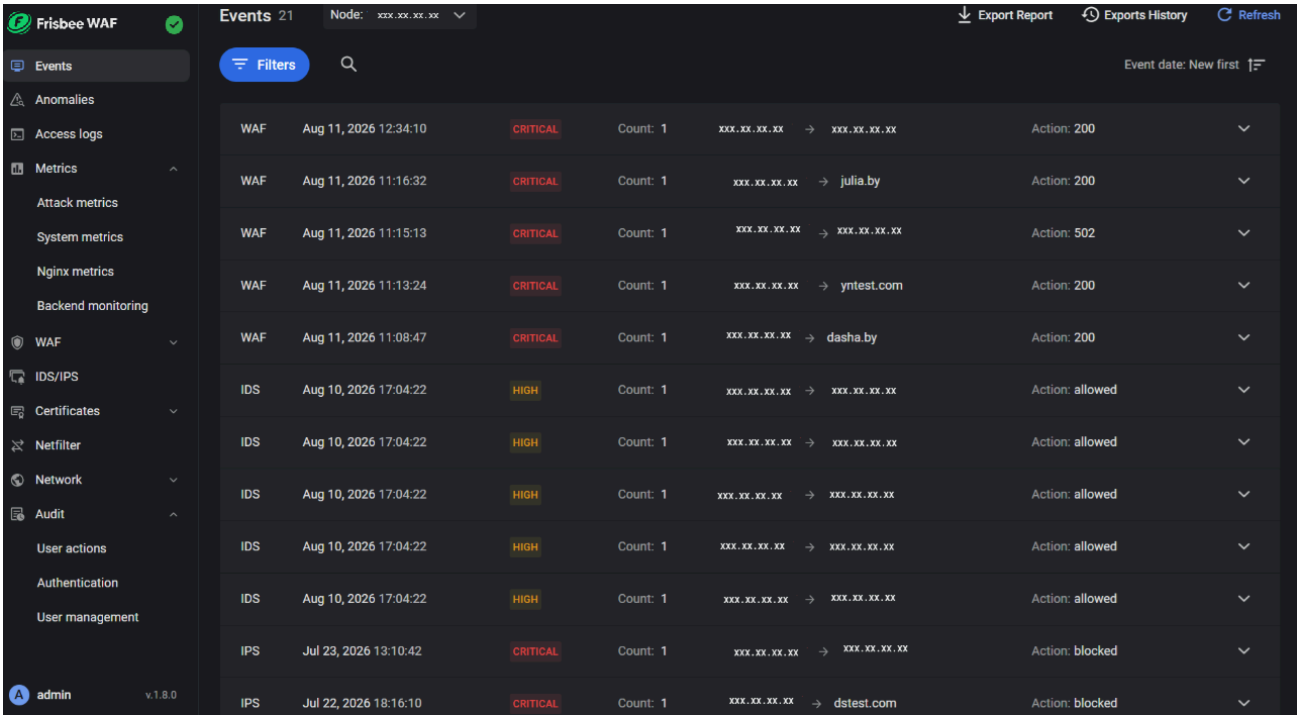
Окно консоли после перезагрузки

14. Интерфейс Frisbee WAF будет доступен по адресу, указанному в окне завершения установки.



Окно входа Frisbee WAF

15. Для авторизации в Frisbee WAF нужно воспользоваться стандартными логином и паролем: admin / admin-waf.



Интерфейс Frisbee WAF

16. В целях безопасности рекомендуется замена всех стандартных паролей после настройки Frisbee WAF.

После установки приложения порт 22 на внешних интерфейсах остается открытым. По лучшим практикам безопасности его рекомендуется закрывать на внешних сетях, за исключением сетей, используемых для администрирования Frisbee WAF.
Для его закрытия необходимо:

- показать все интерфейсы;

- закрыть порт 22 только на внешних интерфейсах. Конкретные действия в UI Frisbee WAF выполняются в разделе Settings → SSH / Netfilter (см. п. 4.3.5 Руководства администратора Frisbee WAF).

После установки рекомендуется изменить пароль администратора через UI Frisbee WAF (см. п. 4.1.2 Руководства администратора Frisbee WAF).

3 Настройка правил безопасности

3.1 Использование правил

В разделе WAF используется система защиты на основе настраиваемых правил ModSecurity, которая надежно защищает веб-приложения от возможных угроз.

ModSecurity – межсетевой экран с открытым исходным кодом, защищающий от ряда атак, направленных на веб-приложения, позволяет осуществлять мониторинг HTTP-трафика и выполнять анализ событий в реальном времени.

Поддерживается свой скриптовый язык, основанный на событиях, обеспечивающий защиту от множества видов атак на веб-приложения и позволяющий контролировать HTTP-трафик, вести логи и анализировать запросы в реальном времени, что делает межсетевой экран для защиты веб-приложений очень гибким инструментом для поиска потенциально небезопасных данных и реакции на них.

Основные возможности:

- анализ запросов – основная функция, особенно актуальная для POST-запросов, в которых получение тела запроса может быть затруднено;
- канонизация и поддержка функции антиуклонения. Выполняется преобразование входных данных в форму, подходящую для анализа, применяется для борьбы с различными методами уклонения;
- выполнение специальных встроенных проверок. Выполняются более сложные проверки правильности, такие как проверка правильности URL и Unicode-кодирования. Можно также контролировать некоторые значения байтов в запросе для борьбы с shellcode;
- запуск входных правил, позволяющих анализировать каждый аспект запроса, используя регулярные выражения. Также здесь могут быть объединены несколько правил для более сложного анализа;
- запуск правил вывода. Правила вывода применяются к телу ответа, полезны для предотвращения утечек информации;
- регистрация запроса – регистрируется окончательный запрос, состоящий из тела запроса, заголовков ввода и вывода. Чтобы предотвращать чрезмерную регистрацию, можно выборочно регистрировать запросы, например, запросы, которые получили ответ.

В следующих подразделах рассмотрены основные подходы при разработке правил. Более подробная информация о правилах доступна по ссылке: **ModSecurity_Manual**.

3.1.1 Структура и разработка правил

Каждое правило безопасности начинается с директивы SecRule. В правилах безопасности Frisbee WAF используются и другие директивы, но они применяются после SecRule. Как правило, SecRule состоит из 4 частей:

- Variables (Переменные) – инструктирует ModSecurity, где искать (может также называться целями);
- Operators (Операторы) – инструктирует ModSecurity, когда инициировать совпадение;
- Transformations (Преобразования) – инструктирует ModSecurity о том, как он должен нормализовать переменные данные;
- Actions (Действия) – инструктирует ModSecurity, что делать, если правило соответствует.

Структура правила следующая:

```
SecRule VARIABLES "OPERATOR" "TRANSFORMATIONS,ACTIONS"
```

Пример базового правила:

```
SecRule REQUEST_URI "@streq /index.php" "id:1,phase:1,t:lowercase,deny"
```

В данном примере правило принимает каждый запрос HTTP и получает только часть URI. После этого значение URI будет преобразовано в строчный регистр. Затем проверяется, соответствует ли это преобразованное значение '/index.php'. В случае совпадения Modsecurity отклонит запрос, то есть остановит обработку дальнейших правил и перехватит запрос.

Уникальность директивы SecRule заключается в том, что каждое SecRule, указанное в конфигурации, оценивается для каждой транзакции. Все остальные директивы оцениваются только при запуске.

Необходимо обратить внимание, что Frisbee WAF поддерживает и другие директивы для правил безопасности. Полный список директив и порядка их использования можно посмотреть по ссылке **ConfigurationDirectives**.

Основные принципы разработки правил:

1. Каждое правило должно иметь ПЕРЕМЕННУЮ (Variables);
2. Каждое правило должно иметь ОПЕРАТОРА (Operator), если он отсутствует, то подразумевается регулярное выражение – @rx;
3. Каждое правило должно иметь ДЕЙСТВИЕ (Action). Единственное обязательное действие – id, однако некоторые действия имплементируются SecDefaultAction (фаза по умолчанию: 2, log, auditlog, pass);
4. Каждое правило должно содержать фазу действия, которая определяет порядок запуска правила. Полный список фаз можно посмотреть по ссылке Processing**_Phases. Если фаз**а не определена, по умолчанию используется фаза: 2.
5. Каждое правило должно иметь прерывание действия. Это условие описывает, что делать с операцией, если она инициирована. Если прерывание работы не включено, по умолчанию используется значение pass;
6. ПРЕОБРАЗОВАНИЯ (Transformations) необязательны, но должны использоваться для предотвращения обхода правила.

Variables (Переменные)

Предусмотрено 105 ПЕРЕМЕННЫХ в 6 категориях. Некоторые примеры:

- Request Variables: ARGS, REQUEST_HEADERS, REQUEST_COOKIES
- Response Variables: RESPONSE_HEADERS, RESPONSE_BODY
- Server Variables: REMOTE_ADDR, AUTH_TYPE
- Time Variables: TIME, TIME_EPOCH, TIME_HOUR
- Collection Variables: TX, IP, SESSION, GEO
- Miscellaneous Variables: HIGHEST_SEVERITY, MATCHED_VAR

Полный список переменных и их синтаксис можно посмотреть по ссылке **Variables**.

Operators (Операторы)

Предусмотрено 36 ОПЕРАТОРОВ в 4 категориях. Некоторые примеры:

- String Operators – rx, pm, beginsWith, contains, endsWith, streq, within
- Numerical Operators – eq, ge, gt, le, lt
- Validation Operators – validateByteRange, validateUrlEncoding, validateSchema
- Miscellaneous Operators – rbl, geoLookup, inspectFile, verifyCC

Полный список операторов и их синтаксис можно посмотреть по ссылке **Operators**.

Transformations (Преобразования)

Предусмотрено 35 функций ПРЕОБРАЗОВАНИЯ в 6 категориях. Некоторые примеры:

- Anti-Evasion Functions – lowercase, normalisePath, removeNulls, replaceComments, compressWhitespace
- Decoding Functions – base64Decode, hexDecode, jsDecode, urlDecodeUni
- Encoding Functions – base64Encode, hexEncode
- Hashing Functions – sha1, md5

Полный список преобразований и их синтаксис можно посмотреть по ссылке **Transformation_functions**.

Actions (Действия)

Предусмотрено 47 ДЕЙСТВИЙ в 6 различных категориях. Некоторые примеры:

- Disruptive Actions – block, drop, deny, proxy
- Flow Actions – chain, skip, skipAfter
- Metadata Actions – phase, id, msg, severity, tag
- Variable Actions – capture, setvar, initcol
- Logging Actions – log, auditlog, nolog, sanitiseArg
- Miscellaneous Actions – ctl, multiMatch, exec, pause, append/prepend

Полный список действий и их синтаксис можно посмотреть по ссылке **Actions**.

3.1.2 Примеры правил безопасности

Пример:

```
SecRule REMOTE_ADDR "@ipMatch 172.20.18.133" \
    "chain, \
    id:400002, \
    phase:1, \
    t:none, \
    log, \
    deny, \
    msg:'IP Allow'"
SecRuleUpdateTargetById 131312 !REQUEST_BODY
SecRuleRemoveById 944110
```

Описание правила:

Наименование	Значение	Примечание
SecRule – директива	REMOTE_ADDR – Переменная содержит IP-адрес удаленного клиента	определяет правило
	"@ipMatch 172.20.18.133" \ – Выполняет быстрое сопоставление ipv4 и ipv6 REMOTE_ADDR переменных данных	
	"chain, \ – Связывает текущее правило с правилом, которое следует за ним, создавая цепочку правил	
	id:400002, \ – Присваивает уникальный числовой идентификатор правилу	

Наименование	Значение	Примечание
	phase:1, \ – Помещает правило в первую фазу обработки	
	t:none, \ – Предотвращает зависимость от конфигурации по умолчанию.	
	log, \ – Указывает, что необходимо зарегистрировать успешное совпадение правила	
	deny, \ – Останавливает обработку правил и перехватывает транзакцию	
	msg:'IP Allow'" – Назначает пользовательское сообщение правилу	
SecRuleUpdateTargetById – директива	131312 – id правила	обновляет целевой список (переменную) указанного правила
	!REQUEST_BODY – удалить переменную	
SecRuleRemoveById – директива	944110 – id правила	удаляет соответствующие правила из текущего контекста конфигурации

Пример:

```
SecRule REQUEST_URI "@beginsWith /admin/" \
    "id:920993,\
    phase:1,\
    block,\
    msg:'Block uri',\
    severity:'NOTICE'"
```

Описание правила:

Наименование	Значение	Примечание
SecRule – директива	REQUEST_URI – Переменная содержит полный URL-адрес запроса, включая данные строки запроса	определяет правило
	"@beginsWith /admin/" \ – Оператор возвращает значение true, если строка параметра найдена в начале ввода	
	"id:920993,\ – Присваивает уникальный числовой идентификатор правилу	

Наименование	Значение	Примечание
	phase:1,\ – Помещает правило в первую фазу обработки	
	block,\ – Выполняет действие прерывания	
	msg:'Block uri',\ – Назначает пользовательское сообщение правилу	
	severity:'NOTICE'" – Назначение серьезности правилу, в котором оно используется	

3.1.3 Простые правила и операторы

Самое простое правило указывает только переменную и регулярное выражение. В примере ниже рассматривается URI запроса и попытка сопоставления шаблона регулярного выражения <script> с ним.

```
SecRule REQUEST_URI <script>
```

В правиле можно не указывать оператор, в этом случае используется оператор регулярного выражения. При этом всегда можно явно указать оператор. Вышеупомянутое правило функционально идентично следующему:

```
SecRule REQUEST_URI "@rx <script>"
```

Обратите внимание, здесь пришлось использовать двойные кавычки, потому что второй параметр теперь содержит пробел. Разные операторы характеризуются разной производительностью. Например, шаблон регулярного выражения выше (<script>) не очень похож на шаблон. Это просто строка, которая не содержит никаких специальных символов. Аналогичного эффекта можно добиться, используя оператор @contains:

```
SecRule REQUEST_URI "@contains <script>"
```

Операторы очень просты, они берут часть транзакции и анализируют ее, обычно сравнивая с параметром, который указан в правиле (<script> в приведенных выше примерах).

3.1.4 Работа с переменными

Можно указать сколько угодно переменных в правиле, если разделять их с помощью символа вертикальной черты:

```
SecRule REQUEST_URI|REQUEST_PROTOCOL <script>
```

Некоторые переменные, называемые коллекциями, потенциально могут содержать более одной части информации. Например, это относится к переменной ARGS, которая содержит все параметры запроса в транзакции. Используйте оператор двоеточия для указания только одного члена коллекции, как показано в следующем правиле, которое рассматривает только параметр с именем p:

```
SecRule ARGS:p <script>
```

Можно использовать одну и ту же коллекцию несколько раз в одном правиле:

```
SecRule ARGS:p|ARGS:q <script>
```

Оператор двоеточие позволяет использовать регулярное выражение для указания имен, что полезно, когда имена параметров изменяются во время выполнения. Следующее правило предназначено для всех параметров, имена которых начинаются с буквы p, перехватывая такие параметры, как password или pea:

```
SecRule ARGS:/^p/ <script>
```

Если не ограничивать правило только некоторыми членами коллекции, предполагается, что будут использоваться все члены. Это очень удобно, если не известно, какие параметры использует страница. Не все коллекции могут использоваться таким образом (например, ARGS может, а ENV – нет). Когда это возможно, ссылка на такую коллекцию будет расширена до отдельных переменных непосредственно перед выполнением правила. Можно наблюдать в журнале отладки, как это работает. Например, для запроса с параметрами p, q и z, ARGS расширяется следующим образом:

```
[4] Expanded "ARGS" to "ARGS:p|ARGS:q|ARGS:z".
```

В противоположность расширению, для исключения параметров из правила, просто поставьте перед ним восклицательный знак. Следующее правило будет просматривать все параметры запроса, кроме одного с именем z:

```
SecRule ARGS|!ARGS:z <script>
```

3.1.5 Объединение правил в цепочки

Можно эффективно комбинировать более одной переменной в правиле, используя логический оператор ИЛИ. Правило будет срабатывать, если любая из переменных совпадает. Также можно использовать логическое И, в результате чего несколько правил объединяется в одно. Например, правило, которое сработает, когда что-то найдено как в параметре p, так и в параметре q:

```
SecRule ARGS:p <script> chain SecRule ARGS:q <script>
```

Это называется цепочкой правил. Действие chain создает цепочку из двух или более правил и эффективно соединяет несколько шагов в оценку.

Первое правило в цепочке всегда будет выполняться, а последующие правила будут выполняться только, если все предыдущие правила в этой цепочке выполнялись. Когда правило, принадлежащее цепочке, не совпадает, выполняется первое правило, которое не является частью данной цепочки.

3.1.6 Оператор отрицания

Результат оператора можно опровергнуть, поставив перед ним восклицательный знак. Например, если правило, срабатывающее на имени пользователя, который не является ни администратором, ни пользователем root (противоположное предыдущему примеру):

```
SecRule ARGS:username "!@rx ^(admin|root)$"
```

Отрицание оператора не следует путать с отрицанием правила. Они одинаковы только тогда, когда правило используется только для одной переменной, но ситуация меняется, когда их больше. Например, следующее правило:

```
SecRule ARGS:p|ARGS:q "!@eq 5"
```

Приведенное выше правило будет срабатывать, если какой-либо один параметр не равен 5. Чтобы написать правило, которое совпадает, когда оба параметра не равны 5, используйте цепочку правил:

```
SecRule ARGS:p "!@eq 5" chain SecRule ARGS:q "!@eq 5"
```

3.1.7 Подсчет переменных

В запросе без каких-либо параметров, ARGS расширяется до нуля переменных. Без каких-либо переменных работать с любым оператором не удастся, и правило (или цепочка) не будет совпадать.

С помощью оператора амперсанд (&) можно подсчитать количество переменных в коллекции и обнаружить случай, когда в ARGS нет параметров.

```
SecRule &ARGS "@eq 0"
```

Оператор амперсанд может применяться к любой коллекции, включая частичную. Следующее правило будет срабатывать каждый раз, когда в запросе более одного параметра username:

```
SecRule &ARGS:username "!@eq 1"
```

3.1.8 Использование действий

Практически невозможно написать правило без указания хотя бы одного действия. Кроме того, хорошей практикой является написание правил, которые являются самодостаточными и не основаны на значениях по умолчанию.

Действия помещаются в третий параметр SecRule и первый параметр SecAction. Правило может иметь ноль, одно или несколько действий. Если существует более одного действия, они разделяются запятой и любым количеством пробелов между ними. Следующее правило определяет два действия:

```
SecRule ARGS K1 log,deny
```

Некоторые действия имеют параметры, и в этом случае нужно ставить двоеточие после имени действия, далее следует параметр. Чтобы запретить со статусом 404, можно использовать:

```
SecRule ARGS K1 log,deny,status:404
```

Наконец, если требуется указать параметр, который использует пробел или содержит запятую, заключите значение в одинарные кавычки. Этот способ обработки параметров часто необходим для сообщений:

```
SecRule ARGS K1 "log,deny,msg:'Acme attack detected'"
```

Помимо использования одинарных кавычек вокруг параметра в действии msg, весь третий параметр директивы заключен в двойные кавычки. Это необходимо для правильного разбора директивной строки в Apache. Некоторые действия принимают сложные параметры (например, ctl и setvar), но данный синтаксис также применим и к ним.

3.1.9 Понятие действий по умолчанию

Поддерживается список действий по умолчанию. Всякий раз, когда новое правило добавляется в конфигурацию, список действий правила объединяется со списком действий по умолчанию. Список действий по умолчанию (в ModSecurity 2.5.11) – это phase:2,log,auditlog,pass, который можно изменить в любое время, используя директиву SecDefaultAction. В простейшем случае, когда добавляемое правило не имеет действия, вместо него используется список действий по умолчанию. Следующее правило (предполагается, что в конфигурации нет других правил или значений по умолчанию):

```
SecRule ARGS K1
```

Эквивалентно:

```
SecRule ARGS K1 phase:2,log,auditlog,pass
```

В общем случае, когда правило имеет одно или больше действий, объединение означает одно из двух:

- действие правила заменяет действие в списке действий по умолчанию – это обычно происходит с разрушительными действиями, которое может быть только одно в правиле. Если в списке действий по умолчанию и в правиле указано разрушающее действие, то действие в правиле, переопределит действие по умолчанию;
- действие правила добавляется к списку действий по умолчанию – некоторые действия могут появляться в списке действий более одного раза. Это имеет место со многими неразрушающими действиями, например, t, setvar, ctl и т. п. В некоторых случаях действия правила могут полностью удалить действия по умолчанию, в зависимости от рассматриваемого действия. Например, при действии преобразования указание none как функции преобразования (t: none) очистит список и начнет его формировать заново.

Идея SecDefaultAction состояла в том, чтобы упростить работу по написанию правил, позволяя указывать часто используемые действия только один раз. Например, можно написать что-то вроде:

```
SecDefaultAction  phase:2,log,deny,status:404 SecRule ARGS K1
SecRule ARGS K2
...
SecRule ARGS K99
```

Приведенный выше подход хорошо работает, когда конфигурация полностью контролируема, но это усложняет ситуацию, потому что правила больше не являются самодостаточными. Возможно, так проще написать конфигурацию за один день, но сложнее понять ее с течением времени. Кроме того, всегда существует опасность непредвиденного взаимодействия между значениями по умолчанию и правилом. Например, правило, основанное на определенных значениях по умолчанию, может вести себя непредсказуемо при изменении этих значений. Это особенно верно, если в список по умолчанию помещаются какие-либо функции преобразования:

```
SecDefaultAction  phase:2,log,pass,t:lowercase SecRule ARGS K1 t:urlDecode
```

> Следует всегда писать правила для указания полного списка функций преобразования, от которых они зависят. Для этого всегда указывайте t: none в качестве первой функции преобразования, которая будет сбрасывать конвейер преобразования.

Еще одна особенность директивы SecDefaultAction состоит в том, что она может использоваться более одного раза. Каждый раз, когда она используется, список действий по умолчанию меняется. Например:

```
# Сначала у нас есть несколько правил, которые только предупреждают SecDefaultAction
phase:2,log,pass
SecRule ARGS W1 SecRule ARGS W2
...
SecRule ARGS W19

# Теперь у нас есть несколько правил, которые блокируют SecDefaultAction
phase:2,log,deny,status:500 SecRule ARGS B1
...
```

```
SecRule ARGS B89
```

Суть в том, что, хотя SecDefaultAction довольно мощная и позволяет указывать любое действие, ее следует использовать только для указания метода блокировки по умолчанию.

3.1.10 Действия в связанных правилах

Особые правила применяются при размещении действий в связанных правилах. Поскольку несколько связанных правил образуют одно сложное правило, для всей цепочки может быть только одно разрушающее действие. Точно так же может быть только один набор правил метаданных. По соглашению разрушающее действие и действия с метаданными помещаются в первое правило цепочки.

```
SecRule ARGS K1 chain,id:1001,log,deny SecRule ARGS k2
```

Вышеприведенный пример выглядит достаточно невинно, но в сложных цепочках правил, когда приходится смешивать работу разрушительных и неразрушительных действий могут возникнуть проблемы, например:

```
SecRule ARGS K1 chain,id:1001,log,deny,setvar:tx.score=+1 SecRule ARGS K2 setvar:tx.score=+1
```

При чтении сложного правила следует помнить, что неразрушительные действия, связанные с правилом, всегда будут выполняться, даже если правило является частью цепочки, а разрушительные действия будут выполняться только при совпадении последнего правила в цепочке.

Последнее правило в цепочке – лучшее место для разрушительных правил и правил метаданных.

3.1.11 Использование функций преобразования

Как правило, прямое сопоставление данных происходит только в простейшем случае. В общем случае данные, обработанные правилом, будут преобразованы одной или несколькими функциями преобразования, прежде чем будут переданы оператору. Функции преобразования часто называют конвейером преобразования.

Например, правило, которое преобразует все символы в нижний регистр и затем сжимает несколько последовательных пробелов:

```
SecRule ARGS "@contains delete from" \
phase:2,t:lowercase,t:compressWhitespace,block
```

В результате правило будет соответствовать всем следующим формам ввода:

```
delete from
DELETE FROM
deLeTe fRoM
Delete From
DELETE\tFROM -- \t represents a TAB character
```

> Рекомендуется всегда начинать список функций преобразования с t: none, которое очищает конвейер преобразования, чтобы начать с нуля. Если этого не сделать, то никогда нельзя быть полностью уверенным, что пользователь не указал функцию преобразования в директиве SecDefaultAction (намеренно или по ошибке), и в этом случае правило, вероятно, будет работать неправильно. Использование t: none гарантирует, что правило всегда использует правильный набор функций преобразования.

Существует несколько причин, по которым можно захотеть применить операторы к чему-то другому, кроме исходных значений переменных:

- Ввод недоступен в требуемой форме. Например, он может быть закодирован в base64, и в этом случае нельзя ничего с ним сделать. Чтобы «открыть» данные для проверки следует применить функцию преобразования, декодирующую данные base64 (t: base64Decode);
- Может понадобиться фрагмент данных в какой-то другой форме. Например, двоичные данные, которые нужно записать в удобной для пользователя форме, закодировав их в шестнадцатеричный вид используя t: hexEncode;
- Иногда правила трудно или невозможно написать, чтобы иметь дело с вводом в его первоначальном виде. Взять, к примеру, чувствительность к регистру. Большинство операторов чувствительны к регистру, но во многих случаях регистр не имеет значения. Например, для сопоставления нетривиальной строки с помощью функции с учетом регистра, потребуется написать значение или правила (каждое из которых должно содержать различную комбинацию строчных и прописных букв), либо правило с очень некрасивым и трудным для восприятия регулярным выражением. Эту проблему можно решить путем преобразования ввода в нижний регистр перед сопоставлением;
- В большинстве случаев функции преобразования используются для противодействия уклонению. Уклонение – метод, часто используемый злоумышленниками для обхода существующего механизма обнаружения и защиты с использованием преимуществ конкретного контекста, в котором обрабатываются полезные данные, путем изменения его таким образом, чтобы избежать обнаружения, но при этом добиться поставленных целей.

3.1.12 Блокировка

Каждое правило всегда связано с одним (и только одним) разрушительным действием. Разрушительные действия прерывают процесс обработки правил в пределах фазы. Поддерживаются следующие три возможности для разрушительных действий:

- Продолжение со следующим правилом – особый случай разрушительного действия, которое не разрушает. Используйте действие pass, если хотите только предупредить о потенциальной проблеме или иметь правило, которое изменяет что-то в транзакции или постоянном состоянии (например, увеличивает счетчик);
- Остановка фазы обработки, но продолжение транзакции – действие allow используется для белого списка, что позволяет проводить транзакции без дальнейшей проверки. В зависимости от того, используется ли allow, можно пропустить только текущую фазу, фазы проверки запроса (фазы 1 и 2) и все оставшиеся фазы проверки (фаза логирования всегда выполняется);
- Остановка этапа обработки и блокировка транзакции. Блокировка – это последнее средство, которое применяется для защиты веб-приложений или отказа для нежелательных клиентов (например, червей, ботов и т. п.). Лучший способ блокировать правило – использовать действие block, которое указывает на блокировку, но не определяет, как она должна осуществляться. Еще одним преимуществом block является то, что блокировка может быть переопределена администратором правила. Если используются какие-либо другие блокирующие действия (deny, drop, redirect и proxy), то, по сути, в правилах жестко программируется политика. Это может быть хорошо для разовых правил, но их, вероятно, придется изменить в соответствии с конкретными обстоятельствами.

3.1.13 Изменение потока правил

Правила будут обрабатываться линейно, начиная с первого правила фазы и заканчивая последним. Если совпадение происходит где-то в фазе и происходит блокировка, обработка фазы

останавливается, но выполнение правил все еще линейно. Также можно сформировать группы правил и создать конструкции if-then-else, для этого используются действия, меняющие ход выполнения правил.

Первым вариантом является использование skip, принимающего один параметр и пропускающего столько правил, сколько указано. Правило не имеет никакого смысла, когда используется разрушительное действие, означающее, что будет использоваться только skip в комбинации с pass. В следующем примере демонстрируется skip:

```
SecRule ARGS K1 id:1,nolog,pass,skip:2 SecRule ARGS K2 id:2,nolog,pass SecRule ARGS K3 id:3,log,block
```

В приведенном выше примере, при соответствии правилу 1 пропускаются два следующих правила.

Следует иметь в виду следующее:

- При формировании цепочки из двух или более отдельных правил, вся цепочка считается одним правилом для пропуска;
- Можно использовать skip в цепочке, но применяются те же правила, что и для разрушающих действий: разрешен только один пропуск, и он должен быть помещен в правило запуска цепочки;
- Пропуск происходит с правилами, которые относятся к той же фазе, что и правило, инициирующее пропуск. Не следует учитывать какие-либо правила, кроме правил фазы, к которой относится правило пропуска. Чтобы избежать непонятных ситуаций, не следует смешивать фазы правил и действия skip.

Пропуск часто используется как метод оптимизации. Иногда выполнение группы правил имеет смысл только при определенных условиях, выполнение которых в противном случае является пустой тратой мощности процессора. В таких случаях, как правило, группе предшествует одно правило, которое проверяет условие и перепрыгивает через всю группу, если условие не выполняется.

Есть также несколько проблем с действием skip. Во-первых, легко ошибиться в подсчете правил, которые требуется пропустить, что также затрудняет обновления. Каждый раз, когда требуется внести изменения в правила, следует помнить, что есть правило пропуска и оценивать, нужно ли обновлять параметр пропуска. Существует также потенциально большая проблема, которая возникает при использовании SecRuleRemoveById или SecRuleRemoveByMsg для удаления пропускаемого правила. Если правил станет на одно меньше, то действие пропуска пропустит и следующее правило. Следующий пример демонстрирует эту проблему:

```
SecRule ARGS K1 id:1,nolog,pass,skip:1 SecRule ARGS K2 id:2,nolog,pass SecRule ARGS K3 id:3,nolog,pass
...
SecRuleRemoveById 2
```

Правило 1 должно пропустить правило 2 при совпадении, но поскольку правило 2 позже в конфигурации удаляется, то правило 1 вместо этого пропустит правило 3.

3.1.14 Умный пропуск

Действия skipAfter и SecLabel обеспечивают улучшенную обработку пропусков. Первый пример, переписанный для использования новых возможностей, выглядит следующим образом:

```
SecRule ARGS K1 id:1,nolog,pass,skipAfter:4 SecRule ARGS K2 id:2,nolog,pass,skipAfter:4
SecRule ARGS K3 id:3,log,block
SecLabel id:4
```

При использовании skipAfter проверяются все правила, которым нужно следовать, чтобы найти правило с указанным идентификатором. После обнаружения выполнение продолжается со

следующего правила. Это действительно означает, что не нужно всегда использовать SecLabel. Во многих случаях достаточно skipAfter. Пример выше можно переписать так:

```
SecRule ARGS K1 id:1,nolog,pass,skipAfter:3 SecRule ARGS K2 id:2,nolog,pass,skipAfter:3
SecRule ARGS K3 id:3,log,block
```

3.1.15 Условные переходы

Можно реализовать примитивную конструкцию if-then-else, если использовать skip и SecAction вместе:

```
SecRule ARGS K1 id:1,nolog,pass,skip:2 SecRule ARGS K2 id:2,block
SecAction nolog,pass,skip:1 SecRule ARGS K3 id:3,block
```

Первое правило в примере определяет, какой из двух вариантов будет обработан. Если оно совпадает, выполняется действие пропуска, чтобы перейти к правилу 3. Однако, если оно не совпадает, будет обработано следующее правило. Безусловное совпадение в SecAction, которое следует за правилом 1, гарантирует, что правило 2 не будет обработано, если правило 1 не совпадает.

Пропуск с использованием меток не облегчает чтение правил, хотя облегчает поддержку больших групп правил:

```
SecRule ARGS K1 id:1,nolog,pass,skipAfter:11 SecRule ARGS K2 id:2,block
SecAction  nolog,pass,skipAfter:12
SecLabel 11
SecRule ARGS K3 id:3,block SecLabel 12
```

3.1.16 Управление ведением логов

Существует несколько действий по ведению логов, которые может использовать правило, они делятся на две группы.

- Первая группа состоит из действий, влияющих только на то, что происходит во время обработки текущего правила; такие действия используются практически в каждом правиле;
- Действия второй группы влияют на то, как ведение логов выполняется на уровне транзакции, они обычно используются только в правилах конфигурации.

Возвращаясь к первой группе, наиболее общее использование выглядит следующим образом:

```
SecRule ARGS K1 log,auditlog,block
```

Если правило выше совпадает, действия log и auditlog сообщают движку об отправке оповещения и регистрируют транзакцию в логе аудита. Действие log на самом деле подразумевает auditlog, поэтому всегда безопасно использовать только первое. То же самое относится и к действиям, которые не требуют регистрации: nolog, противоположность log, подразумевает noauditlog, который противоположен audlog. Следует учитывать два момента:

- Предупреждение – это запись соответствия правила, которое появится в журнале отладки, в журнале ошибок Apache и в разделе H записи журнала аудита. Поскольку существует две пары действий (log и nolog, а также audlog и noauditlog), можно точно решить, что произойдет, с точки зрения ведения журнала, когда правило совпадет. Большинству правил нужны оба варианта, но можно в равной степени регистрировать совпадения только в журнале ошибок и не иметь всей записи журнала аудита (чего достигают с помощью log и noauditlog вместе);
- Когда в правиле указывается auditlog, это не означает, что журнал аудита будет создан. Следует рассматривать auditlog как запрос на запись транзакции, но правило обнаружения

обычно не имеет полного контроля над тем, что на самом деле произойдет. Транзакции делятся на релевантные и нерелевантные. Когда правило совпадает и задает auditlog (явно или неявно через log без noauditlog), то устанавливается флаг релевантности. Обычно это приводит к записи транзакции, но последующее правило может отменить это. Такое разделение интересов является намеренным. Правила должны указывать только необходимое им, но окончательное решение принимает администратор.

3.1.17 Захват данных

В коллекции TX есть 10 переменных, имена которых представляют собой цифры от 0 до 9. Эти переменные зарезервированы для захвата данных, что в первую очередь является функцией оператора @rx. Чтобы использовать эту функцию, следует:

- Использовать круглые скобки в шаблонах регулярных выражений, указывая, где должен происходить захват;
- Добавить действие capture к правилу, в котором требуется захват данных.

Например, веб-приложение помещает идентификаторы сессии в строку запроса. Для поддержки состояния сеанса необходимо извлечь информацию о сеансе и инициализировать состояние сеанса. URI, используемый в приложении и содержащий идентификатор сеанса, может выглядеть следующим образом:

```
http://www.example.com/69d032331009e7b0/index.html
```

Правило для извлечения идентификатора сеанса будет использовать захват данных регулярного выражения:

```
# Initialize session state from the session identifier in URI
SecRule REQUEST_URI ^/([0-9a-fA-f]{16})/ phase:1,nolog,pass,capture,setsid:%{TX.1}
```

Вот что происходит при успешном совпадении:

```
[4] Recipe: Invoking rule 8e8b5c8; [file "/home/ivanr/apache/conf/httpd.conf"] … [line "479"].
[5] Rule 8e8b5c8: SecRule "REQUEST_URI" "@rx ^/([0-9a-fA-f]{16})/" "phase:1,auditlog,nolog,pass,capture,setsid:%{TX.1}"
[4] Transformation completed in 2 usec.
[4] Executing operator "rx" with param "^/([0-9a-fA-f]{16})/" against REQUEST_URI. [9] Target value: "/69d032331009e7b0/index.html"
[9] Added regex subexpression to TX.0: /69d032331009e7b0/ [9] Added regex subexpression to TX.1: 69d032331009e7b0 [4] Operator completed in 63 usec.
```

Переменная TX.0 всегда будет содержать всю часть входа, который был подобран (69d032331009e7b0 в примере, обратите внимание на косые черты в начале и в конце значения). Если регулярное выражение использует якоря ^ и $, TX.0 будет содержать весь ввод. В приведенном выше примере используется только один из якорей, поэтому TX.0 содержит данные от начала ввода, но только до конца соответствующей части (второй косой черты). Переменная TX.1 будет содержать только ту часть, которая была заключена в первых скобках набора из шаблона. Переменная TX.2 будет использовать содержимое из второго набора скобок и т. п. Будет создано до 9 захватов.

