Чеклист безопасности Telegram Store MVP

Версия документа: 1.0
Дата: 2025-11-30
Статус: Production Ready Checklist

Содержание

Аутентификация и авторизация

✅ JWT Токены

- [ ] **JWT Secret Key** - минимум 32 символа, криптографически случайный
- [ ] **Access Token TTL** - не более 30 минут
- [ ] **Refresh Token TTL** - не более 7 дней
- [ ] **Алгоритм подписи** - HS256 или RS256 (предпочтительно RS256 для production)
- [ ] **Token Rotation** - refresh token обновляется при каждом использовании
- [ ] **Token Revocation** - механизм отзыва токенов (blacklist в Redis)
- [ ] **Payload минимален** - только user_id, role, exp, iat

Пример проверки:

```
# Генерация безопасного ключа
python3 -c "import secrets; print(secrets.token_urlsafe(32))"

# Проверка длины ключа в .env
grep JWT_SECRET_KEY .env | wc -c  # Должно быть > 40
```

✅ Пароли администраторов

- [ ] **Хеширование** - bcrypt с cost factor >= 12

- [ ] **Минимальная длина** - 12 символов
- [ ] **Требования к сложности** - буквы, цифры, спецсимволы
- [ ] **Защита от brute-force** - rate limiting на /auth/token
- [ ] **Нет дефолтных паролей** - все пароли меняются при первом входе
- [ ] **Password reset** - безопасный механизм восстановления (email/Telegram)

Пример реализации:

```
from passlib.context import CryptContext

pwd_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

# Хеширование
hashed = pwd_context.hash("SecurePassword123!")

# Проверка
pwd_context.verify("SecurePassword123!", hashed)
```

✅ Авторизация

- [ ] **Role-Based Access Control (RBAC)** - роли: superadmin, manager, viewer
- [ ] **Принцип наименьших привилегий** - каждая роль имеет минимальные права
- [ ] **Защита endpoints** - все /admin/* требуют аутентификации
- [ ] **Проверка прав на уровне ресурсов** - пользователь может редактировать только свои данные
- [ ] **Audit trail** - логирование всех административных действий

Управление секретами

✅ Переменные окружения

- [ ] `.env` **файл** - добавлен в `.gitignore`
- [ ] **Нет секретов в коде** - все чувствительные данные в переменных окружения
- [ ] **Разные секреты для окружений** - dev, staging, production имеют разные ключи
- [ ] **Документация** - `.env.example` с placeholder значениями

Проверка:

```
# Убедиться что .env не в git
git ls-files | grep .env

# Поиск потенциальных секретов в коде
grep -r "password\s*=\s*['\"]" src/
grep -r "api_key\s*=\s*['\"]" src/
```

✅ Secrets Management (Production)

- [ ] **Использование Secrets Manager** - AWS Secrets Manager, Google Secret Manager, HashiCorp Vault
- [ ] **Ротация секретов** - автоматическая ротация каждые 90 дней
- [ ] **Шифрование at rest** - все секреты зашифрованы

- [ ] **Audit logging** - логирование доступа к секретам
- [ ] **Минимальные права доступа** - только необходимые сервисы имеют доступ

✅ Google Service Account

- [ ] **JSON ключ** - хранится вне репозитория
- [ ] **Права доступа** - только необходимые (Sheets API)
- [ ] **Ротация ключей** - каждые 90 дней
- [ ] **Файл защищен** - chmod 600, владелец - app user

Проверка:

```
ls -la /path/to/google-service-account.json
# Должно быть: -rw------- 1 appuser appuser
```

Безопасность API

✅ HTTPS/TLS

- [ ] **Только HTTPS** - HTTP редиректит на HTTPS
- [ ] **TLS 1.2+** - отключены устаревшие протоколы (SSLv3, TLS 1.0, 1.1)
- [ ] **Сильные шифры** - HIGH:!aNULL:!MD5
- [ ] **HSTS заголовок** - Strict-Transport-Security с max-age >= 31536000
- [ ] **Валидный сертификат** - Let’s Encrypt или коммерческий CA
- [ ] **Автообновление сертификатов** - certbot renew в cron

