МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Защита веб-сервера Nginx

КУРСОВАЯ РАБОТА

студента 4 курса 431 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Серебрякова Алексея Владимировича

Научный руководитель
ассистент _____ А. А. Лобов
подпись, дата

Заведующий кафедрой
д. ф.-м. н., доцент _____ М. Б. Абросимов
подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном информационном обществе безопасность веб-серверов является одним из важнейших аспектов информационной безопасности. Веб-серверы играют ключевую роль в предоставлении доступа к веб-ресурсам и обработке пользовательских запросов. В этом контексте особое внимание уделяется защите веб-серверов от различных атак и уязвимостей. Одним из самых популярных веб-серверов на сегодняшний день является Nginx. Nginx («Engine X») представляет собой легкий, высокопроизводительный сервер, который широко используется для обслуживания веб-сайтов, проксирования и балансировки нагрузки. Он также предлагает множество возможностей для обеспечения безопасности веб-сервера и защиты от различных видов атак. Целью данной курсовой работы является исследование и анализ мер безопасности, которые могут быть применены для защиты веб-сервера Nginx. Мы рассмотрим основные уязвимости и атаки, с которыми может столкнуться веб-сервер, а также изучим различные методы и техники, которые могут быть использованы для повышения безопасности сервера Nginx. В ходе исследования будут рассмотрены следующие аспекты защиты веб-сервера Nginx:

1. Конфигурация сервера: изучим важные параметры и настройки, которые могут повлиять на безопасность сервера, такие как ограничение доступа к файлам и директориям, использование SSL-сертификатов и применение правил файервола.
2. Защита от DDoS-атак: рассмотрим различные методы обнаружения и предотвращения распределенных атак отказа в обслуживании (DDoS), которые могут оказаться вредными для сервера и доступности веб-сайта.
3. Фильтрация трафика: изучим возможности фильтрации и обработки трафика, включая использование списка контроля доступа (ACL), блокировку IP-адресов, применение белых и черных списков.
4. Мониторинг и регистрация: рассмотрим важность непрерывного мониторинга и регистрации событий сервера для обнаружения подозрительной активности и быстрого реагирования на потенциальные угрозы.

В заключении работы будут представлены рекомендации и практические рекомендации по улучшению безопасности веб-сервера Nginx на основе полученных результатов и анализа. Кроме того, будет предложена программа-конфигуратор для упрощения составления файла конфигурации веб-сервера.

Безопасность веб-серверов является важной задачей для всех организаций, которые предоставляют свои услуги в сети интернет, и эта курсовая работа представляет собой ценный вклад в области информационной безопасности.

1 Почему необходимо следить за защитой веб-сервера

В последние годы количество атак на веб-серверы значительно выросло. С развитием технологий и все большей зависимости от онлайн-сервисов и электронной коммерции, веб-серверы стали привлекательной целью для злоумышленников. Некоторые факторы, способствующие увеличению количества атак на веб-серверы, включают:

1. Распространение злоумышленников: С возрастанием числа злоумышленников, умеющих проводить атаки на веб-серверы, риск стал более значимым. Легко доступные инструменты и ресурсы в Интернете позволяют даже неопытным злоумышленникам осуществлять атаки на веб-серверы.
2. Уязвимости веб-приложений: часто веб-приложения содержат уязвимости, которые могут быть использованы злоумышленниками для атак на сервер. Отсутствие должной обработки и валидации входных данных, уязвимости в коде приложения или ошибки в конфигурации сервера могут предоставить злоумышленникам доступ к серверу.
3. Развитие новых типов атак: злоумышленники постоянно разрабатывают новые методы и техники для атак на веб-серверы. Это включает в себя более сложные DDoS-атаки, передовые методы SQL-инъекций, улучшенные техники фишинга и другие инновационные способы эксплуатации уязвимостей.
4. Коммерческая ценность данных: веб-серверы часто содержат ценную информацию, такую как финансовые данные, персональные данные пользователей, коммерческие секреты и другие конфиденциальные сведения. Захват или кража таких данных может принести значительную финансовую прибыль злоумышленникам, стимулируя их к проведению атак.
5. Распространение ботнетов: ботнеты, состоящие из множества зараженных компьютеров или устройств, используются для проведения массовых атак на веб-серверы. Эти ботнеты контролируются злоумышленниками и могут использоваться для запуска DDoS-атак или для эксплуатации уязвимостей на сервере. Вот примеры нескольких атак, произведенных на крупные компании за последние годы:
6. Атака на Sony PlayStation Network (2011): в 2011 году Sony PlayStation Network столкнулся с одной из наиболее серьезных атак на веб-инфраструктуру. Злоумышленники скомпрометировали сетевую инфраструктуру, что при-

вело к утечке личных данных более 77 миллионов пользователей, включая имена, адреса электронной почты, пароли и финансовую информацию.

7. Атака на Equifax (2017): в 2017 году компания Equifax, одна из трех крупнейших кредитных бюро в США, столкнулась с серьезной атакой. Злоумышленники эксплуатировали уязвимость веб-приложения, что позволило им получить доступ к личным данным около 147 миллионов человек, включая имена, социальные номера, даты рождения и номера кредитных карт.
8. Атака на Yahoo (2013–2014): в 2013–2014 годах Yahoo столкнулся с масштабной атакой, в результате которой были скомпрометированы данные более 3 миллиардов пользователей. Злоумышленники получили доступ к учетным записям, включая имена, адреса электронной почты, хэшированные пароли и секретные вопросы безопасности.

Это всего лишь некоторые примеры успешных атак на веб-инфраструктуру, и с течением времени появляются новые методы и уязвимости, которые могут быть эксплуатированы злоумышленниками. Важно постоянно обновлять свои знания о безопасности и применять соответствующие меры для защиты веб-серверов и веб-приложений.

2 Основные рекомендации к защите веб-серверов

За стандартами защиты веб-серверов следят различные организации и стандартизационные органы, такие как OWASP (Open Web Application Security Project), NIST (National Institute of Standards and Technology) или ENISA (European Union Agency for Cybersecurity), а также сообщества экспертов в области информационной безопасности. Они играют важную роль в разработке и установлении лучших практик и рекомендаций по обеспечению безопасности веб-серверов. Вот основные из них:

1. Обновляйте программное обеспечение: регулярно обновляйте операционную систему, веб-сервер и все установленные компоненты и приложения до последних версий. Обновления часто содержат исправления уязвимостей, которые могут быть использованы злоумышленниками.
2. Используйте сильные пароли: установите сложные пароли для учетных записей администратора и других пользователей, а также для базы данных. Избегайте использования стандартных паролей и регулярно меняйте пароли.
3. Применяйте фильтрацию трафика: используйте межсетевые экраны (firewalls) и системы обнаружения вторжений (IDS/IPS) для фильтрации и контроля входящего и исходящего сетевого трафика. Это поможет блокировать подозрительные пакеты данных и защитить сервер от многих типов атак.
4. Защитите от DDoS-атак: реализуйте механизмы для обнаружения и смягчения DDoS-атак, такие как использование услуги облачной защиты от DDoS или настройка сетевых устройств для отсеивания вредоносного трафика.
5. Применяйте принцип наименьших привилегий: ограничьте привилегии учетных записей, чтобы минимизировать потенциальные последствия компрометации. Пользовательские учетные записи должны иметь только необходимые права доступа к ресурсам.
6. Фильтруйте входящие данные: валидируйте и санитизируйте входящие данные, особенно если они передаются в базу данных или выполняются на сервере. Это поможет предотвратить SQL-инъекции и XSS-атаки.
7. Шифруйте соединение: используйте протокол HTTPS с помощью сертификатов SSL/TLS для защиты передачи данных между клиентом и сервером. Это поможет предотвратить перехват информации и подделку дан-

ных.

8. Регулярно создавайте резервные копии: регулярное резервное копирование данных позволяет восстановить сервер в случае успешной атаки или сбоя. Убедитесь, что резервные копии хранятся в надежном месте, отдельно от основного сервера.
9. Мониторинг и журналирование: внедрите системы мониторинга, которые следят за активностью сервера и обнаруживают подозрительные или необычные события. Хороший журнал событий поможет вам исследовать инциденты и принять меры по предотвращению будущих атак.

Это лишь некоторые рекомендации, и полная защита веб-сервера требует комплексного подхода, учета специфических потребностей вашего сервера и постоянного обновления знаний о безопасности.

3 О выборе веб-сервера

Правильный выбор веб-сервера обеспечивает оптимальное функционирование веб-приложения, удовлетворяя требованиям бизнеса и ожиданиям пользователей. Он позволяет эффективно обрабатывать запросы, предоставлять контент быстро и без проблем, а также адаптироваться к растущим потребностям и нагрузке.

Кроме того, правильный выбор веб-сервера имеет прямое отношение к безопасности вашего веб-приложения. Надежный веб-сервер обеспечивает защиту от различных угроз и атак, минимизируя риски утечки данных или компрометации системы. В своей работе я буду использовать Nginx (Engine-X) — это мощный веб-сервер и прокси-сервер, который выполняет ряд функций и может быть использован в различных сценариях. Вот некоторые основные области применения Nginx:

1. Веб-сервер: одной из основных функций Nginx является обслуживание веб-содержимого. Он способен обрабатывать статические файлы, такие как HTML, CSS, JavaScript и изображения. Благодаря своей высокой производительности и эффективному использованию ресурсов, Nginx позволяет эффективно обслуживать большое количество запросов, особенно в высоконагруженных средах.
2. Обратный прокси: Nginx часто используется в качестве обратного прокси-сервера, который принимает запросы от клиентов и перенаправляет их на соответствующие веб-серверы. Это позволяет балансировать нагрузку между несколькими серверами и повышает отказоустойчивость, так как приложения на серверах могут быть легко масштабируемы.
3. Балансировка нагрузки: Nginx предоставляет возможности балансировки нагрузки, которые позволяют распределять запросы равномерно между несколькими серверами. Это помогает оптимизировать использование ресурсов и обеспечивает более высокую доступность веб-приложений.
4. Кэширование: Nginx поддерживает функцию кэширования, которая позволяет сохранять статические ресурсы, такие как изображения или файлы CSS/JavaScript, в оперативной памяти или на диске. Это сокращает нагрузку на сервер и ускоряет время загрузки страниц для повторных запросов.
5. SSL/TLS терминирование: Nginx может выполнять функцию терминирования SSL/TLS, что позволяет осуществлять шифрование и расшифровку

данных между клиентом и сервером. Это обеспечивает безопасную передачу данных и защиту от перехвата информации.

6. Проксирование API: Nginx может быть использован для проксирования запросов к внутренним или внешним API. Это позволяет контролировать доступ, управлять авторизацией и маршрутизацией запросов к API.
7. Управление статическими файлами и медиаконтентом: Nginx может использоваться для эффективного доставки статических файлов и медиаконтента, таких как видео или аудиофайлы. Это позволяет обеспечить быструю и надежную доставку контента конечным пользователям.

Сочетание высокой производительности, гибкости и богатого функционала делает Nginx популярным выбором для веб-серверов, обратных прокси и балансировщиков нагрузки во многих веб-приложениях и средах разработки.

Далее будут представлены основные настройки для Nginx сервера, применяемые для защиты сервера.

4 Основные настройки Nginx

В данном разделе будут приведены подобранные для тестового локального сервера настройки конфигурации и детально описано назначение каждой из них. Листинг цельного файла конфигурации nginx.conf приведен в приложении А.

4.1 О структуре файла конфигурации

Файл конфигурации Nginx, известный как nginx.conf, определяет основные настройки и параметры работы веб-сервера Nginx. Вот общая структура файла конфигурации nginx.conf:

1. Директивы глобального блока (http): в этом блоке определяются глобальные настройки для всего веб-сервера. Включает в себя директивы, такие как user, worker_processes, events и другие, которые задают общие параметры работы сервера.
2. Блок events: здесь определяются параметры событийной модели, такие как количество рабочих процессов (worker_processes), метод обработки событий и другие настройки, связанные с обработкой событий сервером.
3. Блок http: в этом блоке определяются основные параметры и настройки HTTP-протокола. Он включает в себя блок server, который может быть повторен несколько раз для определения разных виртуальных хостов и их настроек.
4. Блок server: каждый блок server определяет настройки для конкретного виртуального хоста или сервера. В этом блоке определяются параметры, такие как listen, server_name, location и другие, которые управляют поведением сервера для конкретного хоста.
5. Блок location: блок location определяет настройки для обработки запросов, соответствующих определенному пути URL. Здесь можно определить параметры, такие как root, proxy_pass, rewrite и другие, чтобы настроить обработку запросов для конкретного пути.
6. Другие блоки и директивы: в файле конфигурации nginx.conf могут быть определены и другие блоки и директивы для специфических настроек и модулей, таких как SSL/TLS, кэширование, сжатие и другие.

