ООО «Эйч-Эль-Эль»

УТВЕРЖДАЮ
Генеральный директор
ООО «Эйч-Эль-Эль»
_____ Д.Ф. Ткачев
«9» февраля 2023 г.

Программа «Сенсор аномалий HTTP-трафика QTS»
(Qrator Traffic Sensor)

Описание программы

RU.7704773923. 62.03.11.001-01 13

СОГЛАСОВАНО
Директор по разработке
ООО «Эйч-Эль-Эль»
_____ М.Л. Левитин
«9» февраля 2023 г.

Москва 2023 г.

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

СОДЕРЖАНИЕ

Определения, обозначения и сокращения

Термин	Определение
Веб-ресурс	Веб-сервер Клиента, расположенный в сети интернет
Клиент	Владелец веб-ресурса
ПО	Программное обеспечение
Пользователь	Пользователь, который обращается к веб-ресурсу
API	Application Programming Interface
HTTP	Протокол прикладного уровня передачи данных
HTTPS	Расширение протокола HTTP для поддержки шифрования в целях повышения безопасности
NGINX	HTTP-прокси Клиента
QTS	Программное обеспечение «Сенсор аномалий HTTP-трафика QTS» (Qrator Traffic Sensor)
ZMQ (ZeroMQ, ØMQ, 0MQ)	Интерфейс, сокет, высокопроизводительная асинхронная библиотека обмена сообщениями, ориентированная на использование в распределенных и параллельных вычислениях (ПО с открытым кодом, https://github.com/zeromq)

1 Описание программы

Программа «Сенсор аномалий HTTP-трафика QTS» (Qrator Traffic Sensor) (далее — QTS) является программой выявления аномального поведения пользователей веб-ресурсов Клиента, подключенных к сети интернет, и распознавания потенциальных угроз защищаемым веб-ресурсам.

2 Функциональное назначение программы

2.1 Назначение программы

Программа «Сенсор аномалий HTTP-трафика QTS (Qrator Traffic Sensor)» (далее — «QTS» или «программа») предназначена для анализа информации об HTTP-трафике и выявления IP-адресов, трафик с которых рекомендуется контролировать для ограничения внешних воздействий на веб-ресурсы (серверы) Клиента. QTS используется в связке с файерволом и обратным HTTP-прокси Клиента, находящимися между пользователем и веб-ресурсами Клиента (см. п. 3.1. Структура программы. Руководство системного программиста).

Обратный HTTP-прокси передаёт программе информацию о каждом запросе, включая URL и HTTP-заголовки запроса. Формат передачи информации описан в п. 6.1. Протокол входящих сообщений. Руководство системного программиста.

На основе анализа программа генерирует информацию с аномальными IP-адресами, которые в дальнейшем могут использоваться как для ограничения аномального трафика с помощью внешних средств, так и для его анализа и мониторинга. Системный программист Клиента должен настроить файервол таким образом, чтобы его чёрные и белые списки обновлялись в соответствии с этими командами. Формат передачи команд описан в п. 6.2. Протокол исходящих сообщений. Руководство системного программиста.

2.1.1 Функции программы

К основным функциям QTS относятся:

–анализ HTTP-трафика;

–определение аномального (по ряду критериев) поведения источника HTTP-запросов к веб-ресурсу Клиента;

–регистрация событий возникновения аномалий;

–уведомление подсистемы фильтрации трафика, взаимодействующей с QTS, о времени начала аномалии от одного или группы источников (IP-адресов);

–взаимодействие с внешними системами.

К дополнительным функциям QTS относятся

– Опция проксирования TCP-соединений (Web-сокетов)

–Опция фильтрации трафика по геозонам

–Опция балансировки очищенного трафика между IP адресами

– Опция защиты от ботов

2.2 Общее описание функционирования программы

2.2.1 Структура программы

QTS состоит из ядра и нескольких компонентов, выполняющих анализ входящего трафика.

Ядро осуществляет чтение информации об HTTP-запросах из очереди, данные в которую записывает специально настроенный обратный HTTP-прокси. Информация обрабатывается несколькими фильтрами и преобразуется в события. На основе событий компонент «Менеджер списков» генерирует сообщения об аномалиях и отправляет их файерволу Клиента с помощью другой очереди.

Рекомендуется использовать QTS, установленный в облачной среде, предоставляемой разработчиком, для оптимизированного использования серверных ресурсов. В случае размещения QTS в облачной среде доступ к нему со стороны инфраструктуры Заказчика обеспечивается посредством расширенного функционала QTS, предусматривающих: предоставление одного или нескольких выделенных IP-адресов и/или подключение к инфраструктуре Исполнителя посредством анонса сетевых префиксов Заказчика по протоколу BGP и/или предоставление выделенного порта на оборудовании Исполнителя для организации ВОЛС. Конкретный способ подключения (через выделенный IP-адрес, через BGP, через физическое соединение по ВОЛС либо их комбинацию) определяется по выбору Заказчика и с учётом технической возможности его реализации. Файервол, прокси и QTS могут находиться на одном физическом сервере, но должны быть на разных виртуальных серверах. В последнем случае сетевой администратор Клиента должен самостоятельно настроить сетевые интерфейсы таким образом, чтобы файервол, прокси и QTS имели возможность читать или записывать данные в нужных очередях.

Схема работы QTS приведена на рисунке Рисунок 1.