> Если совпадений нет, переменные захвата данных не будут изменены. Однако, при совпадении неиспользуемые переменные захвата данных будут сброшены.
> Операторы @pm и @pmFromFile имеют ограниченную поддержку захвата данных: если задано действие захвата, переменная TX.0 будет заполнена сопоставляемыми входными данными. В шаблонах отсутствует необходимость использования круглых скобок.

3.1.18 Манипулирование переменными

Хотя большинство данных при анализе транзакций будет доступно только для чтения, существуют определенные переменные и коллекции, которые можно изменять. Коллекция TX – это личное пространство для каждой транзакции, которое могут использовать правила для совместной работы. Переменные, помещенные в TX, могут быть получены с использованием того же подхода, что и для других коллекций. Однако, действие setvar позволяет изменять значения.

Чтобы создать новую переменную, просто установите для нее значение:

```
SecAction  nolog,pass,setvar:tx.score=1
```

Чтобы удалить переменную, поместите восклицательный знак перед ее именем:

```
SecAction  nolog,pass,setvar:!tx.score
```

Цифровые значения могут быть увеличены или уменьшены. В следующем примере сначала переменная увеличивается на 2, а затем уменьшается на 1:

```
SecAction  nolog,pass,setvar:tx.score=+2 SecAction  nolog,pass,setvar:tx.score=-1
```

3.1.19 Расширение переменной

Во многих текстовых контекстах поддерживается функция, известная как расширение переменной, позволяющая помещать данные в текст, что может быть очень полезным, например, при обработке ошибок синтаксического анализа:

```
SecRule REQBODY_PROCESSOR_ERROR "!@eq 0" \
"phase:2,t:none,log,block,msg:'Failed to parse request body: …
%{REQBODY_PROCESSOR_ERROR_MSG}',severity:2"
```

Когда происходит сбой во время синтаксического анализа тела запроса, можно увидеть с чем на самом деле связана ошибка. Расширение переменной происходит всякий раз, когда встречается имя переменной, заключенное в % {...}, что является синтаксисом, принятым из mod_rewrite. Имя переменной может быть любым, и можно получить доступ к коллекциям, используя знакомый синтаксис % {COLNAME.VARNAME}.

> Разница между COLNAME: VARNAME и COLNAME.VARNAME заключается в том, что первый потенциально может возвращать более одного результата, тогда как последний всегда будет возвращать один результат (или вообще никакого результата).

Большинство частей языка правил поддерживают расширение переменных; многие функции действительно требуют этого. Например, отслеживание сеанса или IP-адреса было бы невозможно без возможности каким-либо образом обрабатывать часть данных, полученных от клиента.

В следующем примере используется расширение переменной, где одна часть запроса сравнивается с другой из того же запроса:

```
# If an absolute URI (containing hostname) was given on the request
# line, check that the same hostname is used in the Host header
SecRule REQUEST_URI_RAW "@beginsWith http" "chain,phase:2,block,msg:'Hostname …
mismatch'"
SecRule REQUEST_URI_RAW "!@beginsWith http://%{REQUEST_HEADERS.Host}"
```

Расширение переменной не поддерживается с помощью операторов регулярного выражения и параллельного сопоставления по соображениям производительности. Действия @rx и @pm делят свою работу на два этапа. Они выполняют как можно больше работы заранее, компилируя шаблоны в более эффективные внутренние представления. Затем на втором этапе

выполняется сопоставление. Компиляция шаблонов выполняется только один раз, во время конфигурации, поэтому требуется, чтобы шаблоны были статичными. С другой стороны, сопоставление происходит быстрее. Расширение переменной – это когда строковые операторы, отличные от @rx и @pm, имеют редкое преимущество.

3.1.20 Запись данных в оповещениях

Еще одно действие журнала – это logdata, цель которого – включение указанной части данных вместе с другой информацией в оповещения.

Например, следующее правило, которое ищет обработчики событий JavaScript во входных данных:

```
SecRule ARGS "\bon(abort|blur|change|click|dblclick|dragdrop|end|error|\
focus|keydown|keypress|keyup|load|mousedown|mousemove|mouseout\
mouseover|mouseup|move|readystatechange|reset|resize|select|submit|unload)\b\W*?="  \
phase:1,t:none,t:lowercase,log,deny,capture,logdata:%{TX.0}
```

В данном примере все искомые шаблоны разделяют начало, имеют разные части в середине и разделяют конец. Обратите внимание на следующие моменты:

- В оповещениях не отображаются входные данные. Таким образом, глядя на одно только оповещение нельзя определить, какая часть шаблона соответствует, и потребуется доступ ко всему журналу аудита. Даже если это возможно, то все равно будет занимать много времени;
- Даже при наличии доступа к журналу аудита отслеживание соответствующей части ввода может быть непростым. Когда данный вид правил совпадает, это обычно касается параметров запроса, которые являются довольно длинными. Таким образом, следует сначала понять, что делает правило, а затем провести ручное сопоставление с шаблоном, прочитав каждый параметр;
- Сопоставление происходит с потенциально преобразованным вводом, поэтому часто необработанный ввод не будет содержать данные в форме, используемой для сопоставления;
- Данные проблемы решаются при использовании действия logdata.

Например, следующее оповещение (приведена только часть):

```
[Fri Dec 04 17:00:01 2009] [error] [client 192.168.3.1] ModSecurity: Access denied with code
403 (phase 1). Pattern match "\\bon(abort|blur|change|click|dblclick
|dragdrop|end|error|focus|keydown|keypress|keyup|load|mousedown|mousemove
|mouseoutmouseover|mouseup|move|readystatechange|reset|resize|select
|submit|unload)\\b\\W*?=" at ARGS:p. [file "/home/ivanr/apache/conf/httpd.conf"] [line
"472"] [data "onload="] [hostname "192.168.3.100"] [uri "/"]
[unique_id "SxlAEcCoA2QAABLXHEAAAAAB"]
```

Действие capture, захвата из правила, сообщает оператору регулярного выражения (@rx) поместить всю совпадающую область в переменную TX.0. Часть правила logdata:% {TX.0} указывает движку на включение значения переменной TX.0 в оповещение.

> Основной смысл использования logdata, а не расширения переменных в msg состоит в том, что при помещении части данных в сообщение, программный анализатор не узнает об этом. Все сообщение рассматривается как просто текст. Однако, при включении данных в оповещение с logdata, парсер будет знать, что это что-то, возникшее при вводе, и сможет сделать что-то полезное с этим. Например, выделить часть данных на странице оповещения.

3.1.21 Добавление метаданных

Для повышения уровня понимания правил можно добавить больше метаданных в оповещения. Например, следующее правило:

```
SecRule REQUEST_METHOD "!^(GET|HEAD)$" \
phase:1,t:none,log,block
```

Оно ограничивает методы запроса GET или HEAD, которые подходят только для статических веб-сайтов. При совпадении правило выдаст следующее оповещение:

```
[Thu Dec 03 20:02:50 2009] [error] [client 127.0.0.1] ModSecurity: … Warning. Match of "rx ^(GET|HEAD)$" against "REQUEST_METHOD" required.… [file "/home/ivanr/apache/conf/httpd.conf"] [line "464"]…
[hostname "192.168.3.100"] [uri "/"] [unique_id "SxgbacCoA2QAABC7HMgAAAAB"]
```

По умолчанию оповещения содержат довольно много информации, но их недостаточно. Например, сообщение по умолчанию дает некоторое представление о том, как выглядит правило, но не сообщает о том, чего хотел достичь создатель правила.

Этого позволяют достичь метаданные, которые в основном используются для документирования правил и упрощения их использования. Далее представлено вышеприведенное правило с дополнительными метаданными:

```
SecRule REQUEST_METHOD "!^(GET|HEAD)$" \
"phase:1,t:none,log,block,id:1001,rev:2,\ severity:WARNING,msg:'Request method is not allowed'"
```

Добавлено четыре действия метаданных:

- Действие id назначает правилу уникальный идентификатор, что позволяет сопоставить оповещение с правилом, которое его вызвало. Добавление идентификатора также позволяет манипулировать правилом во время настройки и выполнения. Идентификаторы очень важны, они позволяют настраивать наборы правил, оставляя исходные файлы конфигурации без изменений (например, с использованием директив SecRuleUpdateById и SecRuleRemoveById), что, в свою очередь, позволяет автоматически обновлять наборы правил;
- Действие rev (сокращение от revision) – это, по сути, счетчик изменений или серийный номер, начинающийся с единицы и увеличивающийся на единицу при каждом изменении правила. Идея состоит в том, чтобы дать возможность сразу определить, изменилось ли правило, в том числе для программы (которая иначе не сможет понять различий между двумя версиями правила);
- Действие severity показывает серьезность обнаруженной проблемы, согласно системе уровней серьезности syslog – наименьшая серьезность – DEBUG (7), наибольшая – EMERGENCY (1). При этом нет четких указаний о том, как назначать серьезность правилу, это остается на усмотрение автора правила;
- Наконец, действие msg добавляет еще одно сообщение к правилу, которое должно объяснить цель правила или его результат.

Информация в действиях с метаданными используется в оповещениях. Улучшенное правило создает следующее оповещение:

```
[Thu Dec 03 20:11:25 2009] [error] [client 127.0.0.1] ModSecurity: Warning.… Match of "rx ^(GET|HEAD)$" against "REQUEST_METHOD" required.…
[file "/home/ivanr/apache/conf/httpd.conf"] [line "465"] [id "1001"] [rev "2"]… [msg "Request method is not allowed"] [severity "EMERGENCY"]…
[hostname "192.168.3.100"] [uri "/"] [unique_id "SxgZZsCoA2QAABCYL9IAAAAA"]
```

Это намного лучше, но оповещение все равно не объясняет, почему не разрешается какой-либо метод запроса, кроме GET или HEAD.

Можно добавить подробное описание функциональных возможностей правила, а также улучшить предупреждающее сообщение. Кроме того, можно также использовать действие tag для классификации правил:

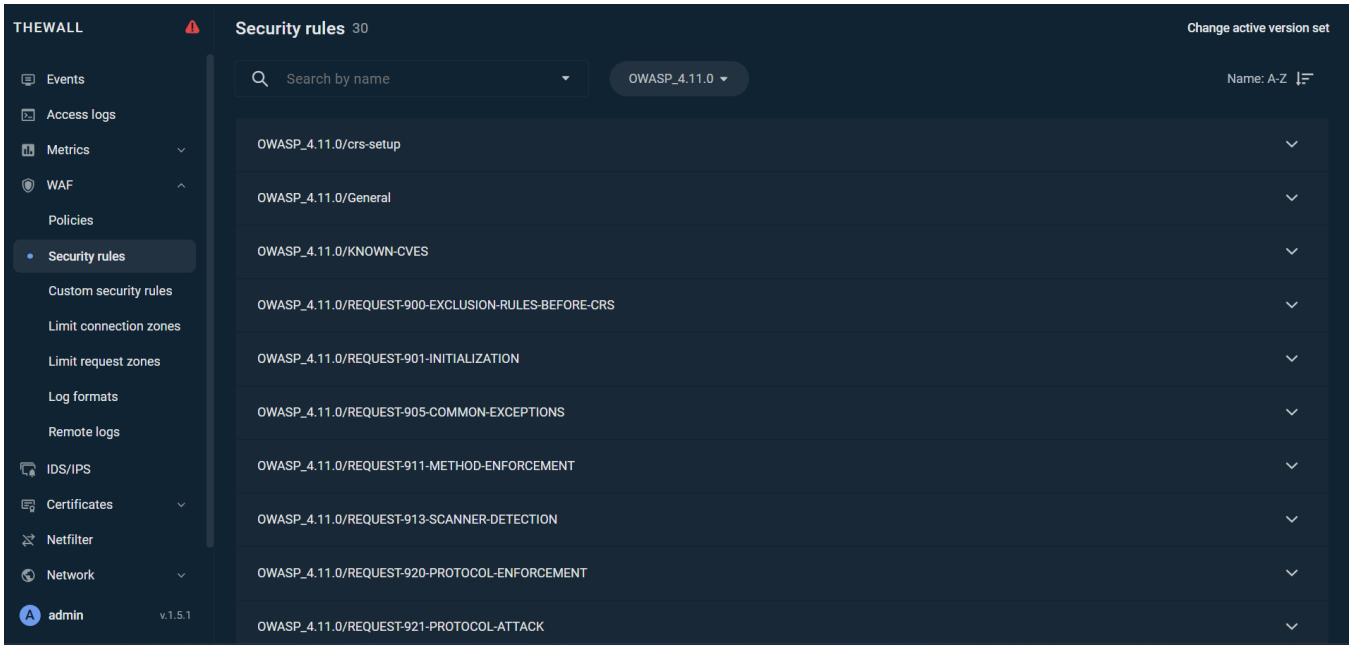
```
# Do not allow request methods other than GET or HEAD. Allow
# site does not currently use any other methods; restricting
# the methods allowed reduces the attack surface.
SecRule REQUEST_METHOD "!^(GET|HEAD)$" \
"phase:1,t:none,log,block,id:1001,rev:2,\ severity:WARNING,msg:'Request method is not allowed because \
it is not used by the application',tag:HARDENING"
```

Теги – это фрагменты текста, которые можно прикрепить к правилам. Можно прикрепить один или несколько тегов. По соглашению первый тег определяет основную категорию правила, а все остальные теги определяют вторичные категории. Знание категории правила очень полезно для понимания того, что оно делает. Категории также позволяют системам мониторинга, которые собирают оповещения, без особых усилий создавать небольшие круговые диаграммы оповещений (например, отображать, сколько оповещений каждой категории произошло за период времени). Для тегов тоже нет четких указаний, как их использовать. Основной набор правил использует их для категоризации правил, но не документирует категории (и не гарантирует, что категории не изменятся).

3.2 Правила безопасности

3.2.1 Общие правила безопасности (по умолчанию)

Список групп общих правил безопасности, созданных по умолчанию, отображается на вкладке **Security rules** в подразделе **WAF**. Перечень действий с данными группами правил ограничен, их нельзя редактировать или удалять.



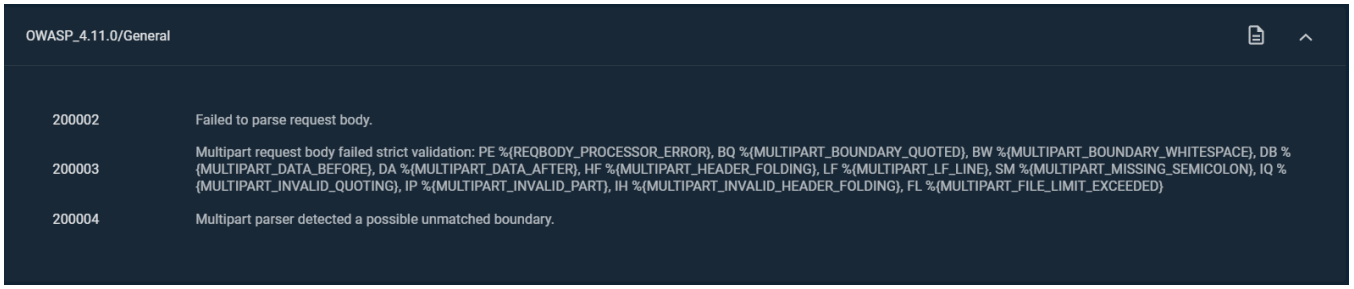
Раздел Security rules

Фильтр обеспечивает возможность фильтрации по видам правил (например, по стандарту OWASP). При нажатии **Change active version set** можно изменить активный набор правил.

Для просмотра набора правил одной группы следует нажать на его заголовок в списке правил, откроется окно просмотра перечня правил, входящих в данную группу.

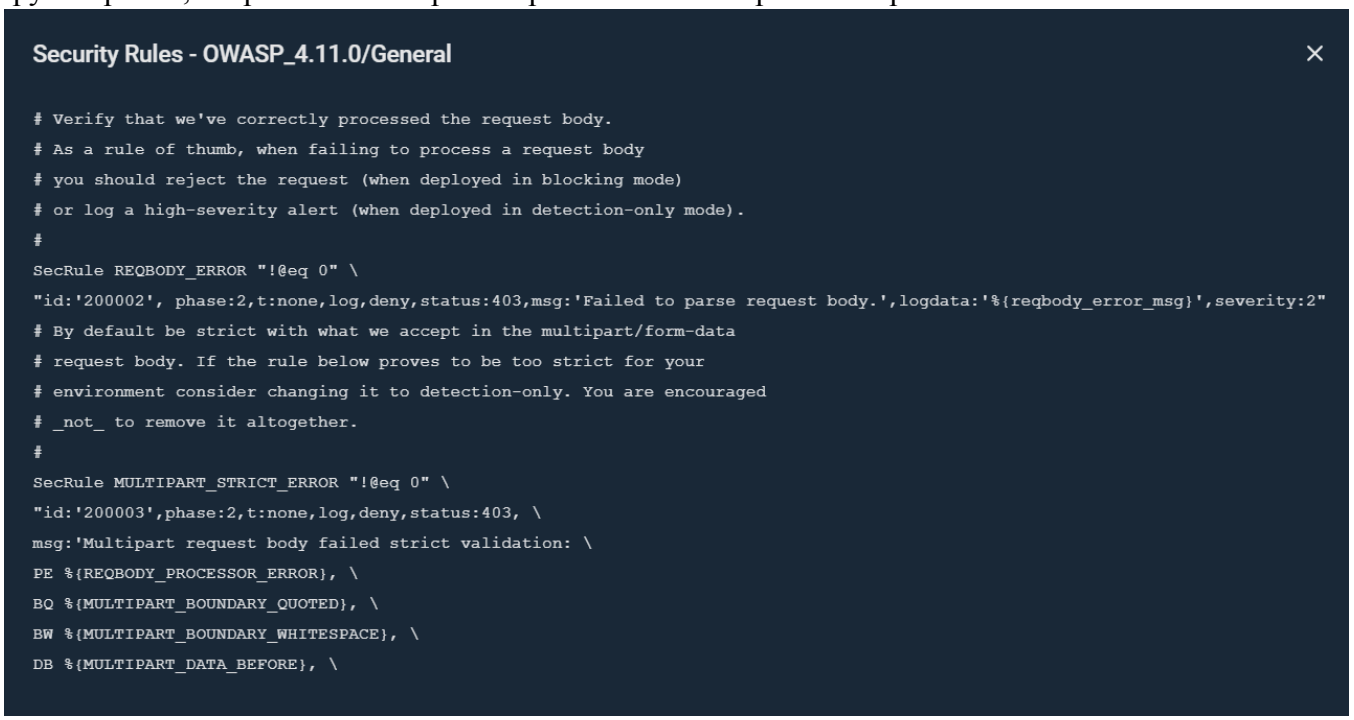
3.2.1.1 Просмотр правила

Для просмотра набора правил одной группы следует нажать на его заголовок в списке правил, откроется окно просмотра перечня правил, входящих в данную группу.



Окно просмотра перечня правил

Для просмотра полного содержания правила следует нажать на значок страницы в списке групп правил, откроется окно просмотра описания выбранного правила.

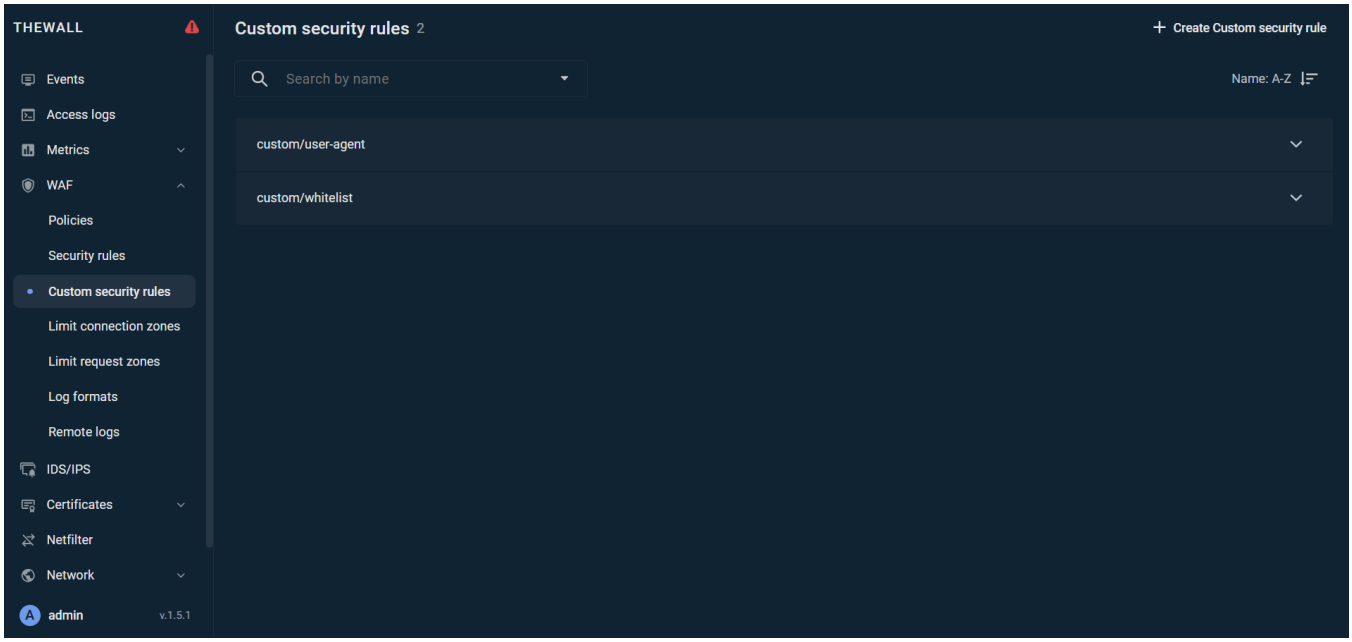


Окно просмотра содержания правила

3.2.2 Кастомные правила безопасности (созданные пользователем)

Пользователь в дополнение к общим правилам безопасности, созданным по умолчанию, может создавать, редактировать и удалять свои кастомные правила безопасности.

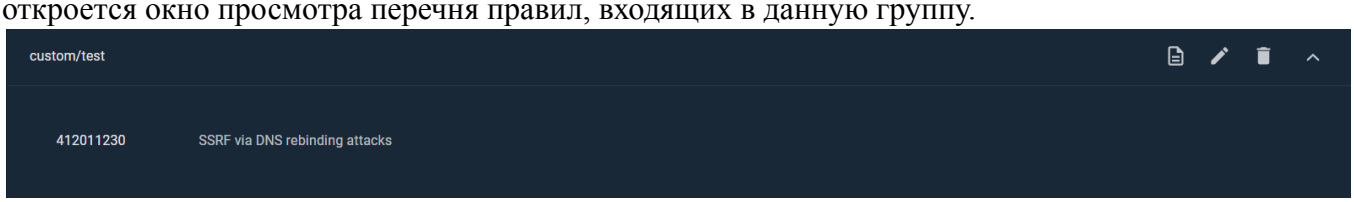
Список групп кастомных правил безопасности, созданных пользователем, отображается на вкладке **Custom security rules** подраздела **WAF**.



Раздел Custom security rules

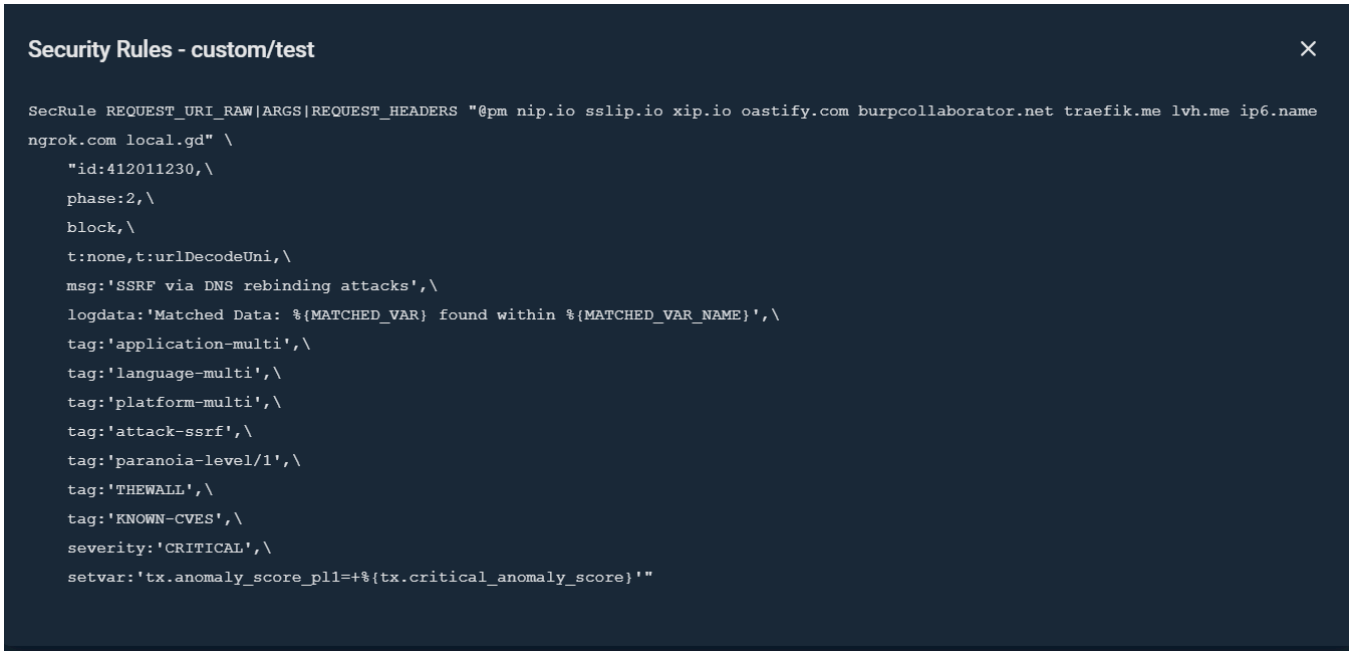
3.2.2.1 Просмотр кастомных правил безопасности

Для просмотра набора правил одной следует нажать на его заголовок в списке правил, откроется окно просмотра перечня правил, входящих в данную группу.



Окно просмотра перечня кастомных правил

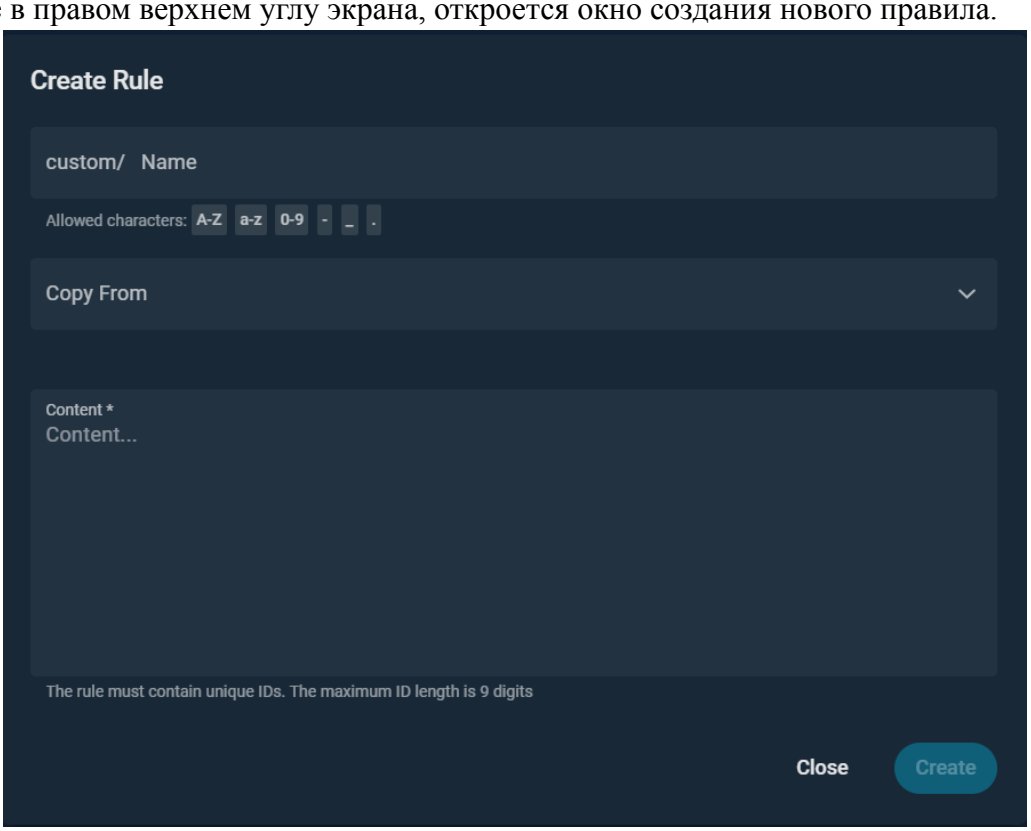
Для просмотра полного содержания правила следует нажать на значок страницы в списке групп правил, откроется окно просмотра описания выбранного правила.



Окно просмотра содержимого кастомного правила

3.2.2.2 Создание кастомных правил безопасности

Для создания нового правила безопасности необходимо нажать кнопку +Create custom security rule в правом верхнем углу экрана, откроется окно создания нового правила.



Окно создания кастомного правила

При создании нового правила следует указать всего два параметра:

- Название правила – указывается в поле Name;
- Содержимое правила – указывается в поле Content.

Поле Copy From позволяет скопировать содержимое другого правила.

> Правило должно содержать уникальные идентификаторы. Максимальная длина идентификатора составляет 9 цифр. При копировании правила все ID должны быть заменены на уникальные. В кастомных правилах рекомендовано использовать только незарезервированные ID. Описание зарезервированных ID доступно по ссылке – **https://coreruleset.org/docs/3-about-rules/ruleid/**.

3.2.2.3 Редактирование кастомных правил безопасности

Для редактирования (обновления) правила нажмите значок карандаша в контекстном меню соответствующего правила. При этом откроется окно редактирования правила. Для сохранения параметров редактирования нужно нажать **Update** в правом нижнем углу.



Окно редактирования правила

3.2.2.4 Удаление кастомных правил безопасности

Удалить кастомное правило можно только если оно не используется в политике. Для удаления кастомного правила безопасности нажмите значок корзины в контекстном меню определенного правила.

4 Настройка фильтрации данных на канальном уровне

Настройка межсетевого экрана канального уровня осуществляется при помощи утилиты arptables из интерфейса командной строки.

4.1 Таблицы

По умолчанию используется только одна таблица ("filter"). Таблица "filter" включает две предустановленные цепочки: INPUT – для входящих ARP-пакетов, OUTPUT – для исходящих ARP-пакетов.

4.2 Опции

Опции, используемые в **arptables**, разделяются на следующие группы:

- Команды
- Параметры
- Дополнительные опции

4.3 Команды

Команды определяют операцию, которую необходимо выполнить. Можно указать только одну из команд, если иное не указано ниже.

-A, --append *цепочка описание_правила*

Добавьте одно или несколько правил в конец выбранной цепочки.

-D, --delete *цепочка описание_правила*

-D, --delete *цепочка номер_правила*

Удалить одно или несколько правил из выбранной цепочки. Существует две версии этой команды: правило может быть указано в виде номера в цепочке (начиная с 1 для первого правила) или в виде правила, которому нужно соответствовать.

-I, --insert *цепочка* [*номер_правила*] *описание_правила*

Вставить одно или несколько правил в выбранную цепочку с заданным номером правила. Таким образом, если номер правила равен 1, правило или правила вставляются в начало цепочки. В случае, если номер правила не указан, то правило или правила будут добавлены в начало цепочки.

-R, --replace *цепочка [номер_правила] описание_правила*

Заменить правило в выбранной цепочке. Правила нумеруются с 1.

-L, --list [*цепочка*]

Вывести все правила из выбранной цепочки. Если ни одна цепочка не выбрана, отображаются все цепочки. Как и любая другая команда, она применяется к указанной таблице.

-F, --flush [*цепочка*]

Удалить все правила в цепочке.

-Z, --zero [*цепочка*]

Обнулить счетчики пакетов и байтов во всех цепочках. Также разрешено указать опцию **-L, --list**, чтобы увидеть счетчики непосредственно перед их очисткой.

-N, --new-chain *цепочка*

Создать новую цепочку. Название цепочки не должно совпадать с названиями уже существующих цепочек.

-X, --delete-chain [*цепочка*]

Удалить цепочку, созданную пользователем. Для успешного удаления цепочки необходимо предварительно удалить или заменить ссылающиеся на нее правила. Если

название цепочки не будет указано, то будет осуществлена попытка удаления всех пользовательских цепочек.

-P, --policy *цепочка действие*

Задать действие по умолчанию для указанной цепочки. Только для встроенных цепочек может использоваться действие по умолчанию.

-E, --rename-chain *старое_название новое_название*

Переименовать цепочку. Применимо только к пользовательским цепочкам

4.4 Параметры

Применяются следующие параметры, используемые для описания правил:

-s, --source-ip [!] *адрес*[/маска]

IP-адрес источника и маска сети (например, 24). Использование оператора ! здесь и далее означает «кроме» (инвертирует значение правила).

-d, --destination-ip [!] *адрес*[/маска]

IP-адрес получателя и маска сети (например, 24).

-- source-mac [!] *MAC-адрес* [маска]

MAC-адрес источника. Должен состоять из одной или нескольких 8-битных шестнадцатеричных цифр, разделенных двоеточиями. Если маска не указана, то она считается равной 0xff, для каждой 8-битной шестнадцатеричной цифры, указанной в адресе.

-- destination-mac [!] *MAC-адрес* [маска]

MAC-адрес получателя. Должен состоять из одной или нескольких 8-битных шестнадцатеричных цифр, разделенных двоеточиями. Если маска не указана, то она считается равной 0xff, для каждой 8-битной шестнадцатеричной цифры, указанной в адресе.

--h-type *тип*

Это поле определяет тип сети, который необходим локальной сети для передачи сообщения ARP. Как правило, используется значение 1 (Ethernet).

-i, --in-interface [!] *имя*

Имя входящего интерфейса (используется только с цепочкой INPUT).

-o, --out-interface [!] *имя*

Имя исходящего интерфейса (используется только с цепочкой OUTPUT).

--opcode *код*

Задает тип ARP-пакета. Для ARP-запроса используется код 1, для ARP-ответа – код 2.

-j, --jump *действие*

Определяет действие, которое необходимо выполнить в случае, если фрейм соответствует правилу. В качестве действия могут быть указаны следующие:

- ACCEPT – пропустить пакет
- DROP – отбросить пакет
- RETURN – прекращение прохождения этой цепочки и возобновление на следующем правиле предыдущей (вызывающей) цепочки
- QUEUE – передача пакета в пространство пользователя (если поддерживается ядром)

-c, --set-counters *пакеты байты*

Позволяет администратору инициализировать счетчики пакетов и байтов правила (во время операций INSERT, APPEND, REPLACE).