Проверка:

```
# Тест SSL конфигурации
curl -I https://api.telegram-shop.example 2>&1 | grep -i "strict-transport"

# Проверка сертификата
openssl s_client -connect api.telegram-shop.example:443 -servername api.telegram-
shop.example < /dev/null 2>&1 | grep -A 2 "Verify return code"
```

✅ CORS

- [ ] **Whitelist доменов** - только разрешенные origins
- [ ] **Credentials** - allow_credentials=True только для доверенных доменов
- [ ] **Методы** - только необходимые (GET, POST, PUT, DELETE, PATCH)
- [ ] **Headers** - только необходимые

Пример конфигурации:

```
from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
    allow_origins=[
        "https://admin.telegram-shop.example",
        "https://telegram-shop.example"
    ],
    allow_credentials=True,
    allow_methods=["GET", "POST", "PUT", "DELETE", "PATCH"],
    allow_headers=["*"],
)
```

✅ Rate Limiting

- [ ] **Глобальный rate limit** - 100 req/min на IP
- [ ] **Auth endpoints** - 5 req/min на /auth/token
- [ ] **Webhook endpoint** - проверка IP Telegram
- [ ] **Использование Redis** - для distributed rate limiting
- [ ] **429 Too Many Requests** - корректный HTTP статус

Пример с slowapi:

```
from slowapi import Limiter, _rate_limit_exceeded_handler
from slowapi.util import get_remote_address

limiter = Limiter(key_func=get_remote_address)
app.state.limiter = limiter
app.add_exception_handler(RateLimitExceeded, _rate_limit_exceeded_handler)

@app.post("/auth/token")
@limiter.limit("5/minute")
async def login(request: Request):
    ...
```

✅ Security Headers

- [ ] **X-Content-Type-Options: nosniff**
- [ ] **X-Frame-Options: DENY** или SAMEORIGIN
- [ ] **X-XSS-Protection: 1; mode=block**
- [ ] **Content-Security-Policy** - для Admin Panel
- [ ] **Referrer-Policy: no-referrer** или strict-origin-when-cross-origin

Nginx конфигурация:

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
```

Безопасность базы данных

✅ PostgreSQL конфигурация

- [ ] **Отдельный пользователь** - не использовать postgres superuser
- [ ] **Сильный пароль** - минимум 16 символов
- [ ] **Ограничение доступа** - pg_hba.conf разрешает только app server IP
- [ ] **SSL соединение** - require ssl в postgresql.conf
- [ ] **Шифрование at rest** - LUKS или managed database encryption
- [ ] **Регулярные обновления** - патчи безопасности PostgreSQL

pg_hba.conf:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
hostssl telegram_shop   telegram_shop_user  10.0.0.5/32         md5
```

✅ SQL Injection защита

- [ ] **ORM использование** - SQLAlchemy для всех запросов
- [ ] **Параметризованные запросы** - никогда не конкатенировать SQL
- [ ] **Валидация входных данных** - Pydantic схемы
- [ ] **Escape специальных символов** - автоматически через ORM
- [ ] **Least privilege** - app user не имеет DROP, CREATE DATABASE прав

Плохо:

```
# НИКОГДА ТАК НЕ ДЕЛАТЬ!
query = f"SELECT * FROM users WHERE id = {user_id}"
```

Хорошо:

```
# Использовать ORM
user = await db.query(User).filter(User.id == user_id).first()

# Или параметризованный запрос
query = "SELECT * FROM users WHERE id = :user_id"
result = await db.execute(query, {"user_id": user_id})
```

✅ Чувствительные данные

- [ ] **Шифрование PII** - персональные данные зашифрованы (phone, address)
- [ ] **Нет паролей в plaintext** - только хеши
- [ ] **Маскирование в логах** - phone, email не попадают в логи
- [ ] **Удаление данных** - механизм безопасного удаления (GDPR right to be forgotten)