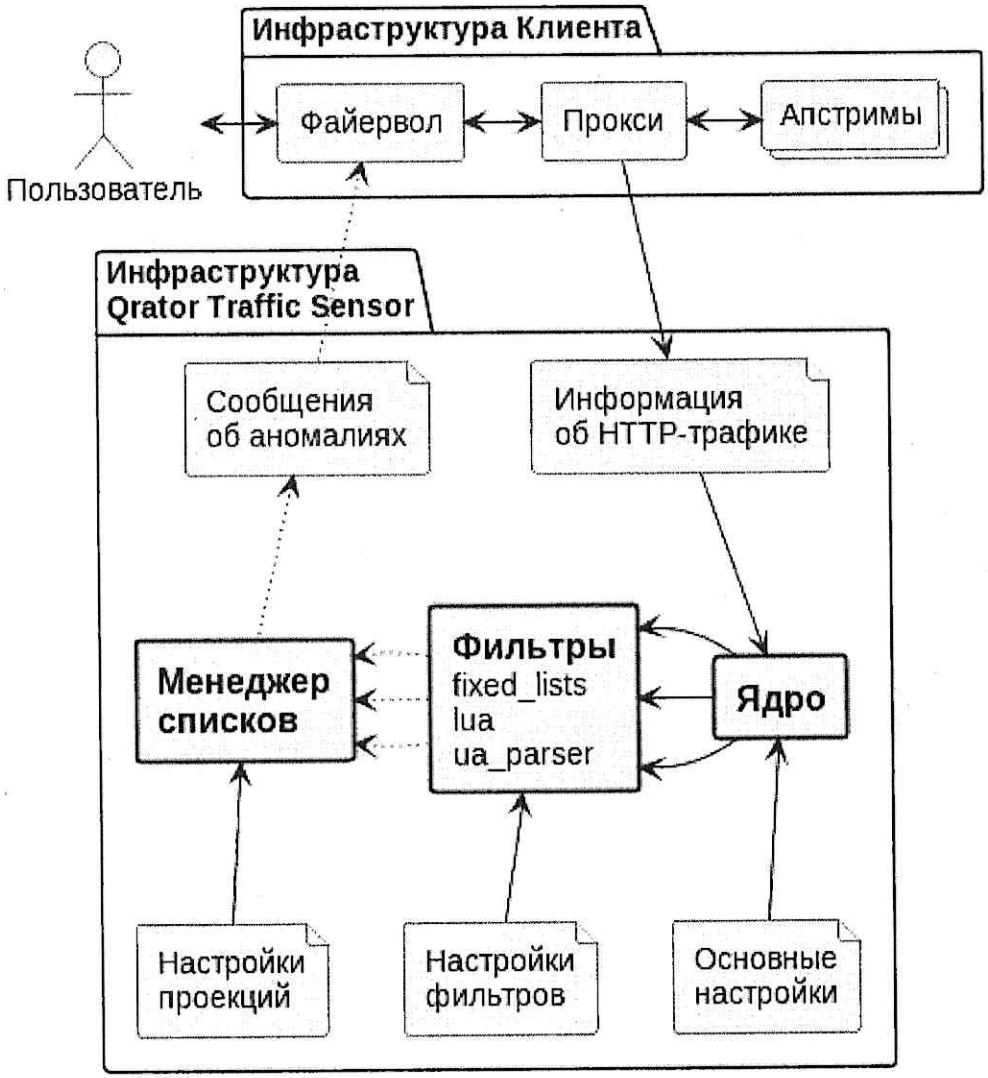

Рисунок 1 — Схема работы QTS

На схеме обозначены:

–Пользователь — пользователь или приложение, отправляющие HTTP-запросы к защищаемому веб-ресурсу Клиента.

–Файервол — сервер в сети Клиента, осуществляющий фильтрацию запросов в соответствии с чёрными и белыми списками. Файервол должен быть настроен таким образом, чтобы обновлять списки согласно сообщениям из очереди Сообщения об аномалиях.

–Прокси — обратный HTTP-прокси в сети Клиента, принимающий запросы от пользователя и пересылающий их на один или несколько апстримов. Прокси должен быть настроен таким образом, чтобы отправлять информацию о каждом обработанном запросе в очередь Информация об HTTP-трафике. Если трафик использует шифрованный протокол HTTPS, то прокси отвечает за его расшифровку.

–Апстримы — сервера на стороне Клиента, обслуживающие веб-ресурс.

–Компоненты программы QTS:

–Ядро — компонент, отвечающий за загрузку файлов настроек, чтение данных из очереди и передачу их другим компонентам программы.

–Фильтры — компоненты, анализирующие данные и генерирующие на их основе события (п. 2.2.3).

–Менеджер списков — компонент, фильтрующий или преобразующий список сгенерированных событий с помощью системы проекций, настроенной системным программистом (см. 4.2.3. Настройки проекций. Руководство системного программиста).

–Настройки программы QTS:

–Основные настройки — настройки, загруженные из файла INI при запуске программы (см. п. 4.1. Основной файл настроек. Руководство системного программиста)).

–Настройки фильтров — настройки, связанные с конкретными фильтрами программы (см. п. 4.2.2. Настройки фильтров. Руководство системного программиста)).

–Настройки проекций — настройки, определяющие правила генерации команд на основе возникших событий (см. п. 4.2.3. Настройки проекций. Руководство системного программиста)).

–Средства коммуникации:

–Информация об HTTP-трафике — очередь ZeroMQ, из которой QTS читает сообщения с информацией об HTTP-запросах.

–Сообщения об аномалиях — очередь ZeroMQ, в которую QTS записывает сообщения о том, какие IP-адреса рекомендуется заблокировать или разблокировать.

2.2.2 Жизненный цикл запроса

QTS выполняет асинхронный анализ запросов. Это означает, что обратный HTTP-прокси, обрабатывающий запрос от пользователя, уведомляет программу о начале и завершении обработки запроса, но не дожидается завершения анализа и отправляет ответ пользователю.

Если, проанализировав запросы, программа обнаружит в них признаки аномалии, она отправит сообщение об этом файерволу. После получения сообщения файервол не должен пропускать последующие запросы от этого IP-адреса до обратного HTTP-прокси.

На рисунке Рисунок 2 изображён порядок обработки HTTP-запросов от ранее не заблокированного IP-адреса.