-t, --table *таблица*

Эта опция указывает таблицу, с которой должна работать команда. По умолчанию используется таблица «filter».

4.5 Дополнительные опции

Могут быть использованы следующие дополнительные опции:

-v, --verbose

Подробный вывод. Эта опция заставляет команду **-L, --list** отображать имена интерфейсов, параметры правил (если есть) и маски TOS. При добавлении, вставке, удалении и замене будет распечатана подробная информация о правиле или правилах.

-n, --numeric

Числовой вывод. IP-адреса и номера портов будут напечатаны в числовом формате. По умолчанию программа попытается отобразить их как имена хостов, имена сетей или служб (если это применимо).

-x, --exact

Отображение точного значения счетчиков пакетов и байтов, а не только округленных чисел в виде K (кратных 1000), M (кратных 1000 КБ) или G (кратных 1000M). Эта опция актуальна только для команды **-L, --list**.

--line-numbers

При перечислении правил указываются номера строк перед каждым правилом, соответствующие положению этого правила в цепочке.

4.6 Примеры использования

Для примера в сети 192.168.16.0/24 имеется 3 хоста (Таблица 2):

Таблица 2 – Формат записей списка интерфейсов

Имя	**IP-адрес**	**MAC-адрес**
HostA (Frisbee WAF)	192.168.16.1	aa:aa:aa:aa:aa:aa
HostB	192.168.16.2	bb:bb:bb:bb:bb:bb
HostC	192.168.16.3	cc:cc:cc:cc:cc:cc

1. Для того чтобы разрешить доступ HostB к HostA и запретить доступ HostC к HostA, выполним следующие команды:

```
sudo arptables -A INPUT --source-mac bb:bb:bb:bb:bb:bb -j ACCEPT
sudo arptables -A INPUT --source-mac cc:cc:cc:cc:cc:cc -j DROP
```

Для просмотра правил выполним:

```
sudo  arptables -L --line-number
```

Вывод должен выглядеть следующим образом:

```
Chain  INPUT (policy ACCEPT)
1 -j ACCEPT --src-mac bb:bb:bb:bb:bb:bb
2  -j DROP --src-mac cc:cc:cc:cc:cc:cc

Chain OUTPUT (policy ACCEPT)
```

Обращаем внимание, что действие по умолчанию для обеих цепочек ACCEPT.

2. Для того чтобы разрешить доступ HostC к HostA, воспользуемся командой замены правила:

```
sudo  arptables -R INPUT 2 --source-mac cc:cc:cc:cc:cc:cc -j ACCEPT
```

Для просмотра правил выполним:

```
sudo arptables -L --line-number
```

Вывод должен выглядеть следующим образом:

```
Chain INPUT (policy ACCEPT)
1 -j ACCEPT --src-mac bb:bb:bb:bb:bb:bb
2 -j ACCEPT --src-mac cc:cc:cc:cc:cc:cc

Chain OUTPUT (policy ACCEPT)
```

3. Для того чтобы запретить доступ всех хостов, кроме HostB к HostA, используем действие по умолчанию для цепочки и удалим разрешающее правило HostC к HostA:

```
sudo arptables -P INPUT DROP
sudo arptables -D INPUT --source-mac cc:cc:cc:cc:cc:cc -j ACCEPT
```

Для просмотра правил выполним (в данном примере номера строк не выводим):

```
sudo arptables -L
```

Вывод должен выглядеть следующим образом:

```
Chain INPUT (policy DROP)
-j ACCEPT --src-mac bb:bb:bb:bb:bb:bb

Chain OUTPUT (policy ACCEPT
```

5 Настройка динамической маршрутизации

В качестве протокола динамической маршрутизации Frisbee WAF поддерживает протокол OSPF.

Протокол OSPF

Протокол OSPF является протоколом, основанным на технологии отслеживания состояния канала (link-state). Это означает, что каждый маршрутизатор передает информацию о состоянии своих интерфейсов и интерфейсов своих соседей своим соседям при помощи Link State Advertisement (LSA). Каждый маршрутизатор по имеющейся информации составляет Link State Database (LSDB). На основе имеющейся информации каждый маршрутизатор строит таблицу маршрутизации на основе алгоритма Дейкстры Shortest Path First (SPF).

Зоны

OSPF позволяет разбивать сети на несколько зон. Каждая такая зона должна быть подключена к backbone зоне с использованием пограничного маршрутизатора Area Boundary Router (ABR). ABR отвечает за суммирование маршрутной информации и обмен ею с другими зонами.

5.1 Настройка протокола OSPF

Настройка протокола OSPF осуществляется из интерфейса командной строки. Для активации протокола OSPF перейдите в файл конфигурации /etc/frr/daemons и укажите:

```
ospfd=yes
```

При необходимости запуска нескольких процессов OSPF добавьте строку вида:

```
ospfd_instances=1,5,6
```

После этого перезапустите daemon динамической маршрутизации:

```
sudo systemctl restart frr
```

Проверить запуск процессов ospfd можно, например, используя команду:

```
ps -ef | grep "ospfd"
frr     11816     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1 -n 1
frr     11822     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1 -n 5
frr     11828     1  0 17:30 ?        00:00:00 /usr/lib/frr/ospfd --daemon -A 127.0.0.1 -n 6
```

Номера процессов в файле конфигурации /etc/frr/daemons должны совпадать с номерами, указанными при настройке OSPF.

Для настройки протокола OSPF перейдите к надстройке daemon маршрутизации (используются Cisco like команды), выполнив:

```
sudo vtysh
```

Для перехода в режим конфигурирования выполните команду:

```
configure terminal
```

Далее проведите настройку. Например:

```
router ospf 5
  ospf router-id 1.2.3.4
```

```
  area 0.0.0.0 authentication message-digest
  ...
```

Для сохранения настроек используется команда:

```
write memory
```

5.2 Общие настройки

router ospf [{(1-65535)|vrf NAME}]

Включить или отключить процесс OSPF. При указании нескольких процессов имена для vrf не поддерживаются.

ospf router-id A.B.C.D

Данная настройка устанавливает *router-id* для процесса OSPF. *Router-id* может быть IP-адресом, но не обязан им быть. Может быть любым 32-битным числом. Данное значение должно быть уникальным во всем OSPF-домене. Если данный параметр не указан, то он будет выбран автоматически.

ospf abr-type TYPE

TYPE может иметь значение *cisco*|*ibm*|*shortcut*|*standart*. Значение *cisco* и *ibm* эквивалентны.

Стандарт OSPF не позволяет ABR-маршрутизатору (Area Border Router) рассматривать маршруты не через backbone-зону, если интерфейс в сторону backbone-зоны выключен. Данное ограничение действует даже в случае, если другие ABR-маршрутизаторы подключены к backbone-зоне. Данное ограничение введено для избегания петель маршрутизации.

При использовании значения *cisco* или *ibm* ограничение, описанное выше, снимается, позволяя ABR-маршрутизатору (Area Border Router) рассматривать маршруты не через backbone-зону, и маршрутизировать трафик не через backbone-зону в качестве шлюза по умолчанию в случае, если линк в сторону backbone-зоны отключен.

Более подробное описание поведения, контролируемого данной командой, можно найти в RFC 3509.

ospf rfc1583compatibility

Поддержка RFC 2328, приемника RFC 1583.

В нормальном состоянии данная команда НЕ ДОЛЖНА использоваться.

log-adjacency-changes [detail]

Используется для журналирования протокола OSPF. В syslog будут выводиться сообщения об изменении состояния канала.

passive-interface default

Сделать все интерфейсы, относящиеся к данному роутеру, пассивными по умолчанию. Пассивный интерфейс означает запрет отправки LSA через указанный интерфейс. Например, если за интерфейсом не находится никаких маршрутизаторов и отсутствует необходимость, чтобы устройства за ним получали апдейты протокола OSPF. При этом, сети, объявленные на этом интерфейсе, будут видны с других маршрутизаторов.

timers throttle spf (0-600000) (0-600000) (0-600000)

Данная команда устанавливает начальную задержку, *initial holdtime* и *maximum holdtime* между событием, которое инициировало расчет SPF и расчетом SPF. Значение указывается в миллисекундах и должно быть в диапазоне от 0 до 600000.

Пример:

```
router ospf
timers throttle spf 200 400 10000
```

В данном примере, начальная задержка установлена 200 мс, *initial holdtime* – 400 мс, *maximum holdtime* – 10. Таким образом всегда будет задержка в минимум 200 мс между событием, инициировавшим расчет SPF и расчетом SPF. Дальнейшие последовательные вычисления SPF всегда будут разделены интервалом от 400 мс до 10 с, увеличиваясь на 400 мс каждый раз, когда будет происходить событие, инициирующее расчет SPF.

auto-cost reference-bandwidth (1-4294967)

Данная команда устанавливает значение *reference-bandwidth*, используемой по умолчанию для расчета стоимости cost. Cost = *reference-bandwidth* / *interface bandwidth*. Значение по умолчанию 100 Мб/с (то есть для интерфейса 100Мб/с стоимость равна 1).

Данную команду необходимо выполнить на всех маршрутизаторах домена, чтобы протокол OSPF мог правильно выбирать кратчайшие пути к сетям назначения.

neighbor A.B.C.D [poll-interval (1-65535)] [priority (0-255)]

Настраивает ip-адрес, *poll interval* и приоритет для OSPF-соседа. *Poll interval* определяет частоту отправки сообщения hello соседу, который неактивен. Когда установлено отношение соседства, то сообщение hello будет отправляться с частотой *hello interval*. Значение по умолчанию для poll interval – 60 секунд. Приоритет используется при выборе designated router (DR).

network A.B.C.D/M area A.B.C.D
network A.B.C.D/M area (0-4294967295)

Данная команда определяет включенные OSPF-интерфейсы. Если на интерфейсе установлен адрес из указанного диапазона, то данная команда активирует OSPF на данном интерфейсе, чтобы роутер мог взаимодействовать с другим роутером с настроенным протоколом OSPF за данным интерфейсом.

proactive-arp

Данная команда включает или отключает отправку ARP-запросов для обновления таблицы соседей. Позволяет улучшить работу протокола для /32 подсетей или P2P-соединений.

Данная функция активирована по умолчанию.

clear ip ospf [(1-65535)] process

Данная команда используется для очистки структуры данных, связанной с указанным OSPF-процессом. Она разорвет отношения соседства, и они будут переинициализированы. LSDB также будет очищена.

Данная команда будет полезной, когда внесены изменения в *router-id*, и требуется, чтобы указанные изменения имели эффект. Данная команда может быть выполнена вместо перезапуска daemon ospfd.

clear ip ospf [(1-65535)] neighbor

Данная команда используется для очистки структуры данных, связанной с указанным OSPF-процессом. Она разорвет отношения соседства, и они будут переинициализированы.

Данная команда может быть использована, если сосед завис в каком-либо состоянии.

maximum-paths (1-64)

Используйте данную команду для контроля максимального количества маршрутов с одинаковой стоимостью (ECMP).

socket buffer <send | recv | all> (1-4000000000)

Данная команда задает размер буферов (отправки или получения) для каждого экземпляра ospf. Использование данной команды с указанием no ведет к сбросу обоих значений к значениям по умолчанию.

no socket-per-interface

Обычно ospfd использует сокет для каждого интерфейса для отправки пакетов. Эта команда отключает эти сокеты для каждого интерфейса и заставляет ospfd использовать один сокет для каждого экземпляра ospf для отправки и получения пакетов.

5.3 Настройки зон

area A.B.C.D range A.B.C.D/M [advertise [cost (0-16777215)]]
area (0-4294967295) range A.B.C.D/M [advertise [cost (0-16777215)]]

Суммирует внутризонные маршруты в один LSA-анонс типа 3. Данная команда может быть использована только для ABR и только для router-LSA (тип 1) и network-LSA (тип 2). AS-external-LSA (тип 5) не может быть суммирован.

Пример:

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/
```

В данной конфигурации будет отправлен один LSA-анонс типа 3 с маршрутной информацией о сети 10.0.0.0/8 в backbone-зоне, если зона 0.0.0.10 содержит как минимум один внутризонный (то есть, описанный с использованием router or network LSA) маршрут из этого диапазона.

area A.B.C.D range A.B.C.D/M not-advertise
area (0-4294967295) range A.B.C.D/M not-advertise

Вместо суммирования маршрутов внутри зоны, фильтрует их. То есть внутренние маршруты не передаются в другие зоны. Данная команда используется только в случае использования ABR.

area A.B.C.D range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}
area (0-4294967295) range A.B.C.D/M {substitute A.B.C.D/M|cost (0-16777215)}

Замена суммарного префикса другим префиксом.

Пример:

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 range 10.0.0.0/8 substitute 11.0.0.0/8
```

Одно LSA-сообщение типа 3 с маршрутной информацией о сети 11.0.0.0/8 будет отправлено в backbone area, если зона 0.0.0.10 содержит хотя бы одну внутреннюю сеть (то есть описанную с помощью router-LSA or network-LSA) из диапазона 10.0.0.0/8.

По умолчанию метрика суммарного маршрута рассчитывается как наивысшая метрика среди суммарных маршрутов. Однако параметр стоимости можно использовать для установки явной метрики.

Данная команда имеет смысл только при использовании ABR.

area A.B.C.D virtual-link A.B.C.D
area (0-4294967295) virtual-link A.B.C.D
area A.B.C.D shortcut
area (0-4294967295) shortcut

Настраивает зону как shortcut-совместимую. Подробное описание доступно в RFC 3509.

area A.B.C.D stub
area (0-4294967295) stub

Данная команда задает stub-зону – зону. Все префиксы, которые будут перераспределены в домен OSPF из другого протокола маршрутизации, не будут объявляться в области stub. Поскольку LSA типа 5 не передаются в области stub, также невозможно иметь пограничный маршрутизатор автономной системы ASBR в области stub. Для доступа к сетям в других областях будет использоваться маршрут по умолчанию.

area A.B.C.D stub no-summary
area (0-4294967295) stub no-summary

Не позволяет ABR передавать суммаризированные внутризонные маршруты в указанную stub-зону.

area A.B.C.D nssa
area (0-4294967295) nssa

Данная команда задает NSSA (Not-So-Stubby Area) зону. Это то же самое, что и область stub, за исключением того, что разрешено иметь ASBR в этой области. Это то место, где включается LSA типа 7. Поскольку не разрешено использовать LSA типа 5, OSPF в области NSSA просто будет использовать новый тип LSA. NSSA зоны описаны в RFC 3101.

area A.B.C.D nssa suppress-fa
area (0-4294967295) nssa suppress-fa

Настраивает установку адреса пересылки 0.0.0.0 во всех LSA типа 5, транслируемых из LSA типа 7. Маршрутизатор должен быть выбран транслирующим в зоне, чтобы команда имела эффект. Данная настройка заставляет маршрутизаторы, настроенные не анонсировать адреса в backbone зону, отправлять их напрямую к NSSA ABR.

area A.B.C.D nssa default-information-originate [metric-type (1-2)] [metric (0-16777214)]
area (0-4294967295) nssa default-information-originate [metric-type (1-2)] [metric (0-16777214)]

NSSA ABR и ASBR могут быть настроены данной командой для создания LSA типа 7 для передачи маршрута по умолчанию в NSSA зону.

area A.B.C.D nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]
area (0-4294967295) nssa range A.B.C.D/M [<not-advertise|cost (0-16777215)>]

Данная команда объединяет набор внешних подсетей в единое LSA типа 7, которое далее транслирует в LSA типа 5 и анонсируется в backbone зону. Данная команда может быть использована на NSSA ABR маршрутизаторе.

По умолчанию метрика суммированного маршрута вычисляется как наибольшая из метрик суммируемых маршрутов. Однако параметр стоимости можно использовать для установки явной метрики.

Опция *not-advertise* предотвращает объявление суммированного маршрута, эффективно фильтруя суммарные маршруты.

area A.B.C.D default-cost (0-16777215)

Данная команда задает стоимость LSA анонсов для stubby зон.

area A.B.C.D export-list NAME
area (0-4294967295) export-list NAME

Фильтрует LSA типа 3 для, анонсированные для других зон, созданные внутри зоны, для указанной зоны.

Пример:

```
router ospf
 network 192.168.1.0/24 area 0.0.0.0
 network 10.0.0.0/8 area 0.0.0.10
 area 0.0.0.10 export-list foo
!
access-list foo permit 10.10.0.0/16
access-list foo deny any
```

В данном примере любые внутренние маршруты из зоны 0.0.0.10 и из диапазона 10.10.0.0/16 (например, 10.10.1.0/24 и 10.10.2.128/30) будут анонсироваться в другие зоны как LSA 3 типа, другие (например 10.11.0.0/16 или 10.128.30.16/30) – нет.

Эта команда относится только к ABR для определенной зоны.

area A.B.C.D import-list NAME
area (0-4294967295) import-list NAME

Аналогично *export-list*, но применяется к маршруту, анонсируемой в указанную зону как LSA 3 типа.

area A.B.C.D filter-list prefix NAME in
area A.B.C.D filter-list prefix NAME out
area (0-4294967295) filter-list prefix NAME in
area (0-4294967295) filter-list prefix NAME out

Фильтрует LSA 3 типа к/из зоны, используя *prefix list*. Данная команда применяется к ABR.

area A.B.C.D authentication
area (0-4294967295) authentication

Задает пароль для аутентификации для указанной области.

area A.B.C.D authentication message-digest
area (0-4294967295) authentication message-digest

Указывает, что пакеты OSPF должны быть аутентифицированы с использованием алгоритма MD5 HMAC в указанной зоне. Ключи должны быть заданы на уровне интерфейсов *(ip ospf message-digest-key*).

5.4 Настройка интерфейсов

ip ospf area AREA [ADDR]

Включает OSPF на интерфейсе, помещая его в указанную зону. Опционально может быть указана подсеть. В таком случае будет применено только к интерфейсам, принадлежащим указанной подсети. В случае большого количества интерфейсов и (или) подсетей используйте данную команду (вместо *network A.B.C.D/M area A.B.C.*D) для повышения производительности.

Стоит обратить внимание, что смешивать команды для сетей (*network*) и для интерфейсов (*ip ospf*) на одном маршрутизаторе нельзя. Если настройка осуществлена командой *ip ospf*, то выполнение команды *network* завершится ошибкой.

ip ospf authentication-key AUTH_KEY

Устанавливает пароль для OSPF. После установки этого параметра все OSPF пакеты будут проходить аутентификацию. Длина AUTH_KEY может быть до 8 символов включительно.

Аутентификация с помощью простого текстового пароля небезопасна и устарела. Рекомендуется использовать аутентификацию MD5 HMAC.

ip ospf authentication message-digest

Определяет, что аутентификация с использованием MD5 HMAC должна использоваться для указанного интерфейса. Ключи должны быть указаны. Приоритет данной команды выше, чем *area A.B.C.D authentication message-digest*.

ip ospf message-digest-key KEYID md5 KEY

Задает имя ключа и ключ для аутентификации. Используется алгоритм хэширования MD5.

Имя ключа должно совпадать на всех соседних маршрутизаторах. Длина ключа не должна превышать 16 символов (в случае превышения будет сокращена до 16 символов).

ip ospf cost (1-65535)

Устанавливает стоимость интерфейса. Используется для расчета SPF.

ip ospf dead-interval (1-65535)
ip ospf dead-interval minimal hello-multiplier (2-20)

Задает количество секунд для *RouterDeadInterval*, используемого для *Wait Timer* и *Inactivity Timer*. Данное значение должно быть одинаковым для всех роутеров в одной сети. Значение по умолчанию – 40 секунд.

Если указано значение *minimal*, то *dead-interval* устанавливается равным 1 секунде. Кроме того требуется указать *hello-multiplier*. *Hello-multiplier* указывает, сколько hello-запросов будет отправляться в секунду. Значение *hello-multiplier* может быть от 2 (каждые 500 мс) до 20 (каждые 50 мс). Таким образом, время сходимости для OSPF может составлять 1 с.

ip ospf hello-interval (1-65535)

Устанавливает количество секунд для *HelloInterval*. При установке данного таймера сообщения Hello будут отправляться 1 раз в указанный промежуток времени для определенного интерфейса. Данное значение должно быть одинаковым для всех роутеров в одной сети. Значение по умолчанию – 10 секунд.

Данная команда не имеет эффекта, если для интерфейса выполнена команда *ip ospf dead-interval minimal hello-multiplier (2-20)*.

ip ospf graceful-restart hello-delay (1-1800)

Установите продолжительность времени, в течение которого Grace-LSA отправляются с интервалом в 1 секунду при восстановлении после незапланированного сбоя. В течение этого времени не отправляются Hello-пакеты.

Более высокая *hello-delay* увеличит вероятность того, что все соседи будут уведомлены о продолжающемся корректном перезапуске до получения пакета приветствия, что важно для корректного перезапуска. Однако hello-delay не следует устанавливать слишком большой, иначе время смежности может истечь. Рекомендуется установить для hello-delay и *hello-interval* одинаковые значения. Значение по умолчанию — 10 секунд.

ip ospf priority (0-255)

Устанавливает значение *RouterPriority*. Маршрутизатор с большим приоритетом с более высокой вероятностью будет выбран в качестве DR. Установка значения 0 для данного параметра лишает маршрутизатор возможности быть выбранным в качестве DR. Значение по умолчанию – 1.

ip ospf retransmit-interval (1-65535)

Устанавливает значение таймера повторной передачи пакетов, если не получен LSAck. Значение по умолчанию 5 секунд.

ip ospf transmit-delay (1-65535) [A.B.C.D]

Устанавливает задержку для передачи состояния канала для интерфейса. Значение по умолчанию – 1 секунда.

ip ospf passive [A.B.C.D]

Делает интерфейс пассивным в рамках OSPF: отправка LSA через указанный интерфейс не производится.

ip ospf prefix-suppression [A.B.C.D]

Все непосредственно подключенные сети, включенные в работу OSPF не попадут в LSA типа 1.

ip ospf neighbor-filter NAME [A.B.C.D]

Настраивает фильтрацию пакетов, получаемых от соседей, на интерфейсе, включенном в OSPF зону. Фильтр должен включать разрешающие или запрещающие правила для соседей по IP адресу источника.

Пример:

```
ip prefix-list nbr-filter seq 10 deny 10.1.0.2/32
ip prefix-list nbr-filter seq 200 permit any
interface eth0
 ip ospf neighbor-filter nbr-filter
```

В данном примере запрещено соседство с маршрутизатором 10.1.0.2.

ip ospf area (A.B.C.D|(0-4294967295))

Активирует OSPF на интерфейсе и указывает зону.

5.5 OSPF route-map

Используется для *route-map*.

set metric [+|-](0-4294967295)

Устанавливает метрику для маршрута при отправке анонса. Используется знак плюс (+) для добавления значения к имеющейся метрике. Используется знак минус (-) для вычитания значения из имеющейся метрики.

5.6 Graceful Restart

graceful-restart [grace-period (1-1800)]

Включает поддержку *graceful restart*, описанную в RFC 3623. Позволяет перезапустить процесс OSPF, не прерывая связь с соседями. Маршрутизатор будет продолжать передавать данные, пока процесс OSPF будет перезапускаться. Когда включен, значение по умолчанию равно 120 секундам.

graceful-restart helper enable [A.B.C.D]

Активирует *graceful restart helper support*: информирование соседей о предстоящей перезагрузке. Перезагружающийся маршрутизатор запрашивает у соседей определенное время на перезагрузку, после которой они заново станут соседями. Может быть указан id определенного соседа.

graceful-restart helper strict-lsa-checking

Если установлена настройка *strict lsa checking*, то *graceful restart* будет прерван, если произойдет изменение LSA, влияющее на перезапуск маршрутизатора. По умолчанию данная опция включена.

graceful-restart helper supported-grace-time (10-1800)

Время, запрашиваемое на перезагрузку в рамках *graceful restart helper*.

graceful-restart helper planned-only

Инициировать *graceful restart helper* только в случае плановой перезагрузки. По умолчанию поддерживается как запланированная, так и незапланированная перезагрузка.

graceful-restart prepare ip ospf

Инициирует *graceful restart* для всех экземпляров OSPF с включенным *graceful restart*. Daemon OSPF должен быть перезапущен в течение *graceful restart grace period*.

5.7 Отображение информации

show ip ospf [vrf <NAME|all>] [json]

Отображает разнообразную общую информацию о настройках OSPF.

show ip ospf interface [INTERFACE] [json]

Отображает состояние и настройки OSPF для указанного интерфейса. В случае, если не указан интерфейс, отображается информация о всех интерфейсах.

show ip ospf neighbor [json]
show ip ospf [vrf <NAME|all>] neighbor INTERFACE [json]
show ip ospf neighbor detail [json]
show ip ospf [vrf <NAME|all>] neighbor A.B.C.D [detail] [json]
show ip ospf [vrf <NAME|all>] neighbor INTERFACE detail [json]

Данные команды отображают информацию о LSDB.

show ip ospf [vrf <NAME|all>] database [self-originate] [json]

Отображает краткую информацию о базе данных OSPF.

show ip ospf [vrf <NAME|all>] database max-age [json]

Отображает наиболее старые LSA, присутствующие в OSPF LSDB.

show ip ospf [vrf <NAME|all>] database detail [LINK-STATE-ID] [adv-router A.B.C.D] [json]
show ip ospf [vrf <NAME|all>] database detail [LINK-STATE-ID] [self-originate] [json]
show ip ospf [vrf <NAME|all>] database (asbr-summary|external|network|router|summary|nssa-external|opaque-link|opaque-area|opaque-as) [LINK-STATE-ID] [adv-router A.B.C.D] [json]
show ip ospf [vrf <NAME|all>] database (asbr-summary|external|network|router|summary|nssa-external|opaque-link|opaque-area|opaque-as) [LINK-STATE-ID] [self-originate] [json]

Данные команды отображают детальную информацию об OSPF LSDB.

show ip ospf route [detail] [json]

Отобразить таблицу маршрутизации OSPF. В случае использования опции detailed отображается более подробная информация (например, *router-id* маршрутизатора, анонсировавшего маршрут и т. д.)

show ip ospf [vrf <NAME|all>] border-routers [json]

Отображает список пограничных маршрутизаторов ABR и ASBR. Данная информация может быть получена в формате json при указании соответствующего параметра.

show ip ospf graceful-restart helper [detail] [json]

Отобразить информацию о graceful restart helper, включая изменение конфигурации.

5.8 Суммирование внешних маршрутов

Эта функция суммирует исходящие внешние LSA (Type-5 and Type-7). Суммарный маршрут будет создан от имени всех совпадающих внешних LSA.

summary-address A.B.C.D/M [tag (1-4294967295)]

Эта команда включает/отключает суммирование для настроенного диапазона адресов. tag необязательный параметр. Если тег настроен, SR будет создан с помощью настроенного тега.

summary-address A.B.C.D/M no-advertise

Эта команда гарантирует, что summary LSA не будет объявлен для соответствующих внешних LSA.

aggregation timer (5-1800)

Настраивает интервал таймера задержки агрегации. Суммирование начинается только после истечения этого таймера задержки. По умолчанию интервал задержки составляет 5 секунд.

Данная команда с префиксом no сбрасывает интервал задержки агрегации на значение по умолчанию.

show ip ospf [vrf <NAME|all>] summary-address [detail] [json]

Отображает конфигурацию всех настроенных суммарных маршрутов с соответствующей внешней информацией LSA.

5.9 TI-LFA

Экспериментальная поддержка Topology Independent LFA (Loop-Free Alternate). Стоит учитывать, что TI-LFA требует корректную настройку сегментной маршрутизации (Segment Routing).

fast-reroute ti-lfa [node-protection]

Настраивается на уровне маршрутизатора (router). Активирует TI-LFA для всех интерфейсов.

На данный момент поддерживаются только P2P интерфейсы.

5.10 Отладка OSPF

debug ospf [(1-65535)] bfd

Включает или отключает уровень отладки для событий BFD (Bidirectional Forwarding Detection). Эта команда покажет сообщения BFD и сообщения интеграции BFD OSPF, которые в основном представляют собой переходы между состояниями и проблемы проверки.

debug ospf [(1-65535)] default-information

Отображает отладочную информацию для информации по умолчанию.

debug ospf [(1-65535)] packet (hello|dd|ls-request|ls-update|ls-ack|all) (send|recv) [detail]

Отображает отладочную информацию по пакетам.

debug ospf [(1-65535)] ism [status|events|timers]

Отображает отладочную информацию о состоянии интерфейсов.

debug ospf [(1-65535)] nsm [status|events|timers]

Отображает отладочную информацию о состоянии сети.

debug ospf [(1-65535)] event

Отображает отладочную информацию событий OSPF.

debug ospf [(1-65535)] nssa

Отображает отладочную информацию о Not So Stub Area.

debug ospf [(1-65535)] ldp-sync

Отображает отладочную информацию для LDP-синхронизации.

debug ospf [(1-65535)] lsa [aggregate|flooding|generate|install|refresh]

Отображает отладочную информацию о LSA-сообщениях.

debug ospf [(1-65535)] sr

Отображает отладочную информацию о сегментной маршрутизации.

debug ospf [(1-65535)] te

Отображает отладочную информацию о Traffic Engineering LSA-сообщениях.

debug ospf [(1-65535)] ti-lfa

Отображает отладочную информацию о SR TI-LFA.

5.11 Примеры настройки

Пример настройки с использованием пароля для аутентификации соседа:

```
interface ens33
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
router ospf
 network 192.168.0.0/16 area 0.0.0.1
 area 0.0.0.1 authentication message-digest
```

Пример настройки ABR с использованием пароля и суммированием маршрутов при взаимодействии между зонами:

```
interface eth0
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 ABCDEFGHIJK
!
interface ppp0
 ip ospf passive
!
interface br0
 ip ospf authentication message-digest
 ip ospf message-digest-key 2 md5 XYZ12345
!
router ospf
 ospf router-id 192.168.0.1
 redistribute connected
 network 192.168.0.0/24 area 0.0.0.0
 network 10.0.0.0/16 area 0.0.0.0
 network 192.168.1.0/24 area 0.0.0.1
 area 0.0.0.0 authentication message-digest
 area 0.0.0.0 range 10.0.0.0/16
 area 0.0.0.0 range 192.168.0.0/24
 area 0.0.0.1 authentication message-digest
 area 0.0.0.1 range 10.2.0.0/16
!
```

6 Настройка ротации логов

По умолчанию логи доступа (access logs) и событий (events) в Frisbee WAF хранятся около года; значение по умолчанию для access logs — 366 дней (CLICKHOUSE_NGINX_ACCESS_TTL_DAYS). Записи старше заданного TTL удаляются. Управление настройками ротации логов осуществляется с помощью изменения или добавления параметров TTL (time to live) в файле /opt/env/admin-console.env.

Предусмотрены следующие параметры редактирования TTL для настройки ротации логов:

CLICKHOUSE_NGINX_ACCESS_TTL_DAYS — TTL логов доступа (access logs);

CLICKHOUSE_WAF_LOGS_TTL_DAYS — TTL логов событий (events).

Таблицы Anomalies и связанные данные аномалий используют тот же TTL, что и access logs (CLICKHOUSE_NGINX_ACCESS_TTL_DAYS), а не CLICKHOUSE_WAF_LOGS_TTL_DAYS.

После изменения данных параметров необходимо перезапустить сервис admin-console. Например, следующей командой:

```
systemctl restart waf@admin-console
```

При перезапуске сервиса admin-console возможно прерывание доступа к UI Frisbee WAF без влияния на работоспособность и защищенность ресурсов.

> При изменении настройки TTL в отказоустойчивом кластере данные параметры необходимо установить на каждой ноде Frisbee WAF.

7 Резервная копия и восстановление

7.1 Создание резервной копии

Перед созданием резервной копии Frisbee WAF предварительно должны быть выполнены следующие условия:

- Frisbee WAF установлен и исправно работает;
- Осуществлены необходимые настройки межсетевого экрана веб-приложений.

> Служба создания резервных копий запускается ежедневно в 00.00 часов. Срок хранения каждой резервной копии - 7 дней. Резервные копии хранятся в каталоге /opt/backups/ в формате «ГГГГММДД».

Для ручного создания резервной копии Frisbee WAF необходимо:

1. Перейти в режим суперпользователя root.
2. Запустить следующие команды:
   - 2.1. systemctl status make-backup – проверка статуса работы службы резервного копирования и восстановления;
   - 2.2. systemctl start make-backup – включение службы. Необходимо запустить службу, если она не была запущена до этого;
   - 2.3. make-backup – сделать резервную копию.
   - 2.4. Созданная резервная копия доступна в каталоге /opt/backups/.

7.2 Восстановление

Восстановление Frisbee WAF проводится в следующих случаях:

- Переустановлена операционная система;
- Переустановлен межсетевой экран веб-приложений;
- Изменены какие-либо настройки или компоненты Frisbee WAF через webUI.

Для восстановления необходимо:

1. Файл с резервной копией следует упаковать и перенести в необходимый каталог. Примеры команд для копирования файла с резервной копией:

   cd /opt/backups/ — переход в каталог с резервными копиями
   tar cf $(date +%Y%m%d).tar $(date +%Y%m%d)/ — создание tar архива резервной копии
   cp $(date +%Y%m%d).tar /home/username — для копирования в домашнюю директорию
   либо scp $(date +%Y%m%d).tar username@hostname: для отправки на удаленный сервер по протоколу scp

2. Скопировать на сервер файл с резервной копией (при необходимости).
3. Запустить команду: restore /путь до файла с резервной копией/ГГГГММДД.tar

> При восстановлении файл с резервной копией обязательно должен быть в формате ГГГГММДД.tar.

8 Отправка событий аудита в SIEM

Frisbee WAF может передавать во внешнюю SIEM новые события из раздела **Audit**.

Настройка отправки выполняется отдельно на каждой ноде кластера. Настройка через веб-интерфейс и API не предусмотрена.

8.1 Категории событий

В SIEM передаются:

- действия пользователей (User actions);
- события аутентификации (Authentication);
- действия по управлению пользователями (User management).

Передаются только новые события. Ранее накопленная история аудита автоматически не отправляется.

8.2 Принцип работы

- Frisbee WAF записывает новые события аудита в файлы на ноде в каталоге /var/log/frisbee-waf/audit-siem/.Файлы событий хранятся ограниченное время — по умолчанию 5 суток, после чего перезаписываются.

- Файлы создаются в формате JSONL: user-actions-YYYY-MM-DD.jsonl, authentication-YYYY-MM-DD.jsonl и user-management-YYYY-MM-DD.jsonl. Каждая строка файла содержит одно событие в формате JSON.

- Служба rsyslog считывает события из этих файлов и отправляет их во внешнюю SIEM.

- По умолчанию rsyslog передаёт события в SIEM с параметрами Facility = local6 *(в SIEM соответствует значению "22")* и Severity = info. При необходимости данные параметры можно изменить в файле конфигурации rsyslog.

- Запись событий в файлы выполняется по умолчанию. Для передачи событий в SIEM требуется ручная настройка.

8.3 Настройка передачи событий на ноде

Для отправки логов аудита в SIEM на каждой ноде WAF необходимо выполнить следующие действия:

- Скопируйте пример конфигурации:

```
sudo cp /etc/rsyslog.d/60-frisbee-waf-audit-siem.conf.example
etc/rsyslog.d/60-frisbee-waf-audit-siem.conf
```

- Откройте конфигурационный файл:

```
sudo nano /etc/rsyslog.d/60-frisbee-waf-audit-siem.conf
```

Укажите **адрес** и **порт** SIEM, **протокол** TCP или UDP, а также **название интерфейса**, через которые планируется отправка логов в .