Защита от атак

✅ OWASP Top 10

Уязвимость	Защита	Статус
A01: Broken Access Control	RBAC, проверка прав на каждом endpoint	[ ]
A02: Cryptographic Failures	TLS, bcrypt, шифрование PII	[ ]
A03: Injection	ORM, валидация, параметризованные запросы	[ ]
A04: Insecure Design	Threat modeling, security by design	[ ]
A05: Security Misconfiguration	Hardened configs, no defaults	[ ]
A06: Vulnerable Components	Регулярные обновления, Dependabot	[ ]
A07: Authentication Failures	JWT, bcrypt, rate limiting	[ ]
A08: Software and Data Integrity	Code signing, SRI для CDN	[ ]
A09: Logging Failures	Centralized logging, alerts	[ ]
A10: SSRF	Валидация URLs, whitelist доменов	[ ]

✅ XSS (Cross-Site Scripting)

- [ ] **Content-Security-Policy** - для Admin Panel
- [ ] **Escape output** - React автоматически экранирует
- [ ] **Валидация input** - sanitize HTML если принимается
- [ ] **HttpOnly cookies** - для refresh token

✅ CSRF (Cross-Site Request Forgery)

- [ ] **CSRF токены** - для state-changing операций
- [ ] **SameSite cookies** - SameSite=Strict для refresh token
- [ ] **Origin/Referer проверка** - для критичных операций
- [ ] **Double Submit Cookie** - альтернативный метод

✅ DDoS защита

- [ ] **Rate limiting** - на всех endpoints
- [ ] **Cloudflare/CDN** - для защиты от L7 DDoS
- [ ] **Connection limits** - nginx limit_conn
- [ ] **Timeout настройки** - предотвращение slowloris
- [ ] **Мониторинг трафика** - алерты на аномалии

Безопасность Telegram Bot

✅ Webhook безопасность

- [ ] **Secret token** - проверка X-Telegram-Bot-Api-Secret-Token заголовка
- [ ] **IP whitelist** - только IP Telegram серверов (опционально)
- [ ] **HTTPS обязателен** - Telegram требует SSL
- [ ] **Валидация payload** - проверка структуры Update объекта

Telegram IP ranges:

```
149.154.160.0/20
91.108.4.0/22
```

Проверка secret token:

```python
from fastapi import Header, HTTPException

@app.post("/webhook")
async def webhook(
    update: dict,
    x_telegram_bot_api_secret_token: str = Header(None)
):
    if x_telegram_bot_api_secret_token != settings.TELEGRAM_WEBHOOK_SECRET:
        raise HTTPException(status_code=403, detail="Invalid secret token")
    # Process update
```

✅ Защита от злоупотреблений

- [ ] **Rate limiting на пользователя** - max 10 сообщений/минуту
- [ ] **Валидация user input** - проверка длины, формата
- [ ] **Защита от спама** - блокировка после N неудачных попыток
- [ ] **Логирование подозрительной активности** - множественные ошибки, необычные паттерны

✅ Данные пользователей

- [ ] **Минимизация сбора** - только необходимые данные
- [ ] **Шифрование в БД** - phone, address
- [ ] **Удаление по запросу** - GDPR compliance
- [ ] **Нет хранения сообщений** - только необходимый контекст в FSM

HTTPS и TLS

✅ SSL/TLS конфигурация

- [ ] **Валидный сертификат** - от доверенного CA
- [ ] **Цепочка сертификатов** - fullchain.pem включает intermediate
- [ ] **Приватный ключ защищен** - chmod 600, только root доступ
- [ ] **TLS 1.2 и 1.3** - отключены SSLv3, TLS 1.0, 1.1
- [ ] **Perfect Forward Secrecy** - ECDHE шифры
- [ ] **OCSP Stapling** - для производительности

Nginx SSL конфигурация:

```
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384';
ssl_prefer_server_ciphers on;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;
ssl_stapling on;
ssl_stapling_verify on;
```