Структура файла конфигурации nginx.conf может различаться в зависимости от конкретных потребностей и настроек веб-сервера. Рекомендуется внимательно изучить документацию Nginx и следовать принятой структуре и

синтаксису для корректной настройки сервера.

4.2 Сетевые ограничения

Данные настройки позволяют вам определить различные параметры работы сервера Nginx, такие как порты, доступ по IP-адресам, размеры буферов, таймауты и другие параметры. Изменение этих настроек позволяет адаптировать сервер к конкретным требованиям приложения и повысить его производительность и безопасность.

Рассмотрим каждую из указанных настроек в контексте Nginx (Рисунок 1):

```
http{

    limit_conn_zone $server_name zone=per_vhost:5m;
    limit_conn_zone $binary_remote_addr zone=per_ip:5m;


        server{
            listen       81;

            deny   192.168.0.170;
            deny   192.168.0.171;
            allow   192.168.0.174;

            server_name  localhost;

            client_body_buffer_size 16k;
            client_header_buffer_size 1k;
            client_max_body_size 8m;
            large_client_header_buffers 2 1k;

            client_body_timeout 12;
            client_header_timeout 12;

            keepalive_timeout 65;
            send_timeout 10;

            server_tokens off;
            ...
            location  ~* \.(css|js|jpg|png|gif)$ {
                    ...
                    limit_conn per_ip 1;
                    ...
            }
    }
}
```

Рисунок 1 – Конфигурация основных сетевых настроек

1. limit_conn_zone $server_name zone=per_vhost:5m и limit_conn_zone $binary_-remote_addr zone=per_ip:5m: эти директивы определяют ограничения на количество одновременных подключений для каждого виртуального хоста (per_vhost) и для каждого IP-адреса клиента (per_ip). Здесь указаны размеры зон памяти (5m), которые используются для отслеживания подключений.
2. listen 81: эта директива указывает, что сервер Nginx будет слушать входящие подключения на порту 81.
3. deny 192.168.0.170, deny 192.168.0.171 и allow 192.168.0.174: эти директивы управляют доступом к серверу на основе IP-адреса клиента. Клиенты

с IP-адресами 192.168.0.170 и 192.168.0.171 будут запрещены, а клиент с IP-адресом 192.168.0.174 будет разрешен.

4. server_name localhost: эта директива определяет имя сервера, к которому применяются настройки в данном блоке конфигурации. В данном случае, сервер будет отвечать на запросы с хостом "localhost".
5. client_body_buffer_size 16k, client_header_buffer_size 1k, client_max_body_size 8m, large_client_header_buffers 2 1k: эти настройки связаны с размерами буферов, используемых для обработки тела запроса клиента (client_body_buffer_size), заголовков клиента (client_header_buffer_size), максимального размера тела запроса клиента (client_max_body_size) и больших буферов для заголовков клиента (large_client_header_buffers).
6. client_body_timeout 12, client_header_timeout 12: эти настройки определяют таймауты ожидания запроса от клиента для тела (client_body_timeout) и заголовков (client_header_timeout).
7. keepalive_timeout 65, send_timeout 10: эти настройки определяют таймауты соединения keep-alive (keepalive_timeout) и отправки данных на сервер (send_timeout).
8. server_tokens off: эта директива отключает отправку информации о версии сервера в заголовках ответа, чтобы уменьшить возможность идентификации сервера в случае потенциальных атак.
9. location ~* \.(css|js|jpg|png|gif)$ { ... } : это блок конфигурации для обработки запросов к статическим файлам с расширениями .css, .js, .jpg, .png и .gif. В данном случае, внутри блока могут быть указаны дополнительные настройки, например, limit_conn per_ip 1, которая ограничивает количество одновременных подключений с одного IP-адреса до 1.

4.3 Кэширование запросов

Кэширование запросов в Nginx — это процесс сохранения результатов запросов на сервере, чтобы при последующих запросах на тот же ресурс сервер мог возвращать результаты из кэша, без необходимости выполнения полной обработки запроса.

При использовании кэширования, Nginx может значительно сократить нагрузку на сервер, улучшить скорость ответа и снизить задержки для конечных пользователей. Когда клиент отправляет запрос, Nginx проверяет наличие соответствующей записи в кэше. Если запись присутствует и не устарела, сервер

может немедленно вернуть результат клиенту без обращения к бэкенд-серверу.

В приведенном примере конфигурации Nginx, рассмотрим несколько соответствующих настроек для кэширования (Рисунок 2):

```
http{

    include             fastcgi_params;
    include             fastcgi.conf;

    #fastCGI
    fastcgi_cache_path /etc/nginx/cache levels=1:2 keys_zone=microcache:10m max_size=500m inactive=10m;
    fastcgi_cache_key "$scheme$request_method$host$request_uri";
    fastcgi_ignore_headers Cache-Control Expires Set-Cookie ;
    add_header caching $upstream_cache_status;

    server{
        ...
        location ~* \.(css|js|jpg|png|gif)$ {
            ...
            expires 1M;
            ...
        }
        ...
        location /testphp {
            fastcgi_cache microcache;
            fastcgi_cache_valid 200 60m;
            fastcgi_pass 127.0.0.1:9000;
            ...
        }
    }
}
```

Рисунок 2 – Конфигурация основных настроек FastCGI

1. fastcgi_cache_path /etc/nginx/cache levels=1:2 keys_zone=microcache:10m max_size=500m inactive=10m: эта настройка определяет путь к директории, где будут храниться кэшированные данные (/etc/nginx/cache), уровни директорий (levels=1:2), зону ключей (keys_zone=microcache:10m), максимальный размер кэша (max_size=500m) и время неактивности после которого записи в кэше считаются устаревшими (inactive=10m).
2. fastcgi_cache_key "$scheme$request_method$host$request_uri": эта директива определяет ключ, по которому происходит кэширование запросов FastCGI. Ключ формируется на основе схемы ($scheme), метода запроса ($request_method), хоста ($host) и URI запроса ($request_uri).
3. fastcgi_cache microcache: эта директива указывает, что запросы, соответствующие данной локации, должны быть кэшированы в зоне ключей microcache.
4. fastcgi_cache_valid 200 60m: эта директива определяет время жизни кэшированной записи с кодом ответа 200 (успешный ответ) в течение 60 минут.

Таким образом, с помощью этих настроек Nginx может кэшировать ответы FastCGI и возвращать их непосредственно из кэша, минуя обращение к бэкенд-серверу, если записи в кэше существуют и не устарели. Это позволяет снизить нагрузку на сервер и сократить время обработки запросов, повышая

производительность и улучшая отзывчивость веб-приложения.

4.4 Шифрование запросов

Шифрование и SSL (Secure Sockets Layer) являются важными аспектами безопасности веб-серверов. Nginx предоставляет возможность использовать SSL/TLS для шифрования соединения между клиентом и сервером. SSL/TLS — это протоколы, обеспечивающие шифрование данных и аутентификацию для безопасной передачи информации по сети. Они используют криптографические алгоритмы для защиты конфиденциальности, целостности и подлинности данных.

В данном примере конфигурации Nginx демонстрируется использование шифрования запросов с помощью протокола SSL/TLS (Рисунок 3).

```
http{
    server{
        listen       443 ssl;
        ssl_certificate /etc/nginx/ssl/nginx.crt;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;
        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout
        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;
    }
}
```

Рисунок 3 – Конфигурация основных настроек шифрования

Давайте рассмотрим каждую из указанных настроек:

1. listen 443 ssl: эта директива указывает на прослушивание порта 443 (стандартный порт для HTTPS) и использование протокола SSL/TLS для шифрования соединения между клиентом и сервером.
2. ssl_certificate /etc/nginx/ssl/nginx.crt и ssl_certificate_key /etc/nginx/ssl/nginx.key: эти директивы указывают пути к сертификату и приватному ключу, необходимым для установки безопасного соединения. В данном случае, указываются пути к файлам сертификата (nginx.crt) и приватного ключа (nginx.key).
3. ssl_session_cache shared:SSL:1m и ssl_session_timeout: эти директивы определяют настройки кэша сеансов SSL/TLS. shared:SSL:1m указывает, что кэш сеансов должен быть разделен между несколькими воркерами и иметь размер 1 мегабайт. ssl_session_timeout 5m определяет время жизни сеансов в кэше.
4. ssl_ciphers HIGH:!aNULL:!MD5: эта директива определяет список шифров, которые могут быть использованы при установке SSL/TLS-соединения.

В данном случае, используются шифры с высоким уровнем безопасности и отключены анонимные шифры и шифры, использующие алгоритм MD5.

5. ssl_prefer_server_ciphers on: эта директива указывает серверу предпочитать шифры, предложенные клиентом, при установке SSL/TLS-соединения.

Данные настройки позволяют использовать SSL/TLS для шифрования запросов между клиентом и сервером, обеспечивая безопасность и защиту передаваемых данных от перехвата или нежелательного доступа.

4.5 Сжатие ответов сервера

Использование сжатия респонзов с помощью gzip позволяет уменьшить размер передаваемых данных между сервером и клиентом, что улучшает скорость загрузки страницы и экономит пропускную способность сети. Это особенно полезно при передаче больших текстовых файлов, стилей CSS и JavaScript, которые часто могут быть сжаты существенно без потери качества.

В данном примере конфигурации Nginx демонстрируется использование сжатия респонзов с помощью gzip (Рисунок 4).

```
http{
    server {
        gzip on;
        gzip_min_length 100;
        gzip_comp_level 3;
        gzip_types text/plain;
        gzip_types text/css;
        gzip_types text/javascript;
        gzip_disable "msie6";
    }
}
```

Рисунок 4 – Конфигурация основных настроек gzip

Давайте рассмотрим каждую из указанных настроек:

1. gzip on: эта директива включает сжатие респонзов с помощью gzip.
2. gzip_min_length 100: эта директива указывает минимальный размер ответа, который будет сжиматься. В данном случае, ответы, размером менее 100 байт, не будут сжиматься.
3. gzip_comp_level 3: эта директива задает уровень компрессии для сжатия gzip. Уровень 3 является стандартным и обеспечивает хороший баланс между скоростью и степенью сжатия.
4. gzip_types text/plain, gzip_types text/css, gzip_types text/javascript: эти директивы указывают типы контента, которые могут быть сжаты с помощью gzip. В данном случае, текстовые файлы (text/plain), CSS-файлы (text/css) и JavaScript-файлы (text/javascript) будут сжиматься.

5. gzip_disable "msie6": эта директива позволяет отключить сжатие для устаревших браузеров, в данном случае для Internet Explorer 6. Такие браузеры не всегда могут правильно обрабатывать сжатые респонзы, поэтому можно отключить сжатие для них.

4.6 Базовая аутентификация

Базовая аутентификация (Basic Authentication) — это простой механизм аутентификации, который использует комбинацию имени пользователя и пароля для ограничения доступа к ресурсам сервера. В Nginx базовая аутентификация может быть легко настроена с использованием модуля ngx_http_auth_basic.

В данной конфигурации Nginx используется базовая аутентификация для ограничения доступа к пути /profile (Рисунок 5).

```
http{
    server{
        location /profile {
            auth_basic "Restricted folder";
            auth_basic_user_file /etc/nginx/creds/.htpasswd;
            access_log /var/log/nginx/new.log upstream_time;
        }
    }
}
```

Рисунок 5 – Конфигурация основных настроек базовой аутентификации

Давайте рассмотрим каждую из указанных настроек:

1. location /profile ... : директива location определяет путь к ресурсу, для которого будет применяться базовая аутентификация. В данном случае, это путь /profile.
2. auth_basic "Restricted folder": эта директива устанавливает сообщение, которое будет отображаться в диалоговом окне аутентификации браузера, при попытке доступа к ограниченной папке /profile. В данном случае, сообщение будет "Restricted folder".
3. auth_basic_user_file /etc/nginx/creds/.htpasswd: эта директива указывает путь к файлу .htpasswd, который содержит пары "имя пользователя:зашифрованный пароль". В данном случае, файл .htpasswd находится по пути /etc/nginx/creds/.htpas Этот файл должен быть создан и содержать допустимые учетные данные для аутентификации пользователей.
4. access_log /var/log/nginx/new.log upstream_time: эта директива настраивает запись журнала доступа в файл /var/log/nginx/new.log с указанием времени выполнения каждого запроса (upstream_time). Журнал доступа содержит информацию о запросах к ресурсу /profile.