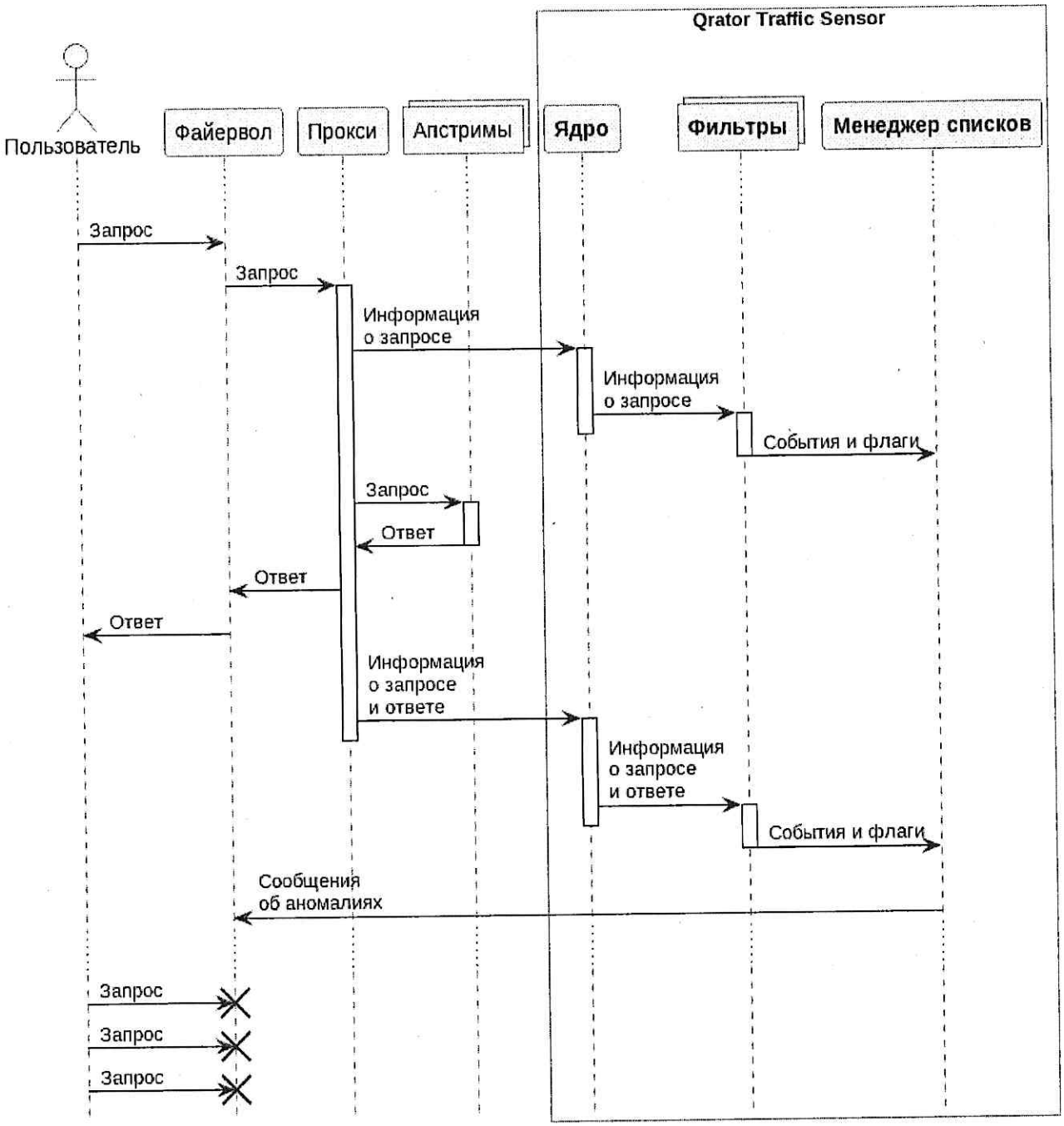

Рисунок 2 — Порядок обработки HTTP-запросов от ранее не заблокированного IP-адреса

1) Пользователь отправляет запрос веб-серверу, находящемуся за файерволом и обратным HTTP-прокси. Поскольку IP-адрес не находится в чёрном списке, файервол пропускает этот запрос.

2) Обратный HTTP-прокси отправляет программе QTS информацию о запросе (см. п. 5.1. Протокол входящих сообщений. Руководство системного программиста)).

3) Ядро QTS передаёт полученную информацию фильтрам (п. 2.2.3).

4) Фильтры анализируют данные и при необходимости генерируют события. Каждое событие содержит IP-адрес пользователя и строковое описание. Дополнительно фильтры могут установить на запрос произвольное количество флагов.

5) Менеджер списков получает информацию обо всех сгенерированных событиях и обрабатывает их с помощью проекций (п. 2.2.4).

6) Обратный HTTP-прокси отправляет запрос одному или нескольким апстримам и получает ответ.

7) Обратный HTTP-прокси отправляет ответ пользователю.

8) Обратный HTTP-прокси отправляет программе QTS информацию о запросе и ответе. QTS обрабатывает информацию аналогично пунктам 3–5.

9) Если необходимо, менеджер списков отправляет сообщения об аномалиях файерволу (см. см. п. 5.2. Протокол исходящих сообщений. Руководство системного программиста)). Дальнейшие запросы от этого пользователя блокируются файерволом и не поступают на веб-сервер.

2.2.3 Фильтры

Фильтры — это компоненты программы QTS, выполняющие анализ HTTP-запросов.

Во время обработки HTTP-запроса программа передаёт информацию о нём всем фильтрам по очереди. Каждый фильтр может:

- сгенерировать связанное с запросом событие,
- установить на запрос произвольное количество флагов.

Системный программист может задать настройки так, чтобы определённый фильтр не использовался, если фильтры до него установили на запрос определённый флаг (см. п. 4.2.1. Включение или отключение фильтров. Руководство системного программиста)).

Все сгенерированные события обрабатываются с помощью проекций (п. 2.2.4).

2.2.3.1 Порядок применения фильтров

Фильтры обрабатывают запросы в следующем порядке:

1) ua_parser;
2) location_flags;
3) fixed_lists (только проверка списков с типом white или непустым массивом flags);
4) lua (только проверка условий с префиксом allow_);
5) fixed_lists (проверка списков, не проверенных ранее);
6) lua (только проверка условий с префиксом deny_);
7) many_uas;
8) limit_req_mc;
9) request_hash_score.