✅ Тестирование SSL

- [ ] **SSL Labs** - оценка A или A+ на https://www.ssllabs.com/ssltest/
- [ ] **testssl.sh** - локальное тестирование
- [ ] **Проверка цепочки** - openssl s_client

Команды:

```
# Проверка сертификата
openssl x509 -in /etc/letsencrypt/live/domain/fullchain.pem -text -noout

# Проверка приватного ключа
openssl rsa -in /etc/letsencrypt/live/domain/privkey.pem -check

# Тест соединения
openssl s_client -connect api.telegram-shop.example:443 -servername api.telegram-
shop.example
```

Логирование и аудит

✅ Что логировать

- [ ] **Аутентификация** - успешные и неудачные попытки входа
- [ ] **Авторизация** - отказы в доступе (403)
- [ ] **Административные действия** - создание/изменение/удаление продуктов, заказов
- [ ] **Ошибки** - все 4xx и 5xx ответы
- [ ] **Подозрительная активность** - множественные неудачные попытки, необычные паттерны
- [ ] **Изменения данных** - кто, что, когда изменил

✅ Что НЕ логировать

- [ ] **Пароли** - даже в хешированном виде
- [ ] **JWT токены** - полные токены
- [ ] **Персональные данные** - phone, address (или маскировать)
- [ ] **Платежные данные** - номера карт, CVV

✅ Структура логов

- [ ] **Структурированные логи** - JSON формат
- [ ] **Timestamp** - ISO 8601 с timezone
- [ ] **Request ID** - для трейсинга
- [ ] **User ID** - для аудита
- [ ] **IP адрес** - для безопасности
- [ ] **User Agent** - для анализа

Пример:

```
{
  "timestamp": "2025-11-30T10:15:30.123Z",
  "level": "INFO",
  "request_id": "abc-123-def",
  "user_id": 123456789,
  "ip": "192.168.1.1",
  "method": "POST",
  "path": "/api/v1/orders",
  "status": 201,
  "duration_ms": 45,
  "message": "Order created successfully"
}
```

✅ Хранение и ротация

- [ ] **Централизованное хранение** - ELK Stack, Loki, CloudWatch
- [ ] **Ротация логов** - logrotate, max 30 дней
- [ ] **Сжатие** - gzip для старых логов
- [ ] **Защита доступа** - только authorized personnel
- [ ] **Backup логов** - для compliance

Backup и восстановление

✅ Backup стратегия

- [ ] **Автоматические backup** - ежедневно в 2:00 AM
- [ ] **Retention policy** - 7 daily, 4 weekly, 12 monthly
- [ ] **Offsite storage** - AWS S3, Google Cloud Storage
- [ ] **Шифрование backup** - AES-256
- [ ] **Тестирование восстановления** - ежемесячно

✅ Что бэкапить

- [ ] **PostgreSQL база данных** - pg_dump

- [ ] **Конфигурационные файлы** - .env, nginx configs
- [ ] **Секреты** - Google Service Account JSON
- [ ] **Загруженные файлы** - если есть user uploads
- [ ] **Логи** - для compliance

Скрипт backup:

```
#!/bin/bash
BACKUP_DIR="/backups"
TIMESTAMP=$(date +"%Y%m%d_%H%M%S")

# Database backup
pg_dump -U user dbname | gzip > "$BACKUP_DIR/db_$TIMESTAMP.sql.gz"

# Encrypt and upload to S3
gpg --encrypt --recipient admin@example.com "$BACKUP_DIR/db_$TIMESTAMP.sql.gz"
aws s3 cp "$BACKUP_DIR/db_$TIMESTAMP.sql.gz.gpg" s3://bucket/backups/

# Cleanup old backups
find $BACKUP_DIR -name "db_*.sql.gz" -mtime +7 -delete
```

✅ Disaster Recovery Plan

- [ ] **RTO (Recovery Time Objective)** - < 4 часа
- [ ] **RPO (Recovery Point Objective)** - < 24 часа
- [ ] **Runbook** - пошаговая инструкция восстановления
- [ ] **Тестирование DR** - ежеквартально
- [ ] **Контакты** - список ответственных лиц