Таким образом, при доступе к пути /profile, браузер будет запрашивать у пользователя имя пользователя и пароль. Введенные учетные данные будут проверяться на соответствие с данными из файла .htpasswd. Если аутентификация прошла успешно, пользователю будет разрешен доступ к ограниченной папке /profile, а информация о запросах будет записываться в указанный журнал доступа.

Эта конфигурация позволяет ограничить доступ к конкретной папке на сервере с использованием базовой аутентификации, что обеспечивает дополнительный уровень защиты и контроля доступа к конфиденциальной информации.

4.7 Ограничения по геоданным

Nginx GeoIP2 — это модуль, который позволяет использовать информацию о географическом расположении клиентов веб-сервера на основе их IP-адресов. Этот модуль использует базу данных GeoIP2, которая содержит информацию о стране, регионе, городе, координатах и других атрибутах, связанных с конкретными IP-адресами. Для работы с данным модулем необходимо загрузить в конфигурацию сервера модуль ngx_http_geoip2, а также предоставить серверу доступ к базе геоданных, в нашем случае использовалась GeoLite2-City.mmdb.

Давайте рассмотрим каждую из указанных настроек (Рисунок 6):

```
http{
    geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {
       auto_reload 60m;
       $geoip2_metadata_city_build metadata build_epoch;
       $geoip2_data_country_name country names en;
       $geoip2_data_country_code country iso_code;
       $geoip2_data_city_name city names en;
       $geoip2_data_region_name subdivisions 0 names en;
       $geoip2_data_state_code  subdivisions 0 iso_code;
    }

    map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {
        default no;
        ~^RU: yes; #Россия
        ~^BY: yes; #Белоруссия
        ~^AM: yes; #Армения
        ~^KZ: yes; #Казахстан
        ~^KG: yes; #Кыргызстан
        UA:40 yes; #Севастополь
        UA:43 yes; #Крым
        UA:14 yes; #Донецкая область
        UA:09 yes; #Луганская область
        UA:23 yes; #Запорожская область
        UA:65 yes; #Херсонская область
        GE:AB yes; #Абхазия
    }
    server{
    }
}
```

Рисунок 6 – Конфигурация основных настроек модуля GeoIP2

1. geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb ... : директива geoip2 указывает путь к файлу базы данных GeoIP2, который содержит информацию о геолокации IP-адресов. В данном случае, файл находится по

пути /usr/share/GeoIP/GeoLite2-City.mmdb. Дополнительно указаны переменные, которые будут содержать информацию о стране, регионе, городе и других атрибутах.

2. auto_reload 60m: эта настройка указывает интервал автоматической перезагрузки базы данных GeoIP2. В данном случае, база данных будет перезагружаться каждые 60 минут.
3. $geoip2_metadata_city_build metadata build_epoch: эта переменная содержит информацию о времени последнего обновления базы данных GeoIP2.
4. $geoip2_data_country_name country names en: эта переменная содержит название страны на основе IP-адреса клиента.
5. $geoip2_data_country_code country iso_code: эта переменная содержит код страны (двухбуквенный ISO-код) на основе IP-адреса клиента.
6. $geoip2_data_city_name city names en: эта переменная содержит название города на основе IP-адреса клиента.
7. $geoip2_data_region_name subdivisions 0 names en: эта переменная содержит название региона на основе IP-адреса клиента.
8. $geoip2_data_state_code subdivisions 0 iso_code: эта переменная содержит код региона (двухбуквенный ISO-код) на основе IP-адреса клиента.
9. geoip blocking: эта директива включает блокировку доступа на основе географической локации. Если клиент находится в запрещенном регионе, его доступ будет заблокирован.
10. map "$geoip2_data_country_code:$geoip2_data_state_code"$allowed_reg ... : эта директива определяет переменную $allowed_reg, которая будет содержать значение yes или no в зависимости от географической локации клиента. В данном случае, используется регулярное выражение для определения разрешенных регионов. Если соответствующий регион найден в базе данных GeoIP2, переменная будет иметь значение yes, в противном случае - no.
11. if ($allowed_reg = no) return 444; : эта конструкция проверяет значение переменной $allowed_reg. Если значение равно no, то выполняется директива return 444, которая прекращает обработку запроса и возвращает ошибку 444 ("No Response") клиенту. Это позволяет блокировать доступ к серверу для клиентов из запрещенных регионов.

Обратите внимание, что данная конфигурация может быть дополнена дру-

гими настройками сервера внутри блока server ... .

4.8 Балансировка нагрузки

Nginx предоставляет несколько алгоритмов балансировки нагрузки, которые определяют способ распределения входящих запросов между серверами в группе. Каждый алгоритм имеет свои особенности и может быть выбран в зависимости от требований вашей системы. Ниже приведены основные алгоритмы балансировки нагрузки, поддерживаемые Nginx:

1. Round Robin (Поочередный выбор):
   а) Алгоритм по умолчанию.
   б) Запросы распределяются по серверам в группе в порядке их указания.
   в) При каждом новом запросе выбирается следующий сервер в порядке списка.
   г) Простой и равномерный способ распределения нагрузки.
2. Least Connections (Выбор сервера с наименьшим количеством активных соединений):
   а) Запросы направляются на сервер с наименьшим количеством активных соединений.
   б) Позволяет распределить нагрузку более равномерно, учитывая текущую нагруженность серверов.
3. IP Hash (Хеширование по IP-адресу):
   а) Каждый клиентский IP-адрес сопоставляется с конкретным сервером.
   б) Позволяет обеспечить сохранение состояния сессии для клиента на протяжении всего времени взаимодействия с сервером.
   в) Гарантирует, что все запросы от одного клиента будут направлены на один и тот же сервер.
4. Generic Hash (Общее хеширование):
   а) Запросы хешируются с использованием произвольного ключа, указанного в конфигурации.
   б) Позволяет гибко настраивать способ хеширования для распределения нагрузки на основе определенных параметров запроса или других данных.

В данном примере конфигурации Nginx используется алгоритм балансировки нагрузки Round Robin, который является алгоритмом по умолчанию. Запросы будут распределяться между серверами в группе в порядке их указания.

ровки нагрузки "Least Connections"для группы серверов php_servers (Рисунок 7).

```
http{
    upstream php_servers {
            least_conn;
            server localhost:10001;
            server localhost:10002;
            server localhost:10003;
        }
    server{
        listen       81;
        listen       443 ssl;
        location /testphp {
            fastcgi_cache microcache;
            fastcgi_cache_valid 200 60m;
            fastcgi_pass 127.0.0.1:9000;
            proxy_set_header proxy_header_to_server nginx;
            add_header proxy_header nginx;
            proxy_pass http://php_servers;
        }
    }
}
```

Рисунок 7 – Конфигурация основных настроек балансировщика нагрузки

Давайте рассмотрим, что делает каждая настройка:

1. least_conn: этот алгоритм направляет запросы на сервер с наименьшим количеством активных соединений. Он распределяет нагрузку равномерно между серверами, учитывая их текущую нагруженность. Серверы localhost:10001, localhost:10002 и localhost:10003 указаны в качестве серверов для балансировки.
2. listen 81, и listen 443 ssl: эти директивы указывают Nginx слушать соединения на портах 81 и 443 с использованием SSL/TLS.
3. location /testphp: в этом блоке настраивается обработка запросов, которые соответствуют пути /testphp. Они проксируются на серверы из группы php_servers с использованием балансировки нагрузки. Также присутствуют дополнительные настройки, такие как кеширование (fastcgi_cache) и передача заголовков (proxy_set_header, add_header).

В данной конфигурации алгоритм балансировки нагрузки "Least Connections"выбр для равномерного распределения запросов между серверами localhost:10001, localhost:10002 и localhost:10003. Это позволяет достичь более эффективного использования ресурсов и более высокой отказоустойчивости системы.

4.9 Логирование

Nginx логирование играет важную роль, позволяя отслеживать и анализировать различные события, ошибки и активность сервера. Nginx предлагает различные типы логов, которые можно настроить в файле конфигурации nginx.conf. Давайте рассмотрим основные типы логов и их назначение:

1. Access logs (логи доступа): они записывают информацию о каждом запросе, поступающем на сервер Nginx. Access logs содержат информацию, такую как IP-адрес клиента, время запроса, HTTP-метод, запрошенный URL, код состояния ответа и объем переданных данных. Эти логи полезны для мониторинга активности сервера и анализа трафика.
2. Error logs (логи ошибок): они регистрируют различные ошибки, возникающие в процессе обработки запросов. Error logs включают сообщения об ошибках, критические события и предупреждения, связанные с работой сервера. Эти логи помогают в выявлении проблем и их диагностике для обеспечения стабильности и безопасности сервера.
3. Application logs (логи приложений): если вы используете Nginx в качестве прокси или обратного прокси для приложений, таких как веб-серверы или приложения на основе фреймворка, то вы можете настроить логирование событий, связанных с вашими приложениями. Это позволяет вам отслеживать действия и проблемы, возникающие в приложениях.

Для каждого из этих типов логов в Nginx можно настроить формат записей, место хранения файлов логов и уровень подробности записываемых сообщений. Это позволяет администраторам настроить логирование согласно своим требованиям и предпочтениям.

В представленной конфигурации Nginx определены два формата логов: upstream_time и main (Рисунок 8).

```
http{
    log_format upstream_time '$remote_addr - $remote_user [$time_local] '
                             '"$request" $status $body_bytes_sent '
                             '"$http_referer" "$http_user_agent"'
                             'rt=$request_time uct="$upstream_connect_time" '
                             'uht="$upstream_header_time" urt="$upstream_response_time"';
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    server{
        error_log off;
        access_log off;
        location /profile {
            error_log /var/log/nginx/errors_profile.log main
            access_log /var/log/nginx/access_profile.log upstream_time;
        }
    }
}
```

Рисунок 8 – Конфигурация основных настроек логирования

Давайте разберем, что делают эти настройки:

1. log_format upstream_time: этот формат логов определяет пользовательский формат записей для логов доступа. Он содержит следующие переменные и данные:
2. $remote_addr: IP-адрес клиента

3. $remote_user: имя пользователя (если используется базовая аутентификация)
4. $time_local: локальное время запроса
5. "$request": сам запрос
6. $status: код состояния ответа сервера
7. $body_bytes_sent: размер ответа в байтах
8. "$http_referer": HTTP-заголовок Referer (если присутствует)
9. "$http_user_agent": HTTP-заголовок User-Agent
10. rt=$request_time: время обработки запроса
11. uct="$upstream_connect_time": время установки соединения с бэкенд-сервером
12. uht="$upstream_header_time": время получения заголовков ответа от бэкенд-сервера
13. urt="$upstream_response_time": время получения ответа от бэкенд-сервера
14. log_format main: этот формат логов также определяет пользовательский формат записей для логов доступа. Он содержит переменные и данные, такие как IP-адрес клиента, имя пользователя (если есть), локальное время запроса, сам запрос, код состояния ответа сервера, размер ответа, HTTP-заголовки Referer и User-Agent, а также заголовок X-Forwarded-For (если присутствует).

Внутри блока server для пути /profile определены настройки логирования:

1. error_log /var/log/nginx/errors_profile.log main: указывает путь к файлу, в который будут записываться ошибки, связанные с обработкой запросов для данного пути.
2. access_log /var/log/nginx/access_profile.log upstream_time: указывает путь к файлу, в который будут записываться логи доступа для данного пути, используя формат upstream_time для форматирования записей логов.

Обратите внимание, что логирование в блоке server может быть включено или выключено с помощью директив error_log и access_log. В представленной конфигурации логирование в целом выключено для данного сервера, но включено и настроено для пути /profile.

Настраивая логирование в Nginx, вы можете контролировать формат записей, выбирать, какие данные включать, и указывать файлы, в которые записывать логи. Это помогает в мониторинге и отладке сервера, а также в получении полезной информации о запросах и ошибках.

5 Описание программы-конфигуратора formatter

Рассмотренный выше файл конфигурации веб-сервера Nginx был прописан вручную. Однако данный подход занял довольно много времени, поскольку стандартный файл конфигурации, созданный при установке веб-сервера, содержал малое количество полезных настроек. Кроме того, было необходимо периодически тестировать файл на соответствие синтаксису Nginx и убеждаться в том, что настройки действительно применяются. Данные проблемы можно решить с помощью применения скрипта-конфигуратора. Написание подобного кода на языке c++ составляет практическую часть данной курсовой работы.