- Проверьте доступность порта SIEM:

```
nc -nvz -w 3 <SIEM_IP> <SIEM_PORT>      # TCP
nc -nvzu -w 3 <SIEM_IP> <SIEM_PORT>     # UDP
```

При доступности порта вы получите соответствующее сообщение.

- Проверьте конфигурацию rsyslog:

sudo rsyslogd -N1

- Перезапустите службу rsyslog и проверьте её состояние:

sudo systemctl restart rsyslog
sudo systemctl status rsyslog --no-pager

- Проведите настройку отправки логов в SIEM на каждой ноде кластера.

```
if ($syslogtag startswith "thewall-audit-") then {
    action(
        type="omfwd"
        target="192.0.2.1"
        port="514"
        protocol="tcp"
        Device="ens224"
        TCP_Framing="traditional"
        template="RSYSLOG_SyslogRFC5424Format"
        action.resumeRetryCount="-1"
        queue.type="LinkedList"
        queue.size="10000"
    )
    stop
```

Пример файла конфигурации

8.4 Резервное копирование

Активный файл /etc/rsyslog.d/60-frisbee-waf-audit-siem.conf сохраняется при создании резервной копии и восстанавливается при восстановлении. Обновление Frisbee WAF этот файл не перезаписывает.

Файл-пример и файлы событий в каталоге /var/log/frisbee-waf/audit-siem/ в резервную копию не входят.

Важно

4. Шифрование канала (TLS) в готовой настройке не предусмотрено.
5. Недоступность SIEM не останавливает работу Frisbee WAF.
6. Если rsyslog долго не работает, неотправленные файлы могут быть удалены по сроку хранения.

-

9 Руководство по правилам IDS/IPS

9.1 IDS/IPS-система

IDS/IPS-система, созданная на движке Suricata, обеспечивает защиту сетей от неавторизованного доступа. Она способна оперативно обнаруживать и эффективно предотвращать вторжения. Среди мер, которые принимаются для достижения ключевых целей IDS/IPS, можно выделить информирование специалистов по информационной безопасности о фактах попыток хакерских атак и внедрения вредоносных программ, обрыв соединения со злоумышленниками и перенастройку сетевого экрана для блокировки доступа к корпоративным данным.

В дополнение к функциям мониторинга и анализа событий по выявлению инцидентов, IDS обеспечивает выполнение следующих функций:

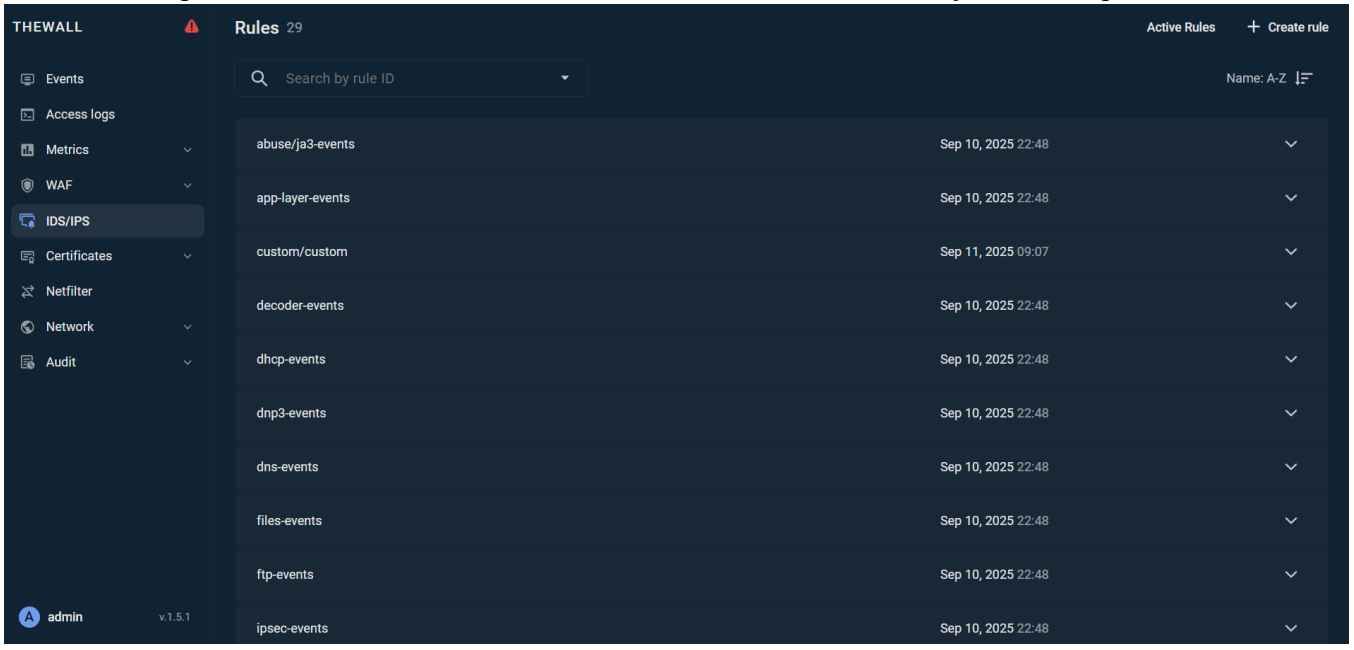
- Запись информации о событиях;
- Уведомление администраторов безопасности об инцидентах;
- Генерация отчетов, отчеты создаются путем суммирования всей информации по запрашиваемому событию (событиям);
- Система IPS дополняет технологию IDS, может самостоятельно не только определить угрозу, но и успешно заблокировать ее. Основные функциональные возможности IPS-системы:
- Блокировка атаки (обрыв сессии пользователя, нарушающего политику безопасности, блокирование доступа к ресурсам, хостам, приложениям);
- Изменение защищаемой среды (изменение конфигурации сетевых устройств для предотвращения атаки);
- Изменение содержания атаки путем анализа входящих запросов и отбрасывания опасных данных в заголовках пакетов.

Основные поддерживаемые возможности управления IDS/IPS-системой включают: просмотр стандартного набора правил IDS/IPS-системы (созданных по умолчанию), создание, изменение и удаление кастомных правил (созданных пользователем).

9.2 Управление правилами IDS/IPS-системы

Управление правилами IDS/IPS-системы осуществляется в подразделе Rules.

Поддерживаются возможности поиска, создания, изменения и удаления правил.



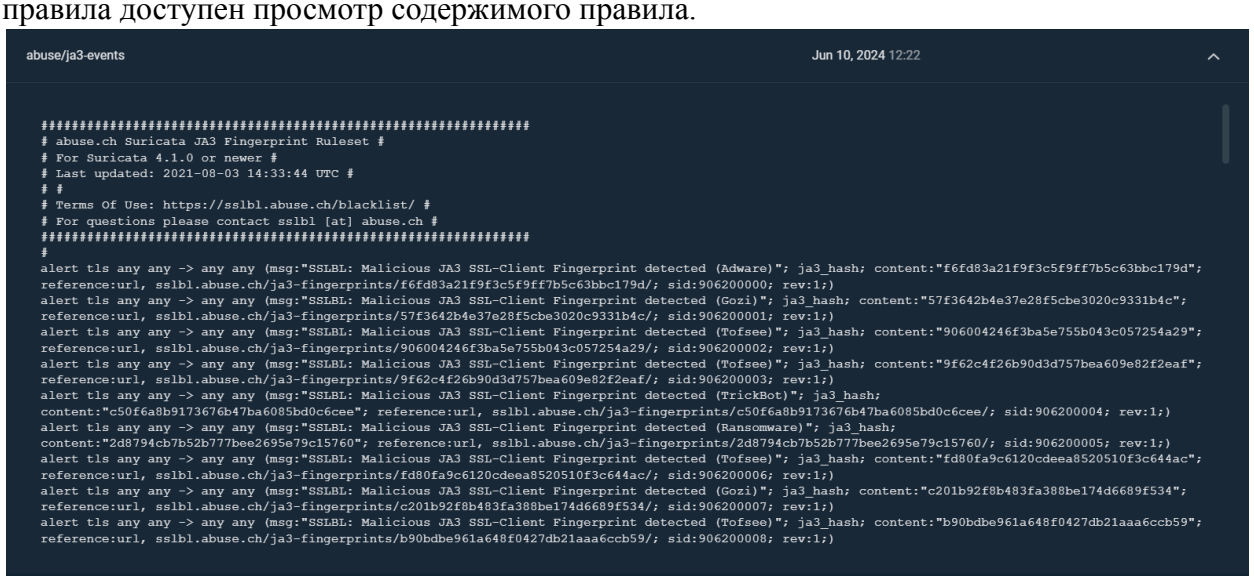
Общий вид окна для управления правилами

Основные разделы окна управления правилами IDS/IPS рассмотрены в таблице 3.

Таблица 3 – Основные разделы окна управления правилами IDS/IPS-системы

Раздел	**Описание**
Rules	Правила IDS/IPS-системы
Active Rules	Активные правила IDS/IPS-системы

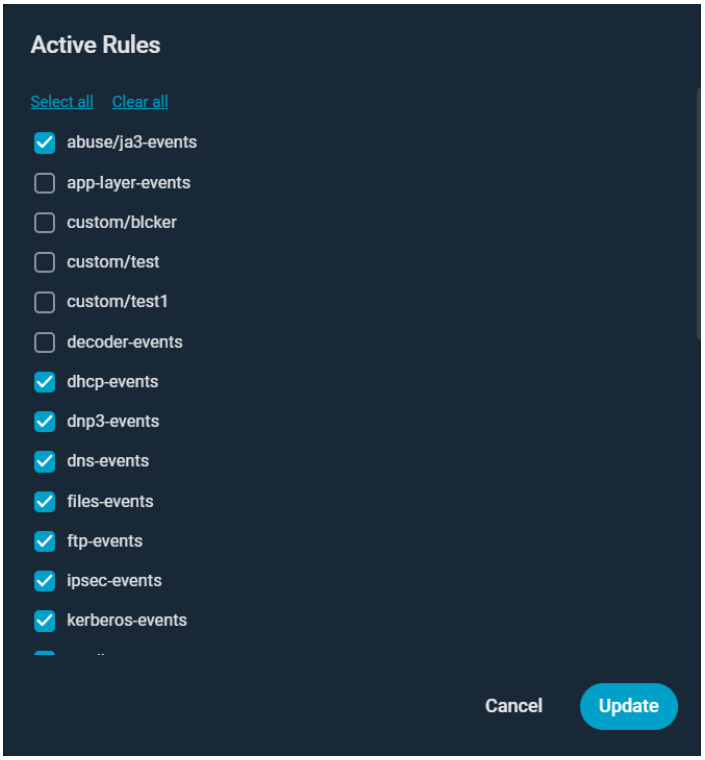
В разделе Rules доступен просмотр списка правил и при нажатии на строку с именем правила доступен просмотр содержимого правила.



Окно просмотра содержимого правила

В разделе Active Rules доступен выбор активных правил. Для изменения конфигурации необходимо выбрать нужные правила и нажать Update. Также доступны следующие кнопки:

- Clear all – очистить список правил. Данная кнопка не активна при полностью очищенном списке правил;
- Select all – выбрать все имеющиеся правила. Кнопка не активна при выборе всех правил в списке.



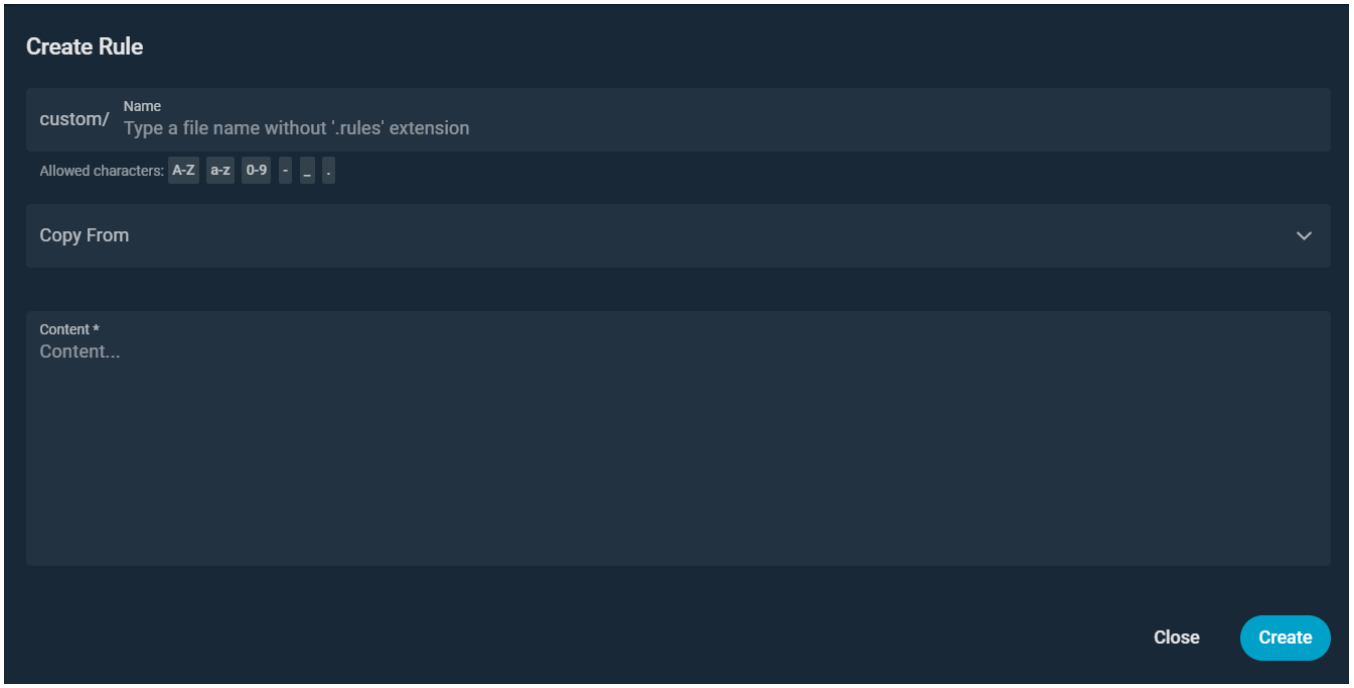
Раздел Active Rules

Доступны следующие функции управления правилами IDS/IPS:

- Поиск правил. Для поиска правила следует указать его наименование (name, id) в строке поиска, при этом будут отображаться только правила, наименование которых соответствует введенному критерию;
- Фильтрация правил по имени;
- Создание правила. В данном разделе создать можно только кастомное правило;

Для создания нового правила следует нажать на кнопку **+Create rule**, расположенную в правом верхнем углу окна. При этом откроется окно нового правила, содержащее следующие разделы:

  - Name – название правила;
  - Copy From – возможно скопировать содержимое из уже созданного правила;
  - Content – содержимое правила.



Окно создания нового правила

В отличие от кастомных правил ModSecurity (WAF), Suricata позволяет создавать кастомные правила с неоригинальными ID (sid), однако при включении правила отображается ошибка. Для включения такого правила необходимо отключить правило, на основе которого оно создано (при создании с помощью Copy From), или указать уникальный ID и потом активировать созданное правило, нажав кнопку **Active Rules** и выбрав нужное правило.

- Просмотр правила. Для просмотра правила следует нажать на его заголовок, при этом откроется окно просмотра без возможности изменения;
- Изменение правила. В данном разделе изменить можно только кастомное правило. Для изменения правила следует нажать на кнопку редактирования (обновления) в контекстном меню напротив соответствующего правила. После этого откроется окно, где можно изменить выбранное правило. После завершения редактирования нажмите на кнопку **Update** и внесенные изменения будут сохранены. Нажатие на кнопку **Close** приведет к закрытию текущего окна без сохранения.
- Удаление правила.

В данном разделе можно удалить только кастомное правило. Для удаления правила нажмите на значок корзины в контекстном меню напротив соответствующего правила. Выбранное правило будет удалено.

9.3 Разработка правил

В разделе IDS/IPS Frisbee WAF используется система управления трафиком на основе настраиваемых правил IPS/IDS-системы Suricata.

Основные возможности Suricata:

- Работа в многопоточном режиме, позволяет наиболее полно задействовать возможности многоядерных и многопроцессорных систем;
- Поддержка автоматического определения протоколов: IP, TCP, UDP, ICMP, HTTP, TLS, FTP и SMB. Возможность определения типа протокола в правилах, без привязки к номеру порта (например, блокировка HTTP-трафика на нестандартном порту);
- Специальная HTTP-библиотека для нормализации и разбора HTTP-трафика;

- Поддержка разбора трафика при сжатии содержимого пакетов методом Gzip;
- Очень быстрый механизм сопоставления по маске с большими наборами IP-адресов;
- Поддержка стандартных интерфейсов для перехвата трафика NFQueue, IPFRing, LibPcap, IPFW. Унифицированный формат вывода результатов проверки позволяет использовать стандартные утилиты для анализа;
- Возможность использования переменных в правилах: можно сохранить информацию из потока и позднее использовать ее в других правилах;
- Наличие модуля для ведения подробного лога транзитных HTTP-пересылок, лог сохраняется в стандартном формате Apache.

Более подробная информация о правилах доступна в Suricata User Guide по ссылке: **https://docs.suricata.io/en/suricata-7.0.8/**.

Пример правила

Пример правила IDS/IPS-системы приведен ниже.

```
# SMTP event  rules
#
# SID's fall in the 2220000+ range. See http://doc.emergingthreats.net/bin/view/Main/SidAllocation
#
# These sigs fire at most once per connection.
#
# A flowint smtp.anomaly.count is incremented for each match. By default it will be 0.
#
alert smtp any any -> any any (msg:"SURICATA SMTP invalid reply"; flow:established,to_client; app-layer-event:smtp.invalid_reply; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220000; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP unable to match reply with request"; flow:established,to_client; app-layer-event:smtp.unable_to_match_reply_with_request; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220001; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP max command line len exceeded"; flow:established; app-layer-event:smtp.max_command_line_len_exceeded; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220002; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP max reply line len exceeded"; flow:established,to_client; app-layer-event:smtp.max_reply_line_len_exceeded; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220003; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP invalid pipelined sequence"; flow:established,to_server; app-layer-event:smtp.invalid_pipelined_sequence; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220004; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP bdat chunk len exceeded"; flow:established; app-layer-event:smtp.bdat_chunk_len_exceeded; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220005; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP no server welcome message"; flow:established,to_client; app-layer-event:smtp.no_server_welcome_message; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220006; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP tls rejected"; flow:established; app-layer-event:smtp.tls_rejected; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220007; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP data command rejected"; flow:established,to_client; app-layer-event:smtp.data_command_rejected; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220008; rev:1;)
```

```
# SMTP MIME events
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime parser failed"; flow:established; app-layer-event:smtp.mime_parse_failed; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220009; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime malformed message found"; flow:established; app-layer-event:smtp.mime_malformed_msg; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220010; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime base64-decoding failed"; flow:established; app-layer-event:smtp.mime_invalid_base64; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220011; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime header name len exceeded"; flow:established; app-layer-event:smtp.mime_long_header_name; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220012; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime header value len exceeded"; flow:established; app-layer-event:smtp.mime_long_header_value; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220013; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime quoted-printable-decoding failed"; flow:established; app-layer-event:smtp.mime_invalid_qp; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220014; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime line len exceeded"; flow:established; app-layer-event:smtp.mime_long_line; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220015; rev:1;)
#alert smtp any any -> any any (msg:"SURICATA SMTP Mime encoded line len exceeded"; flow:established; app-layer-event:smtp.mime_long_enc_line; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220016; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP Mime boundary length exceeded"; flow:established,to_server; app-layer-event:smtp.mime_long_boundary; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220017; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP duplicate fields"; flow:established,to_server; app-layer-event:smtp.duplicate_fields; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220018; rev:1;)
alert smtp any any -> any any (msg:"SURICATA SMTP unparsable content"; flow:established,to_server; app-layer-event:smtp.unparsable_content; flowint:smtp.anomaly.count,+,1; classtype:protocol-command-decode; sid:2220019; rev:1;)
# next sid 2220020
```

9.3.1 Формат правил

Правило/сигнатура включает:

- Действие, определяющее, что происходит, когда сигнатура совпадает;
- Заголовок, определяющий протоколы, IP-адреса, порты и направления правила;
- Опции, определяющие специфику правила.

Пример правила:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

В этом примере, красный – действие, зеленый – заголовок.

Данная сигнатура взята из БД возникающих угроз, открытой базы данных, содержащей множество правил, которые можно свободно скачать и использовать в Suricata.

9.3.1.1 Действие

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

9.3.1.2 Протокол

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

Данное ключевое слово в сигнатуре сообщает Suricata, к какому протоколу это относится. Можно выбрать один из четырех основных протоколов:

- tcp (for tcp-traffic);
- udp;
- icmp;
- ip (ip stands for ‘all’ or ‘any’).

Есть также несколько так называемых протоколов прикладного уровня или протоколов уровня 7, из которых можно выбрать:

- http;
- ftp;
- tls (включая ssl);
- smb;
- dns;
- dcerpc;
- ssh;
- smtp;
- imap;
- modbus (по умолчанию отключено);
- dnp3 (по умолчанию отключено);
- enip (по умолчанию отключено);
- nfs (зависит от наличия rust);
- ikev2 (зависит от наличия rust);
- krb5 (зависит от наличия rust);
- ntp (зависит от наличия rust);
- dhcp (зависит от наличия rust).

Доступность данных протоколов зависит от включения протокола в файле конфигурации suricata.yaml.

При наличии сигнатуры, например, с протоколом http, Suricata гарантирует, что сигнатура может совпасть, только если она касается http-трафика.

9.3.1.3 Источник и цель

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

Первая выделенная часть — это источник, вторая — цель (пункт назначения); обратите внимание на направление стрелки.

В источнике и цели указывайте источник и пункт назначения трафика соответственно. Можно назначить IP-адреса (поддерживаются как IPv4, так и IPv6) и диапазоны IP-адресов. Их можно комбинировать с операторами (таблица 4).

Таблица 4 – Определение источника и пункта назначения трафика

Оператор	**Описание**
./.	Диапазоны IP (нотация CIDR)
!	Исключения / отрицания
[., .]	Группировка

Обычно используются переменные, такие как $HOME_NET и $EXTERNAL_NET. В файле конфигурации указываются IP-адреса, к которым они относятся, и эти параметры будут использоваться вместо переменных в правилах (см. примеры в таблице 5).

Таблица 5 – Пример определения диапазонов адресов

Пример	**Значение**
! 1.1.1.1	Каждый IP-адрес, кроме 1.1.1.1
![1.1.1.1, 1.1.1.2]	Каждый IP-адрес, кроме 1.1.1.1 и 1.1.1.2
$HOME_NET	Ваша настройка HOME_NET в yaml
[$EXTERNAL_NET, !$HOME_NET]	EXTERNAL_NET, а не HOME_NET
[10.0.0.0/24, !10.0.0.5]	10.0.0.0/24 за исключением 10.0.0.5

Порты (источник и цель)

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

Первая выделенная часть — это источник, вторая — цель; обратите внимание на направление стрелки.

Обратите внимание, что порт не определяет, какой протокол используется в связи. Скорее, он определяет, какое приложение получает данные.

Указанные выше порты обычно являются портами назначения. Исходные порты, то есть приложение, которое отправило пакет, обычно использует порт, случайным образом назначенный операционной системой. При написании правила для HTTP-сервиса, обычно пишется any -> 80. Это будет означать, что любой пакет из любого порта в приложении HTTP (работающего на порту 80) будет сопоставлен.

При настройке портов также можно использовать специальные операторы, как описано в таблице 6, примеры настройки с их использованием портов приведены в таблице 7.

Таблица 6 – Специальные операторы для настройки портов

Оператор	**Описание**
:	диапазоны портов
!	исключения / отрицания
[., .]	группировка

Таблица 7 – Примеры описания портов с использованием специальных операторов

Пример	**Значение**
[80, 81, 82]	Порт 80, 81 и 82
[80: 82]	Диапазон от 80 до 82
[1024:]	От 1024 до самого высокого номера порта
!80	Каждый порт, кроме 80
[80:100,!99]	Диапазон от 80 до 100, но 99 исключен
[1:80,![2,4]]	Диапазон от 1 до 80, кроме портов 2 и 4
[., [.,.]]	

9.3.1.4 Направление

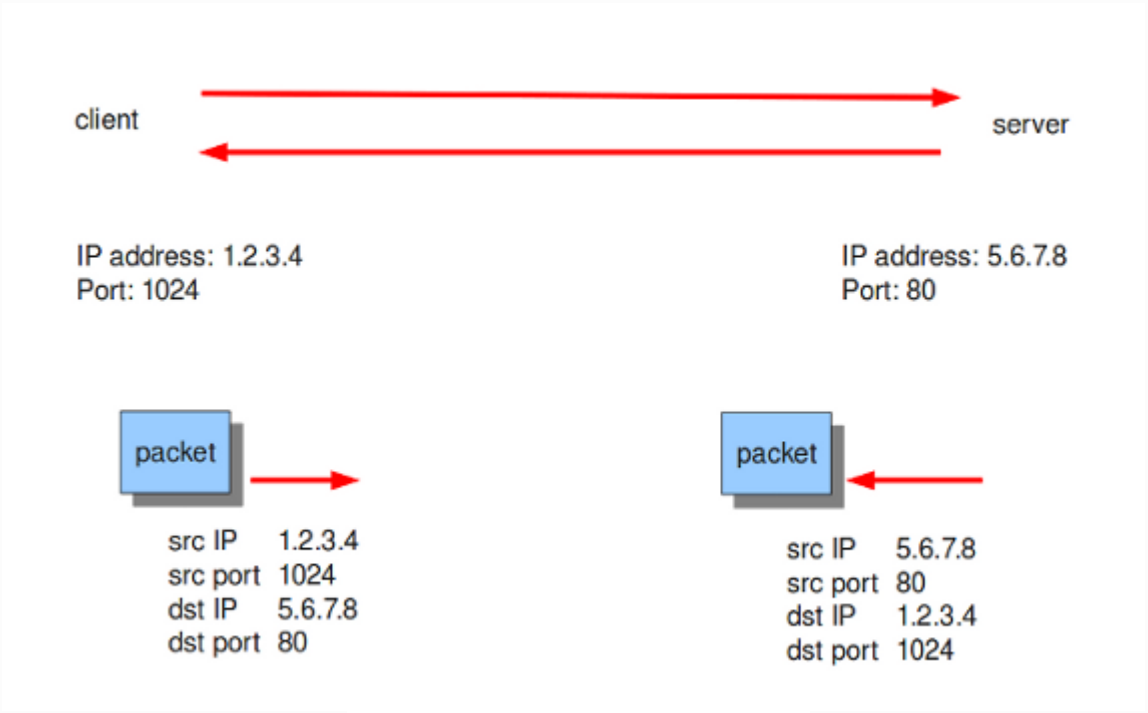
```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

Направление показывает, каким образом должна совпадать сигнатура. Почти каждая сигнатура имеет стрелку вправо (->). Это означает, что могут совпадать только пакеты с одинаковым направлением. Тем не менее, также возможно, чтобы правило соответствовало обоим направлениям (<>).

```
source -> destination
source <> destination  (both directions)
```

Нет «обратного» направления, т. е. нет <-.

Следующий пример иллюстрирует это. Скажем, есть клиент с IP-адресом 1.2.3.4 и портом 1024 и сервер с IP-адресом 5.6.7.8, прослушивающий порт 80 (обычно HTTP). Клиент отправляет сообщение на сервер, а сервер отвечает.



Направление проверки сигнатуры

Предположим, что есть правило со следующим заголовком:

```
alert tcp 1.2.3.4 1024 -> 5.6.7.8 80
```

Только первый пакет будет соответствовать этому правилу, поскольку направление указывает, что для ответного пакета оно не будет совпадать.

9.3.1.5 Опции правила

Остальное правило состоит из опций. Они заключены в круглые скобки и разделены точкой с запятой. Некоторые опции имеют параметры (например, msg), которые задаются ключевым словом опции, после которого следует двоеточие, а затем параметры. Другие не имеют параметров и являются просто ключевыми словами (такими как nocase):

```
<keyword>: <settings>;
<keyword>;
```

Параметры правила имеют определенный порядок, и изменение их порядка изменит смысл правила.

Символы ; и " имеют особое значение в языке правил Suricata и должны быть экранированы при использовании в значении опции правила. Например:

```
msg:"Message with semicolon\;";
```

Также следует экранировать обратную косую черту, поскольку она работает как маска.

9.3.1.6 Ключевые слова модификатора

Некоторые ключевые слова выступают в качестве модификаторов. Есть два типа модификаторов.

Старый стиль «модификаторов контента»:

```
alert http any any -> any any (content:"index.php"; http_uri; sid:1;)
```

В приведенном выше примере шаблон index.php изменен для проверки буфера HTTP URI.

Более поздний тип называется «липким буфером». Сначала он помещает имя буфера, и все ключевые слова, следующие за ним, применяются к этому буферу, например:

```
alert http any any -> any any (http_response_line; content:"403 Forbidden"; sid:1;)
```

В приведенном выше примере шаблон «403 Forbidden» проверяется по строке ответа HTTP, поскольку он следует за ключевым словом http_response_line.

9.3.1.7 Нормализованные буферы

Пакет состоит из необработанных данных. HTTP и повторная сборка делают копии таких пакетов. Они стирают аномальный контент, объединяют пакеты и так далее. То, что остается, называется «нормализованным буфером».

Нормализованные буферы включают все HTTP-ключевые слова, повторно собранные потоки, TLS-, SSL-, SSH-, FTP- и DCE/RPC-буферы.

9.3.2 Мета ключевые слова

Мета-настройки не влияют на проверку Suricata; они влияют на то, как Suricata предупреждает о событиях.

9.3.2.1 msg (сообщение)

Ключевое слово msg используется для вывода текстовой информации о сигнатуре и возможных оповещений.

Формат сообщения:

```
msg: "some Описание";
```

Примеры:

```
msg:"ATTACK-RESPONSES 403 Forbidden";
msg:"ET EXPLOIT SMB-DS DCERPC PnP bind attempt";
```

Продолжая пример из предыдущего раздела:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

По соглашению первую часть сигнатуры пишут заглавными и показывают класс сигнатуры. Также принято, чтобы msg было первым ключевым словом сигнатуры.

Следующие символы должны быть экранированы внутри: ; \ "

9.3.2.2 sid

Ключевое слово sid присваивает каждой сигнатуре свой собственный идентификатор. Этот идентификатор указывается с помощью номера. Формат sid:

```
sid:123;
```

Пример sid в сигнатуре:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

По соглашению sid указывается в качестве последнего ключевого слова (или второго с конца, если есть rev) сигнатуры.

9.3.2.3 rev

Ключевое слово sid почти каждый раз сопровождается rev. Rev представляет версию сигнатуры. Если сигнатура изменена, число rev будет увеличиваться как написано в сигнатуре. Формат rev:

```
rev:123;
```

Пример rev в сигнатуре:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

По соглашению, sid предшествует rev, и оба идут после всех ключевых слов.

9.3.2.4 gid

Ключевое слово gid может использоваться, чтобы дать различным группам сигнатур другое значение идентификатора (как в sid). Suricata по умолчанию использует gid 1. Это можно изменить, хотя изменение может иметь технические последствия, заметить можно только в предупреждении.

Пример гид в оповещении fast.log. В части [1:2008124:2], 1 – gid (2008124 – sid и 2 rev).

```
10/15/09-03:30:10.219671 [**] [1:2008124:2] ET TROJAN Likely Bot Nick in IRC (USA +.) [**] [Classification: A Network Trojan was Detected] [Priority: 3] {TCP} 192.168.1.42:1028 -> 72.184.196.31:6667
```

9.3.2.5 classtype

Ключевое слово classtype предоставляет информацию о классификации правил и оповещений. Он состоит из короткого имени, длинного имени и приоритета. Например, можно определить является ли правило только информационным или имеет отношение к хаку и так далее. Для каждого типа классификации есть приоритет, который будет использоваться в правиле.

Пример определения classtype:

```
config classification: web-application-attack,Web Application Attack,1
config classification: not-suspicious,Not Suspicious Traffic,3
```

Например, правило с типом класса «Атака веб-приложения» будет иметь приоритет 1, а оповещение будет содержать «Атака веб-приложения» (см. таблицу 8).

Таблица 8 – Описание типов трафика

Тип класса	Оповещение	Приоритет
web-application-attack	Атака веб-приложения	1
not-suspicious	Неподозрительный трафик	3

Пример ниже также относится к типам класса «троянская активность»:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

По соглашению, classtype предшествует sid, rev и остальным ключевым словам.

9.3.2.6 reference

Ключевое слово reference указывает на места, где можно найти информацию о сигнатуре и о проблеме, которую пытается решить сигнатура. Reference может появляться в сигнатуре несколько раз. Это ключевое слово предназначено для авторов сигнатур и аналитиков, которые исследуют, почему сигнатура совпадает. Формат:

```
reference: type, reference
```

Типичной ссылкой на www.info.com будет:

```
reference: url, www.info.com
```

Тем не менее, есть также несколько систем, которые можно использовать как справочные. Общеизвестным примером является база данных CVE, которая присваивает номера уязвимостям. Чтобы запретить вам вводить один и тот же URL-адрес снова и снова, можно использовать что-то вроде этого:

```
reference: cve, CVE-2014-1234
```

Это будет ссылка на **http://cve.mitre.org/cgi-bin/cvename .cgi? name = CVE-2014-1234**. Все ссылочные типы определены в файле конфигурации reference.config.

В примере также есть ссылка:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +.)"; flow:established,to_server; flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/i"; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

9.3.2.7 priority

Ключевое слово priority имеет обязательное числовое значение, которое может варьироваться от 1 до 255. Чаще всего используются числа от 1 до 4. Сигнатуры с более высоким приоритетом будут проверены в первую очередь. Наивысший приоритет – 1. Обычно сигнатура уже имеют приоритет, присвоенный через classtype, который может быть переопределен с помощью данного ключевого слова. Формат приоритета:

```
priority:1;
```

9.3.2.8 metadata

Ключевое слово metadata позволяет добавлять дополнительную, нефункциональную информацию к сигнатуре. Хотя формат имеет произвольную форму, рекомендуется придерживаться пары «ключ-значение», поскольку Suricata может включать их в оповещения. Формат:

```
metadata: key value;
metadata: key value, key value;
```

9.3.2.9 target

Ключевое слово target позволяет составителю правил указать, какая сторона оповещения является целью атаки. Если указано, событие оповещения расширяется, чтобы включать информацию об источнике и цели.

Формат:

```
target:[src_ip|dest_ip]
```

Если значением является src_ip, то исходный IP-адрес в сгенерированном событии (поле src_ip в JSON) является целью атаки. Если для target задано значение dest_ip, то целью является IP-адрес цели в сгенерированном событии.

9.3.3 Ключевые слова IP

9.3.3.1 ttl

Ключевое слово ttl используется для проверки конкретного значения времени жизни IP в заголовке пакета. Формат:

```
ttl:<number>
```

Например:

```
ttl:10;
```

В конце ключевого слова ttl можно указать значение для проверки соответствия. Значение времени жизни определяет максимальное время, в течение которого пакет может находиться в интернет-системе. Если это поле установлено в 0, то пакет должен быть уничтожен. Время жизни зависит от количества переходов. Каждый переход / маршрутизатор, через который проходит пакет, вычитает один из счетчиков TTL пакета. Целью этого механизма является ограничение существования пакетов, чтобы пакеты не могли попасть в бесконечные циклы маршрутизации.

Пример ключевого слова ttl в правиле:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL MISC 0 ttl"; ttl:0;
reference:url,support.microsoft.com/default.aspx?scid=kb#-#-EN-US#-#-q138268;
reference:url,www.isi.edu/in-notes/rfc1122.txt; classtype:misc-activity; sid:2101321; rev:9;)
```

9.3.3.2 ipopts

С помощью ключевого слова ipopts можно проверить, установлена ли определенная опция IP. ipopts должен использоваться в начале правила. Можно найти только один вариант для каждого правила. Есть несколько вариантов, которые могут быть сопоставлены, как указано в таблице 9.