2.2.4 Проекции

Проекция — элемент конфигурации компонента «Менеджер списков». Проекция определяет порядок генерации сообщений об аномалиях на основе событий, сгенерированных фильтрами (п. 2.2.3).

Существует три режима работы проекции: «Ясли», «Наблюдение» и «Блокировка». При простейшей настройке проекция работает в режиме «Блокировка».

Системный программист может разрешить работу в одном или двух из предварительных режимов работы проекции: «Ясли» и «Наблюдение». В этих режимах QTS производит анализ количества запросов и сравнение его с допустимыми значениями, но не генерирует сообщения об аномалиях. После выполнения определённых критериев проекция переключается в режим «Блокировка». Такой подход позволяет уменьшить риск ложно-положительных срабатываний за счёт того, что если трафик небольшой и не представляет угрозы, то блокировка трафика производиться не будет.

Подробнее доступные настройки описаны в п. 4.1. Основной файл настроек. Руководство системного программиста.

2.2.4.1 Режимы работы проекции

Режимы работы проекции представлены на рисунке Рисунок 3.

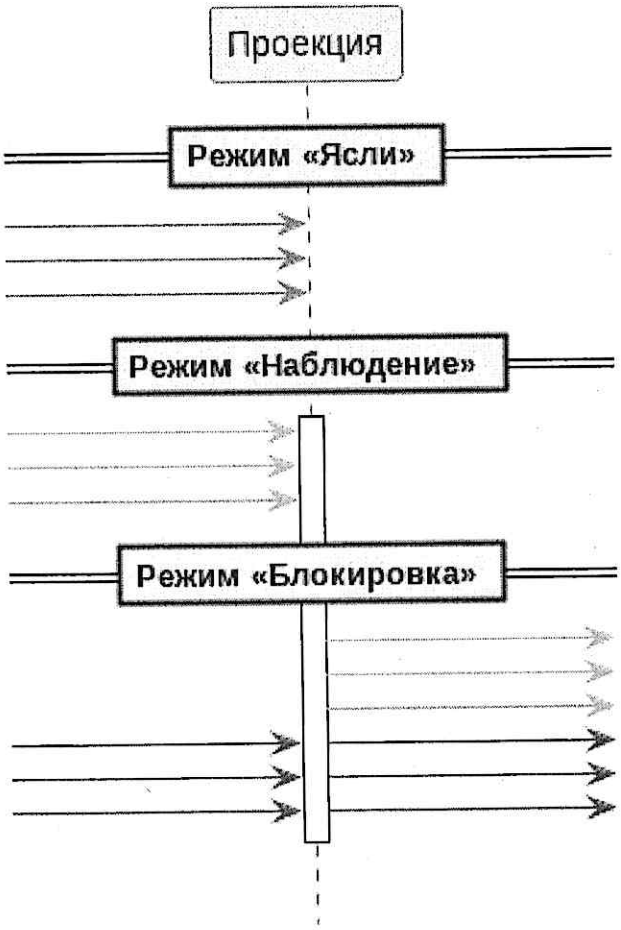

Рисунок 3 — Режимы работы проекции

2.2.4.1.1 Режим «Ясли»

В этом режиме QTS ведёт подсчёт событий, удовлетворяющих регулярному выражению проекции, но не сохраняет конкретные IP-адреса запросов.

Каждые 30 секунд QTS проверяет, достигло ли количество событий за последние 30 секунд порогов, заданных в настройках nursery. Если пороги достигнуты, QTS переводит проекцию в режим «Наблюдение» или «Блокировка».

Чтобы разрешить работу в режиме «Ясли»:

–установите настройку «state» в значение «auto»,

–установите необходимые настройки «nursery».

2.2.4.1.2 Режим «Наблюдение»

В этом режиме QTS рассматривает возникающие события как часть возможной атаки на защищаемый ресурс. QTS сохраняет список IP-адресов, запросы с которых приводят к

возникновению таких событий. IP-адрес, добавленный в список, удаляется из него через некоторое время, если от него не поступает новых запросов.

Сообщения об аномалиях в этом режиме не генерируются, но могут быть сгенерированы позже в режиме «Блокировка».

Каждые 30 секунд QTS проверяет, достигло ли количество событий за последние 30 секунд порогов, заданных в настройках activation. Если пороги достигнуты, QTS переводит проекцию в режим «Блокировка».

Чтобы разрешить работу в режиме «Наблюдение»:

- установите настройку «state» в значение «auto»,
- установите необходимые настройки «activation».

2.2.4.1.3 Режим «Блокировка»

В этом режиме каждое событие, удовлетворяющее регулярному выражению проекции, приводит к генерации сообщения об аномалиях.

Если до этого проекция работала в режиме «Наблюдение», то сразу после её перевода в режим «Блокировка» QTS генерирует сообщения об аномалиях для IP-адресов из списка, составленного за время работы в режиме «Наблюдение».

Чтобы проекция работала в режиме «Блокировка» всегда, без перехода в другие режимы:

- установите настройку «state» в значение «enabled».

2.3 Ограничения на применение программы

На использование QTS распространяются следующие ограничения:

- QTS не является непосредственно средством предотвращения аномалий сетевой активности на уровне протокола HTTP. QTS не собирает, не поддерживает и не обрабатывает персональные данные.
- QTS обрабатывает любой декодированный с HTTP-сервера Клиента трафик.

2.4 Технические средства

При развертывании QTS на виртуальных серверах соблюдаются следующие минимальные технические требования:

–x86/64-совместимый многоядерный процессор с тактовой частотой не менее 2.4 Гц;

–минимальное количество оперативной памяти — 16 гигабайт;

–минимальный объем дисковой памяти — 24 гигабайта.

3 Описание логики программы

3.1 Описание функций составных частей ПО и связей между ними

QTS состоит из следующих основных функциональных модулей:

–Модуль обнаружения аномалий (п. 3.1.1).