Программу-конфигуратор formatter.cpp (листинг которой приведен в приложении Б) необходимо скомпиллировать и запустить в терминале с помощью следующей команды (Рисунок 9):

```
g++ formatter.cpp -o formatter &amp;amp; ./formatter
```

Рисунок 9 – Команда для запуска программы formatter.cpp

После ее выполнения в консоли появится приветствие, в ходе которого будет необходимо ввести адрес корневой папки веб-сервера и выбрать режим работы:

1. default – по указанному адресу будет создан файл конфигурации nginx.conf, содержащий в себе стандартные настройки (версия для Ubuntu LTS 22.04)
2. recommended – по указанному адресу будет создан файл конфигурации nginx.conf, содержащий в себе все настройки, описанные выше
3. custom – режим конфигурации (заполнение пустого файла конфигурации nginx.conf)

В режиме конфигурации пользователю предоставлена возможность самому прописывать необходимые для его приложения настройки или частично использовать рекоммендованные параметры.

В результате работы программы в режиме конфигурации, будет создан синтаксически верный и рабочий файл конфигурации nginx.conf по адресу, указанному пользователем во время приветствия. Давайте рассмотрим работу данной программы более детально.

6 Тестирование программы-конфигуратора formatter

Запустим в терминале программу-конфигуратор formatter.cpp с помощью указанной в предыдущем разделе команды. После введем необходимые настройки, используя предлагаемый интерфейс (Рисунки 10–18).

```
[alse0722@gavno pract]$ g++ formatter.cpp -o formatter & ./formatter
[1] 94505

        Приветствуем в конфигураторе nginx!

        Укажите желаемое расположение nginx.conf:

        Какую конфигурацию необходимо применить? (default/preset/custom):
                [default] Применить стандартную конфигурацию для Ubuntu LTS 22.04
                [preset]  Применить рекомендованную конфигурацию, разработанную в рамках курса
                [custom]  Составить собственную конфигурацию

        Ваш выбор: custom

        Давайте приступим к персональной настройке файла конфигурации!
```

Рисунок 10 – Запуск программы и приветствие

```
                        [Конфигурация блока events]

        ! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter
        ! Если хотите установить настройку по умолчанию, Укажите default

        Укажите количество одновременных соединений для каждого процесса: 1024

        Укажите, должен ли рабочий процесс принимать несколько соединений одновременно [yes/no]: yes

        Укажите, должен ли рабочий процесс использовать мьютекс при приеме новых соединений [yes/no]: no

        Укажите дополнительные настройки для блока events [no чтобы закончить ввод]:
                Доп. настройка 1: #test events

                Доп. настройка 2: no

                                [Конец конфигурации блока events]
```

Рисунок 11 – Конфигурация блока events

```
                        [Конфигурация блока http]

        ! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter
        ! Если хотите установить настройку по умолчанию, Укажите default

        Укажите настройку глобальных хэдеров [default/custom]: default

        Укажите используемые форматы логов [default/custom]: default

        В файл конфигурации были добавлены следующие виды форматирования:
log_format upstream_time '$remote_addr - $remote_user [$time_local] '
                                '"$request" $status $body_bytes_sent '
                                '"$http_referer" "$http_user_agent"'
                                'rt=$request_time uct="$upstream_connect_time" '
                                'uht="$upstream_header_time" urt="$upstream_response_time"';

log_format  main '$remote_addr - $remote_user [$time_local] "$request" '
                        '$status $body_bytes_sent "$http_referer" '
                        '"$http_user_agent" "$http_x_forwarded_for"';


        Укажите зоны ограничения соединений [default/custom]: default

        В файл конфигурации добавлена зона ограничения соединений с именем per_ip и размером 5 мегабайт.
                Данная зона ставит ограничения на количество соединений для каждого IP-адреса по-отдельности
```

Рисунок 12 – Конфигурация блока http

```
Использовать модуль FastCGI? [yes/no]: yes

        Укажите тип настроек FastCGI [default/custom]: default

                Какой путь использовать для кэша FastCGI? [default/custom]: custom

                        Укажите кастомный путь: /custom/path/to/fastcgi/folder

Создать группу серверов для балансировщика нагрузки? [yes/no]: yes

        Укажите имя группы серверов: balancer_group_1

        Укажите алгоритм балансировки: least_conn

        Укажите сервера группы (ip:port) [no чтобы закончить ввод]:
                Server 1: 192.168.0.1

                Server 2: 192.168.0.2

                Server 3: 192.168.0.3

                Server 4: no
```

Рисунок 13 – Конфигурация блока http (продолжение)

```
Использовать модуль Geoip2? [yes/no]: yes

Какой путь использовать для доступа к GeoLite2-City.mmdb? [default/custom]: custom

Введите путь до GeoLite2-City.mmdb: /custom/path/to/geoipdb

Текущие разрешенные регионы:
        [1]  RU --> Россия
        [2]  BY --> Белоруссия
        [3]  AM --> Армения
        [4]  KZ --> Казахстан
        [5]  KG --> Кыргызстан
        [6]  UA:4 --> Севастополь
        [7]  UA:4 --> Крым
        [8]  UA:1 --> Донецкая область
        [9]  UA:0 --> Луганская область
        [10] UA:2 --> Запорожская область
        [11] UA:6 --> Херсонская область
        [12] GE:A --> Абхазия
Укажите id регионов, которые будут иметь доступ при внештатных ситуциях [no чтобы закончить ввод]: 1 2 7 8 9

Укажите дополнительные настройки для блока http [no чтобы закончить ввод]:
        Доп. настройка 1: #test http

        Доп. настройка 2: no

                        [Конец конфигурации блока http]
```

Рисунок 14 – Конфигурация блока http (продолжение)

```
                [Конфигурация блока server]

! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter
! Если хотите установить настройку по умолчанию, Укажите default

Укажите прослушиваемые порты [через пробел]: 80 81 82

Укажите разрешенные хосты [через пробел]: 192.168.0.4 192.168.0.5 localhost

Укажите запрещенные хосты [через пробел]: all

Укажите имя сервера [default]: server_custom_name

Укажите корневой каталог сайта [default]: /custom/path/to/sites/root

Включить логирование ошибок для всех директорий? [yes/no]: no

Включить логирование доступа для всех директорий? [yes/no]: no

Включить шифрование SSL/TLS? [yes/no]: yes

        Укажите прослушивающий порт: 443

        Укажите файл ssl-сертификата (.crt): path/to/ssl/data/nginx.cert

        Укажите файл ssl-ключа (.key): path/to/ssl/data/nginx.key
```

Рисунок 15 – Конфигурация блока server

```
Включить ограничения буферизации сообщений? [yes/no]: yes

Включить таймаут ожидания для передачи запроса клиента? [yes/no]: yes

        Укажите таймаут для хэдера запроса: 10

        Укажите таймаут для тела запроса: 20

Включить таймаут ожидания для активного соединения? [yes/no]: yes

Включить таймаут ожидания для отправки данных клиенту? [yes/no]: yes

Отключить отображение информации о версии Nginx в хэдерах? [yes/no]: yes

Включить сжатие ответов сервера с помощью gzip? [yes/no]: yes

        Укажите минимальный размер сжимаемых файлов: 100

        Укажите степень сжатия файлов: 3

        Укажите типы сжимаемых файлов через пробел(Пр.: css plain): css html php

Ограничитить кэширование FastCGI по определенным правилам? [yes/no]: yes

Укажите правила ограничений [no чтобы закончить ввод]:
        Правило 1: test_rule_1

        Правило 2: test_rule_2

        Правило 3: no

Укажите дополнительные настройки для блока server [no чтобы закончить ввод]:
        Доп. настройка 1: #test server

        Доп. настройка 2: no

                        [Конец конфигурации блока server]
```

Рисунок 16 – Конфигурация блока server (продолжение)

```
                [Конфигурация блоков location]

! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter
! Если хотите установить настройку по умолчанию, Укажите default

Создать новую локацию? [yes/no]: yes

        Укажите имя локации: /location_one

Включить логирование ошибок для данной директории? [yes/no]: yes

        Укажите каталог для хранения логов ошибок [default]: default

        Укажите имя логов: location_one.log

        Укажите формат логов ошибок [default]: default

Включить логирование доступа для данной директории? [yes/no]: no

Включить базовую аутентификацию? [yes/no]: no

Включить кэширование на стороне клиента? [yes/no]: no

Включить кэширование FastCGI? [yes/no]: no

Добавить кастомные хэдеры? [yes/no]: no

Использовать балансировщик? [yes/no]: no

Укажите дополнительные настройки для блока server [no чтобы закончить ввод]:
        Доп. настройка 1: #test location_one

        Доп. настройка 2: no
```

Рисунок 17 – Конфигурация блоков location (директория /location_one)

```
Создать новую локацию? [yes/no]: yes
        Укажите имя локации: /location_two
Включить логирование ошибок для данной директории? [yes/no]: no
Включить логирование доступа для данной директории? [yes/no]: yes
        Укажите каталог для хранения логов ошибок [default]: /path/to/logs
        Укажите имя логов: location_two_access.log
        Укажите формат логов ошибок [default]: upstream_time
Включить базовую аутентификацию? [yes/no]: yes
                Укажите файл с учетными данными: /path/to/credentials/.lgnpsswd
Включить кэширование на стороне клиента? [yes/no]: yes
        Укажите время валидности кэша: 60m
Включить кэширование FastCGI? [yes/no]: yes
        Укажите зону ключей кэша FastCGI: microcache
        Укажите время валидности кэша FastCGI: 1M
        Укажите прокси сервер FastCGI: localhost:9000
        Использовать дополнительные правила кэширования FastCGI? [yes/no]: yes
```

Рисунок 18 – Конфигурация блоков location (директория /location_two)

Проверим файл nginx.conf. Очевидно что файл был сгенерирован (Рисунки 19–23), можно просмотреть его содержимое в любом доступном текстовом редакторе (в данном случае использовался Visual Code с рисширением Nginx Configuration для наглядности).

```
Добавить кастомные хэдеры? [yes/no]: yes
        Укажите хэдер 1:
                Имя: custom_header_name_1
                Данные: custom_header_data_1
        Укажите хэдер 2:
                Имя: custom_header_name_2
                Данные: custom_header_data_2
        Укажите хэдер 3:
                Имя: no
Использовать балансировщик? [yes/no]: yes
        Укажите имя группы серверов: balancer_group_1
Укажите дополнительные настройки для блока server [no чтобы закончить ввод]:
        Доп. настройка 1: #test location_two
        Доп. настройка 2: no
Создать новую локацию? [yes/no]: no
                        [Конец конфигурации блоков locations]
```

Рисунок 19 – Конфигурация блоков location (директория /location_two - продолжение)

pract > nginx.conf
You, 3 seconds ago | 1 author (You)

```
events {
    worker_connections  1024;
    multi_accept on;
    accept_mutex off;
    #test events;
}

http {
    include mime.types;
    default_type application/octet-stream;
    include fastcgi_params;
    include fastcgi.conf;
    log_format upstream_time '$remote_addr - $remote_user [$time_local] '
                             '"$request" $status $body_bytes_sent '
                             '"$http_referer" "$http_user_agent"'
                             'rt=$request_time uct="$upstream_connect_time" '
                             'uht="$upstream_header_time" urt="$upstream_response_time"';

    log_format  main '$remote_addr - $remote_user [$time_local] "$request" '
                     '$status $body_bytes_sent "$http_referer" '
                     '"$http_user_agent" "$http_x_forwarded_for"';

    limit_conn_zone $binary_remote_addr zone=per_ip:5m;
    fastcgi_cache_path /custom/path/to/fastcgi/folder/cache levels=1:2 keys_zone=microcache:10m max_size=500m inactive=10m;
    fastcgi_cache_key "$scheme$request_method$host$request_uri";
    fastcgi_ignore_headers Cache-Control Expires Set-Cookie ;
    upstream balancer_group_1 {
        least_conn;
        server 192.168.0.1;
        server 192.168.0.2;
        server 192.168.0.3;
    }
    geoip2 /custom/path/to/geoipdb;GeoLite2-City.mmdb {
        auto_reload 60m;
        $geoip2_metadata_city_build metadata build_epoch;
        $geoip2_data_country_name country names en;
        $geoip2_data_country_code country iso_code;
        $geoip2_data_city_name city names en;
        $geoip2_data_region_name subdivisions 0 names en;
        $geoip2_data_state_code  subdivisions 0 iso_code;
    }
```

