SSL/TLS ? Let's Encrypt

??? 1: ????????? Certbot

Certbot — это инструмент для автоматического получения и обновления сертификатов Let's Encrypt. Установите его и плагин для Nginx.

```
sudo apt install -y certbot python3-certbot-nginx
```

??? 2: ????????? ???????????

Запустите Certbot для вашего домена. Он автоматически изменит вашу конфигурацию Nginx, чтобы использовать HTTPS.

```
sudo certbot --nginx -d example.com -d www.example.com
```

??? 3: ???????? ?????????????

Certbot автоматически настраивает системный таймер (в systemd или cron), который будет продлевать сертификаты за 30 дней до их истечения. Вы можете протестировать этот процесс.

```
sudo certbot renew --dry-run
```

??????? ???????????? HTTPS

1. Стандартная конфигурация от Certbot

Это базовый и надежный вариант, который Certbot создает автоматически. Он указывает пути к сертификатам и подключает рекомендованные Certbot'ом параметры безопасности.

```
server {
    listen 443 ssl http2;
    server_name example.com www.example.com;
    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
        # ... остальная конфигурация (root, location)
```

```
}
```

2. Редирект с HTTP на HTTPS

Это обязательный блок для любого сайта на HTTPS. Он "слушает" небезопасный 80-й порт и, получив запрос, немедленно отправляет клиенту ответ с кодом 301 (Permanent Redirect), указывая новый адрес на `https://`. Переменные `$host` и `$request_uri` сохраняют исходный домен и путь, чтобы пользователь попал на нужную страницу, но уже по защищенному протоколу.

```
server {
    listen 80;
    server_name example.com www.example.com;
    return 301 https://$host$request_uri;
}
```

3. Повышенная безопасность (Strong Ciphers + TLS 1.3)

Здесь мы явно указываем, какие протоколы и шифры разрешено использовать. `TLSv1.2` и `TLSv1.3` — современные безопасные протоколы. Длинный список `ssl_ciphers` перечисляет надежные наборы шифров и задает их приоритет. Это защищает от использования устаревших и уязвимых алгоритмов шифрования.

```
server {
    # ...
    listen 443 ssl http2;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
    # ...
}
```

4. С HSTS (HTTP Strict Transport Security)

Это мощный механизм защиты. Когда пользователь впервые заходит на сайт, сервер отправляет заголовок `Strict-Transport-Security`. Браузер пользователя "запоминает" эту инструкцию и в течение указанного времени (`max-age` в секундах, здесь — 2 года) будет автоматически и принудительно выполнять все запросы к этому домену (и поддоменам, если указано `includeSubDomains`) только по HTTPS, даже если пользователь введет в адресной строке `http://`.

```
server {
    # ...
    listen 443 ssl http2;
    # ...
    add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"
always;
    # ...
}
```

5. Оптимизация производительности SSL

Установка SSL-соединения — ресурсоемкий процесс. Чтобы ускорить повторные подключения от одного и того же клиента, Nginx может кэшировать параметры сессии. `ssl_session_cache` создает в памяти кэш размером 10 МБ, а `ssl_session_timeout` указывает, как долго хранить сессию. Это значительно снижает нагрузку на сервер и ускоряет загрузку сайта для вернувшихся посетителей.

```
server {
    # ...
    listen 443 ssl http2;
    # ...
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 1d;
    ssl_session_tickets off; # Рекомендуется для Perfect Forward Secrecy
    # ...
}
```

6. Полная конфигурация с лучшими практиками

Этот пример объединяет все лучшее: указывает пути к сертификатам, использует современные протоколы и шифры, добавляет заголовок HSTS для защиты и включает кэширование сессий для производительности. Это хороший, сбалансированный шаблон для большинства современных сайтов.

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name example.com;
    # SSL
    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/example.com/chain.pem;
```

```
    # Security
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384';
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    # Performance
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 1d;
    root /var/www/example.com;
    index index.html;
    # ...
}
```

Revision #2
Created 5 October 2025 12:31:49 by Admin
Updated 5 October 2025 12:33:39 by Admin