–Модуль данных для статистики и мониторинга (п. 3.1.2).

На рисунке Рисунок 4 представлена функциональная архитектура QTS при взаимодействии с внешними системами.

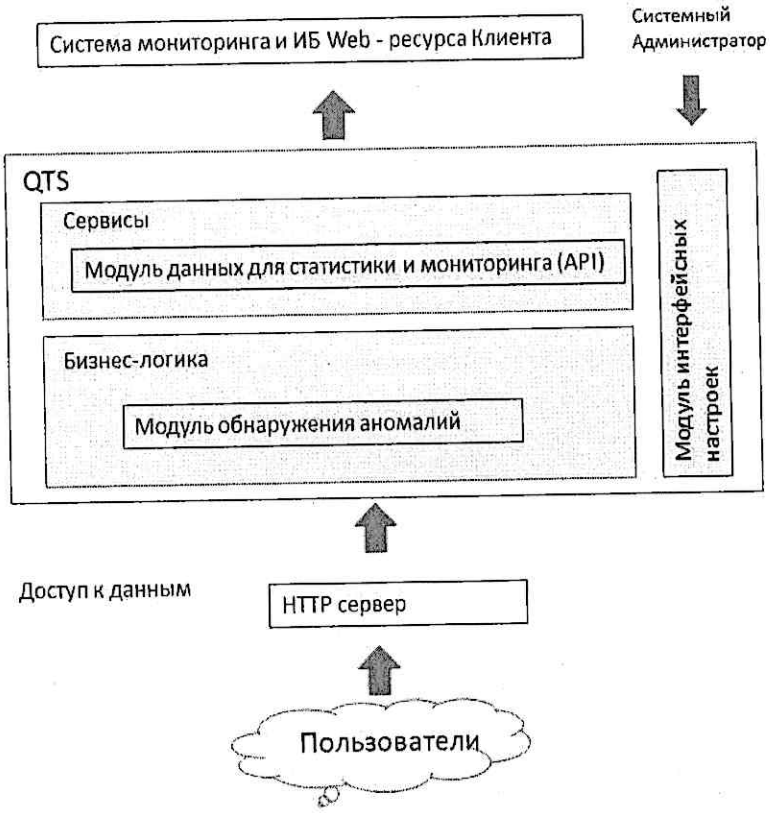

Рисунок 4 — Функциональная архитектура QTS

3.1.1 Модуль обнаружения аномалий

Модуль обнаружения аномалий использует ПО «Программа управления автоматической фильтрацией нелегитимного HTTP-трафика» и наследует алгоритмы, существующие в указанном ПО. При этом функция управления (т. е. выдача команд для перезаписи списков легитимных IP-адресов для сети Клиента) в указанном ПО блокируется

при использовании в QTS. Кроме того, модуль позволяет добавлять новые алгоритмы с учетом особенностей веб-ресурса Клиента. Модуль доступен по интерфейсу ZMQ.

Модуль распознает следующие виды аномалий:

–Содержимое HTTP-заголовков, являющееся аномальным статистически либо по шаблонам.

–Превышение заданного количества запросов в единицу времени и/или к заданному ресурсу (URL) для данного единичного либо группы IP-адресов источника.

3.1.2 Модуль данных для статистики и мониторинга

Модуль данных для статистики и мониторинга обеспечивает формирование сообщений об обнаружении аномалий и осуществляет передачу сообщений об обнаружении поступления аномальных HTTP-запросов.

3.2 Сведения о языке программирования

Исходя из общего подхода к развитию платформы доставки трафика Qrator и унификации используемого ПО, используется следующее окружение для работы QTS:

–Системное ПО — операционная система Linux, дистрибутив Gentoo, версия ядра ОС 4.14.

- Компилятор C++ и набор библиотек:
  - boost (https://www.boost.org/);
  - spdlog (https://github.com/gabime/spdlog)
  - simple-web-server (https://github.com/eidheim/Simple-Web-Server, MIT);
  - rapidjson (https://rapidjson.org/);
  - https://github.com/Tencent/rapidjson/blob/master/license.txt);
  - msgpack (https://github.com/msgpack/msgpack-c/tree/cpp_master, boost);
  - json11 (https://github.com/dropbox/json11, MIT);
  - re2 (https://github.com/google/re2, BSD 3-Clause);
  - luajit (https://luajit.org/luajit.html, MIT);
  - cppzmq (https://github.com/zeromq/cppzmq, MIT);
  - uap-cpp (https://github.com/ua-parser/uap-cpp, MIT);
  - yaml-cpp (https://github.com/jbeder/yaml-cpp, MIT).

–

3.3 Описание входных и выходных данных

Источником данных для QTS является поток информации об HTTP-запросах, направляемых на ресурсы, за которыми наблюдает QTS. В состав информации об HTTP-запросах входит вся необходимая атрибутика запроса:

–URL, к которому обращается источник запроса;

–метаданные HTTP-запроса в виде заголовков;

–точная отметка времени запроса.

На основании этих входных данных QTS, используя набор алгоритмов, выявляет наличие аномалий и формирует выходной поток IP-адресов, являющихся источниками «подозрительного трафика». Поток IP-адресов, для которых распознаны аномалии, используется внешними системами для аналитики и/или управления задачами фильтрации трафика.

3.4 Описание логики составных частей

3.4.1 Входные данные

QTS принимает информацию об HTTP-трафике через очередь ZeroMQ, адрес которой указывается в настройке «tsng.input».

Сообщения, поступающие в очередь, должны быть multipart-сообщениями из двух частей (см. http://api.zeromq.org/2-1:zmq-send#toc3). В этом документе первая часть multipart-сообщения называется заголовком, а вторая — содержимым.