Рисунок 20 – Обзор сгенерированного файла nginx.conf (блоки events и http)

```
    map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {
        default no;
        default no;
        ~^RU:  yes;
        ~^BY:  yes;
        ~^AM:  yes;
        ~^KZ:  yes;
        ~^KG:  yes;
        UA:40  yes;
        UA:43  yes;
        UA:14  yes;
        UA:09  yes;
        UA:23  yes;
        UA:65  yes;
        GE:AB  yes;
    }
    # map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {
    #   default no;
    #   ~^RU:  yes;
    #   ~^BY:  yes;
    #   UA:43  yes;
    #   UA:14  yes;
    #   UA:09  yes;
    # }
```

Рисунок 21 – Обзор сгенерированного файла nginx.conf (блок http - продолжение)

```
    server {
        listen 80;
        listen 81;
        listen 82;
        listen 443 ssl;
        allow 192.168.0.4;
        allow 192.168.0.5;
        allow localhost;
        deny all;
        server_name server_custom_name
        root /custom/path/to/sites/root
        server_tokens off;
        error_log off;
        access_log off;
        ssl_certificate path/to/ssl/data/nginx.cert;
        ssl_certificate_key path/to/ssl/data/nginx.key;
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout 5m;
        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;
        client_body_buffer_size 16k;
        client_header_buffer_size 1k;
        client_max_body_size 8m;
        large_client_header_buffers 2 1k;
        client_header_timeout 10;
        client_body_timeout 20;
        keepalive_timeout 65;
        send_timeout 10;
        gzip on;
        gzip_min_length 100;
        gzip_comp_level 3;
        gzip_types text/css;
        gzip_types text/html;
        gzip_types text/php;
        gzip_disable "msie6";
        set $no_cache 0;
        if (test_rule_1) { set $no_cache 1; }
        if (test_rule_2) { set $no_cache 1; }
        if ($allowed_reg = no) {
            return 444;
        }
        #test server;
```

Рисунок 22 – Обзор сгенерированного файла nginx.conf (блок server)

```
        location  ~ /\.ht {
            deny  all;
        }

        location /location_one {
            access_log off;
            error_log /etc/nginx/logs//location_one.log yes;
            #test location_one;
        }

        location /location_two {
            error_log off;
            error_log /path/to/logs/location_two_access.log yes;
            auth_basic "Restricted access";
            auth_basic_user_file /path/to/credentials/.lgnpsswd;
            fastcgi_cache microcache;
            fastcgi_cache_valid 200  1M;
            fastcgi_pass localhost:9000;
            fastcgi_no_cache $no_cache;
            expires yes;
            add_header custom_header_name_1 custom_header_data_1;
            add_header custom_header_name_2 custom_header_data_2;
            http://balancer_group_1;
            #test location_two;
        }

    }
    #test http;
}
```

Рисунок 23 – Обзор сгенерированного файла nginx.conf (блоки location)

Проверим правильность синтаксиса файла конфигурации и его работоспособность с помощью встроенных функций проверки синтаксиса и запуска бесшовной реконфигурации сервера (Рисунок 24).

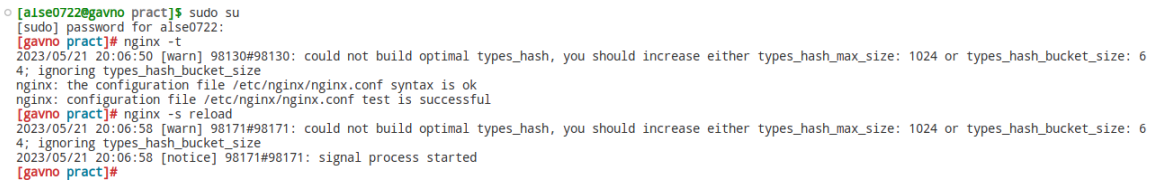

Рисунок 24 – Проверка сгенерированного файла конфигурации nginx.conf

Очевидно, что синтаксис конфигурационного файла в порядке. Кроме того, веб-сервер успешно применил его настройки и работает в штатном режиме.

ЗАКЛЮЧЕНИЕ

В заключение, данная курсовая работа по теме "Защита веб-сервера Nginx"была направлена на исследование и практическую реализацию методов и мер безопасности, связанных с конфигурацией веб-сервера Nginx.

В рамках практической части работы была разработана программа, способная конфигурировать файл nginx.conf, который является основным файлом конфигурации Nginx. Программа предоставляет пользователю интерактивный интерфейс для выбора типа конфигурации: стандартной, рекомендованной или кастомной. В зависимости от выбора пользователя, программа создает или изменяет файл nginx.conf соответствующим образом.

Полученные результаты позволили продемонстрировать возможности программы в автоматизации процесса конфигурирования Nginx и обеспечении необходимых настроек безопасности. Это может включать установку корректных значений для директив, таких как ограничение подключений, контроль времени ожидания, установка заголовков безопасности, скрытие информации о сервере и других аспектов, которые способствуют общей защите веб-сервера.

Важно отметить, что представленная программа является примером реализации и может быть доработана и расширена в соответствии с потребностями и требованиями конкретного сервера и его окружения. Также следует отметить, что конфигурация веб-сервера Nginx должна рассматриваться как одна из многих мер безопасности, а полная защита веб-сервера требует комплексного подхода, включая обновление программного обеспечения, управление доступом, мониторинг и другие стратегии и механизмы безопасности.

В целом, данная курсовая работа по защите веб-сервера Nginx и разработка программы для конфигурирования nginx.conf позволила изучить и применить практические методы обеспечения безопасности, связанные с одним из самых популярных веб-серверов, что является важной составляющей для обеспечения защиты веб-приложений и данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Что такое Nginx и как правильно его настроить [Электронный ресурс] - URL: https://www.nic.ru/help/chto-takoe-nginx-i-kak-pravil6no-ego-nastroit6_-11046.html (дата обращения 02.05.2023) Загл. с экрана. Яз. рус.

2 Модуль Nginx для борьбы с DDoS [Электронный ресурс] - URL: https://habr.com/ru/articles/139931/ (дата обращения 02.05.2023) Загл. с экрана. Яз. рус.

3 Защита Nginx от DDoS атак [Электронный ресурс] - URL: https://amkolomna.ru/content/zashchita-nginx-ot-ddos-atak (дата обращения 02.05.2023) Загл. с экрана. Яз. рус.

4 Настройка защиты от DDoS-атак [Электронный ресурс] - URL: https://rudocs.ispmanager.com/ispmanager-lite/nastrojka-zashchity-ot-ddos-atak (дата обращения 03.05.2023) Загл. с экрана. Яз. рус.

5 Установка Nginx 1.18 на Manjaro [Электронный ресурс] - URL: https://onedev.net/post/1025 (дата обращения 04.05.2023) Загл. с экрана. Яз. рус.

6 Как настроить Nginx в качестве балансировщика нагрузки [Электронный ресурс] - URL: https://habr.com/ru/companies/first/articles/683870/ (дата обращения 08.05.2023) Загл. с экрана. Яз. рус.

7 Nginx cache: всё новое — хорошо забытое старое [Электронный ресурс] - URL: https://habr.com/ru/articles/428127/ (дата обращения 08.05.2023) Загл. с экрана. Яз. рус.

8 Updating GeoIP and GeoLite Databases [Электронный ресурс] - URL: https://dev.maxmind.com/geoip/updating-databases?lang=en (дата обращения 10.05.2023) Загл. с экрана. Яз. англ.

9 Настройка логов nginx [Электронный ресурс] - URL: https://ixnfo.com/nastroyka-logov-nginx.html (дата обращения 10.05.2023) Загл. с экрана. Яз. рус.

10 Простая аутентификация на NGINX с помощью LUA [Электронный ресурс] - URL: https://habr.com/ru/articles/351904/ (дата обращения 12.05.2023) Загл. с экрана. Яз. рус.

ПРИЛОЖЕНИЕ А

Разработанные настройки для веб-сервера nginx

```
events {
        worker_connections  1024;
}

http {
        include             mime.types;
        include             fastcgi_params;
        include             fastcgi.conf;
        default_type        application/octet-stream;

        add_header Access-Control-Allow-Origin;
        add_header Caching $upstream_cache_status;
        add_header Cache-Control public;
        add_header Vary Accept-Encoding;
        add_header X-Content-Type-Options nosniff;
        add_header X-XSS-Protection "1; mode=block";

        log_format upstream_time '$remote_addr - $remote_user [$time_local] '
        '"$request" $status $body_bytes_sent '
        '"$http_referer" "$http_user_agent"'
        'rt=$request_time uct="$upstream_connect_time" uht="$upstream_header_time"
↪   urt="$upstream_response_time"';

        log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

        limit_conn_zone $server_name zone=per_vhost:5m;
        limit_conn_zone $binary_remote_addr zone=per_ip:5m;

        fastcgi_cache_path /etc/nginx/cache levels=1:2 keys_zone=microcache:10m max_size=500m
↪   inactive=10m;
        fastcgi_cache_key "$scheme$request_method$host$request_uri";
        fastcgi_ignore_headers Cache-Control Expires Set-Cookie ;

        upstream php_servers {
                least_conn;
                server localhost:10001;
                server localhost:10002;
                server localhost:10003;
        }

        geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {
                auto_reload 60m;
                $geoip2_metadata_city_build metadata build_epoch;
                $geoip2_data_country_name country names en;
                $geoip2_data_country_code country iso_code;
                $geoip2_data_city_name city names en;
                $geoip2_data_region_name subdivisions 0 names en;
                $geoip2_data_state_code  subdivisions 0 iso_code;
        }

        map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {
                default no;
                ~^RU: yes; #Россия
                ~^BY: yes; #Белоруссия
                ~^AM: yes; #Армения
```

```
        ~^KZ: yes; #Казахстан
        ~^KG: yes; #Кыргызстан
        UA:40 yes; #Севастополь
        UA:43 yes; #Крым
        UA:14 yes; #Донецкая область
        UA:09 yes; #Луганская область
        UA:23 yes; #Запорожская область
        UA:65 yes; #Херсонская область
        GE:AB yes; #Абхазия
}

server {

        listen       81;
        listen       443 ssl;

        allow   192.168.0.174;
        deny all;

        server_name  localhost;

        root /etc/nginx/site/gaming/;

        error_log off;
        access_log off;

        ssl_certificate /etc/nginx/ssl/nginx.crt;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;
        ssl_session_cache    shared:SSL:1m;
        ssl_session_timeout     5m;
        ssl_ciphers  HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers  on;

        client_body_buffer_size 16k;
        client_header_buffer_size 1k;
        client_max_body_size 8m;
        large_client_header_buffers 2 1k;

        client_body_timeout 12;
        client_header_timeout 12;

        keepalive_timeout 65;
        send_timeout 10;

        server_tokens off;

        gzip on;
        gzip_min_length 100;
        gzip_comp_level 3;

        gzip_types text/plain;
        gzip_types text/css;
        gzip_types text/javascript;

        gzip_disable "msie6";

        set $no_cache 0;
        if ($request_method = POST) { set $no_cache 1; }
        if ($query_string != "") { set $no_cache 1; }
        if ($request_uri ~* "/profile") { set $no_cache 1; }
```

```
        if ($allowed_reg = no) {
                return 444;
        }


        location /test_auth {
                #base auth
                auth_basic "Restricted access";
                auth_basic_user_file /etc/nginx/creds/.htpasswd;
                access_log /var/log/nginx/new.log upstream_time;
        }

        location ~* \.(css|js|jpg|png|gif)$ {

                fastcgi_cache microcache;
                fastcgi_cache_valid 200 60m;
                expires 1M;

                access_log off;
                error_log off;

                limit_conn per_ip 1;
        }

        location /test_balancer {
                fastcgi_cache microcache;
                fastcgi_cache_valid 200 60m;
                fastcgi_pass 127.0.0.1:9000;
                fastcgi_no_cache $no_cache;

                proxy_set_header proxy_header_to_server nginx;
                add_header proxy_header nginx;

                proxy_pass http://php_servers;
        }

        location ~ /\.ht {
                deny  all;
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

Листинг программы-конфигуратора formatter.cpp

```
#include <iostream>
#include <cstdio>
#include <fstream>
#include <istream>
#include <vector>
#include <regex>
#include <string>
#include <iterator>

using namespace std;

typedef vector<string> vs;
typedef string ss;

struct events_block
{
	ss start;
	ss tab;
	vs workers;
	vs multi;
	vs mutex;
	vs custom;
	ss end;
};
struct http_block
{
	ss start;
	ss tab;
	vs include;
	vs global_headers;
	vs log_formatting;
	vs limit_concurrency;
	vs fast_cgi;
	vs load_balancer;
	vs geoip;
	vs geoip_blocks_allow;
	vs geoip_blocks_deny;
	vs custom;
	ss end;
};
struct server_block
{
	ss start;
	ss tab;
	vs listen;
	vs allow;
	vs deny;
	ss server_name;
	ss root_folder;
	vs global_logs_status;
	vs global_logs_settings;
	vs ssl;
	vs buffers;
	vs timeouts;
	vs keepalive;
	ss server_token;
	vs gzip;
```