Таблица 9 – Варианты для сопоставления опций IP

Опция IP	**Описание**
rr	Запись маршрута
eol	Конец списка
nop	Нет опций

ts	Штамп времени
sec	IP-безопасность
esec	Расширенная IP-безопасность
lsrr	Маршрутизация по свободным источникам
ssrr	Строгая маршрутизация от источника
satid	Идентификатор потока
any	Любые установленные IP-опции

Формат ключевого слова ipopts:

```
ipopts: <name>
```

Например:

```
ipopts: lsrr;
```

Пример ipopts в правиле:

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL MISC source route ssrr"; ipopts:ssrr; reference:arachnids,422; classtype:bad-unknown; sid:2100502; rev:3;)
```

9.3.3.3 sameip

Каждый пакет имеет IP-адрес источника и IP-адрес назначения. Может быть так, что IP-адрес источника совпадает с IP-адресом назначения. С помощью ключевого слова sameip можно проверить, совпадает ли IP-адрес источника с IP-адресом получателя. Формат ключевого слова sameip:

```
sameip;
```

Пример sameip в правиле:

```
alert ip any any -> any any (msg:"GPL SCAN same SRC/DST"; sameip; reference:bugtraq,2666; reference:cve,1999-0016; reference:url,www.cert.org/advisories/CA-1997-28.html; classtype:bad-unknown; sid:2100527; rev:9;)
```

9.3.3.4 ip_proto

С ключевым словом ip_proto можно сопоставить по IP-протоколу в заголовке пакета. Можно использовать имя или номер протокола. Можно сопоставить по следующим протоколам:

- 1 – ICMP – Internet Control Message;
- 6 – TCP – Transmission Control Protocol;
- 17 – UDP – User Datagram;
- 47 – GRE – General Routing Encapsulation;
- 50 – ESP – Encap Security Payload для IPv6;
- 51 – AH – Authentication Header для Ipv6;
- 58 – IPv6-ICMP – ICMP для Ipv6.

Пример ip_proto в правиле:

```
alert ip any any -> any any (msg:"GPL MISC IP Proto 103 PIM"; ip_proto:103; reference:bugtraq,8211; reference:cve,2003-0567; classtype:non-standard-protocol; sid:2102189; rev:4;)
```

Вариант этого примера с именем:

```
ip_proto:PIM
```

9.3.3.5 id

С помощью ключевого слова id можно сопоставить конкретное значение IP ID. Идентификатор идентифицирует каждый пакет, отправленный хостом, и обычно увеличивается на единицу с каждым отправляемым пакетом. Идентификатор IP используется в качестве идентификационного номера фрагмента. Каждый пакет имеет IP-идентификатор, и когда пакет становится фрагментированным, все фрагменты этого пакета имеют одинаковый идентификатор. Таким образом, получатель пакета знает, какие фрагменты принадлежат одному и тому же пакету. (IP ID не заботится о порядке, в этом случае используется смещение, уточняющее порядок фрагментов).

Формат id:

```
id:<number>;
```

Пример id в правиле:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET DELETED F5 BIG-IP 3DNS TCP Probe 1"; id: 1; dsize: 24; flags: S,12; content:"|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|"; window: 2048; reference:url,www.f5.com/f5products/v9intro/index.html; reference:url,doc.emergingthreats.net/2001609; classtype:misc-activity; sid:2001609; rev:13;)
```

9.3.3.6 geoip

Ключевое слово geoip позволяет сопоставлять адреса источника, места назначения сетевого трафика и определять, к какой стране он принадлежит. Чтобы сделать это, Suricata использует GeoIP2 API MaxMind.

Синтаксис geoip:

```
geoip: src,RU;
geoip: both,CN,RU;
geoip: dst,CN,RU,IR;
geoip: both,US,CA,UK;
geoip: any,CN,IR;
```

Можно использовать следующие опции (см. таблицу 10), чтобы сопоставить по направлению.

Таблица 10 – Опции для сопоставления по гео-направлению

Опция	**Описание**
both	Оба направления должны соответствовать данному geoip
any	Одно из направлений должно соответствовать данному geoip
dest	Если пункт назначения совпадает с заданным geoip

src	Источник совпадает с данным geoip

Ключевое слово поддерживает только IPv4. Поскольку использует GeoIP2 API MaxMind, libmaxminddb должен быть скомпилирован. Следует загрузить и установить нужные версии базы данных GeoIP2 или GeoLite2. Посетите сайт MaxMind по адресу **https://dev.maxmind.com/geoip/geoip2/geolite2/** для получения подробной информации.

Также необходимо указать расположение файла базы данных GeoIP2 или GeoLite2 в локальной системе в конфигурационном файле YAML. Например:

```
geoip-database: /usr/local/share/GeoIP/GeoLite2-Country.mmdb
```

9.3.3.7 fragbits

С помощью ключевого слова fragbits можно проверить, установлена ли фрагментация и зарезервированные биты в заголовке IP. Ключевое слово fragbits должно быть помещено в начало правила. Fragbits используется для модификации механизма фрагментации. Во время маршрутизации сообщений может случиться, что пакет больше максимального размера, который может обработать сеть. В этом случае пакет может быть отправлен фрагментарно. Максимальный размер пакета называется максимальной единицей передачи (MTU).

Для сопоставления можно использовать следующие биты:

- M – больше фрагментов;
- D – нет фрагментов;
- R – зарезервировано.

Соответствие этим битам может быть определено с помощью следующих модификаторов:

- + сопоставление с указанными битами плюс любые другие;
- * совпадение, если установлены какие-либо указанные биты;
- ! совпадение, если указанные биты не установлены.

Формат:

```
fragbits:[*+!]<[MDR]>;
```

Пример fragbits в правиле:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"ET EXPLOIT Invalid non-fragmented packet with fragment offset>0"; fragbits: M; fragoffset: >0; reference:url,doc.emergingthreats.net/bin/view/Main/2001022; classtype:bad-unknown; sid:2001022; rev:5; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

9.3.3.8 fragoffset

Ключевое слово fragoffset позволяет сопоставлять конкретные десятичные значения поля смещения фрагмента IP. Если следует проверить первые фрагменты сеанса, то можно объединить fragoffset 0 с опцией More Fragment. Поле смещения фрагмента удобно для повторной сборки. Идентификатор используется для определения того, какие фрагменты принадлежат какому пакету, а поле смещения уточняет порядок фрагментов.

Можно использовать следующие модификаторы:

- < совпадает, если значение меньше указанного значения;
- > совпадает, если значение больше указанного значения;
- ! совпадает, если указанное значение отсутствует.

Формат fragoffset:

```
fragoffset:[!|<|>]<number>;
```

Пример fragoffset в правиле:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”ET EXPLOIT Invalid non-fragmented packet with fragment offset>0”; fragbits: M; fragoffset: >0; reference:url,doc.emergingthreats.net/bin/view/Main/2001022; classtype:bad-unknown; sid:2001022; rev:5; metadata:created_at 2010_07_30, updated_at 2010_07_30;)
```

9.3.3.9 tos

Ключевое слово tos может соответствовать определенным десятичным значениям поля TOS IP-заголовка. Ключевое слово tos может иметь значение от 0 до 255.

Формат tos:

```
tos:[!]<number>;
```

Пример tos в правиле:

```
alert ip any any -> any any (msg:”Differentiated Services Codepoint: Class Selector 1 (8)”; flow:established; tos:8; classtype:not-suspicious; sid:2600115; rev:1;)
```

Пример tos с отрицательными значениями:

```
alert ip any any -> any any (msg:”TGI HUNT non-DiffServ aware TOS setting”; flow:established,to_server; tos:!0; tos:!8; tos:!16; tos:!24; tos:!32; tos:!40; tos:!48; tos:!56; threshold:type limit, track by_src, seconds 60, count 1; classtype:bad-unknown; sid:2600124; rev:1;)
```

9.3.4 Ключевые слова TCP

9.3.4.1 seq

Ключевое слово seq можно использовать в сигнатуре для проверки определенного порядкового номера TCP. Порядковый номер – это число, которое генерируется практически случайным образом обеими конечными точками TCP-соединения. Клиент и сервер создают порядковый номер, который увеличивается на единицу с каждым отправляемым байтом. Этот порядковый номер отличается для обеих сторон и должен быть подтвержден обеими сторонами соединения. С помощью порядковых номеров TCP обрабатывает подтверждение, заказ и повторную передачу. Номер увеличивается с каждым байтом данных, отправленным отправителем. Ключевое слово seq помогает отслеживать, к какому месту в потоке данных принадлежит байт. Если флаг SYN установлен в 1, то порядковый номер первого байта данных равен этому числу плюс 1 (т. е. 2).

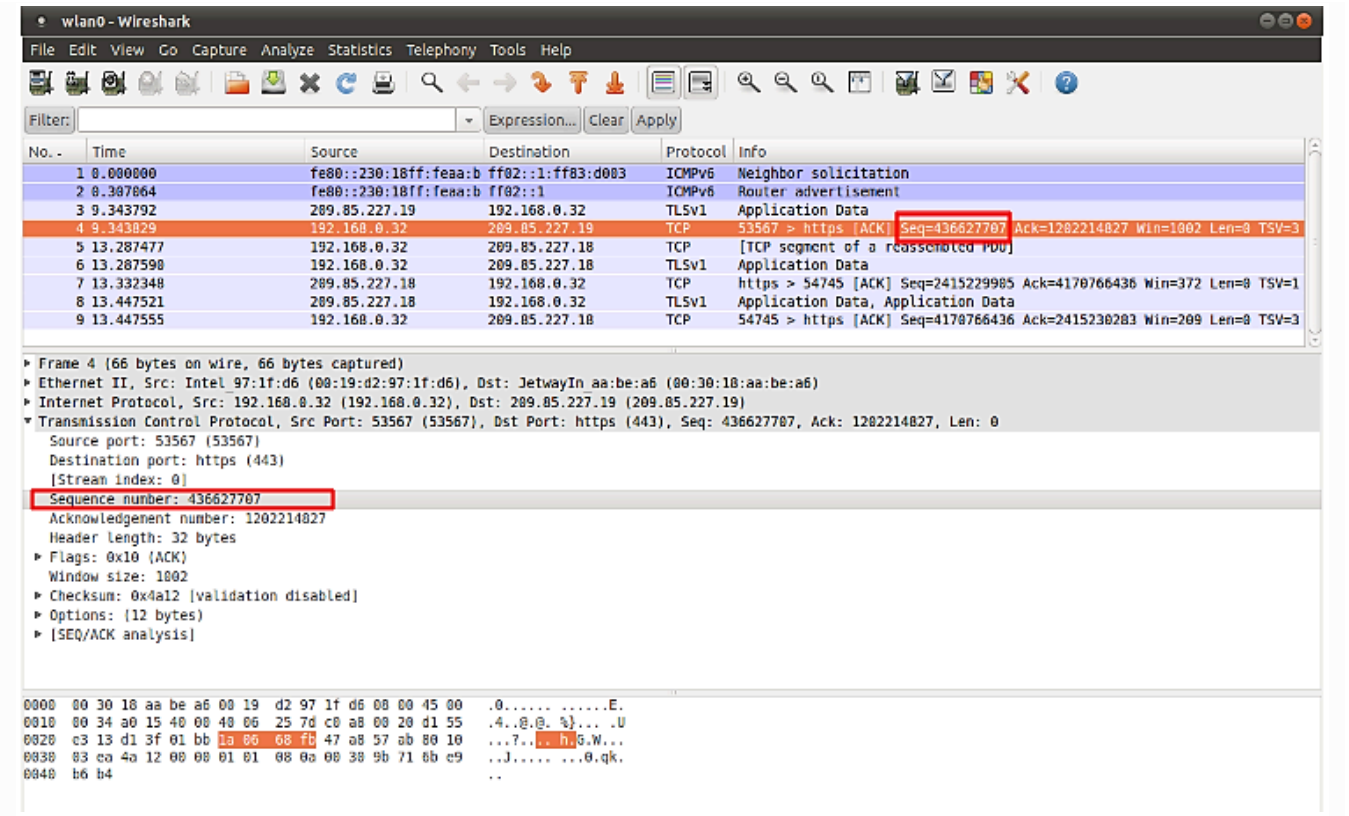
Пример:

```
seq:0;
```

Пример seq в сигнатуре:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:”GPL SCAN NULL”; flow:stateless; ack:0; flags:0; seq:0; reference:arachnids,4; classtype:attempted-recon; sid:2100623; rev:7;)
```

Пример seq в пакете (Wireshark) представлен ниже.



Пример seq в пакете

9.3.4.2 ack

Ack – это подтверждение получения всех предыдущих байтов (данных), отправленных другой стороной TCP-соединения. В большинстве случаев каждый пакет TCP-соединения имеет флаг ACK после первого SYN и номер подтверждения, который увеличивается с получением каждого нового байта данных. Ключевое слово ack можно использовать в сигнатуре для проверки определенного номера подтверждения TCP.

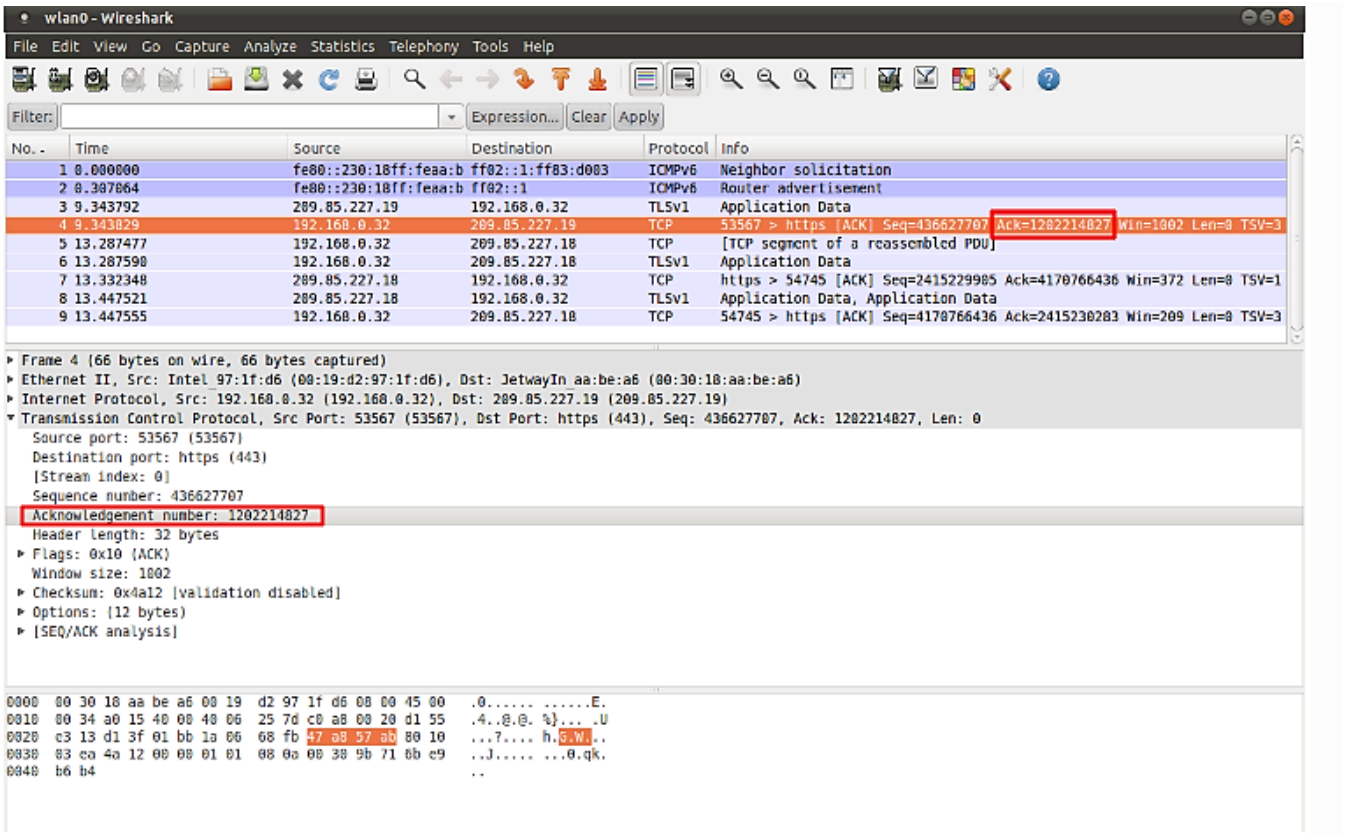
Формат ack:

```
ack:1;
```

Пример ack в сигнатуре:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL SCAN NULL"; flow:stateless; ack:0; flags:0; seq:0; reference:arachnids,4; classtype:attempted-recon; sid:2100623; rev:7;)
```

Пример ack в пакете (Wireshark) представлен ниже.



Пример ack в пакете

9.3.4.3 window

Ключевое слово window используется для проверки определенного размера окна TCP. Размер окна TCP – это механизм, который контролирует поток данных. Окно устанавливается получателем (размер окна, объявленный получателем) и указывает количество байтов, которое может быть получено. Объем данных должен быть подтвержден получателем в первую очередь, прежде чем отправитель сможет отправить такое же количество новых данных. Данный механизм используется для предотвращения переполнения. Значение размера окна ограничено и может составлять от 2 до 65 535 байт. Чтобы полнее использовать пропускную способность, можно использовать большее TCP-окно.

Формат ключевого слова window:

```
window:[!]<number>;
```

Пример window в правиле:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL DELETED typot trojan traffic"; flow:stateless; flags:S,12; window:55808; reference:mcafee,100406; classtype:trojan-activity; sid:2182; rev:8;)
```

9.3.4.4 tcp.mss

Соответствует значению опции TCP MSS. Не будет совпадать, если опция отсутствует.

Формат ключевого слова:

```
tcp.mss:<min>-<max>;
tcp.mss:[<|>]<number>;
tcp.mss:<value>;
```

Пример правила:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (flow:stateless; flags:S,12; tcp.mss:<536; sid:1234; rev:5;)
```

9.3.4.5 tcp.hdr

Липкий буфер для соответствия целому заголовку TCP.

Пример правила:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET any (flags:S,12; tcp.hdr; content:"|02 04|"; offset:20; byte_test:2,<,536,0,big,relative; sid:1234; rev:5;)
```

Этот пример начинает искать после фиксированной части заголовка, то есть в опциях переменного размера. Там будет искаться опция MSS (тип 2, опция len 4) и, используя byte_test, определять, будет ли значение опции меньше 536. Опция *tcp.mss* будет более эффективной, данное ключевое слово предназначено для использования в случаях, когда другие варианты недоступны.

9.3.5 Ключевые слова UDP

9.3.5.1 udp.hdr

Липкий буфер для соответствия всему заголовку UDP.

Пример правила:

```
alert udp any any -> any any (udp.hdr; content:"|00 08|"; offset:4; depth:2; sid:1234; rev:5;)
```

Этот пример определяет соответствие по полю длины заголовок UDP. В этом случае длина 8 означает, что полезной нагрузки нет. Также можно сопоставить с помощью *dsize: 0*.

9.3.6 Ключевые слова ICMP

ICMP является частью IP, который сам по себе не является надежным, когда речь идет о доставке данных (датаграмма). ICMP дает обратную связь в случае возникновения проблем. Это не предотвращает проблемы, но помогает понять, что пошло не так и где. Если необходима надежность, протоколы, использующие IP, должны сами позаботиться о надежности. В разных ситуациях будут отправляться ICMP-сообщения. Например, когда пункт назначения недоступен, если не хватает емкости буфера для пересылки данных, или, когда дейтаграмма отправляется фрагментировано в тех случаях, когда это неприемлемо, и так далее.

В сообщении ICMP имеется четыре важных вида контента, которые можно сопоставить с соответствующими ключевыми словами ICMP: тип, код, идентификатор и последовательность сообщения.

9.3.6.1 itype

Ключевое слово itype предназначено для сопоставления с конкретным типом ICMP (числом). ICMP имеет несколько видов сообщений и использует коды для уточнения этих сообщений. Разные сообщения различаются по именам, но важнее по числовым значениям. Для получения дополнительной информации см. таблицу 11 с типами сообщений и кодами.

Формат ключевого слова itype:

```
itype:min<>max;
itype:[<|>]<number>;
```

Пример, который ищет тип ICMP больше 10:

```
itype:>10;
```

Пример ключевого слова itype в сигнатуре:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL SCAN Broadscan Smurf Scanner"; dsize:4; icmp_id:0; icmp_seq:0; itype:8; classtype:attempted-recon; sid:2100478; rev:4;)
```

Ниже перечислены все типы ICMP, известные на момент написания. Самую актуальную таблицу можно найти **на веб-сайте IANA.**

Таблица 11 – Типы сообщений и коды

Тип ICMP	Наименование
0	Эхо-ответ
3	Адресат недоступен
4	Источник источника
5	Перенаправление
6	Альтернативный адрес хоста
8	Эхо
9	Объявление маршрутизатора
10	Запрос маршрутизатора
11	Превышено время
12	Проблема параметра
13	Отметка времени
14	Ответ с отметкой времени
15	Запрос информации
16	Ответ с информацией
17	Запрос маски адреса
18	Ответ маски адреса
30	Трассировка маршрута
31	Ошибка преобразования дейтаграммы
32	Перенаправление мобильного хоста
33	IPv6 Где вы
34	IPv6 Я здесь

Тип ICMP	Наименование
35	Запрос мобильной регистрации
36	Ответ мобильной регистрации
37	Запрос доменного имени
38	Ответ доменного имени
39	Пропустить
40	Photuris (протокол управления сеансовыми ключами)
41	Протоколы экспериментальной мобильности, такие как Seamoby

9.3.6.2 icode

С ключевым словом icode можно сопоставить определенный код ICMP. Код сообщения ICMP уточняет сообщение и вместе с ICMP-типом указывает на тип возникшей проблемы. Код имеет различное назначение для каждого типа ICMP.

Формат ключевого слова icode:

```
icode:min<>max;
icode:[<|>]<number>;
```

Данный пример ищет код ICMP больше 5:

```
icode:>5;
```

Пример ключевого слова icode в правиле:

```
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"GPL MISC Time-To-Live Exceeded in Transit"; icode:0; itype:11; classtype:misc-activity; sid:2100449; rev:7;)
```

Ниже перечислены значения всех типов ICMP (см. таблицу 12). Если код отсутствует в списке, в таблице 12 определен тип 0, который имеет значение кода ICMP. Свежую таблицу можно найти **на веб-сайте IANA.**

Таблица 12 – Значения всех типов ICMP

Код ICMP	Тип ICMP	Описание
3	0	Нет доступа к сети
	1	Нет доступа к хосту
	2	Нет доступа к протоколу
	3	Порт недоступен
	4	Необходима фрагментация и не задан фрагмент
	5	Исходный маршрут не пройден

Код ICMP	Тип ICMP	Описание
	6	Сеть назначения неизвестна
	7	Хост назначения неизвестен
	8	Исходный хост изолирован
	9	Связь с целевой сетью административно запрещена
	10	Связь с конечным хостом административно запрещена
	11	Целевая сеть недоступна для типа услуг
	12	Целевой хост недоступен для типа услуг
	13	Связь административно запрещена
	14	Нарушение приоритета хоста
	15	Действует прерывание приоритета
5	0	Перенаправление дейтаграммы для сети (или подсети)
	1	Перенаправление дейтаграммы для хоста
	2	Перенаправление дейтаграммы для типа обслуживания и сети
	3	Перенаправление дейтаграммы для типа обслуживания и хоста
9	0	Обычное объявление маршрутизатора
	16	Не маршрутизируется общий трафик
11	0	При пересылке превышено время жизни
	1	Время сборки фрагмента превышено
12	0	Указатель указывает на ошибку
	1	Отсутствует обязательный параметр
	2	Ошибка длины
40	0	Ошибка SPI
	1	Ошибка аутентификации
	2	Ошибка декомпрессии
	3	Ошибка дешифрования
	4	Требуется аутентификация

Код ICMP	Тип ICMP	Описание
	5	Требуется авторизация

9.3.6.3 icmp_id

С ключевым словом icmp_id можно сопоставить конкретные значения ICMP-идентификатора. Каждый ICMP-пакет получает идентификатор при отправке. В момент получения пакет отправляется ответ, используя тот же идентификатор, чтобы отправитель узнал его и связал с ICMP-запросом.

Формат ключевого слова icmp_id:

```
icmp_id:<number>;
```

Пример, который ищет ICMP-идентификатор 0:

```
icmp_id:0;
```

Пример ключевого слова icmp_id в правиле:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:”GPL SCAN Broadscan Smurf Scanner”; dsize:4; icmp_id:0; icmp_seq:0; itype:8; classtype:attempted-recon; sid:2100478; rev:4;)
```

9.3.6.4 icmp_seq

Можно использовать ключевое слово icmp_seq для проверки порядкового номера ICMP. Все сообщения ICMP имеют порядковые номера. Оно может использоваться (вместе с идентификатором) для проверки того, какое ответное сообщение принадлежит какому сообщению запроса.

Формат ключевого слова icmp_seq:

```
icmp_seq:<number>;
```

Пример, который ищет последовательность ICMP 0:

```
icmp_seq:0;
```

Пример icmp_seq в правиле:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:”GPL SCAN Broadscan Smurf Scanner”; dsize:4; icmp_id:0; icmp_seq:0; itype:8; classtype:attempted-recon; sid:2100478; rev:4;)
```

9.3.7 Ключевые слова полезной нагрузки

Ключевые слова полезной нагрузки проверяют содержимое полезной нагрузки пакета или потока.

9.3.7.1 content

Ключевое слово content используется для сопоставления содержимого сигнатур, указанного между кавычками. Самый простой формат:

```
content: ".....";
```

Данное ключевое слово можно несколько раз использовать в сигнатуре.

Содержимое сопоставляется по байтам. Есть 256 различных значений байта (0-255). Можно проверять соответствие для всех символов от a до z, в верхнем и нижнем регистре, а также всех специальных знаков. Но не все байты являются печатными символами. Для этих байтов используются шестнадцатеричные обозначения. Многие языки программирования используют 0x00 в качестве нотации, где 0x означает, что это касается двоичного значения, однако язык правил использует | 00 | в качестве обозначения. Это обозначение также может использоваться для печатных символов.

Пример:

```
|61| is a
|61 61| is aa
|41| is A
|21| is !
|0D| is carriage return
|0A| is line feed
```

Есть символы, которые нельзя использовать для проверки содержимого, потому что они играют важную роль в сигнатуре. Для сопоставления по этим символам следует использовать шестнадцатеричное обозначение:

```
"    |22|
;    |3B|
:    |3A|
|    |7C|
```

Соглашение предписывает записывать шестнадцатеричные обозначения в верхнем регистре.

Например, чтобы написать http:// в содержимом сигнатуры, следует написать это так: content: "http | 3A | //"; Если вы используется шестнадцатеричное обозначение в сигнатуре, убедитесь, что оно помещено между вертикальными чертами. В противном случае нотация будет восприниматься буквально как часть содержимого.

Несколько примеров:

```
content:"a|0D|bc";
content:"|61 0D 62 63|";
content:"a|0D|b|63|";
```

Можно разрешить сигнатуре проверять всю полезную нагрузку на соответствие содержимому или разрешать проверять определенные части полезной нагрузки. Если ничего особенного не добавить к сигнатуре, то она попытается найти совпадение во всех байтах полезной нагрузки.

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

По умолчанию сопоставление с образцом чувствительно к регистру. Содержание должно быть точным, иначе совпадения не будет.

Также можно использовать символ ‘!’ для исключений в содержимом.

Например:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"Outdated Firefox on Windows"; content:"User-Agent|3A| Mozilla/5.0 |28|Windows|3B| ";
```

```
content:"Firefox/3."; distance:0; content:!"Firefox/3.6.13";
distance:-10; sid:9000000; rev:1;)
```

Для content:!"Firefox/3.6.13"; будет сгенерировано оповещение, если версия Firefox отлична от 3.6.13.

Следует исключить следующие символы внутри содержимого: ; \ ".

9.3.7.2 nocase

Если не следует вводить различия между прописными и строчными буквами, можно использовать nocase. Ключевое слово nocase – это модификатор содержимого.

Формат данного ключевого слова:

```
nocase;
```

Его следует поместить после содержимого, которое необходимо изменить, например:

```
content: "abc"; nocase;
```

Оно не влияет на другое содержимое в сигнатуре.

9.3.7.3 depth

Ключевое слово depth является абсолютным модификатором содержимого, его указывают после содержимого. Модификатор глубины содержимого имеет обязательное числовое значение, например:

```
depth:12;
```

Число после depth определяет сколько байтов после начала полезной нагрузки будет проверено.

9.3.7.4 startswith

Ключевое слово startswith похоже на depth. Оно не требует аргументов и должно следовать за ключевым словом content. Оно изменяет content так, чтобы он точно совпадал с началом буфера.

Пример:

```
content:"GET|20|"; startswith;
```

Краткая запись startswit:

```
content:"GET|20|"; depth:4; offset:0;
```

startswith не может смешиваться с depth, offset, within или distance в одном шаблоне.

9.3.7.5 endswith

Ключевое слово endswith похоже на isdataat:!1,relative;. Оно не требует аргументов и должно следовать за ключевым словом content. Оно изменяет content так, чтобы он точно совпадал с концом буфера.

Пример:

```
content:".php"; endswith;
```

Краткая запись endswith:

```
content:".php"; isdatat:!1,relative;
```

endswith не может смешиваться с offset, within или distance в одном шаблоне.

9.3.7.6 offset

Ключевое слово offset указывает с какого байта полезная нагрузка будет проверяться на соответствие. Например, для смещения 3; будет проверен четвертый байт и далее.

Ключевые слова offset и depth могут быть объединены и часто используются вместе.

Например:

```
content:"def"; offset:3; depth:3;
```

Такая сигнатура проверяла бы полезную нагрузку с третьего до шестого байта.

9.3.7.7 distance

Ключевое слово distance является относительным модификатором содержимого. Оно указывает на связь между ключевым словом content и предшествующим ему содержимым. Distance имеет значение после предыдущего совпадения. Ключевое слово distance имеет обязательное числовое значение, определяющее байт в полезной нагрузке, с которого будет проводиться сопоставление относительно предыдущего совпадения. Расстояние только определяет, где Suricata начнет искать шаблон. Итак, расстояние: 5; означает, что шаблон может быть где угодно после предыдущего совпадения +5 байт. Для ограничения области поиска совпадения после последнего совпадения, используйте ключевое слово within.

Distance также может быть отрицательным числом. Оно может использоваться для проверки совпадений частично в том же самом содержимом (см. пример) или даже в содержимом до него. Хотя это не очень часто используется, так как того же результата можно достичь с помощью других ключевых слов.

9.3.7.8 within

Ключевое слово within относится к предыдущему совпадению, оно имеет обязательное числовое значение. Использование within гарантирует, что совпадение будет только в том случае, если содержимое совпадает с полезной нагрузкой в пределах установленного количества байтов. Within не может быть 0.

Второе содержимое должно находиться в пределах 3 байт от первого содержимого.

Как упоминалось ранее, distance и within могут очень удобно комбинироваться в сигнатуре.

9.3.7.9 isdataat

Ключевое слово isdataat предназначено для проверки наличия данных в определенной части полезной нагрузки. Ключевое слово начинается с числа (позиция), затем следует опция relative и отделенная запятой опциональная часть необработанных данных. Используйте слово ‘relative’, чтобы определить, есть ли еще данные в определенной части полезной нагрузки относительно последнего совпадения.

Таким образом, можно использовать оба примера:

```
isdataat:512;

isdataat:50, relative;
```

Первый пример иллюстрирует сигнатуру, которая ищет 512-й байт полезной нагрузки. Второй пример иллюстрирует поиск сигнатуры для байта 50 после последнего совпадения.

Также можно использовать отрицание (!) перед isdataat.

9.3.7.10 dsize

С помощью ключевого слова dsize можно сопоставить размер полезной нагрузки пакета. Можно использовать это ключевое слово, например, чтобы искать аномальные размеры полезных нагрузок, что может быть удобно для обнаружения переполнений буфера.