1) Заголовок сообщения должен представлять собой строку вида http:⟨ДОМЕН⟩:, где ⟨ДОМЕН⟩ — идентификатор домена. Пример первой части сообщения: http:8984:.
2) QTS обрабатывает только запросы к доменам, перечисленным в параметре force_domains основного файла настроек. Если в заголовке запроса указан неизвестный идентификатор домена, такой запрос обработан не будет.
3) При обработке запроса могут быть использованы настройки, заданные для конкретного домена.
4) Тело сообщения должно представлять собой последовательность полей, описанных в таблице

5) Таблица 1.
6) Обратный HTTP-прокси, обрабатывающий запросы от пользователей, может направить QTS два сообщения об одном и том же запросе: до и после обработки запроса апстримом. При отправке сообщения до обработки запроса во всех полях, отмеченных звёздочкой в таблице ниже, должно передаваться значение «0». Остальные поля должны быть одинаковыми в первом и втором

сообщениях.

Таблица 1 — Поля тела сообщения

Название поля	Описание поля	Тип	Пример
req_type*	1, если запрос уже обработан Веб-сервером	uint32	1
timestamp	Время отправки сообщения в формате UNIX time	uint32	1670368996
remote_ip	IP-адрес пользователя	uint32	198.51.100.1
RESERVED	Не используется, должно быть равно «0»	uint32	0
request_time*	Общее время обработки запроса (мс)	uint32	60150
response_time*	Суммарное время ожидания ответа от всех апстримов (мс)	uint32	60100
last_response_time*	Время ожидания ответа от последнего апстрима (мс)	uint32	100
n_responses*	Количество апстримов, которым был передан запрос. Если Веб-сервер обслуживается только одним апстримом, значение должно быть равно 1	uint16	3
status*	Код ответа	uint16	200
body_bytes_sent	Размер ответа, отправленного пользователю (байт)	uint32	12345
supp_bits	Не используется, должно быть равно «0»	uint32	0
failed_servers*	Не используется, должно быть равно «0»	uint64	0
responded_server*	Не используется, должно быть равно «0»	int8	0
geo	Код страны, к которой принадлежит IP-адрес пользователя	char[2]	US
method	Метод запроса, например: GET	char[8]	GET
url	Путь, к которому сделан запрос	char[132]	/page.html
version	Версия протокола HTTP	char[9]	HTTP/1.1
referer	Значение заголовка Referer	char[64]	https://example.com/
host	Значение заголовка Host	char[32]	example.com
accept	Значение заголовка Accept	char[16]	text/html, application/xhtml+xml

Название поля	Описание поля	Тип	Пример
accept_language	Значение заголовка Accept-Language	char[8]	ru-RU
accept_encoding	Значение заголовка Accept-Encoding	char[24]	deflate, gzip
forwarded_for	Значение заголовка X-Forwarded-For	char[16]	203.0.113.1
cookie	Значение заголовка Cookie	char[128]	name=value; name2=value2; name3=value3
http_user_agent	Значение заголовка User-Agent	uint32 + char[]	Mozilla/5.0 (X11; Linux x86_64; rv:10.0) Gecko/20100101 Firefox/10.0
http_x_requested_with	Значение заголовка X-Requested-With	uint32 + char[]	XMLHttpRequest
upstream_addr*	IP-адрес апстрима, обработавшего запрос	uint32 + char[]	203.0.113.3
upstream_statuses*	Коды ответов от апстримов	uint32 + char[]	504, 504, 200
upstream_durations*	Время ответов апстримов (мс)	uint32 + char[]	30000, 30000, 100
request	Первая строка HTTP-запроса	uint32 + char[]	GET / HTTP/1.1
qrator_request_id	Уникальный идентификатор запроса	uint32 + char[]	abcd1234
server_port	Порт, на который сделан запрос	uint32 + char[]	443
remote_port	Порт, с которого сделан запрос	uint32 + char[]	12345
connection_requests	Порядковый номер запроса в рамках TCP-соединения	uint32 + char[]	1
msec	Время отправки сообщения в формате UNIX time в миллисекундах	uint32 + char[]	1670368996.6454
ssl_cipher	Используемый шифр	uint32 + char[]	ECDHE-RSA-AES256-GCM-SHA384
ssl_ciphers	Шифры, поддерживаемые пользователем	uint32 + char[]	0xc030:0xc02c:0xc028:0xc024:0xc014:0xc00a:0xc022:0xc021:0x00a3:0x009f
RESERVED	Не используется, должно быть равно «0»	uint32	0
RESERVED	Не используется, должно быть равно «0»	uint32	0
sent_http_location*	Значение заголовка Location	uint32 + char[]	https://www.example.com/

Название поля	Описание поля	Тип	Пример
sent_http_content_type*	Значение заголовка Content-Type	uint32 + char[]	text/html; charset=utf-8
headers	Все заголовки запроса от пользователя	uint32 + char[]	Accept: */*\r\nAccept-Encoding: gzip, deflate
body	Тело запроса	uint32 + char[]	Hello.
RESERVED	Не используется, должно быть равно «0»	uint32	0
request_length	Размер запроса пользователя (байт)	uint32 + char[]	6
bytes_sent*	Размер ответа, отправленного пользователю (байт)	uint32 + char[]	12345
upstream_bytes_sent*	Размер запроса, отправленного на апстрим (байт)	uint32 + char[]	12345
upstream_bytes_received*	Размер ответа, полученного от апстрима (байт)	uint32 + char[]	12345

Обозначения типов полей:

–int8 — 8-битное целое число со знаком.

–uint16 — 16-битное беззнаковое целое число.

–uint32 — 32-битное беззнаковое целое число.

–uint64 — 64-битное беззнаковое целое число.

–char[⟨РАЗМЕР⟩] — строка размером не более ⟨РАЗМЕР⟩ байт. Кодировка строки — UTF-8. В качестве последнего символа должен использоваться нулевой байт. Чтобы оставить поле пустым, передайте нулевой байт в качестве первого байта.

–uint32 + char[] — 32-битное беззнаковое число, обозначающее длину строки, за которым следует указанное в нём количество байт. Кодировка строки — UTF-8. Чтобы оставить поле пустым, передайте «0» в качестве длины строки.