```
        vs caching;
        vs custom;
        ss end;
};
struct location_block
{
        ss start;
        ss tab;
        ss location_name;
        vs auth;
        vs fast_cgi;
        ss expires;
        vs custom_logs_status;
        vs custom_logs_settings;
        vs custom_headers;
        ss proxy_pass;
        vs custom;
        ss end;
};
struct nginx_conf
{
        events_block events;
        http_block http;
        server_block server;
        vector<location_block> locations;
        ss location;
};

nginx_conf config;

vs readInputValues()
{
        vs values;
        string input;

        getline(cin, input);

        istringstream iss(input);
        string value;
        while (iss >> value)
        {
                values.push_back(value);
        }

        return values;
}

void setEventsBlock()
{
        ss workers, multi, mutex, custom;
        int cnt(1);

        config.events.start = "\n\tevents {";
                config.events.tab = "\n\t\t";
                config.events.end = "\n\t}\n";

        cout << "\n\t\n\t\t\t[Конфигурация блока events]\n";
        cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";
        cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";
```

```
	cout << "\n\tУкажите количество одновременных соединений для каждого процесса: ";
	getline(cin, workers);
	if (workers != "")
	if (workers != "default")
	config.events.workers.push_back("worker_connections\t" + workers + ";");
	else
	config.events.workers.push_back("worker_connections 1024;");

	cout << "\n\tУкажите, должен ли рабочий процесс принимать несколько соединений
↪   одновременно [yes/no]: ";
	getline(cin, multi);
	if (multi != "")
	{
		if (multi == "default")
		config.events.multi.push_back("multi_accept on;");
		if (multi == "yes")
		config.events.multi.push_back("multi_accept on;");
		if (multi == "no")
		config.events.multi.push_back("multi_accept off;");
	}

	cout << "\n\tУкажите, должен ли рабочий процесс использовать мьютекс при приеме новых
↪   соединений [yes/no]: ";
	getline(cin, mutex);
	if (mutex != "")
	{
		if (mutex == "default")
		config.events.mutex.push_back("accept_mutex on;");
		if (mutex == "yes")
		config.events.mutex.push_back("accept_mutex on;");
		if (mutex == "no")
		config.events.mutex.push_back("accept_mutex off;");
	}

	cout << "\n\tУкажите дополнительные настройки для блока events [no чтобы закончить ввод]: ";
	cout << "\n\t\tДоп. настройка " << cnt << ": ";
	getline(cin, custom);
	while (custom != "no")
	{
		config.events.custom.push_back(custom + (custom.back() == ';' ? "" : ";"));
		cnt++;
		cout << "\n\t\tДоп. настройка " << cnt << ": ";
		getline(cin, custom);
	}

	cout << "\n\t\t\t\t[Конец конфигурации блока events]\n";
}

void setHttpBlock()
{
	config.http.start = "\n\thttp {";
		config.http.tab = "\n\t\t";
		config.http.end = "\n\t}";
	config.http.include.push_back("include mime.types;");
	config.http.include.push_back("default_type application/octet-stream;");

	vs servers, regions, selected_regions;
	ss header_st, gl_header_name, gl_header_data,
	log_st, log_format_name, log_format_data,
	limc_st, lim_conn_param, lim_conn_name, lim_conn_size,
```

```
        fcgi_st, fcgi_path, fcgi_custom,
        ups_st, ups_name, ups_type, ups_server,
        geo_st, geo_path, geobl_regs,
        custom;
        ss upstream_time_format = R"(log_format upstream_time '$remote_addr - $remote_user
↪   [$time_local] '
        '"$request" $status $body_bytes_sent '
        '"$http_referer" "$http_user_agent"'
        'rt=$request_time uct="$upstream_connect_time" '
        'uht="$upstream_header_time" urt="$upstream_response_time"';
        )";
        ss main_format = R"(log_format  main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';
        )";
        regions = {
                "default no;",
                "~^RU:  yes;",
                "~^BY:  yes;",
                "~^AM:  yes;",
                "~^KZ:  yes;",
                "~^KG:  yes;",
                "UA:40  yes;",
                "UA:43  yes;",
                "UA:14  yes;",
                "UA:09  yes;",
                "UA:23  yes;",
                "UA:65  yes;",
                "GE:AB  yes;"};

        int cnt(1);

        cout << "\n\t\n\t\t\t[Конфигурация блока http]\n";
        cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";
        cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

        cout << "\n\tУкажите настройку глобальных хэдеров [default/custom]: ";
        getline(cin, gl_header_name);

        if (header_st != "no")
        {
                if (header_st == "default")
                {
                        config.http.global_headers.push_back(R"(add_header Access-Control-Allow-Origin;)");
                        config.http.global_headers.push_back(R"(add_header Caching
↪   $upstream_cache_status;)");
                        config.http.global_headers.push_back(R"(add_header Cache-Control public;)");
                        config.http.global_headers.push_back(R"(add_header Vary Accept-Encoding;)");
                        config.http.global_headers.push_back(R"(add_header X-Content-Type-Options nosniff;)");
                        config.http.global_headers.push_back(R"(add_header X-XSS-Protection "1;
↪   mode=block";)");
                }

                if (header_st == "custom")
                {
                        cout << "\n\tУкажите хэдер " << cnt << ": ";
                        cout << "\n\t\tИмя: ";
                        getline(cin, gl_header_name);
                        while (gl_header_name != "no")
                        {
```

```
                        cout << "\n\t\tДанные: ";
                        getline(cin, gl_header_data);
                        config.http.global_headers.push_back(
                        "add_header " + gl_header_name + " " + gl_header_data + ";");
                        cnt++;
                        cout << "\n\tУкажите хэдер " << cnt << ": ";
                        cout << "\n\t\tИмя: ";
                        getline(cin, gl_header_name);
                }
        }
}

cout << "\n\tУкажите используемые форматы логов [default/custom]: ";
getline(cin, log_st);
if (log_st != "")
{
        if (log_st == "default")
        {
                config.http.log_formatting.push_back(upstream_time_format);
                config.http.log_formatting.push_back(main_format);
                cout << "\n\tВ файл конфигурации были добавлены следующие виды
↪   форматирования:\n";
                cout << upstream_time_format << endl;
                cout << main_format << endl;
        }

        if (log_st == "custom")
        {
                cnt = 1;
                config.http.log_formatting.push_back(upstream_time_format);
                config.http.log_formatting.push_back(main_format);
                cout << "\n\tВ файл конфигурации были добавлены следующие виды
↪   форматирования:\n";
                cout << upstream_time_format << endl;
                cout << main_format << endl;

                cout << "\n\tНовый формат " << cnt << ": ";
                cout << "\n\t\tИмя: ";
                getline(cin, log_format_name);

                while (gl_header_name != "no")
                {
                        cout << "\n\t\tДанные: ";
                        getline(cin, log_format_data);

                        config.http.log_formatting.push_back("log_format " + log_format_name + " " +
↪   log_format_data + ";");
                        cnt++;

                        cout << "\n\tНовый формат" << cnt << ": ";
                        cout << "\n\t\tИмя: ";
                        getline(cin, gl_header_name);
                }
        }
}

cout << "\n\tУкажите зоны ограничения соединений [default/custom]: ";
getline(cin, limc_st);
if (limc_st != "")
{
```

```
        if (limc_st == "default")
        {
                config.http.limit_concurrency.push_back(
                "limit_conn_zone $binary_remote_addr zone=per_ip:5m;");
                cout << "\n\tВ файл конфигурации добавлена зона ограничения соединений с именем
↪   per_ip и размером 5 мегабайт.";
                cout << "\n\t\tДанная зона ставит ограничения на количество соединений для
↪   каждого IP-адреса по-отдельности\n";
        }
        if (limc_st == "custom")
        {
                cnt = 1;
                cout << "\n\tУкажите зону ограничения соединений " << cnt << " [no чтобы
↪   закончить ввод]: ";
                cout << "\n\t\tИмя: ";
                getline(cin, lim_conn_name);
                cout << "\n\t\tПеременная ограничения: ";
                getline(cin, lim_conn_param);
                cout << "\n\t\tРазмер зоны: ";
                getline(cin, lim_conn_size);
                while (gl_header_name != "no")
                {
                        config.http.limit_concurrency.push_back(
                        "limit_conn_zone" + lim_conn_param + " zone=" + lim_conn_name + ":" +
↪   lim_conn_size + ";");
                        cnt++;
                        cout << "\n\tУкажите зону ограничения соединений " << cnt << " [no чтобы
↪   закончить ввод]: ";
                        cout << "\n\t\tИмя: ";
                        getline(cin, lim_conn_name);
                        cout << "\n\t\tПеременная ограничения: ";
                        getline(cin, lim_conn_param);
                        cout << "\n\t\tРазмер зоны: ";
                        getline(cin, lim_conn_size);
                }
        }
 }

 cout << "\n\tИспользовать модуль FastCGI? [yes/no]: ";
 getline(cin, fcgi_st);
 if (fcgi_st == "yes")
 {
        config.http.include.push_back("include fastcgi_params;");
        config.http.include.push_back("include fastcgi.conf;");

        cout << "\n\t\tУкажите тип настроек FastCGI [default/custom]: ";
        getline(cin, fcgi_st);

        if (fcgi_st == "default")
        {
                cout << "\n\t\t\tКакой путь использовать для кэша FastCGI? [default/custom]: ";
                getline(cin, fcgi_path);
                if (fcgi_path == "default" || fcgi_path == "")
                {
                        config.http.fast_cgi.push_back(
                        "fastcgi_cache_path " +
                        config.location + "/cache levels=1:2 keys_zone=microcache:10m max_size=500m
↪   inactive=10m;");
                        config.http.fast_cgi.push_back(
                        R"(fastcgi_cache_key "$scheme$request_method$host$request_uri";)");
```

```
                        config.http.fast_cgi.push_back(
                        "fastcgi_ignore_headers Cache-Control Expires Set-Cookie ;");
                }
                else
                {
                        cout << "\n\t\t\t\tУкажите кастомный путь: ";
                        getline(cin, fcgi_path);
                        config.http.fast_cgi.push_back(
                        "fastcgi_cache_path " +
                        fcgi_path + "/cache levels=1:2 keys_zone=microcache:10m max_size=500m
↪   inactive=10m;");
                        config.http.fast_cgi.push_back(
                        R"(fastcgi_cache_key "$scheme$request_method$host$request_uri";)");
                        config.http.fast_cgi.push_back(
                        "fastcgi_ignore_headers Cache-Control Expires Set-Cookie ;");
                }
        }

        if (fcgi_st == "custom")
        {
                cnt = 1;
                cout << "\n\tУкажите настройку FastCGI " << cnt << " [no чтобы закончить ввод]: ";
                getline(cin, fcgi_custom);
                while (fcgi_custom != "no")
                {
                        config.http.fast_cgi.push_back(
                        fcgi_custom + (fcgi_custom.back() == ';' ? "" : ";"));
                        cnt++;
                        cout << "\n\tУкажите настройку FastCGI " << cnt << " [no чтобы закончить
↪   ввод]: ";
                        getline(cin, fcgi_custom);
                }
        }
 }

 cout << "\n\tСоздать группу серверов для балансировщика нагрузки? [yes/no]: ";
 getline(cin, ups_st);
 if (ups_st == "yes")
 {
        cout << "\n\t\tУкажите имя группы серверов: ";
        getline(cin, ups_name);
        cout << "\n\t\tУкажите алгоритм балансировки: ";
        getline(cin, ups_type);
        cout << "\n\t\tУкажите сервера группы (ip:port) [no чтобы закончить ввод]: ";

        cnt = 1;
        cout << "\n\t\t\tServer " << cnt << ": ";
        getline(cin, ups_server);
        while (ups_server != "no" && ups_server != "")
        {
                cnt++;
                servers.push_back(ups_server);
                cout << "\n\t\t\tServer " << cnt << ": ";
                getline(cin, ups_server);
        }

        config.http.load_balancer.push_back(
        "upstream " + ups_name + " {");
                config.http.load_balancer.push_back(
                "\t" + ups_type + ";");
```