Формат:

```
dsize:<number>;
```

Пример dsize в правиле:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 65535 (msg:"GPL DELETED EXPLOIT LANDesk Management Suite Alerting Service buffer overflow"; dsize:>268; reference: bugtraq,23483; reference: cve,2007-1674; classtype: attempted-admin; sid:100000928; rev:1;)
```

9.3.7.11 byte_test

Ключевое слово byte_test извлекает <num of bytes> и выполняет операцию выбранную в <operator> со значением в <test value> с определенным <offset>.

Формат (значения параметров представлены в таблице 13):

```
byte_test:<num of bytes>, [!]<operator>, <test value>, <offset> [,relative] \
[,<endian>][, string, <num type>][, dce][, bitmask <bitmask value>];
```

Таблица 13 – Значения параметров ключевого слова byte_test

<num of bytes>	Число байт, выбранных из пакета для преобразования
<operator>	- [!] Отрицание, может предшествовать оператору - < меньше чем - > больше чем - = равно - <= меньше или равно - >= больше или равно - & побитовое И - ^ побитовое ИЛИ
<value>	Значение для проверки преобразованного значения по отношению к [принятому шестнадцатеричному или десятичному]
<offset>	Число байтов в полезной нагрузке
[relative]	Смещение относительно последнего совпадения содержимого
[endian]	Тип читаемого числа: - больший (старший значащий байт по меньшему адресу) - меньший (старший значащий байт по большему адресу)

[string] <num>	– hex – Преобразование строки в шестнадцатеричный формат – dec – Преобразование строки в десятичный формат – oct – Преобразование строки в восьмеричный формат
[dce]	Разрешить DCE модуль определения порядка байтов
[bitmask]	Применяет оператор AND к преобразованным байтам.

Пример:

```
alert tcp any any -> any any \
    (msg:"Byte_Test Пример – Num = Value"; \
    content:"|00 01 00 02|"; byte_test:2,=,0x01;)

alert tcp any any -> any any \
    (msg:"Byte_Test Пример – Num = Value relative to content"; \
    content:"|00 01 00 02|"; byte_test:2,=,0x03,relative;)

alert tcp any any -> any any \
    (msg:"Byte_Test Пример – Num != Value"; content:"|00 01 00 02|"; \
    byte_test:2,!=,0x06;)

alert tcp any any -> any any \
    (msg:"Byte_Test Пример – Detect Large Values"; content:"|00 01 00 02|"; \
    byte_test:2,>,1000,relavtive;)

alert tcp any any -> any any \
    (msg:"Byte_Test Пример – Lowest bit is set"; \
    content:"|00 01 00 02|"; byte_test:2,&,0x01,relative;)

alert tcp any any -> any any (msg:"Byte_Test Пример – Compare to String"; \
    content:"foobar"; byte_test:4,=,1337,1,relative,string,dec;)
```

9.3.7.12 byte_jump

Ключевое слово byte_jump позволяет возможность выбора <num of bytes> по смещению <offset> и перемещает указатель обнаружения на эту позицию. Совпадение содержимого будет основано на новой позиции.

Формат (значения параметров представлены в таблице 14):

```
byte_jump:<num of bytes>, <offset> [, relative][, multiplier <mult_value>] \
    [, <endian>][, string, <num_type>][, align][, from_beginning][, from_end] \
    [, post_offset <value>][, dce][, bitmask <value>];
```

Таблица 14 – Значения параметров ключевого слова byte_jump

<num of bytes>	Число байт, выбранных из пакета для преобразования
<offset>	Количество байтов в полезной нагрузке

[relative]	Смещение относительно последнего совпадения содержимого
[multiplier] <value>	Умножение преобразованных байт на <value>
[endian]	– больший (старший значащий байт по меньшему адресу) – меньший (старший значащий байт по большему адресу)
[string] <num_type>	– hex – Преобразование данных в шестнадцатеричный формат – dec – Преобразование данных в десятичный формат – oct – Преобразование данных в восьмеричный формат
[align]	Округляет число до следующей 32-битной границы
[from_beginning]	Переходит вперед от начала пакета вместо того, где установлен указатель обнаружения
[from_end]	Переход начнется в конце полезной нагрузки, а не там, где установлена точка обнаружения
[post_offset] <value>	После выполнения операции перехода будет выполнен переход на дополнительное количество байтов, указанное в <value>
[dce]	Разрешить модулю DCE определять порядок байтов
[bitmask] <value>	Оператор AND будет применяться к <value> и преобразованными байтами, затем выполняется операция перехода

Пример:

```
alert tcp any any -> any any \
    (msg:"Byte_Jump Пример"; \
    content:"Alice"; byte_jump:2,0; content:"Bob";)

alert tcp any any -> any any \
    (msg:"Byte_Jump Multiple Jumps"; \
    byte_jump:2,0; byte_jump:2,0,relative; content:"foobar"; distance:0; within:6;)

alert tcp any any -> any any \
    (msg:"Byte_Jump From the End -8 Bytes"; \
    byte_jump:0,0, from_end, post_offset -8; \
    content:"|6c 33 33 74|"; distance:0 within:4;)
```

9.3.7.13 byte_extract

Ключевое слово byte_extract извлекает число байт <num of bytes> по смещению <offset> и сохраняет его в <var_name>. Значение в <var_name> может использоваться в любом

модификаторе, который принимает число в качестве опции, а в случае byte_test его можно использовать в качестве значения.

Формат (модификаторы для ключевых слов представлены в таблице 15):

```
byte_extract:<num of bytes>, <offset>, <var_name>, [, relative];
```

Таблица 15 – Модификаторы для ключевых слов

Ключевое слово	**Модификатор**
content	offset, depth, distance, within
byte_test	offset, value
byte_jump	offset
isdataat	offset

Пример:

```
alert tcp any any -> any any \
      (msg:"Byte_Extract пример использования distance"; \
      content:"Alice"; byte_extract:2,0,size; content:"Bob"; distance:size; within:3; sid:1;)
alert tcp any any -> any any \
      (msg:"Byte_Extract пример использования within"; \
      flow:established,to_server; content:"|00 FF|"; \
      byte_extract:1,0,len,relative; content:"|5c 00|"; distance:2; within:len; sid:2;)
alert tcp any any -> any any \
      (msg:"Byte_Extract пример сравнения"; \
      flow:established,to_server; content:"|00 FF|"; \
       byte_extract:2,0,cmp_ver,relative; content:"FooBar"; distance:0; byte_test:2,=,cmp_ver,0;
sid:3;)
```

9.3.7.14 rpc

Ключевое слово rpc можно использовать для сопоставления в SUNRPC CALL по номерам процедур RPC и версии RPC.

Можно изменить ключевое слово, используя подстановочный знак, определенный с помощью *. С помощью этого подстановочного знака можно сопоставить все номера версий и / или процедур.

RPC– это приложение, которое позволяет компьютерной программе выполнять процедуру на другом компьютере (или в адресном пространстве), используется для межпроцессного взаимодействия.

Формат:

```
rpc:<application number>, [<version number>|*], [<procedure number>|*]>;
```

Пример ключевого слова rpc в правиле:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg:”RPC portmap request yppasswdd”; rpc:100009,*,*; reference:bugtraq,2763; classtype:rpc-portmap-decode; sid:1296; rev:4;)
```

9.3.7.15 replace

Модификатор содержимого replace можно использовать только в ips. Регулирует сетевой трафик. Он изменяет содержимое, за которым следует ('abc'), на другое ('def'):



Модификатор replace должен содержать столько символов, сколько заменяется. Он может использоваться только с отдельными пакетами и не будет работать для нормализованных буферов, таких как HTTP URI или для совпадения содержимого в повторно собранном потоке.

Контрольные суммы будут пересчитаны Suricata и изменены после использования ключевого слова replace.

9.3.7.16 pcre (Perl-совместимые регулярные выражения)

Ключевое слово pcre соответствует определенным регулярным выражениям.

Сложность pcre приводит к тому, что его использование негативно влияет на производительность. Таким образом, чтобы избавить Suricata от необходимости часто проверять pcre, его в основном комбинируют с content. В этом случае содержимое сначала сопоставляется, прежде чем оно будет проверено с помощью pcre.

Формат pcre:

```
pcre:"/<regex>/opts";
```

Пример pcre. Этот пример будет совпадать, если полезная нагрузка содержит шесть следующих чисел:

```
pcre:"/[0-9]{6}/";
```

Пример pcre в сигнатуре:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:”ET TROJAN Likely Bot Nick in IRC (USA +.)”; flow:established,to_server; flowbits:isset,is_proto_irc; content:”NICK “; pcre:”/NICK .*USA.*[0-9]{3,}/i”; reference:url,doc.emergingthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

Есть несколько свойств pcre, которые можно изменить:

- по умолчанию pcre чувствительно к регистру;
- . (точка) является частью регулярного выражения, она соответствует каждому байту, кроме символов новой строки;
- по умолчанию полезная нагрузка будет проверяться одной строкой.

Эти свойства могут быть изменены с использованием следующих символов:

- i – нечувствительность к регистру;
- s – проверка символов новой строки;
- m – превращение одной строки (из полезной нагрузки) в две строки.

Эти параметры являются perl-совместимыми модификаторами. Чтобы использовать эти модификаторы, следует добавить их в pcre после регулярного выражения: Например:

```
pcre: "/<regex>/i";
```

Существует несколько pcre-совместимых модификаторов, которые также могут изменять свойства pcre:

- A: шаблон должен совпадать в начале буфера. (в pcre ^ похоже на A);
- E: игнорирует символы новой строки в конце буфера / полезной нагрузки;
- G: инвертирование потребления ресурсов.

Следует исключить следующие символы внутри содержимого: ; \ ".

9.3.7.17 Модификаторы Suricata

Suricata имеет свои собственные модификаторы pcre:

- R: совпадение относительно последнего совпадения с образцом. Оно похоже на расстояние: 0;
- U: pcre-сопоставление нормализованного URI. uri_buffer, как и uricontent и content, объединенный с http_uri.U, можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-uri;
- I: pcre-сопоставление в HTTP-raw-uri. Сопоставляет с тем же буфером, что и http_raw_uri. Может быть объединено с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-raw-uri;
- P: pcre-сопоставление в теле HTTP-запроса. Сопоставляет с тем же буфером, что и http_client_body. P можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в теле HTTP-запроса;
- Q: pcre соответствие в теле HTTP-ответа. Сопоставляет с тем же буфером, что и http_server_body. Q можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в теле HTTP-ответа;
- H: pcre-сопоставление в HTTP-заголовке. H можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в теле заголовка HTTP;
- D: pcre-сопоставление в ненормализованном заголовке. Сопоставляет с тем же буфером, что и http_raw_header. D можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в HTTP-raw-header;
- M: pcre-сопоставление по методу запроса. Сопоставляет с тем же буфером, что и http_method. M можно комбинировать с /p. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-метода;
- C: pcre-сопоставление в HTTP-cookie. Сопоставляет с тем же буфером, что и http_cookie. C можно сочетать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-cookie;
- S: Определяет pcre-сопоставление в HTTP-stat-коде. Сопоставляет с тем же буфером, что и http_stat_code. S можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-stat-code;
- Y: Определяет pcre-сопоставление в HTTP-stat-msg. Сопоставляет с тем же буфером, что и http_stat_msg. Y можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-stat-msg;
- B: В сигнатуре можете встретить B, но это только для совместимости. Итак, Suricata не использует B, но поддерживает его, чтобы не возникало ошибок.
- O: Переопределяет предел соответствия pcre;
- V: Определяет pcre-сопоставление в HTTP-User-Agent. Сопоставляет с тем же буфером, что и http_user_agent. V можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-User-Agent;
- W: Определяет pcre-сопоставление в HTTP-хосте. Сопоставляет с тем же буфером, что и http_host. W можно комбинировать с /R. Обратите внимание, что R относительно предыдущего совпадения, поэтому оба совпадения должны быть в буфере HTTP-Host.

9.3.8 Трансформации

Ключевое слово transformation превращают данные из липкого буфера в нечто другое.

Пример:

```
alert http any any -> any any (file_data; strip_whitespace; \
  content:"window.navigate("; sid:1;)
```

Этот пример будет проверять трафик на соответствие, даже если есть один или несколько пробелов между navigate и (.

Преобразования могут быть связаны. Они обрабатываются в том порядке, в котором появляются в правиле. Вывод каждого преобразования является входом для следующего.

Пример:

```
alert http any any -> any any (http_request_line; compress_whitespace; to_sha256; \
    content:"|54A9 7A8A B09C 1B81 3725 2214 51D3 F997 F015 9DD7 049E E5AD CED3
945A FC79 7401|"; sid:1;)
```

9.3.8.1 dotprefix

Заполняет буфер и добавляет символ ., чтобы облегчить краткую проверку домена. Например, строка ввода hello.google.com будет изменена и станет .hello.google.com. Кроме того, добавление точки позволяет google.com сопоставлят content:".google.com"

Пример:

```
alert dns any any -> any any (dns.query; dotprefix; \
  content:".microsoft.com"; sid:1;)
```

Этот пример будет соответствовать для windows.update.microsoft.com и maps.microsoft.com.au, но не для windows.update.fakemicrosoft.com.

Это правило можно использовать только для сопоставления домена; пример

```
alert dns any any -> any any (dns.query; dotprefix; \
  content:".microsoft.com"; endswith; sid:1;)
```

Этот пример будет соответствовать для windows.update.microsoft.com, но не для windows.update.microsoft.com.au.

Наконец, это правило может использоваться для сопоставления только в TLD; пример:

```
alert dns any any -> any any (dns.query; dotprefix; \
  content:".co.uk"; endswith; sid:1;)
```

Этот пример будет соответствовать для maps.google.co.uk, но не для maps.google.co.nl.

9.3.8.2 strip_whitespace

Удаляет все пробелы наподобие isspace() в C.

Пример:

```
alert http any any -> any any (file_data; strip_whitespace; \
  content:"window.navigate("; sid:1;)
```

9.3.8.3 compress_whitespace

Сжатие всех последовательных пробелов в один пробел.

9.3.8.4 to_md5

Занимает буфер, вычисляет хеш MD5 и передает значение сырого хэша (зависит от компиляции libnss в Suricata).

Пример:

```
alert http any any -> any any (http_request_line; to_md5; \
   content:"|54 A9 7A 8A B0 9C 1B 81 37 25 22 14 51 D3 F9 97|"; sid:1;)
```

9.3.8.5 to_sha1

Занимает буфер, вычисляет хеш SHA-1 и передает значение сырого хэша (зависит от компиляции libnss в Suricata).

Пример:

```
alert http any any -> any any (http_request_line; to_sha1; \
   content:"|54A9 7A8A B09C 1B81 3725 2214 51D3 F997 F015 9DD7|"; sid:1;)
```

9.3.8.6 to_sha256

Занимает буфер, вычисляет хеш SHA-256 и передает значение сырого хэша (зависит от компиляции libnss в Suricata).

Пример:

```
alert http any any -> any any (http_request_line; to_sha256; \
    content:"|54A9 7A8A B09C 1B81 3725 2214 51D3 F997 F015 9DD7 049E E5AD CED3
945A FC79 7401|"; sid:1;)
```

9.3.9 Ключевые слова для предварительной фильтрации

9.3.9.1 fast_pattern

Для Multi Pattern Matcher (MPM) используется только одно содержимое сигнатуры. Если содержимых несколько, Suricata использует самое «сильное» содержимое – сочетание длины, разнообразия и просматриваемого буфера. Как правило, чем длиннее и разнообразнее, тем лучше.

Иногда автор сигнатуры приходит к тому чтобы Suricata использовала другое содержимое, чем по умолчанию.

Например:

```
User-agent: Mozilla/5.0 Badness;

content:"User-Agent|3A|";
content:"Badness"; distance:0;
```

В этом примере видно, что первое содержимое длиннее и разнообразнее второго, поэтому Suricata будет использовать его для MPM. Поскольку «User-Agent:» будет совпадать очень часто, а «Badness» встречается реже в сетевом трафике, можно заставить Suricata использовать второй контент с помощью «fast_pattern».

```
content:"User-Agent|3A|";
content:"Badness"; distance:0; fast_pattern;
```

Ключевое слово fast_pattern заменяет предшествующее ему содержимое.

Fast-pattern также можно комбинировать со всеми ранее упомянутыми ключевыми словами и всеми упомянутыми HTTP-модификаторами.

fast_pattern:only

Иногда сигнатура содержит только одно содержимое. В этом случае нет необходимости, чтобы Suricata проверяла его еще раз после того, как в MPM будет найдено совпадение. Если есть только одно содержимое, вся сигнатура совпадает. Suricata замечает это автоматически. В некоторых сигнатурах это все еще обозначается как 'fast_pattern: only;'. Хотя Suricata не нужен fast_pattern: only, она поддерживает его.

fast_pattern:’chop’

Если не требуется, чтобы MPM использовал весь контент, можно использовать fast_pattern 'chop'.

Например:

```
content: "aaaaaaaaabc"; fast_pattern:8,4;
```

Таким образом, MPM использует только последние четыре символа.

9.3.9.2 prefilter

Механизмы prefilter для других, отличных от MPM, ключевых слов. Его можно включить в определенных правилах с помощью ключевого слова prefilter.

В следующем правиле TTL-тест будет использоваться при предварительной фильтрации вместо однобайтового шаблона:

```
alert ip any any -> any any (ttl:123; prefilter; content:"a"; sid:1;)
```

9.3.9.3 Объяснение быстрого определения шаблона в Suricata

Если ключевое слово fast_pattern явно задано в правиле, Suricata будет использовать его для быстрого сопоставления с шаблоном. Ключевое слово fast_pattern может быть установлено только один раз для каждого правила. Если 'fast_pattern' не установлен, Suricata автоматически определяет содержимое, которое будет использоваться для быстрого сопоставления с шаблоном.

Далее объясняется логика, которую использует Suricata для автоматического сопоставления при использовании 'fast_pattern'.

Имейте в виду, что если есть положительные (то есть неотрицательные) совпадения содержимого, то отрицательные совпадения игнорируются для быстрого определения шаблона. В противном случае рассматриваются отрицательные совпадения содержимого.

Критерии выбора шаблона для быстрого сопоставления следующие:

1. Suricata сначала идентифицирует все совпадения содержимого, имеющие самый высокий «приоритет», которые используются в сигнатуре. Приоритет основывается на сопоставляемом буфере, и, как правило, буферы 'http_*' имеют более высокий приоритет (чем меньше число, тем выше приоритет).
2. В совпадениях содержимого, определенных на шаге 1 (совпадения содержимого с наивысшим приоритетом), самое длинное (с точки зрения длины в символах/байтах) используется как шаблон для быстрого сопоставления.
3. Если несколько совпадений содержимого имеют одинаковый наивысший приоритет и самую большую длину, то в качестве шаблона для быстрого сопоставления используется тот, который имеет наивысший показатель разнообразия символов/байтов («сила шаблона»).
4. Если несколько совпадений содержимого имеют одинаковый наивысший приоритет, самую большую длину и одинаковую силу шаблона, то буфер («list_id»), который был

зарегистрирован последним, используется в качестве шаблона для быстрого сопоставления.

5. Если несколько совпадений содержимого имеют одинаковый наивысший приоритет, самую большую длину, одинаковую наивысшую силу шаблона и имеют одинаковый list_id (т. е. ищут в одном и том же буфере), то тот, который идет первым (слева направо) в правиле используется в качестве шаблона для быстрого сопоставления.

Стоит отметить, что для совпадений содержимого, имеющих одинаковый приоритет, длину и силу шаблона, «http_stat_msg», «http_stat_code» и «http_method» имеют приоритет над обычными совпадениями «content».

9.3.9.3.1 Буферы, значения list_id и порядок регистрации для Suricata 1.3.4

Для Suricata 1.1.x – 1.4.x см. таблицу 16.

Таблица 16 – Буферы, значения list_id и порядок регистрации для Suricata 1.3.4

list_id	**Модификатор содержимого / ключевое слово**	**Наименование буфера**	**Порядок регистрации**
1	<none> (совпадение содержимого)	DETECT_SM_LIST_PMATCH	1 (первый)
2	http_uri	DETECT_SM_LIST_UMATCH	2
6	http_client_body	DETECT_SM_LIST_HCBDMATCH	3
7	http_server_body	DETECT_SM_LIST_HSBDMATCH	4
8	http_header	DETECT_SM_LIST_HHDMATCH	5
9	http_raw_header	DETECT_SM_LIST_HRHDMATCH	6
10	http_method	DETECT_SM_LIST_HMDMATCH	7
11	http_cookie	DETECT_SM_LIST_HCDMATCH	8
12	http_raw_uri	DETECT_SM_LIST_HRUDMATCH	9
13	http_stat_msg	DETECT_SM_LIST_HSMDMATCH	10
14	http_stat_code	DETECT_SM_LIST_HSCDMATCH	11
15	http_user_agent	DETECT_SM_LIST_HUADMATCH	12 (последний)

Порядок регистрации не влияет на определение шаблона для быстрого поиска в Suricata 1.3.4, но list_id влияет.

9.3.9.3.2 Буферы, значения list_id, приоритеты и порядок регистрации для Suricata 2.0.7

Применимо для Suricata 2.0.x см. таблицу 17.

Таблица 17 – Буферы, значения list_id и порядок регистрации для Suricata 2.0.x

Приоритет (меньший номер имеет более высокий приоритет)	Порядок регистрации	Модификатор содержимого / ключевое слово	Наименование буфера	list_id
3	11	<first> (совпадение содержимого)	DETECT_SM_LIST_PMATCH	1
3	12	http_method	DETECT_SM_LIST_HMDMATCH	12
3	13	http_stat_code	DETECT_SM_LIST_HSCDMATCH	9
3	14	http_stat_msg	DETECT_SM_LIST_HSMDMATCH	8
2	1 (первый)	http_client_body	DETECT_SM_LIST_HCBDMATCH	4
2	2	http_server_body	DETECT_SM_LIST_HSBDMATCH	5
2	3	http_header	DETECT_SM_LIST_HHDMATCH	6
2	4	http_raw_header	DETECT_SM_LIST_HRHDMATCH	7
2	5	http_uri	DETECT_SM_LIST_UMATCH	2
2	6	http_raw_uri	DETECT_SM_LIST_HRUDMATCH	3
2	7	http_host	DETECT_SM_LIST_HHHDMATCH	10
2	8	http_raw_host	DETECT_SM_LIST_HRHHDMATCH	11
2	9	http_cookie	DETECT_SM_LIST_HCDMATCH	13
2	10	http_user_agent	DETECT_SM_LIST_HUADMATCH	14
2	15 (последний)	dns_query	DETECT_SM_LIST_DNSQUERY_MATCH	20

Значение list_id не влияет на определение шаблона для быстрого поиска в Suricata 2.0.7, но порядок регистрации влияет.

9.3.9.3.3 Алгоритм силы шаблона

Из detect-engine-mpm.c: Оценка силы шаблона начинается с нуля, просматривается каждый символ/байт в переданном байтовом массиве слева направо. Если символ/байт ранее не встречался в массиве, то к счету добавляется 3, если это альфа-символ; иначе к счету добавляется 4, если это печатный символ, 0x00, 0x01 или 0xFF; иначе к счету добавляется 6. Если символ/байт встречался раньше, то добавляется 1. Возвращается окончательный результат.

```
/** \brief Predict a strength value for patterns
 *
 *  Patterns with high character diversity score higher.
 *  Alpha chars score not so high
 *  Other printable + a few common codes a little higher
 *  Everything else highest.
 *  Longer patterns score better than short patters.
 *
 *  \param pat pattern
 *  \param patlen length of the pattern
 *
 *  \retval s pattern score
 */
uint32_t PatternStrength(uint8_t *pat, uint16_t patlen) {
    uint8_t a[256];
    memset(&a, 0 ,sizeof(a));
    uint32_t s = 0;
    uint16_t u = 0;
    for (u = 0; u < patlen; u++) {
        if (a[pat[u]] == 0) {
            if (isalpha(pat[u]))
                s += 3;
            else if (isprint(pat[u]) || pat[u] == 0x00 || pat[u] == 0x01 || pat[u] == 0xFF)
                s += 4;
            else
                s += 6;
            a[pat[u]] = 1;
        } else {
            s++;
        }
    }
    return s;
}
```

9.3.10 Ключевые слова потока

9.3.10.1 flowbits

Flowbits состоит из двух частей. Первая часть описывает действие, которое планируется выполнить, вторая часть – имя потока.

Есть несколько пакетов, которые принадлежат одному потоку. Suricata хранит эти потоки в памяти. Flowbits может гарантировать, что будет сгенерировано оповещение, когда, например, два разных пакета совпадают. Оповещение будет генерироваться только тогда, когда оба пакета совпадают. Таким образом, когда второй пакет совпадает, Suricata должна знать совпадал ли первый пакет. Flowbits отмечает поток, если пакет совпадает, поэтому Suricata «знает», что он должен генерировать оповещение, когда второй пакет также совпадает.

Flowbits имеют разные действия:

flowbits: set, name
Устанавливает условие / «имя», если оно присутствует, в потоке.

flowbits: isset, name

Может использоваться в правиле, чтобы убедиться, что оно генерирует оповещение, когда правило соответствует и в потоке задано условие.

flowbits: toggle, name

Инвертирует текущую настройку. Например, если задано условие, оно будет отменено, и наоборот.

flowbits: unset, name

Может использоваться для сброса условия в потоке.

flowbits: isnotset, name

Может использоваться в правиле, чтобы убедиться, что оно генерирует оповещение, когда оно соответствует и в потоке не задано условие.

flowbits: noalert

Это правило не генерирует оповещений

В правиле можно использовать flowbits несколько раз и комбинировать различные функции.

9.3.10.2 flow

Ключевое слово flow может использоваться для сопоставления направления потока клиента или с сервера. Также можно сопоставлять установлен поток или нет. Ключевое слово flow также можно использовать, чтобы указать должна ли сигнатура совпадать только в потоке (only_stream) или только в пакете (no_stream).

Таким образом, с ключевым словом flow можно:

to_client

Проверить совпадение пакетов от сервера к клиенту.

to_server

Проверить совпадение пакетов от клиента к серверу.

from_client

Проверить совпадение пакетов от клиента к серверу (аналогично to_server).

from_server

Проверить совпадение пакетов от сервера к клиенту (аналогично to_client).

established

Проверить установление соединения.

not_established

Проверить совпадение пакетов, которые не являются частью установленного соединения.

stateless

Проверить совпадение пакетов, которые являются и не являются частью установленного соединения.

only_stream

Проверить совпадение пакетов, которые были повторно собраны потоковым движком.

no_stream

Проверить совпадение пакетов, которые не были повторно собраны потоковым движком. Не будет соответствовать пакетам, которые были повторно собраны.

only_frag

Проверить совпадение пакетов, которые были собраны из фрагментов.

no_frag
Проверить совпадение пакетов, которые не были собраны из фрагментов.
Несколько параметров flow могут быть объединены, например:

```
flow:to_client, established
flow:to_server, established, only_stream
flow:to_server, not_established, no_frag
```

Определение *установленного соединения* зависит от протокола:

- Для TCP соединение будет установлено после трехстороннего рукопожатия (начала сеанса);
- Для других протоколов (например, UDP), соединение будет считаться установленным после просмотра трафика с обеих сторон соединения.

9.3.10.3 flowint

Flowint позволяет хранить и выполнять математические операции с использованием переменных. Работает также, как flowbits, но с добавлением математических возможностей и того факта, что можно хранить и манипулировать целыми числами, а не просто устанавливать флаг. Может использоваться для подсчета вхождений, добавления или вычитания вхождений, или выполнения установки пороговых значений в потоке в зависимости от множества факторов.

Синтаксис выглядит следующим образом:

```
flowint: name, modifier[, value];
```

Определение переменной, проверка, что она установлена или не установлена.

```
flowint: name, < +,-,=,>,<,>=,<=,==, != >, value;
flowint: name, (isset|isnotset);
```

Сравните или измените переменную. Доступны сложение, вычитание, сравнение больше или меньше, больше или равно и меньше или равно. Элемент для сравнения может быть целым числом или другой переменной.

Например, если следует подсчитать, сколько раз имя пользователя отображается в определенном потоке, и предупредить, если это число превышает 5.

```
alert tcp any any -> any any (msg:"Counting Usernames"; content:"jonkman"; \
    flowint: usernamecount, +, 1; noalert;)
```

Будет подсчитываться каждое вхождение и увеличивать переменную usernamecount, а не создавать оповещение каждый раз.

Теперь, если требуется генерировать оповещение, если в потоке более пяти вхождений.

```
alert tcp any any -> any any (msg:"More than Five Usernames!"; content:"jonkman"; \
    flowint: usernamecount, +, 1; flowint:usernamecount, >, 5;)
```

Оповещение появится ТОЛЬКО, если usernamecount превышает пять.

Например, оповещение, как указано выше, но, если было больше больше пяти активных логинов пользователя. Предположим, что протокол указывает на выход с помощью «jonkman logout»:

```
alert tcp any any -> any any (msg:"Username Logged out"; content:"logout jonkman"; \
    flowint: usernamecount, -, 1; flowint:usernamecount, >, 5;)
```

Теперь оповещение возникнет ТОЛЬКО, если больше пяти активных логинов для определенного пользователя.

Это довольно упрощенный пример, но он показывает всю мощь такой простой функции для написания правил. Существует множество применений для таких вещей, как отслеживание

входа в систему, машины состояния IRC, отслеживание вредоносных программ и обнаружение входа в систему методом грубой силы.

Например, отслеживание протокола, который обычно допускает пять неудачных попыток входа в систему на каждое соединение, но есть уязвимость, из-за которой злоумышленник может продолжить вход после пяти попыток:

```
alert tcp any any -> any any (msg:"Start a login count"; content:"login failed"; \
    flowint:loginfail, notset; flowint:loginfail, =, 1; noalert;)
```

Таким образом, можно обнаружить первоначальный сбой, если переменная еще не установлена установите в 1, если так. Первое попадание.

```
alert tcp any any -> any any (msg:"Counting Logins"; content:"login failed"; \
    flowint:loginfail, isset; flowint:loginfail, +, 1; noalert;)
```

Увеличение счетчика, если он установлен.

```
alert tcp any any -> any any (msg:"More than Five login fails in a Stream"; \
    content:"login failed"; flowint:loginfail, isset; flowint:loginfail, >, 5;)
```

Теперь оповещение генерируется при пяти неудачных логинов в том же потоке.
Теперь оповещение, если есть два успешных входа и неудачный вход после этого.

```
alert tcp any any -> any any (msg:"Counting Good Logins";          \
    content:"login successful"; flowint:loginsuccess, +, 1; noalert;)
```

Здесь подсчет успешных логинов, соответствующих сбоям:

```
alert tcp any any -> any any (msg:"Login fail after two successes";   \
    content:"login failed"; flowint:loginsuccess, isset;          \
    flowint:loginsuccess, =, 2;)
```

Вот некоторые другие общие примеры:

```
alert tcp any any -> any any (msg:"Setting a flowint counter"; content:"GET"; \
    flowint:myvar, notset; flowint:maxvar,notset;                  \
    flowint:myvar,=,1; flowint: maxvar,=,6;)
alert tcp any any -> any any (msg:"Adding to flowint counter";          \
    content:"Unauthorized"; flowint:myvar,isset; flowint: myvar,+,2;)
alert tcp any any -> any any (msg:"when flowint counter is 3 create new counter"; \
    content:"Unauthorized"; flowint:myvar, isset; flowint:myvar,==,3; \
    flowint:cntpackets,notset; flowint:cntpackets, =, 0;)
alert tcp any any -> any any (msg:"count the rest without generating alerts"; \
    flowint:cntpackets,isset; flowint:cntpackets, +, 1; noalert;)
alert tcp any any -> any any (msg:"fire this when it reach 6";          \
    flowint: cntpackets, isset;                                    \
    flowint: maxvar,isset; flowint: cntpackets, ==, maxvar;)
```

9.3.10.4 stream_size

Опция stream_size проверяет трафик на соответствие с зарегистрированным количеством байтов по порядковым номерам. Для этого ключевого слова есть несколько модификаторов:

- > больше чем;
- < меньше чем;
- = равно;
- != не равно;
- >= больше или равно;

- <= меньше или равно.
  Формат:

```
stream_size:<server|client|both|either>, <modifier>, <number>;
```

  Пример ключевого слова stream_size в правиле:

```
alert tcp any any -> any any (stream_size:both, >, 5000; sid:1;)
```

9.3.11 Ключевое слово bypass

В Suricata есть ключевое слово bypass, которое можно использовать в сигнатуре, чтобы исключить трафик из дальнейшей оценки.

Ключевое слово bypass полезно, когда ожидается большой поток (например, Netflix, Spotify, Youtube).

Данное ключевое слово bypass используются после сопоставления.

9.3.11.1 bypass

Обход потока при совпадении http-трафика.
Пример:

```
alert http any any -> any any (content:"suricata-ids.org"; \
   http_host; bypass; sid:10001; rev:1;)
```

9.3.12 Ключевые слова HTTP

Существуют дополнительные модификаторы контента, которые могут предоставлять специфичные для протокола возможности на уровне приложений. Эти ключевые слова обеспечивают проверку только определенных частей сетевого трафика. Например, чтобы специально проверить URI запроса, файлы cookie или тело HTTP-запроса или ответа и т. п.

Все ключевые слова HTTP являются модификаторами. Обратите внимание на разницу между модификаторами контента и липкими буферами:

- **«липкие буферы»** размещаются первыми, и все ключевые слова, следующие за ними, применяются к этому буферу, например:

```
alert http any any -> any any (http.response_line; content:"403 Forbidden"; sid:1;)
```

- **«модификаторы контента»** возвращаются в правило, например:

```
alert http any any -> any any (content:"index.php"; http_uri; sid:1;)
```

Доступны следующие ключевые слова **запроса** (см. таблицу 18).

Таблица 18 – Доступные ключевые слова запроса

Ключевое слово	**Липкий буфер или модификатор**	**Направление**
http.uri	Липкий буфер	Запрос
http.uri.raw	Липкий буфер	Запрос
http.method	Липкий буфер	Запрос
http.request_line	Липкий буфер	Запрос
http.request_body	Липкий буфер	Запрос

Ключевое слово	Липкий буфер или модификатор	Направление
http.header	Липкий буфер	Оба
http.header.raw	Липкий буфер	Оба
http.cookie	Липкий буфер	Оба
http.user_agent	Липкий буфер	Запрос
http.host	Липкий буфер	Запрос
http.host.raw	Липкий буфер	Запрос
http.accept	Липкий буфер	Запрос
http.accept_lang	Липкий буфер	Запрос
http.accept_enc	Липкий буфер	Запрос
http.referer	Липкий буфер	Запрос
http.connection	Липкий буфер	Запрос
http.content_type	Липкий буфер	Оба
http.content_len	Липкий буфер	Оба
http.start	Липкий буфер	Оба
http.protocol	Липкий буфер	Оба
http.header_names	Липкий буфер	Оба

Доступны следующие ключевые слова ответа (см. таблицу 19).

Таблица 19 – Доступные ключевые слова ответа

Ключевое слово	Липкий буфер или модификатор	Направление
http.stat_msg	Липкий буфер	Ответ
http.stat_code	Липкий буфер	Ответ
http.response_line	Липкий буфер	Ответ
http.header	Липкий буфер	Оба
http.header.raw	Липкий буфер	Оба
http.cookie	Липкий буфер	Оба
http.server_body	Липкий буфер	Ответ

Ключевое слово	Липкий буфер или модификатор	Направление
http.server	Липкий буфер	Ответ
http.location	Липкий буфер	Ответ
file_data	Липкий буфер	Ответ
http.content_type	Липкий буфер	Оба
http.content_len	Липкий буфер	Оба
http.start	Липкий буфер	Оба
http.protocol	Липкий буфер	Оба
http.header_names	Липкий буфер	Оба

9.3.12.1 Первичный HTTP

Важно понимать структуру HTTP-запросов и ответов. Ниже приведен простой пример HTTP-запроса и ответа:

Запрос HTTP

```
GET /index.html HTTP/1.0\r\n
```

GET – это метод запроса. Примеры использования атрибута: GET, POST, PUT, HEAD и т. п. Путь URI – /index.html а версия HTTP – HTTP/1.0. На протяжении многих лет использовалось несколько версий HTTP, включая 0.9, 1.0 и 1.1, 1.0 и 1.1 являются наиболее часто используемыми сегодня.

Ответ HTTP

```
HTTP/1.0 200 OK\r\n
<html>
<title> some page </title>
</HTML>
```

В этом примере HTTP / 1.0 – это версия HTTP, 200 – код состояния ответа и OK – сообщение о состоянии ответа.

Хотя файлы cookie отправляются в заголовке HTTP их нельзя сопоставить с ключевым словом http.header. Файлы cookie соответствуют их собственному ключевому слову, а именно http.cookie.

Каждая часть таблицы принадлежит так называемому *буферу*. HTTP-метод относится к буферу методов, HTTP-заголовки – к буферу заголовков и т. п. Буфер – это определенная часть запроса или ответа, которую Suricata извлекает в памяти для проверки.

Все ранее описанные ключевые слова могут быть использованы в сочетании с буфером в сигнатуре. Ключевые слова distance и within являются относительными модификаторами, поэтому они могут использоваться только в одном и том же буфере. Нельзя связать совпадения содержимого в различных буферах с относительными модификаторами.

9.3.12.2 http.method

С модификатором содержимого http.method можно сопоставлять данные только в буфере метода HTTP. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как: depth, distance, offset, nocase и within.

9.3.12.3 http.uri и http.uri.raw

С модификаторами содержимого http.uri и http.uri.raw можно сопоставить только буфер URI-запроса. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как depth, distance, offset, nocase и within.

Uri имеет два вида в Suricata: uri.raw и нормализованный uri. Пробел для примера может быть указан в шестнадцатеричной записи %20. Чтобы преобразовать это обозначение в пространство, его следует нормализовать. Тем не менее, можно указать конкретные символы %20 в uri. Это означает соответствие uri.raw. Uri.raw и нормализованный URI являются отдельными буферами. Поэтому uri.raw проверяет буфер uri.raw и не может проверить нормализованный буфер.