При анализе сообщения, отправленного после обработки запроса, QTS использует уникальный идентификатор запроса (поле «qrator_request_id»), чтобы сопоставить его с полученной ранее информацией и дополнить её.

Если обратный HTTP-прокси последовательно предпринял попытки отправить запрос на несколько апстримов, то в сообщении должна содержаться информация о результатах всех этих попыток. Например, в поле «response_time» должно быть передано суммарное время ожидания ответа от всех апстримов, а в поле «upstream_statuses» — коды ответов от всех апстримов.

3.4.2 Выходные данные

Исходящие сообщения содержат:

Информацию о начале и окончании аномалий и отправляются файрволу через очередь ZeroMQ. Адрес очереди указывается в параметре «tsng:qpc.outbound» основного INI-файла. Каждое сообщение представляет собой словарь JSON.

Каждое сообщение представляет собой словарь JSON. Пример сообщения:

```
{
  "domain_id": 8984,
  "comment": "loc_rate_def1_GET/",
  "rule_name": "tsng.location_rate",
  "method": "add",
  "ip_addr": "1.2.3.4",
  "geo": "US",
  "timestamp": 1669808971.2739999,
  "timeout": 5183,
  "worker": "wrk_1",
  "broadcast": false,
  "hoplite": "1c30c1ea0869",
}
```

Доступные поля:

–domain_id (целое число) — идентификатор домена.

–comment (строка) — комментарий, который может быть использован при добавлении правила в файервол.

–Если команда сгенерирована на основе одного события, то в комментарии указывается название этого события.

–Если команда сгенерирована на основе двух или более событий, то в качестве комментария используется строка "_multiple_".

–rule_name (строка) — название списка файервола, в котором необходимо добавить или удалить IP-адрес.

–method (строка) — действие:

–"add" — добавить IP-адрес в список файервола.

–"del" — удалить IP-адрес из списка файервола.

–ip_addr (строка) — IP-адрес, добавляемый или удаляемый из списка файервола.

–geo (строка) — код страны, к которой принадлежит IP-адрес. Страна определяется по IP-адресу с помощью базы данных MaxMind GeoIP.

–timestamp (вещественное число) — время генерации команды в формате UNIX time.

–timeout (целое число) — срок жизни добавляемой записи в секундах. По истечении этого количества секунд запись следует удалить из файервола. Это поле присутствует только в командах на добавление.

–worker (строка) — уникальный идентификатор потока, в котором был обработан запрос.

–broadcast (булево значение) — не используется, всегда равно false.

–hoplite (строка) — доменное имя (hostname) машины, на которой запущена программа.

4 API для управления настройками

Программа предоставляет REST API, с помощью которого возможно:

–получать и изменять настройки программы в формате JSON без прямого редактирования файлов;

–принудительно применять изменения, внесённые в файлы настроек во время работы программы.

Все методы API доступны по протоколу HTTP на доменном имени и порту, определёнными в настройках api_server_host и api_server_port.

При успешном выполнении запроса программа возвращает статус 200 OK и JSON-объект с полем status, равным success. В зависимости от конкретного метода, в объекте могут присутствовать дополнительные поля.

При ошибке программа возвращает статус 400 Bad Request и JSON-объект с полем «status», равным «error». Также в объекте присутствует поле exception с описанием произошедшей ошибки.

4.1 GET /filters

Получить перечень доменов, используемых в текущей конфигурации.

Ответ содержит поля, представленные в таблице Таблица 2.

Таблица 2 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки
data	массив чисел	Идентификаторы доменов, используемых в текущей конфигурации

Пример ответа:

```
{
  "status": "success",
  "data": [8984]
}
```

4.2 GET /filters/⟨ДОМЕН⟩

Получить текущую конфигурацию всех компонентов для домена ⟨ДОМЕН⟩.

Ответ содержит поля, представленные в таблице Таблица 3.

Таблица 3 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки
data	объект	Ключами являются названия всех доступных компонентов, а значениями — объекты с конфигурацией этих компонентов для домена. Описания полей конфигурации конкретных компонентов (п. **Ошибка! Источник ссылки не найден.**)

Пример ответа:


```
{
 "status": "success",
 "data": {
   "fixed_lists": {"enabled": true, "lists": [], "use": [], "use_component_use":
true},
   "limit_req_mc": {"configurations": [], "enabled": false},
   "lm": {"lists": [], "projections": [], "use_component_projections": true},
   "location_flags": {"locations": [{"flags": [{"flag": "static", "value":
false}], "host": "", "method": "", "path": "", "stop": false}, {"flags": [{"flag":
"static", "value": true}], "host": "", "method": "", "path":
".*(\\.css|\\.js|\\.jpg|\\.jpeg|\\.tiff|\\.png|\\.ico|\\.xml|\\.swf|\\.cur|\\.ogg|\\.mp
3|\\.pgif|\\.woff|\\.svg|\\.gif|\\.ttf|\\.eot|\\.flv|\\.txt|\\.pdf|\\.otf|\\.js\\.map|\
\.css\\.map|\\.woff2|\\.webp)$", "stop": false}]},
   "lua": {"enabled": true, "scripts": [{"file": "default.lua"}], "skip_if": []},
   "many_uas": {"enabled": true, "max_uas": 10, "skip_if": ["whitelisted"]},
   "qpc": {},
   "request_hash_score": {"enabled": false, "per_ip": {"history_size": 40,
"max_score": 20, "min_repeats": 3, "timeout": 300}, "per_useragent": {"history_size": 20,
"max_score": 10, "min_repeats": 3, "timeout": 300}, "skip_if": ["static",
"whitelisted"]},
   "statstore_http_stats": {"enabled": true},
   "telemetry_http_sampler": {"enabled": true},
```

```
  "ua_parser": {"enabled": true, "skip_if": []}
 }
}
```

4.3 PUT /filters/⟨ДОМЕН⟩?rewrite=⟨0|1⟩

Добавить или заменить конфигурацию всех компонентов для домена ⟨ДОМЕН⟩.