```
                for (auto server : servers)
                config.http.load_balancer.push_back(
                "\tserver " + server + ";");
                config.http.load_balancer.push_back(
                "}");
        }

        cout << "\n\tИспользовать модуль Geoip2? [yes/no]: ";
        getline(cin, geo_st);
        if (geo_st == "yes")
        {

                cout << "\n\tКакой путь использовать для доступа к GeoLite2-City.mmdb? [default/custom]:
↪   ";
                getline(cin, geo_path);
                if (geo_path != "")
                {
                        if (geo_path == "default")
                        config.http.geoip.push_back(
                        "geoip2 /usr/share/GeoIP/GeoLite2-City.mmdb {");
                                if (geo_path != "default")
                                {
                                        cout << "\n\tВведите путь до GeoLite2-City.mmdb: ";
                                        getline(cin, geo_path);
                                        config.http.geoip.push_back(
                                        "geoip2 " + geo_path + (geo_path.back() == ';' ? "" : ";") +
↪   "GeoLite2-City.mmdb {");
                                        }
                                        config.http.geoip.push_back("\tauto_reload 60m;");
                                        config.http.geoip.push_back("\t$geoip2_metadata_city_build metadata
↪   build_epoch;");
                                        config.http.geoip.push_back("\t$geoip2_data_country_name country names
↪   en;");
                                        config.http.geoip.push_back("\t$geoip2_data_country_code country
↪   iso_code;");
                                        config.http.geoip.push_back("\t$geoip2_data_city_name city names en;");
                                        config.http.geoip.push_back("\t$geoip2_data_region_name subdivisions 0
↪   names en;");
                                        config.http.geoip.push_back("\t$geoip2_data_state_code  subdivisions 0
↪   iso_code;");
                                        config.http.geoip.push_back("}");
                        }

                        cout << "\n\tТекущие разрешенные регионы: ";
                        cout << "\n\t\t[1]  RU --> Россия";
                        cout << "\n\t\t[2]  BY --> Белоруссия";
                        cout << "\n\t\t[3]  AM --> Армения";
                        cout << "\n\t\t[4]  KZ --> Казахстан";
                        cout << "\n\t\t[5]  KG --> Кыргызстан";
                        cout << "\n\t\t[6]  UA:4 --> Севастополь";
                        cout << "\n\t\t[7]  UA:4 --> Крым";
                        cout << "\n\t\t[8]  UA:1 --> Донецкая область";
                        cout << "\n\t\t[9]  UA:0 --> Луганская область";
                        cout << "\n\t\t[10] UA:2 --> Запорожская область";
                        cout << "\n\t\t[11] UA:6 --> Херсонская область";
                        cout << "\n\t\t[12] GE:A --> Абхазия";

                        cout << "\n\tУкажите id регионов, которые будут иметь доступ при внештатных
↪   ситуциях [no чтобы закончить ввод]: ";
                        getline(cin, geobl_regs);
```

```
                string delimiter = " ";
                size_t pos = 0;
                string token;

                while ((pos = geobl_regs.find(delimiter)) != string::npos)
                {
                        token = geobl_regs.substr(0, pos);
                        int index = stoi(token);

                        if (index >= 0 && index < regions.size())
                        selected_regions.push_back(regions[index]);

                        geobl_regs.erase(0, pos + delimiter.length());
                }

                if (!geobl_regs.empty())
                {
                        int index = stoi(geobl_regs);
                        if (index >= 0 && index < regions.size())
                        selected_regions.push_back(regions[index]);
                }

                config.http.geoip_blocks_allow.push_back(
                R"(map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {)");
                        config.http.geoip_blocks_allow.push_back(
                        "\tdefault no;");
                        for (auto reg : regions)
                        config.http.geoip_blocks_allow.push_back(
                        "\t" + reg);
                        config.http.geoip_blocks_allow.push_back(
                        "}");

                config.http.geoip_blocks_deny.push_back(
                R"(map "$geoip2_data_country_code:$geoip2_data_state_code" $allowed_reg {)");
                        config.http.geoip_blocks_deny.push_back(
                        "\tdefault no;");
                        for (auto reg : selected_regions)
                        config.http.geoip_blocks_deny.push_back(
                        "\t" + reg);
                        config.http.geoip_blocks_deny.push_back(
                        "}");
        }

        cnt = 1;
        cout << "\n\tУкажите дополнительные настройки для блока http [no чтобы закончить
↪   ввод]: ";
        cout << "\n\t\tДоп. настройка " << cnt << ": ";
        getline(cin, custom);
        while (custom != "no")
        {
                config.http.custom.push_back(custom + (custom.back() == ';' ? "" : ";"));
                cnt++;
                cout << "\n\t\tДоп. настройка " << cnt << ": ";
                getline(cin, custom);
        }

        cout << "\n\t\t\t\t[Конец конфигурации блока http]\n";
    }
```

```
void setServerBlock()
{
	config.server.start = "\n\t\tserver {";
		config.server.tab = "\n\t\t\t";
		config.server.end = "\n\t\t}";

	int cnt(1);
	vs listening_ports, allow_hosts, deny_hosts, gzip_files, gzip_disable;
	ss custom, server_name, root_dir,
	alog_st, elog_st, alog_path, elog_path, alog_type, elog_type,
	ssl_st, ssl_crt, ssl_key, ssl_listen,
	buf_st, tmout_st, kpalive_st, ver_st,
	gzip_st, cache_st;

	cout << "\n\t\n\t\t\t[Конфигурация блока server]\n";
	cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";
	cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

	cout << "\n\tУкажите прослушиваемые порты [через пробел]: ";
	listening_ports = readInputValues();
	for (auto port : listening_ports)
	config.server.listen.push_back("listen " + port + ";");

	cout << "\n\tУкажите разрешенные хосты [через пробел]: ";
	allow_hosts = readInputValues();
	for (auto host : allow_hosts)
	config.server.allow.push_back("allow " + host + ";");

	cout << "\n\tУкажите запрещенные хосты [через пробел]: ";
	deny_hosts = readInputValues();
	for (auto host : deny_hosts)
	config.server.deny.push_back("deny " + host + ";");

	cout << "\n\tУкажите имя сервера [default]: ";
	getline(cin, server_name);
	if (server_name == "default")
	cout << "\n\tИмя сервера установлено как: localhost";
	config.server.server_name = config.server.tab + "server_name ";
	config.server.server_name += server_name == "" ? "localhost" : server_name;

	cout << "\n\tУкажите корневой каталог сайта [default]: ";
	if (root_dir == "default")
	cout << "\n\tКорневой каталог сайта установлен как: /etc/nginx/site/";
	getline(cin, root_dir);
	config.server.root_folder = config.server.tab + "root ";
	config.server.root_folder += root_dir == "" ? "/etc/nginx/site/" : root_dir;

	cout << "\n\tВключить логирование ошибок для всех директорий? [yes/no]: ";
	getline(cin, elog_st);
	if (elog_st != "")
	{
		if (elog_st == "yes")
		{
			config.server.global_logs_status.push_back("error_log on;");
			cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";
			getline(cin, elog_path);
			elog_path = elog_path == "default" ? "/etc/nginx/logs/" : elog_path;

			cout << "\n\t\tУкажите формат логов ошибок [default]: ";
			getline(cin, elog_type);
```

```
                elog_type = elog_type == "default" ? "main" : elog_type;
                config.server.global_logs_settings.push_back(
                "error_log " + elog_path + (elog_path.back() == '/' ? "" : "/") +
                "error_base.log " + elog_type + ";");
        }
        if (elog_st == "no")
        config.server.global_logs_status.push_back("error_log off;");
}

cout << "\n\tВключить логирование доступа для всех директорий? [yes/no]: ";
getline(cin, alog_st);
if (alog_st != "")
{
        if (alog_st == "yes")
        {
                config.server.global_logs_status.push_back("access_log on;");

                cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";
                getline(cin, alog_path);
                alog_path = alog_path == "default" ? "/etc/nginx/logs/" : alog_path;

                cout << "\n\t\tУкажите формат логов ошибок [default]: ";
                getline(cin, alog_type);
                alog_type = alog_type == "default" ? "main" : alog_type;

                config.server.global_logs_settings.push_back(
                "access_log " + alog_path + (alog_path.back() == '/' ? "" : "/") +
                "access_base.log " + alog_type + ";");
        }
        if (alog_st == "no")
        config.server.global_logs_status.push_back("access_log off;");
}

cout << "\n\tВключить шифрование SSL/TLS? [yes/no]: ";
getline(cin, ssl_st);
if (ssl_st == "yes")
{
        cout << "\n\t\tУкажите прослушивающий порт: ";
        getline(cin, ssl_listen);
        cout << "\n\t\tУкажите файл ssl-сертификата (.crt): ";
        getline(cin, ssl_crt);
        cout << "\n\t\tУкажите файл ssl-ключа (.key): ";
        getline(cin, ssl_key);

        config.server.listen.push_back("listen " + ssl_listen + " ssl;");
        config.server.ssl.push_back("ssl_certificate " + ssl_crt + ";");
        config.server.ssl.push_back("ssl_certificate_key " + ssl_key + ";");
        config.server.ssl.push_back("ssl_session_cache shared:SSL:1m;");
        config.server.ssl.push_back("ssl_session_timeout 5m;");
        config.server.ssl.push_back("ssl_ciphers HIGH:!aNULL:!MD5;");
        config.server.ssl.push_back("ssl_prefer_server_ciphers on;");
}

cout << "\n\tВключить ограничения буферизации сообщений? [yes/no]: ";
getline(cin, buf_st);
if (ssl_st == "yes")
{
        config.server.buffers.push_back("client_body_buffer_size 16k;");
        config.server.buffers.push_back("client_header_buffer_size 1k;");
        config.server.buffers.push_back("client_max_body_size 8m;");
```

```
        config.server.buffers.push_back("large_client_header_buffers 2 1k;");
}

cout << "\n\tВключить таймаут ожидания для передачи запроса клиента? [yes/no]: ";
getline(cin, tmout_st);
if (ssl_st == "yes")
{
        cout << "\n\t\tУкажите таймаут для хэдера запроса: ";
        getline(cin, tmout_st);
        config.server.timeouts.push_back("client_header_timeout " + tmout_st + ";");
        cout << "\n\t\tУкажите таймаут для тела запроса: ";
        getline(cin, tmout_st);
        config.server.timeouts.push_back("client_body_timeout " + tmout_st + ";");
}

cout << "\n\tВключить таймаут ожидания для активного соединения? [yes/no]: ";
getline(cin, kpalive_st);
if (kpalive_st == "yes")
config.server.keepalive.push_back("keepalive_timeout 65;");

cout << "\n\tВключить таймаут ожидания для отправки данных клиенту? [yes/no]: ";
getline(cin, kpalive_st);
if (kpalive_st == "yes")
config.server.keepalive.push_back("send_timeout 10;");

cout << "\n\tОтключить отображение информации о версии Nginx в хэдерах? [yes/no]: ";
getline(cin, ver_st);
if (ver_st == "yes")
config.server.server_token = config.server.tab + "server_tokens off;";

cout << "\n\tВключить сжатие ответов сервера с помощью gzip? [yes/no]: ";
getline(cin, gzip_st);
if (gzip_st == "yes")
{
        config.server.gzip.push_back("gzip on;");

        cout << "\n\t\tУкажите минимальный размер сжимаемых файлов: ";
        getline(cin, gzip_st);
        config.server.gzip.push_back("gzip_min_length " + gzip_st + ";");

        cout << "\n\t\tУкажите степень сжатия файлов: ";
        getline(cin, gzip_st);
        config.server.gzip.push_back("gzip_comp_level " + gzip_st + ";");

        cout << "\n\t\tУкажите типы сжимаемых файлов через пробел(Пр.: css plain): ";
        gzip_files = readInputValues();
        for (auto file : gzip_files)
        config.server.gzip.push_back("gzip_types text/" + file + ";");

        config.server.gzip.push_back(R"(gzip_disable "msie6";)");
}

cout << "\n\tОграничитить кэширование FastCGI по определенным правилам? [yes/no]: ";
getline(cin, cache_st);
if (cache_st == "yes")
{
        config.server.caching.push_back("set $no_cache 0;");
        cout << "\n\tУкажите правила ограничений [no чтобы закончить ввод]: ";
        cnt = 1;
        cout << "\n\t\tПравило " << cnt << ": ";
```