9.3.12.4 uricontent

Ключевое слово uricontent действует как модификатор содержимого http.uri. uricontent является устаревшим (хотя все еще поддерживаемый) способ сопоставления специально и только для буфера URI-запроса

Пример uricontent:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:”ET TROJAN Possible Vundo Trojan Variant reporting to Controller”; flow:established,to_server; content:”POST “; depth:5; uricontent:”/frame.html?”; urilen: > 80; classtype:trojan-activity; reference:url,doc.emergingthreats.net/2009173; reference:url,www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/VIRUS/TROJAN_Vundo; sid:2009173; rev:2;)
```

Разница между http.uri и uricontent заключается в синтаксисе:

При разработке новых правил рекомендуется использовать липкий буфер содержимого http.uri вместо устаревшего ключевого слова uricontent.

9.3.12.5 urilen

Ключевое слово urilen используется для проверки соответствия длине URI-запроса. Можно использовать < и > операторы, которые указывают соответственно *меньше* и *больше чем*.

Формат urilen:

```
urilen:3;
```

Другие возможности:

```
urilen:1;
urilen:>1;
urilen:<10;
urilen:10<>20;       (bigger than 10, smaller than 20)
```

Пример urilen в сигнатуре:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:”ET TROJAN Possible Vundo Trojan Variant reporting to Controller”; flow:established,to_server; content:”POST “; depth:5; uricontent:”/frame.html?”; urilen: > 80; classtype:trojan-activity; reference:url,doc.emergingthreats.net/2009173;
```

```
reference:url,www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/VIRUS/TROJAN_Vundo;
sid:2009173; rev:2;)
```

Также можно добавить norm или raw, чтобы определить, какой тип буфера следует использовать (нормализованный или необработанный буфер).

9.3.12.6 http.protocol

http.protocol проверяет поле протокола в HTTP-запросе или строке ответа. Если строка запроса – «GET / HTTP / 1.0rn», то этот буфер будет содержать «HTTP / 1.0».

Пример:

```
alert http any any -> any any (flow:to_server; http.protocol; content:"HTTP/1.0"; sid:1;)
```

http.protocol заменяет предыдущее имя ключевого слова: `http_protocol. Можно продолжить использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

Пример:

```
alert http any any -> any any (flow:to_server; http.protocol; content:"HTTP/1.0"; sid:1;)
```

9.3.12.7 http.request_line

http.request_line принудительно проверяет всю строку HTTP-запроса.

Пример:

```
alert http any any -> any any (http.request_line; content:"GET / HTTP/1.0"; sid:1;)
```

9.3.12.8 http.header and http.header.raw

С помощью модификатора содержимого http.header можно специально сопоставлять только в буфере заголовка HTTP. Он содержит все извлеченные заголовки в одном буфере, за исключением указанных в документации, которые не могут быть сопоставлены этим буфером и имеют свой собственный модификатор содержимого (например, http.cookie). Модификатор может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как depth, distance, offset, nocase и within.

Буфер заголовка *нормализован*. Все завершающие пробелы и символы табуляции удаляются. Чтобы избежать этого, используйте ключевое слово http.header.raw.

9.3.12.9 http.cookie

С помощью модификатора содержимого http.cookie можно сопоставлять данные только в буфере cookie. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами контента, такими как depth, distance, offset, nocase и within.

Обратите внимание, что файлы cookie передаются в заголовках HTTP, но извлекаются в выделенный буфер и сопоставляются с использованием их собственного определенного модификатора содержимого.

9.3.12.10 http.user_agent

Модификатор содержимого http.user_agent является частью заголовка HTTP-запроса, позволяет точно сопоставить значение заголовка User-Agent. Он нормализован в том смысле, что он не включает в имя заголовка _”User-Agent: “_ и разделитель, а также не содержит возврат каретки и перевод строки (CRLF). Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами контента, такими как depth, distance, offset, nocase и within. Обратите внимание, что ключевое слово pcre также может проверять этот буфер при использовании модификатора /V.

Нормализация: начальные пробелы не являются частью этого буфера. Таким образом, «User-Agent: rn» приведет к очистке буфера http.user_agent.

9.3.12.10.1 Примечания

- Буфер http.user_agent не будет включать имя заголовка, двоеточие или ведущие пробелы, то есть он не будет включать «User-Agent:».
- Буфер http.user_agent не включает CRLF (0x0D 0x0A) в конце. Если требуется сопоставить конец буфера, используйте относительный isdataat или PCRE (хотя PCRE будет хуже по производительности).
- Если запрос содержит несколько заголовков «User-Agent», значения будут объединены в буфере http.user_agent в порядке сверху вниз, с запятой и пробелом («, ») между каждым из них.

  Пример запроса:

  ```
  GET /test.html HTTP/1.1
  User-Agent: SuriTester/0.8
  User-Agent: GGGG
  ```

  http.user_agent содержимое буфера:

  ```
  SuriTester/0.8, GGGG
  ```

- Соответствующий PCRE-модификатор: V;
- Использование буфера http.user_agent более эффективно (~ 10% лучше), когда речь идет о производительности, чем использование буфера http.header.

9.3.12.11 http.accept

Липкий буфер для проверки соответствия в заголовке HTTP Accept. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (http.accept; content:"image/gif"; sid:1;)
```

9.3.12.12 http.accept_enc

Липкий буфер для проверки соответствия в заголовке HTTP Accept-Encoding. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (http.accept_enc; content:"gzip"; sid:1;)
```

9.3.12.13 http.accept_lang

Липкий буфер для проверки соответствия в заголовке HTTP Accept-Language. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (http.accept_lang; content:"en-us"; sid:1;)
```

9.3.12.14 http.connection

Липкий буфер для проверки соответствия в заголовке HTTP Connection. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (http.connection; content:"keep-alive"; sid:1;)
```

9.3.12.15 http.content_type

Липкий буфер для проверки соответствия в заголовках HTTP Content-Type. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Используйте flow:to_server или flow:to_client для принудительной проверки запроса или ответа.

Примеры:

```
alert http any any -> any any (flow:to_server; \
    http.content_type; content:"x-www-form-urlencoded"; sid:1;)

alert http any any -> any any (flow:to_client; \
    http.content_type; content:"text/javascript"; sid:2;)
```

9.3.12.16 http.content_len

Липкий буфер для проверки соответствия в заголовке HTTP Content-Length. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Используйте flow:to_server или flow:to_client для принудительной проверки запроса или ответа.

Примеры:

```
alert http any any -> any any (flow:to_server; \
    http.content_len; content:"666"; sid:1;)

alert http any any -> any any (flow:to_client; \
    http.content_len; content:"555"; sid:2;)
```

Чтобы выполнить числовую проверку длины содержимого можно использовать byte_test.

Пример, соответствует, если CL больше или равен 8079:

```
alert http any any -> any any (flow:to_client; \
    http.content_len; byte_test:0,>=,8079,0,string,dec; sid:3;)
```

9.3.12.17 http.referer

Липкий буфер для проверки соответствия в заголовке HTTP Referer. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (http.referer; content:".php"; sid:1;)
```

9.3.12.18 http.start

Проверяет начало HTTP-запроса или ответа. Будет содержать строку запроса/ответа плюс заголовки запроса/ответа. Используйте flow:to_server или flow:to_client для принудительной проверки запроса или ответа.

Пример:

```
alert http any any -> any any (http.start; content:"HTTP/1.1|0d 0a|User-Agent"; sid:1;)
```

Буфер содержит нормализованные заголовки и завершается дополнительным \r\n для указания конца заголовков.

9.3.12.19 http.header_names

Проверяет буфер, содержащий только имена заголовков HTTP. Полезно для проверки отсутствия заголовка или проверки определенного порядка заголовков.

Буфер начинается с \r\n и заканчивается дополнительным \r\n.

Пример буфера:

```
\\r\\nHost\\r\\n\\r\\n
```

Пример правила:

```
alert http any any -> any any (http.header_names; content:"|0d 0a|Host|0d 0a|"; sid:1;)
```

Пример, проверяющий, что присутствует *только* Host

```
alert http any any -> any any (http.header_names; \
    content:"|0d 0a|Host|0d 0a 0d 0a|"; sid:1;)
```

Пример, проверяющий, что *User-Agent* находится сразу после *Host*:

```
alert http any any -> any any (http.header_names; \
    content:"|0d 0a|Host|0d 0a|User-Agent|0d 0a|"; sid:1;)
```

Пример, проверяющий, что *User-Agent* находится после *Host*, но не обязательно сразу после:

```
alert http any any -> any any (http.header_names; \
    content:"|0d 0a|Host|0d 0a|"; content:"|0a 0d|User-Agent|0d 0a|"; \
    distance:-2; sid:1;)
```

9.3.12.20 http.request_body

Модификатор содержимого http.request_body позволяет выполнять сопоставление только с телом HTTP-запроса. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как distance, offset, nocase, within и т. п.

> Какая часть запроса/клиента проверяется, определяется в **разделе конфигурации libhtp** через параметр request-body-limit.

http.request_body заменяет предыдущее имя ключевого слова: `http_client_body. Можно продолжить использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.12.21 http.stat_code

Модификатор содержимого http.stat_code позволяет выполнять сопоставление только с буфером кода состояния HTTP. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как distance, offset, nocase, within и т. п.

9.3.12.22 http.stat_msg

С помощью модификатора содержимого http.stat_msg можно проверять совпадение и только в буфере сообщений о состоянии HTTP. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами контента, такими как depth, distance, offset, nocase и within.

9.3.12.23 http.response_line

http.response_line принудительно проверяет всю строку HTTP-ответа.

Пример:

```
alert http any any -> any any (http.response_line; content:"HTTP/1.0 200 OK"; sid:1;)
```

9.3.12.24 http.response_body

Модификатор содержимого http.response_body позволяет выполнять сопоставление только с телом HTTP-ответа. Ключевое слово может использоваться в сочетании со всеми ранее упомянутыми модификаторами содержимого, такими как distance, offset, nocase, within и т. п.

> Объем проверки тела ответа/сервера, контролируется в **разделе конфигурации libhtp** с помощью параметра response-body-limit.

9.3.12.24.1 Примечания

- Использование http.response_body аналогично проверке совпадений содержимого, которые идут после file_data, за исключением того, что он не устанавливает постоянно (если не сброшено) указатель обнаружения на начало тела ответа сервера. т. е. это не липкий буфер;
- http.response_body будет совпадать с декодированными в gzip данными, аналогично file_data.
- Поскольку http.response_body совпадает с ответом сервера, его нельзя использовать с директивами потока to_server или from_client.
- Соответствующий PCRE-модификатор: Q
- Дополнительные примечания в разделе file_data ниже.

http.response_body заменяет предыдущее имя ключевого слова: http_server_body. Можно продолжить использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.12.25 http.server

Липкий буфер для соответствия заголовкам HTTP-сервера. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (flow:to_client; \
    http.server; content:"Microsoft-IIS/6.0"; sid:1;)
```

9.3.12.26 http.location

Липкий буфер для соответствия в заголовках HTTP Location. Содержит только значение заголовка. \r\n после заголовка не являются частью буфера.

Пример:

```
alert http any any -> any any (flow:to_client; \
    http.location; content:"http://www.google.com"; sid:1;)
```

9.3.12.27 http.host и http.host.raw

С модификатором содержимого http.host можно сопоставить только определенное имя хоста. Http.host.raw проверяет необработанное имя хоста.

Ключевое слово может использоваться в сочетании с большинством модификаторов содержимого, таких как distance, offset, within и т. п.

Ключевое слово nocase больше не разрешено. Имейте в виду, что следует использовать строчные буквы.

9.3.12.27.1 Примечания

- Буферы http.host и http.host.raw заполняются из URI (если полный URI присутствует в запросе как в прокси-запросе) или заголовка HTTP-хоста. Если присутствуют оба, используется URI.
- Буферы http.host и http.host.raw не будут включать имя заголовка, двоеточие или предшествующие пробелы, если заполняются из заголовка хоста, то есть они не будут включать «Host: ».
- Буферы http.host и http.host.raw не включают CRLF (0x0D 0x0A) в конце. Если следует соответствовать концу буфера, используйте относительный «isdataat» или PCRE (хотя PCRE будет хуже по производительности).
- Буфер http.host нормализован, чтобы быть в нижнем регистре.
- Совпадение содержимого, к которому применяется http.host должно быть в нижнем регистре или иметь установленный флаг nocase.
- http.host.raw сопоставляет ненормализованный буфер, поэтому сопоставление будет чувствительным к регистру (если не установлен nocase).
- Если запрос содержит несколько заголовков «Host», значения будут объединены в буферах http.host и http.host.raw в порядке сверху вниз, с запятой и пробелом («, ») между каждым из них.

  Пример запроса:

  ```
  GET /test.html HTTP/1.1
  Host: ABC.com
  Accept: */*
  Host: efg.net
  ```

  http.host содержимое буфера:

  ```
  abc.com, efg.net
  ```

  http.host.raw содержимое буфера:

  ```
  ABC.com, efg.net
  ```

- Соответствующий PCRE-модификатор (http_host): W
- Соответствующий PCRE-модификатор (http_raw_host): Z

9.3.12.28 file_data

При использовании file_data проверяется тело ответа HTTP, как и в случае с http.response_body. Ключевое слово file_data является липким буфером.

Пример:

```
alert http any any -> any any (file_data; content:"abc"; content:"xyz";)
```

Ключевое слово file_data влияет на все последующие совпадения содержимого, пока не встретится ключевое слово pkt_data или не будет достигнут конец правила. Это делает его полезным для применения для множества совпадений контента в теле ответа HTTP, устраняя необходимость индивидуальной модификации каждого совпадения контента.

Поскольку тело HTTP-ответа может быть очень большим, оно проверяется небольшими частями.

Количество проверяемого тела ответа/сервера контролируется в **разделе конфигурации libhtp** с помощью параметра response-body-limit.

Если тело HTTP представляет собой флэш-файл, сжатый с помощью 'deflate' или 'lzma', его можно распаковать, и file_data может соответствовать данным распаковки. Декомпрессия Flash должна быть включена в конфигурации libhtp:

```
# Decompress SWF files.
# 2 types: 'deflate', 'lzma', 'both' will decompress deflate and lzma
```

```
# compress-depth:
# Specifies the maximum amount of data to decompress,
# set 0 for unlimited.
# decompress-depth:
# Specifies the maximum amount of decompressed data to obtain,
# set 0 for unlimited.
swf-decompression:
  enabled: yes
  type: both
  compress-depth: 0
  decompress-depth: 0
```

9.3.12.28.1 Примечания

- Если тело HTTP использует gzip или deflate, file_data будет соответствовать для распакованных данных.
- Отрицательное совпадение зависит от проверки по частям. Например, ‘content:!”<html”;’ не может совпасть с первой частью, но затем может совпасть со вторым. Чтобы избежать этого, используйте настройку глубины. Настройка глубины учитывает размер тела. Предполагая, что response-body-minimal-inspect-size больше 1k, ‘content:!”<html”; depth:1024;’ может совпасть только в том случае, если шаблон '<html' отсутствует в первом проверенном фрагменте.
- file_data также может использоваться с SMTP.

9.3.13 Ключевые слова операций с файлами

Suricata поставляется с несколькими ключевыми словами для проверки соответствия свойствам файла. Они зависят от правильно настроенного **извлечения файлов**.

9.3.13.1 filename

Проверяет совпадение с именем файла.
Синтаксис:

```
filename:<string>;
```

Пример:

```
filename:"secret";
```

9.3.13.2 fileext

Проверяет совпадение с расширением имени файла.
Синтаксис:

```
fileext:<string>;
```

Пример:

```
fileext:"jpg";
```

9.3.13.3 filemagic

Проверяет совпадение с информацией, которую libmagic возвращает о файле.
Синтаксис:

```
filemagic:<string>;
```

Пример:

```
filemagic:"executable for MS Windows";
```

> Поскольку версии libmagic различаются в зависимости от установки, возвращаемая информация также может немного измениться. Смотрите также # 437.

9.3.13.4 filestore

Сохраняет файлы на диск, если сигнатура совпадает.
Синтаксис:

```
filestore:<direction>,<scope>;
```

Направление может быть:

- request /to_server: сохранить файл в запросе в направлении /to_server;
- response /to_client: сохранить файл в направлении ответа /to_client;
- both: сохранить обоих направлениях.

Scope может быть:

- file: сохранить только соответствующий файл (для совпадений filename, fileext, filemagic);
- tx: сохранить все файлы из соответствующей HTTP-транзакции;
- ssn/flow: сохранить все файлы из TCP-сессии/потока.

Если направление и область действия не указаны, направление будет таким же, как правило, и область действия будет для каждого файла.

9.3.13.5 filemd5

Сопоставляет файл **MD5** со списком контрольных сумм MD5.
Синтаксис:

```
filemd5:[!]filename;
```

Имя файла расширено, чтобы включить правило dir. По умолчанию это будет /etc/suricata/rules/filename. Используйте восклицательный знак, чтобы получить отрицание соответствия. Действует для белого списка.

Примеры:

```
filemd5:md5-blacklist;
filemd5:!md5-whitelist;
```

Формат файла

Формат файла прост. Это текстовый файл с одним md5 на строку, в начале строки, в шестнадцатеричной записи. Если в строке есть дополнительная информация, она игнорируется.

Вывод из md5sum:

```
2f8d0355f0032c3e6311c6408d7c2dc2  util-path.c
b9cf5cf347a70e02fde975fc4e117760  util-pidfile.c
02aaa6c3f4dbae65f5889eeb8f2bbb8d  util-pool.c
dd5fc1ee7f2f96b5f12d1a854007a818  util-print.c
```

Просто MD5 также хорош:

```
2f8d0355f0032c3e6311c6408d7c2dc2
b9cf5cf347a70e02fde975fc4e117760
02aaa6c3f4dbae65f5889eeb8f2bbb8d
```

```
dd5fc1ee7f2f96b5f12d1a854007a818
```

Требования к оперативной памяти:

Каждый MD5 использует 16 байт памяти. 20 миллионов MD5 используют около 310 МБ памяти.

9.3.13.6 filesha1

Сопоставляет файл SHA1 со списком контрольных сумм SHA1.

Синтаксис:

```
filesha1:[!]filename;
```

Имя файла расширено, чтобы включить правило dir. По умолчанию это будет /etc/suricata/rules/filename. Используйте восклицательный знак, чтобы получить отрицание соответствия. Действует для белого списка.

Примеры:

```
filesha1:sha1-blacklist;
filesha1:!sha1-whitelist;
```

Формат файла

Формат файла аналогичен md5.

9.3.13.7 filesha256

Сопоставляет файл SHA256 со списком контрольных сумм SHA256.

Синтаксис:

```
filesha256:[!]filename;
```

Имя файла расширено, чтобы включить правило dir. По умолчанию это будет /etc/suricata/rules/filename. Используйте восклицательный знак, чтобы получить отрицание соответствия. Действует для белого списка.

Примеры:

```
filesha256:sha256-blacklist;
filesha256:!sha256-whitelist;
```

Формат файла

Формат файла аналогичен md5.

9.3.13.8 filesize

Сопоставляет размер файла с размером файла при его передаче.

Синтаксис:

```
filesize:<value>;
```

Возможные единицы: КБ, МБ и ГБ, без единиц по умолчанию – байты.

Примеры:

```
filesize:100; # exactly 100 bytes
filesize:100<>200; # greater than 100 and smaller than 200
filesize:>100MB; # greater than 100 megabytes
filesize:<100MB; # smaller than 100 megabytes
```

Для файлов, которые не полностью отслеживаются из-за потери пакета или отмечается, что stream.reassembly.depth достигает значения «больше чем», это связано с тем, что Suricata

может знать, что файл больше, чем значение, но не может знать, был бы конечный размер в пределах диапазона, соответствовал значению или был меньше его.

9.3.14 Ключевые слова DNS

Есть еще несколько модификаторов содержимого, гарантирующих, что сигнатура проверяет определенную часть сетевого трафика.

9.3.14.1 dns.opcode

Это ключевое слово сопоставляет по **opcode** в флагах заголовка DNS.

9.3.14.1.1 Синтаксис

```
dns.opcode:[!]<number>
```

9.3.14.1.2 Примеры

Сопоставление запросов DNS и ответов с **opcode** 4:

```
dns.opcode:4;
```

Сопоставление запросов DNS, где **opcode** не равен 0:

```
dns.opcode:!0;
```

9.3.14.2 dns.query

С помощью dns.query проверяются DNS-запросы. Ключевое слово dns.query немного отличается от обычных модификаторов содержимого. При использовании в правиле, оно влияет на все содержимое после него. Пример:

```
alert dns any any -> any any (msg:”Test dns.query опция”; dns.query; content:”google”; nocase; sid:1;)
```

Ключевое слово **dns.query** влияет на все следующее содержимое, пока pkt_data не используется или не достигает конца правила.

9.3.14.2.1 Нормализованный буфер

Буфер содержит буквенное имя домена:

- Значения <длина> (как видно из сырого DNS-запроса), являются буквенными '.' символами;
- Нет начального значения <длина>;
- Нет завершающего байта NULL (0x00) (используется отрицательный относительный isdataat для соответствия концу).

Пример DNS-запроса для «mail.google.com» (для удобства чтения шестнадцатеричные значения кодируются между вертикальными линиями):

DNS-запрос (фрагмент)

```
|04|mail|06|google|03|com|00|
```

Буфер: dns.query:

```
mail.google.com
```

9.3.15 Ключевые слова SSL/TLS

Suricata поддерживает несколько ключевых слов правил для сопоставления различным свойствам TLS/SSL-рукопожатия. Совпадения – это совпадения по включенным строкам.

9.3.15.1 tls.cert_subject

Сопоставление полю темы сертификата TLS/SSL.
Примеры:

```
tls.cert_subject; content:"CN=*.googleusercontent.com"; isdataat:!1,relative;
tls.cert_subject; content:"google.com"; nocase; pcre:"/google.com$/";
```

tls.cert_subject является «липким буфером»
tls.cert_subject может использоваться как fast_pattern.
tls.cert_subject заменяет предыдущее имя ключевого слова: tls_cert_subject. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.15.2 tls.cert_issuer

Сопоставляет поле эмитента сертификата TLS/SSL.
Примеры:

```
tls.cert_issuer; content:"WoSign"; nocase; isdataat:!1,relative;
tls.cert_issuer; content:"StartCom"; nocase; pcre:"/StartCom$/";
```

tls.cert_issuer является «липким буфером».
tls.cert_issuer может использоваться как fast_pattern.
tls.cert_issuer заменяет предыдущее имя ключевого слова: tls_cert_issuer. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.15.3 tls.cert_serial

Сопоставляет по серийному номеру сертификата.
Пример:

```
alert tls any any -> any any (msg:"match cert serial"; \
 tls.cert_serial; content:"5C:19:B7:B1:32:3B:1C:A1"; sid:200012;)
```

tls.cert_serial является «липким буфером».
tls.cert_serial может использоваться как fast_pattern.
tls.cert_serial заменяет предыдущее имя ключевого слова: tls_cert_serial. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.15.4 tls.cert_fingerprint

Сопоставление по отпечатку SHA-1 в сертификате.
Пример:

```
alert tls any any -> any any (msg:"match cert fingerprint"; \
 tls.cert_fingerprint; \
 content:"4a:a3:66:76:82:cb:6b:23:bb:c3:58:47:23:a4:63:a7:78:a4:a1:18"; \
 sid:200023;)
```

tls.cert_fingerprint является «липким буфером».
tls.cert_fingerprint может использоваться как fast_pattern.
tls.cert_fingerprint заменяет предыдущее имя ключевого слова: tls_cert_fingerprint можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.15.5 tls.sni

Сопоставление по полю индикации имени сервера TLS/SSL.
Примеры:

```
tls.sni; content:"oisf.net"; nocase; isdataat:!1,relative;
tls.sni; content:"oisf.net"; nocase; pcre:"/oisf.net$/";
```

tls.sni является «липким буфером».
tls.sni может использоваться как fast_pattern.
tls.sni заменяет предыдущее имя ключевого слова: tls_sni. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.15.6 tls_cert_notbefore

Сопоставление по полю NotBefore в сертификате.
Пример:

```
alert tls any any -> any any (msg:"match cert NotBefore"; \
  tls_cert_notbefore:1998-05-01<>2008-05-01; sid:200005;)
```

9.3.15.7 tls_cert_notafter

Сопоставление по полю NotAfter в сертификате.
Пример:

```
alert tls any any -> any any (msg:"match cert NotAfter"; \
  tls_cert_notafter:>2015; sid:200006;)
```

9.3.15.8 tls_cert_expired

Сопоставление возвращает true, если срок действия сертификата истек. Оценивает срок действия сертификата.
Применение:

```
tls_cert_expired;
```

9.3.15.9 tls_cert_valid

Сопоставление возвращает true, если срок действия сертификата не истек. Только оценивает срок действия. Н*е* проверяет цепочку сертификатов, является противоположностью tls_cert_expired.

Применение:

```
tls_cert_valid;
```

9.3.15.10 tls.certs

Выполняет «сырое» сопоставление для каждого из сертификатов в цепочке сертификатов TLS.
Пример:

```
alert tls any any -> any any (msg:"match bytes in TLS cert"; tls.certs; \
  content:"|06 09 2a 86|"; sid:200070;)
```

tls.certs является «липким буфером».
tls.certs может использоваться как fast_pattern.

9.3.15.11 tls.version

Сопоставление по согласованной версии TLS / SSL.
Поддерживаемые значения: “1.0”, “1.1”, “1.2”, “1.3”.
Также возможно сопоставление версий с использованием шестнадцатеричной строки.
Примеры:

```
tls.version:1.2;
tls.version:0x7f12;
```

Первый пример сопоставляет TLS v1.2, последний пример сопоставляет TLS v1.3 draft 16.

9.3.15.12 ssl_version

Сопоставлении по записи версии SSL/TLS.
Поддерживаемые значения “sslv2”, “sslv3”, “tls1.0”, “tls1.1”, “tls1.2”, “tls1.3”
Пример:

```
alert tls any any -> any any (msg:"match TLSv1.2"; \
  ssl_version:tls1.2; sid:200030;)
```

Также можно сопоставить несколько версий одновременно.
Пример:

```
alert tls any any -> any any (msg:"match SSLv2 and SSLv3"; \
  ssl_version:sslv2,sslv3; sid:200031;)
```

9.3.15.13 tls.subject

Сопоставление по полю темы TLS/SSL-.
Пример:

```
tls.subject:"CN=*.googleusercontent.com"
```

Чувствительно к регистру, нельзя использовать ‘nocase’.
Унаследованное ключевое слово tls.cert_subject является заменой.

9.3.15.14 tls.issuerdn

Сопоставление по полю IssuerDN TLS/SSL-сертификат.
Пример:

```
tls.issuerdn:!"CN=Google-Internet-Authority"
```

Чувствительно к регистру, нельзя использовать ‘nocase’.
Унаследованное ключевое слово tls.cert_issuer является заменой.

9.3.15.15 tls.fingerprint

Сопоставление по отпечатку SHA1 TLS/SSL-сертификата
Пример:

```
tls.fingerprint:!"f3:40:21:48:70:2c:31:bc:b5:aa:22:ad:63:d6:bc:2e:b3:46:e2:5a"
```

Чувствительно к регистру, нельзя использовать ‘nocase’.
Буфер tls.fingerprint имеет нижний регистр, поэтому следует использовать строчные буквы для сопоставления.

9.3.15.16 tls.store

Сохраняет TLS/SSL-сертификат на диск.

9.3.15.17 ssl_state

Ключевое слово ssl_state сопоставляет состояние соединения SSL. Возможные состояния: client_hello, server_hello, client_keyx, server_keyx и unknown. Можно указать несколько состояний с помощью | (ИЛИ) для проверки любого из указанных состояний.

Поддержка отрицания пока недоступна.

9.3.16 Ключевые слова SSH

Suricata имеет несколько ключевых слов для правил, которые соответствуют различным элементам SSH-соединений.

9.3.16.1 ssh.proto

Сопоставление используемой версии протокола SSH. ssh.proto является липким буфером и может использоваться в качестве быстрого шаблона. ssh.proto заменяет предыдущее имя буфера: ssh_proto. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать существующие правила для использования нового имени.

Формат:

```
ssh.proto;
```

Пример:

```
alert ssh any any -> any any (msg:"match SSH protocol version"; ssh.proto; content:"2.0"; sid:1000010;)
```

Приведенный выше пример сопоставляет SSH-соединения с SSH версии 2.0.

9.3.16.2 ssh.software

Сопоставляет строку ПО из баннера SSH. ssh.software является липким буфером и может использоваться в качестве быстрого шаблона.

ssh.software заменяет предыдущее имя ключевого словas: ssh_software & ssh.softwareversion. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

Формат:

```
ssh.software;
```

Пример:

```
alert ssh any any -> any any (msg:"match SSH software string"; ssh.software; content:"openssh"; nocase; sid:1000020;)
```

Пример выше сопоставляет соединениям SSH, где строка ПО содержит “openssh”.

9.3.16.3 ssh.protoversion

Сопоставляет версию используемого протокола SSH. Значение 2_compat включает SSH версии 1.99.

Формат:

```
ssh.protoversion:[0-9](\.[0-9])?|2_compat;
```

Пример:

```
alert ssh any any -> any any (msg:"SSH v2 compatible"; ssh.protoversion:2_compat; sid:1;)
```

Приведенный выше пример сопоставляет SSH-соединения с SSH версии 2 или 1.99.

```
alert ssh any any -> any any (msg:"SSH v1.10"; ssh.protoversion:1.10; sid:1;)
```

Приведенный выше пример сопоставляет SSH-соединения только с SSH версии 1.10.

9.3.16.4 ssh.softwareversion

Данное ключевое слово устарело. Используйте ssh.software вместо него. Сопоставляет со строкой ПО из баннера SSH.

Пример:

```
alert ssh any any -> any any (msg:”match SSH software string”; ssh.softwareversion:”OpenSSH”; sid:10000040;)
```

9.3.17 Ключевые слова JA3

Suricata поставляется с интеграцией JA3 **(https://github.com/salesforce/ja3).** JA3 используется для идентификации клиентов TLS.

JA3 должен быть включен в конфигурационном файле Suricata (установите для «app-layer.protocols.tls.ja3-fingerprints» значение «yes»)

JA3 также требует сборки Suricata с поддержкой LibNSS.

9.3.17.1 ja3.hash

Сопоставляет по хэшу JA3 (md5).

Пример:

```
alert tls any any -> any any (msg:"match JA3 hash"; \
   ja3.hash; content:"e7eca2baf4458d095b7f45da28c16c34"; \
   sid:100001;)
```

ja3.hash является «липким буфером».

ja3.hash может использоваться как fast_pattern.

ja3.hash заменяет предыдущее имя ключевого слова: ja3_hash. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.17.2 ja3.string

Сопоставляет по строке JA3.

Пример:

```
alert tls any any -> any any (msg:"match JA3 string"; \
   ja3.string; content:"19-20-21-22"; \
   sid:100002;)
```

ja3.string является «липким буфером».

ja3.string может использоваться как fast_pattern.

ja3.string заменяет предыдущее имя ключевого слова: ja3_string. Можно продолжать использовать предыдущее имя, но рекомендуется преобразовать правила для использования нового имени.

9.3.17.3 ja3s.hash

Сопоставляет по хешу JA3S (md5).

Пример:

```
alert tls any any -> any any (msg:"match JA3S hash"; \
   ja3s.hash; content:"b26c652e0a402a24b5ca2a660e84f9d5"; \
```

```
  sid:100003;)
```

ja3s.hash является «липким буфером».
ja3s.hash может использоваться как fast_pattern.

9.3.17.4 ja3s.string

Сопоставляет по строке JA3S.
Пример:

```
alert tls any any -> any any (msg:"match on JA3S string"; \
  ja3s.string; content:"771,23-35"; sid:100004;)
```

ja3s.string является «липким буфером».
ja3s.string может использоваться как fast_pattern.

9.3.18 Ключевое слово Modbus

Ключевое слово modbus можно использовать для сопоставления различных свойств запросов Modbus.

Существует три способа использования этого ключевого слова:

- Сопоставление свойств функций с настройкой “function”;
- Сопоставление непосредственно при доступе к данным с настройкой “access”;
- Сопоставление идентификатора устройства только с настройкой «unit» или с предыдущей настройкой “function” или “access”.

С помощью настройки function можно сопоставить:

- Действие на основе поля кода функции и кода подфункции, когда это применимо;
- Одна из трех категорий функций Modbus;
- Общедоступные функции, которые определяются в качестве общедоступными (установка “public”);
- Пользовательские функции (установка “user”)
- Зарезервированные функции, которые предназначены для проприетарных расширений Modbus (ключевое слово “reserved”)
- Одна из двух подгрупп общедоступных функций:
  - назначенные функции, определение которых уже дано в спецификации Modbus (ключевое слово “assigned”);
  - неназначенные функции, которые зарезервированы для будущего использования (ключевое слово “unassigned”).

Синтаксис:

```
modbus: function <value>
modbus: function <value>, subfunction <value>
modbus: function [!] <assigned | unassigned | public | user | reserved | all>
```

Символ '!' означает отрицание.

Примеры:

```
modbus: function 21               # Write File record function
modbus: function 4, subfunction 4  # Force Listen Only Mode (Diagnostics) function
modbus: function assigned          # defined by Modbus Application Protocol Specification
V1.1b3
modbus: function public          # validated by the Modbus.org community
modbus: function user            # internal use and not supported by the specification
```

```
modbus: function reserved          # used by some companies for legacy products and not available for public use
modbus: function !reserved         # every function but reserved function
```

С помощью настройки access можно сопоставить:

- Тип доступа к данным (чтение или запись);
- Доступ к одной из основных таблиц (дискретные входные данные, свертки, входные регистры и регистры хранения);
- Диапазон адресов доступа;
- Значение в письменной форме.

Синтаксис:

```
modbus: access <read | write>
modbus: access read <discretes | coils | input | holding>
modbus: access read <discretes | coils | input | holding>, address <value>
modbus: access write < coils | holding>
modbus: access write < coils | holding>, address <value>
modbus: access write < coils | holding>, address <value>, value <value>
```

С настройкой _<value>_ сопоставляется с адресом или значением, когда к ним обращаются или пишут следующим образом:

```
address 100     # exactly address 100
address 100<>200 # greater than address 100 and smaller than address 200
address >100    # greater than address 100
address <100    # smaller than address 100
```

Примеры:

```
modbus: access read                          # Read access
modbus: access write                         # Write access
modbus: access read input                    # Read access to Discretes Input table
modbus: access write coils                   # Write access to Coils table
modbus: access read discretes, address <100          # Read access at address smaller than 100 of Discretes Input table
modbus: access write holding, address 500, value >200  # Write value greater than 200 at address 500 of Holding Registers table
```

С помощью настройки unit, можно сопоставить ведомый адрес MODBUS удаленного устройства, подключенного к подсети за мостом или шлюзом. IP-адрес назначения идентифицирует сам мост, и мост использует идентификатор модуля MODBUS для пересылки запроса на правильное подчиненное устройство.