Тело запроса должно представлять собой JSON-объект, в котором ключами являются названия всех доступных компонентов, а значениями — объекты с конфигурацией этих компонентов для домена. Если для некоторых компонентов конфигурация не предоставлена, для них будет использована конфигурация по умолчанию из файла default.conf.

Если после замены конфигурации одна или несколько проекций получают статус disabled, то программа сгенерирует команды об окончании всех аномалий, обнаруженных ранее этими проекциями. В частности, если тело запроса не содержит ни одной проекции (пустой JSON-объект {}), и при этом все проекции в default.conf имеют статус disabled, программа сообщит об окончании аномалий для всех IP-адресов.

По умолчанию изменения сохраняются только в оперативной памяти и могут быть потеряны при перезапуске программы. При включённой настройке autodiscover изменения могут быть также потеряны во время автоматического перечитывания настроек каждые несколько минут. Чтобы избежать этого, передайте в запросе GET-параметр rewrite=1. В этом случае программа сохранит новую конфигурацию домена в файл ⟨ДОМЕН⟩.json в первой из директорий, определённых настройкой filters_config_dir.

Пример запроса:

```
{
 "fixed_lists": {"enabled": true, "lists": [], "use": [], "use_component_use": true},
 "limit_req_mc": {"configurations": [], "enabled": false},
 "lm": {"lists": [], "projections": [], "use_component_projections": true},
 "location_flags": {"locations": [{"flags": [{"flag": "static", "value": false}], "host": "", "method": "", "path": "", "stop": false}, {"flags": [{"flag": "static", "value": true}], "host": "", "method": "", "path": ".*(\\.css|\\.js|\\.jpg|\\.jpeg|\\.tiff|\\.png|\\.ico|\\.xml|\\.swf|\\.cur|\\.ogg|\\.mp3|\\.pgif|\\.woff|\\.svg|\\.gif|\\.ttf|\\.eot|\\.flv|\\.txt|\\.pdf|\\.otf|\\.js\\.map|\\.css\\.map|\\.woff2|\\.webp)$", "stop": false}]},
 "lua": {"enabled": true, "scripts": [{"file": "default.lua"}], "skip_if": []},
 "many_uas": {"enabled": true, "max_uas": 10, "skip_if": ["whitelisted"]},
 "qpc": {},
```

"request_hash_score": {"enabled": false, "per_ip": {"history_size": 40, "max_score": 20, "min_repeats": 3, "timeout": 300}, "per_useragent": {"history_size": 20, "max_score": 10, "min_repeats": 3, "timeout": 300}, "skip_if": ["static", "whitelisted"]},
"statstore_http_stats": {"enabled": true},
"telemetry_http_sampler": {"enabled": true},
"ua_parser": {"enabled": true, "skip_if": []}
}

Ответ содержит поля, представленные в таблице Таблица 4.

Таблица 4 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки

Пример ответа:

{
"status": "success"
}

4.4 DELETE /filters/⟨ДОМЕН⟩

Удалить конфигурацию для домена ⟨ДОМЕН⟩.

Ответ содержит поля, представленные в таблице Таблица 5.

Таблица 5 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки

Пример ответа:

{
"status": "success"
}

4.5 GET /filters/⟨ДОМЕН⟩/⟨КОМПОНЕНТ⟩

Получить текущую конфигурацию компонента ⟨КОМПОНЕНТ⟩ для домена ⟨ДОМЕН⟩.

Ответ содержит поля, представленные в таблице Таблица 6.

Таблица 6 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки
data	объект	Конфигурация компонента. Описания полей конфигурации конкретных компонентов см. в п. 5.2. Настройки компонентов

Пример ответа:

```
{
 "status": "success",
 "data": {"lists": [], "projections": [], "use_component_projections": true}
}
```

4.6 PUT /filters/⟨ДОМЕН⟩/⟨КОМПОНЕНТ⟩?rewrite=⟨0|1⟩

Добавить или заменить конфигурацию компонента ⟨КОМПОНЕНТ⟩ для домена ⟨ДОМЕН⟩. Тело запроса должно представлять собой JSON-объект с конфигурацией компонента.

Если после замены конфигурации проекция получает статус disabled, то программа сгенерирует команды об окончании всех аномалий, обнаруженных ранее этой проекцией.

По умолчанию изменения сохраняются только в оперативной памяти и могут быть потеряны при перезапуске программы. При включённой настройке autodiscover изменения могут быть также потеряны во время автоматического перечитывания настроек каждые несколько минут. Чтобы избежать этого, передайте в запросе GET-параметр rewrite=1. В этом случае программа сохранит новую конфигурацию домена в файл ⟨ДОМЕН⟩.json в первой из директорий, определённых настройкой filters_config_dir.

Пример запроса:

```
{"lists": [], "projections": [], "use_component_projections": true}
```

Ответ содержит поля, представленные в таблице Таблица 7.

Таблица 7 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки

Пример ответа:

```
{
 "status": "success"
}
```

4.7 POST /reload_filters

Перечитать конфигурацию всех доменов.

Данный метод рекомендуется вызывать после каждого изменения файлов в директории, определённой настройкой filters_config_dir, если не включена настройка autoreload_filters.

Ответ содержит поля, представленные в таблице Таблица 8.

Таблица 8 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки

Пример ответа:

```
{
 "status": "success"
}
```

4.8 POST /reload_components

Перечитать глобальную конфигурацию всех компонентов.

Данный метод рекомендуется вызывать после каждого изменения файлов в директории, определённой настройкой components_config_dir, если не включена настройка autoreload_components.

Ответ содержит поля, представленные в таблице Таблица 9.

Таблица 9 — Поля, содержащиеся в ответе

Название	Тип	Описание
status	строка	success в случае успешного выполнения запроса, error в случае ошибки

Пример ответа:

```
{
 "status": "success"
}
```

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (стр.) в документе	№ документа	Входящий № сопроводите ль-ного документа	Под-пись	Дата
	изме-ненны х	Заме-нен-ных	новых	Аннулиро -ванных					