```
                getline(cin, custom);

                while (custom != "no")
                {
                        config.server.caching.push_back("if (" + custom + ") { set $no_cache 1; }");
                        cnt++;
                        cout << "\n\t\tПравило " << cnt << ": ";
                        getline(cin, custom);
                }
        }

        if (config.http.geoip.size() != 0)
        {
                config.server.custom.push_back("if ($allowed_reg = no) {");
                        config.server.custom.push_back("\treturn 444;");
                        config.server.custom.push_back("}");
        }

        cnt = 1;
        cout << "\n\tУкажите дополнительные настройки для блока server [no чтобы закончить
↪   ввод]: ";
        cout << "\n\t\tДоп. настройка " << cnt << ": ";
        getline(cin, custom);
        while (custom != "no")
        {
                config.server.custom.push_back(custom + (custom.back() == ';' ? "" : ";"));
                cnt++;
                cout << "\n\t\tДоп. настройка " << cnt << ": ";
                getline(cin, custom);
        }

        cout << "\n\t\t\t\t[Конец конфигурации блока server]\n";
}

void setLocations()
{

        int cnt(1);

        ss loc_st, exp_st, prx_st, prx_name, bal_st,
        auth_st, auth_path, auth_log, cgi_st,
        celog_st, celog_path, celog_type, celog_name,
        prxhc_st, prxhs_st, custom, hd_name, hd_status;

        location_block default_loc;

        cout << "\n\t\n\t\t\t[Конфигурация блоков location]\n";
        cout << "\n\t! Если хотите пропустить настройку, оставьте поле пустым и нажмите Enter";
        cout << "\n\t! Если хотите установить настройку по умолчанию, Укажите default\n";

        default_loc.location_name = R"( ~ /\.ht )";
        default_loc.start = "\n\t\t\tlocation " + default_loc.location_name + "{";
                default_loc.tab = "\n\t\t\t\t";
                default_loc.end = "\n\t\t\t}\n";
        default_loc.custom.push_back("deny  all;");

        config.locations.push_back(default_loc);

        cout << "\n\tСоздать новую локацию? [yes/no]: ";
        getline(cin, loc_st);
```

```
while (loc_st == "yes")
{
	location_block nloc;

	cout << "\n\t\tУкажите имя локации: ";
	getline(cin, nloc.location_name);

	nloc.start = "\n\t\t\tlocation " + nloc.location_name + " {";
		nloc.tab = "\n\t\t\t\t";
		nloc.end = "\n\t\t\t}\n";

	cout << "\n\tВключить логирование ошибок для данной директории? [yes/no]: ";
	getline(cin, celog_st);
	if (celog_st != "")
	{
		if (celog_st == "yes")
		{

			cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";
			getline(cin, celog_path);
			celog_path = celog_path == "default" ? "/etc/nginx/logs/" : celog_path;

			cout << "\n\t\tУкажите имя логов: ";
			getline(cin, celog_name);

			cout << "\n\t\tУкажите формат логов ошибок [default]: ";
			getline(cin, celog_type);
			celog_type = celog_type == "default" ? "main" : celog_type;

			nloc.custom_logs_settings.push_back(
			"error_log " + celog_path + (celog_st.back() == '/' ? "" : "/") +
			celog_name + " " + celog_st + ";");
		}
		if (celog_st == "no")
		nloc.custom_logs_status.push_back("error_log off;");
	}

	cout << "\n\tВключить логирование доступа для данной директории? [yes/no]: ";
	getline(cin, celog_st);
	if (celog_st != "")
	{
		if (celog_st == "yes")
		{

			cout << "\n\t\tУкажите каталог для хранения логов ошибок [default]: ";
			getline(cin, celog_path);
			celog_path = celog_path == "default" ? "/etc/nginx/logs/" : celog_path;

			cout << "\n\t\tУкажите имя логов: ";
			getline(cin, celog_name);

			cout << "\n\t\tУкажите формат логов ошибок [default]: ";
			getline(cin, celog_type);
			celog_type = celog_type == "default" ? "main" : celog_type;

			nloc.custom_logs_settings.push_back(
			"error_log " + celog_path + (celog_st.back() == '/' ? "" : "/") +
			celog_name + " " + celog_st + ";");
		}
		if (celog_st == "no")
```

```
                nloc.custom_logs_status.push_back("access_log off;");
        }

        cout << "\n\tВключить базовую аутентификацию? [yes/no]: ";
        getline(cin, auth_st);
        if (auth_st == "yes")
        {
                cout << "\n\t\t\tУкажите файл с учетными данными: ";
                getline(cin, auth_path);
                nloc.auth.push_back(R"(auth_basic "Restricted access";)");
                nloc.auth.push_back("auth_basic_user_file " + auth_path + ";");
        }

        cout << "\n\tВключить кэширование на стороне клиента? [yes/no]: ";
        getline(cin, exp_st);
        if (auth_st == "yes")
        {
                cout << "\n\t\tУкажите время валидности кэша: ";
                getline(cin, exp_st);
                nloc.expires = "expires " + auth_st + ";";
        }

        if (config.http.fast_cgi.size() > 0)
        {
                cout << "\n\tВключить кэширование FastCGI? [yes/no]: ";
                getline(cin, cgi_st);
                if (auth_st == "yes")
                {
                        cout << "\n\t\tУкажите зону ключей кэша FastCGI: ";
                        getline(cin, cgi_st);
                        nloc.fast_cgi.push_back("fastcgi_cache " + cgi_st + ";");
                        cout << "\n\t\tУкажите время валидности кэша FastCGI: ";
                        getline(cin, cgi_st);
                        nloc.fast_cgi.push_back("fastcgi_cache_valid 200  " + cgi_st + ";");
                        cout << "\n\t\tУкажите прокси сервер FastCGI: ";
                        getline(cin, cgi_st);
                        if (cgi_st != "" && cgi_st != "no")
                        nloc.fast_cgi.push_back("fastcgi_pass " + cgi_st + ";");
                        cout << "\n\t\tИспользовать дополнительные правила кэширования
↪   FastCGI? [yes/no]: ";
                        getline(cin, cgi_st);
                        if (cgi_st == "yes")
                        nloc.fast_cgi.push_back("fastcgi_no_cache $no_cache;");
                }
        }

        cout << "\n\tДобавить кастомные хэдеры? [yes/no]: ";
        getline(cin, prxhc_st);
        if (prxhc_st == "yes")
        {
                cnt = 1;
                cout << "\n\t\tУкажите хэдер " << cnt << ": ";
                cout << "\n\t\t\tИмя: ";
                getline(cin, hd_name);
                while (hd_name != "no")
                {
                        cout << "\n\t\t\tДанные: ";
                        getline(cin, hd_status);
                        nloc.custom_headers.push_back(
                        "add_header " + hd_name + " " + hd_status + ";");
```

```
                        cnt++;
                        cout << "\n\t\tУкажите хэдер " << cnt << ": ";
                        cout << "\n\t\t\tИмя: ";
                        getline(cin, hd_name);
                    }
                }

                if (config.http.load_balancer.size() > 0)
                {
                    cout << "\n\tИспользовать балансировщик? [yes/no]: ";
                    getline(cin, bal_st);
                    if (bal_st == "yes")
                    {
                        cout << "\n\t\tУкажите имя группы серверов: ";
                        getline(cin, bal_st);
                        nloc.proxy_pass = "http://" + bal_st + ";";
                    }
                }

                cnt = 1;
                cout << "\n\tУкажите дополнительные настройки для блока server [no чтобы
↪   закончить ввод]: ";
                cout << "\n\t\tДоп. настройка " << cnt << ": ";
                getline(cin, custom);
                while (custom != "no")
                {
                    nloc.custom.push_back(custom + (custom.back() == ';' ? "" : ";"));
                    cnt++;
                    cout << "\n\t\tДоп. настройка " << cnt << ": ";
                    getline(cin, custom);
                }

                config.locations.push_back(nloc);

                cout << "\n\tСоздать новую локацию? [yes/no]: ";
                getline(cin, loc_st);
            }
            cout << "\n\t\t\t\t[Конец конфигурации блоков locations]\n";
        }

        void pushAllBlocks()
        {
            ofstream config_file(config.location + (config.location.back() == '/' ? "" : "/") + "nginx.conf");

            config_file << config.events.start;

            for (auto params : config.events.workers)
            if (params != "")
            config_file << config.events.tab + params;

            for (auto params : config.events.multi)
            if (params != "")
            config_file << config.events.tab + params;

            for (auto params : config.events.mutex)
            if (params != "")
            config_file << config.events.tab + params;

            for (auto params : config.events.custom)
            if (params != "")
```

```
config_file << config.events.tab + params;

config_file << config.events.end;

config_file << config.http.start;

for (auto params : config.http.include)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.global_headers)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.log_formatting)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.limit_concurrency)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.fast_cgi)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.load_balancer)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.geoip)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.geoip_blocks_allow)
if (params != "")
config_file << config.http.tab + params;

for (auto params : config.http.geoip_blocks_deny)
if (params != "")
config_file << config.http.tab + "# " + params;

config_file << config.server.start;

for (auto params : config.server.listen)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.allow)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.deny)
if (params != "")
config_file << config.server.tab + params;

config_file << config.server.server_name;

config_file << config.server.root_folder;
```

```
config_file << config.server.server_token;

for (auto params : config.server.global_logs_status)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.global_logs_settings)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.ssl)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.buffers)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.timeouts)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.keepalive)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.gzip)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.caching)
if (params != "")
config_file << config.server.tab + params;

for (auto params : config.server.custom)
if (params != "")
config_file << config.server.tab + params;

for (auto location : config.locations)
{
        config_file << location.start;
        for (auto params : location.custom_logs_status)
        if (params != "")
        config_file << location.tab + params;

        for (auto params : location.custom_logs_settings)
        if (params != "")
        config_file << location.tab + params;

        for (auto params : location.auth)
        if (params != "")
        config_file << location.tab + params;

        for (auto params : location.fast_cgi)
        if (params != "")
        config_file << location.tab + params;

        if (location.expires != "")
        config_file << location.tab + location.expires;
```

```
                for (auto params : location.custom_headers)
                if (params != "")
                config_file << location.tab + params;

                if (location.proxy_pass != "")
                config_file << location.tab + location.proxy_pass;

                for (auto params : location.custom)
                if (params != "")
                config_file << location.tab + params;

                config_file << location.end;
        }

        config_file << config.server.end;

        for (auto params : config.http.custom)
        if (params != "")
        config_file << config.http.tab + params;

        config_file << config.http.end;

        config_file.close();
    }

    void applyConfiguration()
    {
        string userInput;
        cout << "\n\tКакую конфигурацию необходимо применить? (default/preset/custom): ";
        cout << "\n\t\t[default] Применить стандартную конфигурацию для Ubuntu LTS 22.04";
        cout << "\n\t\t[preset]  Применить рекомендованную конфигурацию, разработанную в
↪   рамках курса";
        cout << "\n\t\t[custom]  Составить собственную конфигурацию";
        cout << "\n\n\tВаш выбор: ";
        getline(cin, userInput);

        if (userInput == "default")
        {
            ifstream srcFile("nginx.conf.ubuntu_template", ios::binary);
            ofstream dstFile(config.location + (config.location.back() == '/' ? "" : "/") +
↪   "nginx.conf", ios::binary);

            if (srcFile && dstFile)
            {
                dstFile << srcFile.rdbuf();
                cout << "\n\tФайл конфигурации успешно создан.";
            }
            else
            cout << "\n\tНе удалось создать файл конфигурации.";
        }
        else if (userInput == "preset")
        {
            ifstream srcFile("nginx.conf.mine", ios::binary);
            ofstream dstFile("nginx.conf", ios::binary);

            if (srcFile && dstFile)
            {
                dstFile << srcFile.rdbuf();
                cout << "\n\tФайл конфигурации успешно создан.";
            }
```

```
                else
                cout << "\n\tНе удалось создать файл конфигурации.";
        }
        else if (userInput == "custom")
        {
                ofstream customFile("nginx.conf");

                if (customFile)
                {
                        cout << "\n\tДавайте приступим к персональной настройке файла
↪   конфигурации!";
                        setEventsBlock();
                        setHttpBlock();
                        setServerBlock();
                        setLocations();
                        pushAllBlocks();
                }
                else
                cout << "\n\tНе удалось создать файл конфигурации.";
        }
        else
        cout << "\n\tНекорректный ввод.";
}

int main()
{
        cout << "\n\tПриветствуем в конфигураторе nginx!\n\n\tУкажите желаемое расположение
↪   nginx.conf:";
        cin >> config.location;
        cout << "\n\t";

        config.location = "/home/alse0722/Desktop/kurs4/pract";
        applyConfiguration();

        return 0;
}
```