Синтаксис:

```
modbus: unit <value>
modbus: unit <value>, function <value>
modbus: unit <value>, function <value>, subfunction <value>
modbus: unit <value>, function [!] <assigned | unassigned | public | user | reserved | all>
modbus: unit <value>, access <read | write>
modbus: unit <value>, access read <discretes | coils | input | holding>
modbus: unit <value>, access read <discretes | coils | input | holding>, address <value>
modbus: unit <value>, access write < coils | holding>
modbus: unit <value>, access write < coils | holding>, address <value>
modbus: unit <value>, access write < coils | holding>, address <value>, value <value>
```

С настройкой _<value>_ сопоставляется с адресом или значением, когда к ним обращаются или пишут следующим образом:

```
unit 10     # exactly unit identifier 10
unit 10<>20 # greater than unit identifier 10 and smaller than unit identifier 20
unit >10    # greater than unit identifier 10
unit <10    # smaller than unit identifier 10
```

Примеры:

```
modbus: unit 10                                          # Unit identifier 10
modbus: unit 10, function 21                             # Unit identifier 10 and write File record function
modbus: unit 10, function 4, subfunction 4               # Unit identifier 10 and force Listen Only Mode (Diagnostics) function
modbus: unit 10, function assigned                       # Unit identifier 10 and assigned function
modbus: unit 10, function !reserved                      # Unit identifier 10 and every function but reserved function
modbus: unit 10, access read                             # Unit identifier 10 and Read access
modbus: unit 10, access write coils                      # Unit identifier 10 and Write access to Coils table
modbus: unit >10, access read discretes, address <100    # Greater than unit identifier 10 and Read access at address smaller than 100 of Discretes Input table
modbus: unit 10<>20, access write holding, address 500, value >200    # Greater than unit identifier 10 and smaller than unit identifier 20 and Write value greater than 200 at address 500 of Holding Registers table
```

Адреса чтения и записи начинаются с 1. Поэтому, если система использует начало с 0, то необходимо добавить 1 к значению адреса.

В соответствии с Руководством по внедрению обмена сообщениями MODBUS по TCP/IP V1.0b рекомендуется держать TCP-соединение открытым с удаленного устройства, а не открывать и закрывать его для каждой транзакции MODBUS/TCP. В этом случае важно установить неограниченную глубину повторной сборки потока (stream.reassembly.depth: 0).

В соответствии с Руководством по внедрению обмена сообщениями MODBUS в TCP/IP V1.0b адреса подчиненного устройства MODBUS на последовательной линии назначаются от 1 до 247 (десятичное число). Адрес 0 используется как широковещательный адрес.

9.3.19 Ключевые слова DNP3

Ключевые слова DNP3 могут использоваться для сопоставления полей в декодированных сообщениях DNP3. Ключевые слова основаны на ключевых словах Snort DNP3 и нацелены на 100% совместимость.

9.3.19.1 dnp3_func

Ключевое слово будет сопоставлять по коду функции приложения, найденному в запросе и ответах DNP3. Может быть указано как целочисленное значение или символическое имя кода функции.

9.3.19.1.1 Синтаксис

```
dnp3_func:<value>;
```

Где значение value является одним из:

- Целочисленное значение от 0 до 255 включительно.
- Кодовое название функции:
  - confirm
  - read

- write
- select
- operate
- direct_operate
- direct_operate_nr
- immed_freeze
- immed_freeze_nr
- freeze_clear
- freeze_clear_nr
- freeze_at_time
- freeze_at_time_nr
- cold_restart
- warm_restart
- initialize_data
- initialize_appl
- start_appl
- stop_appl
- save_config
- enable_unsolicited
- disable_unsolicited
- assign_class
- delay_measure
- record_current_time
- open_file
- close_file
- delete_file
- get_file_info
- authenticate_file
- abort_file
- activate_config
- authenticate_req
- authenticate_err
- response
- unsolicited_response
- authenticate_resp

9.3.19.2 dnp3_ind

Ключевое слово сопоставляет флагам внутреннего индикатора DNP3 в заголовке приложения ответа.

9.3.19.2.1 Синтаксис

```
dnp3_ind:<flag>{,<flag>..}
```

Где flag – это имя следующего внутреннего индикатора:

- all_stations
- class_1_events
- class_2_events
- class_3_events
- need_time
- local_control
- device_trouble

- device_restart
- no_func_code_support
- object_unknown
- parameter_error
- event_buffer_overflow
- already_executing
- config_corrupt
- reserved_2
- reserved_1

Ключевое слово будет сопоставлять любой из флагов в списке установлены. Чтобы сопоставить несколько флагов (совпадение типа AND), используйте dnp3_ind для каждого установленного флага.

9.3.19.2.2 Примеры

```
dnp3_ind:all_stations;
dnp3_ind:class_1_events,class_2_events;
```

9.3.19.3 dnp3_obj

Ключевое слово сопоставляет объекты данных приложения DNP3.

9.3.19.3.1 Синтаксис

```
dnp3_obj:<group>,<variation>
```

Где <group> и <variation> – целочисленные значения от 0 до 255 включительно.

9.3.19.4 dnp3_data

Ключевое слово приводит к сопоставлению следующих параметров содержимого в повторно собранном буфере приложения. Пересобранный буфер приложения представляет собой фрагмент DNP3 с удаленными CRC (которые появляются каждые 16 байтов) и будет полным фрагментом, возможно, повторно собранным из нескольких кадров канального уровня DNP3.

9.3.19.4.1 Синтаксис

```
dnp3_data;
```

9.3.19.4.2 Пример

```
dnp3_data; content:"|c3 06|";
```

9.3.20 Ключевые слова ENIP/CIP

Ключевые слова enip_command и cip_service могут использоваться для сопоставления различных свойств запросов ENIP.

Существует три способа использования этого ключевого слова:

- Сопоставление по команде ENIP с параметром “enip_command”;
- Сопоставление в CIP Service с параметром “cip_service”;
- Сопоставление команды ENIP и службы CIP с “enip_command” и “cip_service” вместе.

Для команды ENIP сопоставляется поле команды, найденное в инкапсуляции ENIP.

Для сервиса CIP используется максимум 3 значения, разделенные запятыми, представляющие Сервис, Класс и Атрибут. Эти значения описаны в спецификации CIP. Классы

CIP связаны с их Сервисом, а Атрибуты CIP связаны с их Сервисом. Если нужно сопоставлять только до Сервиса, укажите только значение Сервиса. Если требуется сопоставлять Атрибут CIP, то следует указать все 3 значения.

Синтаксис:

```
enip_command:<value>
cip_service:<value(s)>
enip_command:<value>, cip_service:<value(s)>
```

Примеры:

```
enip_command:99
cip_service:75
cip_service:16,246,6
enip_command:111, cip_service:5
```

9.3.21 Ключевые слова FTP/FTP-DATA

9.3.21.1 ftpdata_command

Фильтрует канал данных ftp на основе команды, используемой на канале команд FTP. В настоящее время поддерживаются команды RETR (получить в файл) и STOR (положить в файл).

Синтаксис:

```
ftpdata_command:(retr|stor)
```

Примеры:

```
ftpdata_command:retr
ftpdata_command:stor
```

Пример сигнатуры:

```
alert ftp-data any any -> any any (msg:"FTP store password"; filestore; filename:"password";
ftpdata_command:stor; sid:3; rev:1;)
```

9.3.21.2 ftpbounce

Обнаружение атак FTP-отказов.

Синтаксис:

```
ftpbounce
```

9.3.22 Ключевые слова Kerberos

9.3.22.1 krb5_msg_type

Тип сообщения Kerberos (целое число).

Значения определены в RFC4120. Общие значения:

- 10 (AS-REQ)
- 11 (AS-REP)
- 12 (TGS-REQ)
- 13 (TGS-REP)
- 14 (AP-REQ)
- 15 (AP-REP)
- 30 (ERROR)

Синтаксис:

```
krb5_msg_type:<number>
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"Kerberos 5 AS-REQ message"; krb5_msg_type:10; sid:3; rev:1;)
```

9.3.22.2 krb5_cname

Имя клиента Kerberos (для сообщений AS-REQ и TGS-REQ).

Если имя клиента из сообщения Kerberos состоит из нескольких частей, имя сравнивается с каждой частью, и сопоставление будет успешным, если они совпадут.

Сравнение чувствительно к регистру.

Синтаксис:

```
krb5_cname; content:"name";
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"Kerberos 5 des server name"; krb5_cname; content:"des"; sid:4; rev:1;)
```

krb5_cname является «липким буфером».

krb5_cname может использоваться как fast_pattern.

9.3.22.3 krb5_sname

Имя сервера Kerberos указанное в заявке (для сообщений AS-REQ и TGS-REQ) или в сообщении об ошибке.

Если имя сервера из сообщения Kerberos состоит из нескольких частей, имя сравнивается с каждой частью, и сопоставление будет успешным, если они совпадут.

Сравнение чувствительно к регистру.

Синтаксис:

```
krb5_sname; content:"name";
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"Kerberos 5 krbtgt server name"; krb5_sname; content:"krbtgt"; sid:5; rev:1;)
```

krb5_sname является «липким буфером».

krb5_sname может использоваться как fast_pattern.

9.3.22.4 krb5_err_code

Код ошибки Kerberos (целое число). Это поле сопоставляется только с сообщениями об ошибках Kerberos.

Список кодов ошибок приведен в RFC 4120.

Синтаксис:

```
krb5_err_code:<number>
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"Kerberos 5 error C_PRINCIPAL_UNKNOWN"; krb5_err_code:6; sid:6; rev:1;)
```

9.3.22.5 krb5.weak_encryption (событие)

Событие возникает, если выбранные сервером параметры шифрования являются слабыми или устарели. Например, используя размер ключа меньше 128 или устаревшие шифры, такие как DES.

Синтаксис:

```
app-layer-event:krb5.weak_encryption
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"SURICATA Kerberos 5 weak encryption parameters"; flow:to_client; app-layer-event:krb5.weak_encryption; classtype:protocol-command-decode; sid:2226001; rev:1;)
```

9.3.22.6 krb5.malformed_data (событие)

Событие, возникает при ошибке декодирования протокола.
Синтаксис:

```
app-layer-event:krb5.malformed_data
```

Пример сигнатуры:

```
alert krb5 any any -> any any (msg:"SURICATA Kerberos 5 malformed request data"; flow:to_server; app-layer-event:krb5.malformed_data; classtype:protocol-command-decode; sid:2226000; rev:1;)
```

9.3.23 Ключевые слова SNMP

9.3.23.1 snmp.version

Версия протокола SNMP (целое число). Ожидаемые значения: 1, 2 (для версии 2c) или 3.
Синтаксис:

```
snmp.version:[op]<number>
```

Версия может быть точно сопоставлена или сравнена при использовании параметра _op_ setting:

```
snmp.version:3    # exactly 3
snmp.version:<3   # smaller than 3
snmp.version:>=2  # greater or equal than 2
```

Пример сигнатуры:

```
alert snmp any any -> any any (msg:"old SNMP version (<3)"; snmp.version:<3; sid:1; rev:1;)
```

9.3.23.2 snmp.community

Строки SNMP похожи на пароли для сообщений SNMP в версиях 1 и 2c. В версии 3 строка, вероятно, будет зашифрована. Это ключевое слово не будет сопоставлять, если значение недоступно

Значением по умолчанию для строки сообщества, доступной только для чтения, часто является «public» и «private» для строки сообщества «read-write».

Сравнение чувствительно к регистру.
Синтаксис:

```
snmp.community; content:"private";
```

Пример сигнатуры:

```
alert snmp any any -> any any (msg:"SNMP community private"; snmp.community; content:"private"; sid:2; rev:1;)
```

snmp.community является «липким буфером».
snmp.community может использоваться как fast_pattern.

9.3.23.3 snmp.pdu_type

SNMP PDU type (целое число).

Общие значения:

- 0: GetRequest
- 1: GetNextRequest
- 2: Response
- 3: SetRequest
- 4: TrapV1 (obsolete, was the old Trap-PDU in SNMPv1)
- 5: GetBulkRequest
- 6: InformRequest
- 7: TrapV2
- 8: Report

Это ключевое слово не будет сопоставлять, если значение недоступно внутри (например, зашифрованное сообщение SNMP v3).

Синтаксис:

```
snmp.pdu_type:<number>
```

Пример сигнатуры:

```
alert snmp any any -> any any (msg:"SNMP response"; snmp.pdu_type:2; sid:3; rev:1;)
```

9.3.24 Ключевые слова Base64

Suricata поддерживает декодирование данных Base64 из буферов и сопоставление с декодированными данными.

Это достигается с помощью двух ключевых слов, base64_decode и base64_data. Оба ключевых слова должны использоваться для создания оповещения.

9.3.24.1 base64_decode

Декодирует данные base64 из буфера и делает их доступными для функции base64_data.

Синтаксис:

```
base64_decode:bytes <value>, offset <value>, relative;
```

Опция bytes указывает, сколько байтов Suricata должна декодировать и сделать доступным для base64_data. Декодирование остановится в конце буфера.

Опция offset определяет, сколько байтов Suricata должна пропустить перед декодированием. Байты пропускаются относительно начала буфера полезной нагрузки, если relative не установлен.

Опция relative определяет начало декодирования относительно предыдущего совпадения содержимого. Поведение по умолчанию – запуск в начале буфера. Данная опция пропускает offset байтов относительно предыдущего совпадения.

Совпадение содержимого, которое следует относительно декодировать, должно быть первым совпадением в потоке.

9.3.24.2 base64_data

base64_data является липким буфером (sticky buffer).

Включает сопоставление содержимого для данных, ранее декодированных с помощью base64_decode.

9.3.24.3 Пример

Вот пример правила сопоставления для строки «test» в кодировке base64, которая находится внутри буфера http_uri.

Декодирование начинается относительно известной строки «somestring» с известным смещением 1. Это должно быть первое появление «somestring» в буфере.

Пример:

```
Buffer content:
http_uri = "GET /en/somestring&dGVzdAo=&not_base64"

Rule:
alert http any any -> any any (msg:"Пример"; http.uri; content:"somestring"; \
    base64_decode:bytes 8, offset 1, relative; \
    base64_data; content:"test"; sid:10001; rev:1;)

Buffer content:
http_uri = "GET /en/somestring&dGVzdAo=&not_base64"

Rule:
alert http any any -> any any (msg:"Пример"; content:"somestring"; http_uri; \
    base64_decode:bytes 8, offset 1, relative; \
    base64_data; content:"test"; sid:10001; rev:1;)
```

9.3.25 Ключевые слова SIP

Ключевые слова SIP реализованы в виде липких буферов и могут использоваться для сопоставления полей в сообщениях SIP, как показано в таблице 20.

Таблица 20 – Ключевые слова для сопоставления полей в сообщениях SIP

Ключевое слово	**Направление**
sip.method	Запрос
sip.uri	Запрос
sip.request_line	Запрос
sip.stat_code	Ответ
sip.stat_msg	Ответ
sip.response_line	Ответ
sip.protocol	Оба

9.3.25.1 sip.method

Это ключевое слово соответствует методу, найденному в SIP-запросе.

9.3.25.1.1 Синтаксис

```
sip.method; content:<method>;
```

Примеры использования метода:

- INVITE
- BYE

- REGISTER
- CANCEL
- ACK
- OPTIONS

9.3.25.1.2 Примеры

```
sip.method; content:"INVITE";
```

9.3.25.2 sip.uri

Это ключевое слово сопоставляет URI, найденный в SIP-запросе.

9.3.25.2.1 Синтаксис

```
sip.uri; content:<uri>;
```

Где <uri> – это uri, который следует схеме SIP URI.

9.3.25.2.2 Примеры

```
sip.uri; content:"sip:sip.url.org";
```

9.3.25.3 sip.request_line

Ключевое слово принудительно проверяет всю строку SIP-запроса.

9.3.25.3.1 Синтаксис

```
sip.request_line; content:<request_line>;
```

Где <request_line> является частичной или полной строкой.

9.3.25.3.2 Примеры

```
sip.request_line; content:"REGISTER sip:sip.url.org SIP/2.0"
```

9.3.25.4 sip.stat_code

Это ключевое слово сопоставляет код статуса, найденный в ответе SIP.

9.3.25.4.1 Синтаксис

```
sip.stat_code; content:<stat_code>
```

Где <status_code> принадлежит к одной из следующих групп кодов:

- 1xx – предварительные ответы
- 2xx – успешные ответы
- 3xx – ответы перенаправления
- 4xx – ответы сбоя клиента
- 5xx – ответы сбоя сервера
- 6xx – ответы глобального сбоя

9.3.25.4.2 Примеры

```
sip.stat_code; content:"100";
```

9.3.25.5 sip.stat_msg

Ключевое слово сопоставляет сообщение о статусе, найденное в ответе SIP.

9.3.25.5.1 Синтаксис

```
sip.stat_msg; content:<stat_msg>
```

Где <stat_msg> – фраза причины, связанная с кодом состояния.

9.3.25.5.2 Примеры

```
sip.stat_msg; content:"Trying";
```

9.3.25.6 sip.response_line

Ключевое слово принудительно проверяет всю строку SIP-ответа.

9.3.25.6.1 Синтаксис

```
sip.response_line; content:<response_line>;
```

Где <response_line> является частичной или полной строкой.

9.3.25.6.2 Примеры

```
sip.response_line; content:"SIP/2.0 100 OK"
```

9.3.25.7 sip.protocol

Это ключевое слово сопоставляет по полю протокола из строки запроса или ответа SIP. Если строка ответа «SIP/2.0 100 OK», то этот буфер будет содержать «SIP/2.0».

9.3.25.7.1 Синтаксис

```
sip.protocol; content:<protocol>
```

Где <protocol> – версия протокола SIP.

9.3.25.7.2 Пример

```
sip.protocol; content:"SIP/2.0"
```

9.3.26 Ключевые слова уровня универсальных приложений

9.3.26.1 app-layer-protocol

Сопоставление определенного протокола уровня приложений.
Синтаксис:

```
app-layer-protocol:[!]<protocol>;
```

Примеры:

```
app-layer-protocol:ssh;
app-layer-protocol:!tls;
app-layer-protocol:failed;
```

Специальное значение 'fail' может использоваться для сопоставления потоков, в которых не удалось обнаружить протокол. Это может произойти, если Suricata не знает протокол или, при определенных ‘bail out’ условиях.

9.3.26.1.1 Условия предоставления помощи

Обнаружение протокола прекращается в нескольких случаях:

- Обе стороны проверены,а совпадение не найдено;
- Стороне A не удалось обнаружить, у стороны B вообще нет трафика (например, канал данных FTP);
- Стороне A не удалось обнаружить, сторона B обнаружила слишком мало данных для убедительных выводов.

В последних 2 случаях устанавливается app-layer-event:applayer_proto_detection_skipped.

9.3.26.2 app-layer-event

Сопоставление событий, генерируемых анализаторами уровня приложений и механизмом обнаружения протоколов.

Синтаксис:

```
app-layer-event:<event name>;
```

Примеры:

```
app-layer-event:applayer_mismatch_protocol_both_directions;
app-layer-event:http.gzip_decompression_failed;
```

9.3.26.2.1 Обнаружение протокола

9.3.26.2.1.1 applayer_mismatch_protocol_both_directions

Направления в сторону сервера и клиента имеют разные протоколы. Например, клиент использующий HTTP в направлении SSH-сервера.

9.3.26.2.1.2 applayer_wrong_direction_first_data

Некоторые реализации протокола в Suricata имеют требование в отношении первого направления данных. Анализатор HTTP является примером этого.

9.3.26.2.1.3 applayer_detect_protocol_only_one_direction

Обнаружение протокола успешно выполнено только в одном направлении. Это ожидается для FTP и SMTP.

9.3.26.2.1.4 applayer_proto_detection_skipped

Обнаружение протокола было пропущено из-за **условий Bail out**.

9.3.27 Ключевое слово Xbits

Устанавливает, переключает и проверяет биты, хранящиеся на хосте или ip_pair.

Синтаксис:

```
xbits:<set|unset|isset|isnotset|toggle>,<name>,track <ip_src|ip_dst|ip_pair>;
xbits:<set|unset|isset|toggle>,<name>,track <ip_src|ip_dst|ip_pair> \
   [,expire <seconds>];
```

```
xbits:<set|unset|isset|toggle>,<name>,track <ip_src|ip_dst|ip_pair> \
   [,expire <seconds>];
```

9.3.27.1 Примечания

- Нет разницы между использованием hostbits и xbits с track ip_<src|dst>;
- Если установлен set на запрос клиента и используется track ip_dst, если требуется сопоставить ответ сервера, то проверяйте его (isset) с track ip_src;
- Чтобы исключить оповещения, используйте noalert;
- Параметр toggle перевернет значение xbits.

9.3.27.1.1 Настройки YAML

Биты, которые хранятся на хосте? в таблице хостов. Это означает, что настройки таблицы хостов влияют на hostsbits, а xbits на хост.

Биты, которые хранятся для каждой пары IP в таблице IPPair. Это означает, что настройки таблицы ippair, особенно memcap, влияют на xbits для ip_pair.

9.3.27.1.2 Потоки

Из-за тонких проблем синхронизации между потоками порядок наборов и проверок может быть немного непредсказуемым.

9.3.27.1.3 Unix-сокет

Hostbits можно добавлять, удалять и помещать в списки с использованием сокета unix.

Добавление:

```
suricatasc -c "add-hostbit <ip> <bit name> <expire in seconds>"
suricatasc -c "add-hostbit 1.2.3.4 blacklist 3600"
```

Если hostbit добавляется для существующего hostbit, его таймер истечения будет обновлен.
Удаление:

```
suricatasc -c "remove-hostbit <ip> <bit name>"
suricatasc -c "remove-hostbit 1.2.3.4 blacklist"
```

Список:

```
suricatasc -c "list-hostbit <ip>"
suricatasc -c "list-hostbit 1.2.3.4"
```

Это приводит к:

```
{
   "message":
   {
      "count": 1,
      "hostbits":
         [{
            "expire": 89,
            "name": "blacklist"
         }]
   },
   "return": "OK"
}
```

9.3.27.1.4 Пример

9.3.27.1.4.1 Создание черного списка SSH

Ниже приведен пример правил, поступающих на сервер SSH.

Первые два правила сопоставляют версию программного обеспечения SSH с версиями, часто используемыми ботами. Они отбрасывают трафик и создают 'xbit' 'badssh' для исходного IP-адреса. Срок действия истекает через час:

```
drop ssh any any -> $MYSERVER 22 (msg:"DROP libssh incoming";   \
 flow:to_server,established; ssh.softwareversion:"libssh";     \
 xbits:set, badssh, track ip_src, expire 3600; sid:4000000005;)
drop ssh any any -> $MYSERVER 22 (msg:"DROP PUTTY incoming";    \
 flow:to_server,established; ssh.softwareversion:"PUTTY";      \
 xbits:set, badssh, track ip_src, expire 3600; sid:4000000007;)
```

Тогда следующее правило просто отбрасывает любой входящий трафик на этот сервер, который находится в списке «badssh»:

```
drop ssh any any -> $MYSERVER 22 (msg:"DROP BLACKLISTED";       \
 xbits:isset, badssh, track ip_src; sid:4000000006;)
```

9.3.28 Пороговые ключевые слова

Пороговые значения могут быть настроены для каждого правила, а также глобально, см. **Global-Thresholds.**

9.3.28.1 threshold

Ключевое слово threshold можно использовать для управления частотой оповещений со стороны правила. Поддерживаются 3 режима: threshold, limit и both.

Синтаксис:

```
threshold: type <threshold|limit|both>, track <by_src|by_dst>, count <N>, seconds <T>
```

9.3.28.1.1 Тип “threshold”

Этот тип может использоваться для установки минимального порога для правила до того, как оно генерирует оповещения. Установка порога N означает, что, когда правило соответствует, оповещение генерируется N раз,

Пример:

```
alert tcp !$HOME_NET any -> $HOME_NET 25 (msg:"ET POLICY Inbound Frequent
Emails – Possible Spambot Inbound"; \
flow:established; content:"mail from|3a|"; nocase;                    \
threshold: type threshold, track by_src, count 10, seconds 60;        \
reference:url,doc.emergingthreats.net/2002087; classtype:misc-activity; sid:2002087; rev:10;)
```

Данная сигнатура генерирует оповещение, только если получит 10 входящих сообщений электронной почты или более с одного и того же сервера в течение периода времени в одну минуту.

Если сигнатура устанавливает flowbit, flowint и т. п., то эти действия выполняются для каждого из совпадений.

Действия правил отбрасывания (режим IPS) и отклонения применяются к каждому пакету (а не только к тому, который соответствует пороговому условию).

9.3.28.1.2 Тип “limit”

Этот тип можно использовать против излишнего количества оповещений. Если установлено ограничение N, то оповещения генерируются не более N раз.

Пример:

```
alert http $HOME_NET any -> any $HTTP_PORTS (msg:"ET USER_AGENTS Internet Explorer 6 in use – Significant Security Risk"; \
flow:to_server,established; content:"|0d 0a|User-Agent|3a| Mozilla/4.0 (compatible|3b| MSIE 6.0|3b|"; \
threshold: type limit, track by_src, seconds 180, count 1; \
reference:url,doc.emergingthreats.net/2010706; classtype:policy-violation; sid:2010706; rev:7;)
```

В этом примере самое большее 1 оповещение генерируется для хоста в течение 3 минут, если обнаружен MSIE 6.0.

Если сигнатура устанавливает flowbit, flowint и т. п., то эти действия выполняются для каждого из совпадений.

Действия правил отбрасывания (режим IPS) и отклонения применяются к каждому пакету (а не только к тому, который соответствует пороговому условию).

9.3.28.1.3 Тип “both”

Этот тип является комбинацией типов “threshold” и “limit”. Применяется как пороговое значение, так и ограничение.

Пример:

```
alert tcp $HOME_NET 5060 -> $EXTERNAL_NET any (msg:"ET VOIP Multiple Unauthorized SIP Responses TCP"; \
flow:established,from_server; content:"SIP/2.0 401 Unauthorized"; depth:24; \
threshold: type both, track by_src, count 5, seconds 360; \
reference:url,doc.emergingthreats.net/2003194; classtype:attempted-dos; sid:2003194; rev:6;)
```

Генерируется оповещение только в том случае, если в течение 6 минут было 5 или более ответов «SIP/2.0 401 Unauthorized», и оно будет предупреждаться только один раз за эти 6 минут.

Если сигнатура устанавливает flowbit, flowint и т. п., то эти действия выполняются для каждого из совпадений.

Действия правил отбрасывания (режим IPS) и отклонения применяются к каждому пакету.

9.3.28.2 detection_filter

Ключевое слово detection_filter может использоваться для оповещения о каждом совпадении после достижения порогового значения. Оно отличается от threshold с типом threshold тем, что генерирует оповещение для каждого совпадения правила после достижения начального порогового значения, при котором последнее сбрасывает свой внутренний счетчик и снова выдает оповещение до следующего достижении порогового значения.

Синтаксис:

```
detection_filter: track <by_src|by_dst>, count <N>, seconds <T>
```

Пример:

```
alert http $EXTERNAL_NET any -> $HOME_NET any \
    (msg:"ET WEB_SERVER WebResource.axd access without t (time) parameter – possible ASP padding-oracle exploit"; \
```

```
flow:established,to_server; content:"GET"; http_method; content:"WebResource.axd"; http_uri; nocase; \
content:!"&t="; http_uri; nocase; content:!"&amp|3b|t="; http_uri; nocase; \
detection_filter:track by_src,count 15,seconds 2; \
reference:url,netifera.com/research/; reference:url,www.microsoft.com/technet/security/advisory/2416728.mspx; \
classtype:web-application-attack; sid:2011807; rev:5;)
```

Оповещения каждый раз после 15 или более совпадений в течение 2 секунд.

Если сигнатура устанавливает flowbit, flowint и т. п., эти действия продолжают выполняться для каждого из совпадений.

Действия правил отбрасывания (режим IPS) и отклонения применяются к каждому пакету, который генерирует оповещение.

9.3.29 Ключевое слово IP Reputation

IP Reputation может использоваться в правилах с помощью нового ключевого слова для правил “iprep”.

9.3.29.1 iprep

Директива iprep сопоставляет информацию о репутации IP для хоста.

```
iprep:<side to check>,<category>,<operator>,<reputation score>
```

side to check (сторона для проверки): <any|src|dst|both>
category: краткое имя категории
operator: <, >, =
reputation score (оценка репутации): 1-127
Пример:

```
alert ip $HOME_NET any -> any any (msg:"IPREP internal host talking to CnC server"; flow:to_server; iprep:dst,CnC,>,30; sid:1; rev:1;)
```

Данное правило будет предупреждать, когда система в $HOME_NET выступает в качестве клиента при обмене данными с любым IP-адресом в категории CnC, который имеет показатель репутации выше 30.

9.3.29.1.1 IP-only

Ключевое слово “iprep” совместимо с правилами «только IP». Это означает, что правило вроде:

```
alert ip any any -> any any (msg:"IPREP High Value CnC"; iprep:src,CnC,>,100; sid:1; rev:1;)
```

Будет проверяться только один раз для каждого направления потока.

9.3.30 Наборы данных

Использование ключевых слов dataset и datarep может быть сопряжено с большими объемами данных в отношении любого липкого буфера.

Например, для сравнения с черным списком dns с именем dns-bl:

```
dns.query; dataset:isset,dns-bl;
```

Чтобы посмотреть DNS-запрос по черному списку MD5:

```
dns.query; to_md5; dataset:isset,dns-bl;
```

9.3.30.1 Глобальная конфигурация (опционально)

Наборы данных могут быть опционально определены в основной конфигурации. Наборы также могут быть объявлены из правил.

Пример наборов для отслеживания уникальных значений:

```
datasets:
  ua-seen:
    type: string
    state: ua-seen.lst
  dns-sha256-seen:
    type: sha256
    state: dns-sha256-seen.lst
```

Правила, соответствующие приведенному выше:

```
alert dns any any -> any any (msg:"dns list test"; dns.query; to_sha256; dataset:isset,dns-sha256-seen; sid:123; rev:1;)
alert http any any -> any any (msg: "http user-agent test"; http.user_agent; dataset:set,ua-seen; sid:234; rev:1;)
```

9.3.30.2 Ключевые слова правила

9.3.30.2.1 dataset

Наборы данных являются двоичными.

Синтаксис:

```
dataset:<cmd>,<name>,<options>;

dataset:<set|isset|isnotset>,<name> \
   [, type <string|md5|sha256>, save <file name>, load <file name>, state <file name>];
```

type <type>
Тип данных: строка, md5, sha256.
save <file name>
Имя файла для сохранения данных в памяти при выходе из Suricata.
load <file name>
Имя файла для загрузки данных при запуске Suricata.
state
Устанавливает 'save' и 'load' для одного и того же значения.
Нельзя смешивать 'load' или 'load' плюс 'save' и 'state'.

9.3.30.2.2 datarep

Позволяет сопоставлять данные со списком репутации.
Синтаксис:

```
datarep:<name>,<Оператор>,<value>, \
   [, load <file name>, type <string|md5|sha256>];
```

Пример правил может выглядеть так:

```
alert dns any any -> any any (dns.query; to_md5; datarep:dns_md5, >, 200, load dns_md5.rep, type md5; sid:1;)
alert dns any any -> any any (dns.query; to_sha256; datarep:dns_sha256, >, 200, load dns_sha256.rep, type sha256; sid:2;)
alert dns any any -> any any (dns.query; datarep:dns_string, >, 200, load dns_string.rep, type string; sid:3;)
```

В этих примерах строка запроса DNS проверяется по трем различным спискам репутации. Список MD5, список SHA256 и список необработанных строк (буферов). Правила будут совпадать только в том случае, если данные присутствуют в списке и значение репутации выше 200.

9.3.30.3 Unix-сокет

9.3.30.3.1 dataset-add

Команда Unix Socket для добавления данных в набор. В случае успеха дополнение мгновенно становится активным.

Синтаксис:

```
dataset-add <set name> <set type> <data>
```

set name
Присвоение имени уже определенному набору данных.
type
Тип данных: строка, md5, sha256.
data
Данные для добавления в последовательной форме (base64 для строки, шестнадцатеричная нотация для md5/sha256).
Пример добавления ‘google.com’ в набор ‘myset’:

```
dataset-add myset string Z29vZ2xlLmNvbQ==
```

9.3.30.4 Форматы файлов

Наборы данных используют простой формат CSV, в котором данные находятся в строке в файле.

9.3.30.4.1 data types

string
В файле в виде строки в кодировке base64.
md5
В файле в виде строки в шестнадцатеричной кодировке.
sha256
В файле в виде строки в шестнадцатеричной кодировке.

9.3.30.4.2 dataset

Наборы данных имеют простую структуру, где в файле содержится один фрагмент данных на строку.

Синтаксис:

```
<data>
```

Например, для типа string:

```
TW96aWxsYS80LjAgKGNvbXBhdGlibGU7ICk=
```

Который при base64 -d раскрывает свое значение:

```
Mozilla/4.0 (compatible; )
```

9.3.30.4.3 datarep

Формат datarep следует за набором данных, ожидается, что есть более 1 поля CSV.

Синтаксис:

```
<data>,<value>
```

9.3.31 Lua-скрипты

Синтаксис:

```
lua:[!]<scriptfilename>;
```

Имя файла скрипта будет добавлено к местоположению правил по умолчанию.

Скрипт состоит из 2 частей, функции инициализации и функции сопоставления. Сначала идет инициализация.

9.3.31.1. Функция Init

```
function init (args)
   local needs = {}
   needs["http.request_line"] = tostring(true)
   return needs
end
```

Функция init регистрирует буфер (ы), которые требуют проверки. В настоящее время доступны следующие:

- packet – весь пакет, включая заголовки
- payload – полезная нагрузка пакета (не поток)
- buffer – текущий липкий буфер
- http.uri
- http.uri.raw
- http.request_line
- http.request_headers
- http.request_headers.raw
- http.request_cookie
- http.request_user_agent
- http.request_body
- http.response_headers
- http.response_headers.raw
- http.response_body
- http.response_cookie

Все HTTP-буферы имеют ограничение: только один из них может быть проверяться скриптом в один момент времени.

9.3.31.2 Функция Match

```
function match(args)
```

```
    a = tostring(args["http.request_line"])
    if #a > 0 then
        if a:find("^POST%s+/.*%.php%s+HTTP/1.0$") then
            return 1
        end
    end

    return 0
end
```

Скрипт может возвращать 1 или 0. Он должен возвращать 1, если проверяемое условие (я) выполняется, если нет – 0.

Весь скрипт:

```
function init (args)
    local needs = {}
    needs["http.request_line"] = tostring(true)
    return needs
end

function match(args)
    a = tostring(args["http.request_line"])
    if #a > 0 then
        if a:find("^POST%s+/.*%.php%s+HTTP/1.0$") then
            return 1
        end
    end

    return 0
end

return 0
```

Дополнительная документация

Дополнительная информация по программному комплексу Frisbee WAF приведена по следующим ссылкам:

- ModSecurity Documentation, **ModSecurity_Manual**.
- Suricata User Guide, **https://docs.suricata.io/en/suricata-7.0.8/**

Использование стороннего программного обеспечения

Apache License, Version 2.0

Licensed under the Apache License, Version 2.0 (the "License"; you may not use this file except in compliance with the License.
You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

GNU GPL License, Version 2

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program, if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

BSD-2-Clause License

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the copyright holder or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

The MIT License

The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

The ISC License

The software is provided "as is" and the author disclaims all warranties with regard to this software including all implied warranties of merchantability and fitness. In no event shall the author be liable for any special, direct, indirect, or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

The PostgreSQL License

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL ORGANISATION BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF ORGANISATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ORGANISATION SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND ORGANISATION HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.