Compliance и регуляторные требования

✅ 152-ФЗ “О персональных данных” (РФ)

- [ ] **Локализация данных** - серверы в РФ
- [ ] **Согласие пользователя** - явное согласие на обработку ПДн
- [ ] **Политика конфиденциальности** - доступна в боте
- [ ] **Права субъекта** - доступ, исправление, удаление данных
- [ ] **Шифрование ПДн** - phone, address, full_name
- [ ] **Уведомление Роскомнадзора** - если обрабатывается > 1000 субъектов

✅ GDPR (если есть EU пользователи)

- [ ] **Lawful basis** - consent, contract, legitimate interest
- [ ] **Data minimization** - только необходимые данные
- [ ] **Right to be forgotten** - механизм удаления
- [ ] **Data portability** - экспорт данных пользователя
- [ ] **Breach notification** - уведомление в течение 72 часов
- [ ] **DPO (Data Protection Officer)** - если требуется

✅ PCI DSS (если обрабатываются платежи)

- [ ] **Не хранить CVV** - никогда

- [ ] **Токенизация карт** - использовать payment gateway
- [ ] **Шифрование** - все платежные данные
- [ ] **Регулярные аудиты** - ежегодно
- [ ] **Сегментация сети** - изоляция платежной системы

Мониторинг безопасности

✅ Метрики безопасности

- [ ] **Failed login attempts** - алерт при > 10/минуту
- [ ] **403/401 errors** - алерт при аномалиях
- [ ] **Slow queries** - потенциальная SQL injection
- [ ] **Unusual traffic patterns** - DDoS индикаторы
- [ ] **Certificate expiration** - алерт за 30 дней

✅ Инструменты мониторинга

- [ ] **SIEM** - Security Information and Event Management
- [ ] **IDS/IPS** - Intrusion Detection/Prevention System
- [ ] **Vulnerability scanning** - Nessus, OpenVAS
- [ ] **Dependency scanning** - Dependabot, Snyk
- [ ] **SAST/DAST** - Static/Dynamic Application Security Testing

✅ Алерты

- [ ] **Critical alerts** - Telegram/Email немедленно
- [ ] **Warning alerts** - Email в течение часа
- [ ] **Info alerts** - Daily digest
- [ ] **On-call rotation** - 24/7 для critical

Инцидент-менеджмент

✅ Процесс реагирования

1. **Detection** - обнаружение инцидента (мониторинг, пользователи)
2. **Containment** - изоляция (отключение скомпрометированного сервиса)
3. **Eradication** - устранение угрозы (патчи, изменение паролей)
4. **Recovery** - восстановление сервиса
5. **Post-Incident** - анализ, улучшение процессов

✅ Контакты

Роль	Имя	Телефон	Email
Security Lead	TBD	+7...	security@example.com
DevOps Lead	TBD	+7...	devops@example.com
CEO/CTO	TBD	+7...	cto@example.com

✅ Runbook для типовых инцидентов

Компрометация JWT Secret:
1. Немедленно изменить JWT_SECRET_KEY
2. Инвалидировать все активные токены
3. Уведомить всех админов о необходимости re-login
4. Проверить логи на подозрительную активность
5. Провести security audit

SQL Injection обнаружена:
1. Отключить уязвимый endpoint
2. Проверить БД на изменения
3. Восстановить из backup если необходимо
4. Исправить код, добавить тесты
5. Deploy патча
6. Уведомить пользователей если данные скомпрометированы

DDoS атака:
1. Включить Cloudflare “Under Attack” режим
2. Увеличить rate limits
3. Блокировать IP атакующих
4. Масштабировать инфраструктуру если необходимо
5. Связаться с хостинг-провайдером

Чеклист перед Production запуском

Критические проверки

- [ ] Все секреты в Secrets Manager, не в .env
- [ ] SSL сертификаты валидны и автообновляются
- [ ] Firewall настроен, только необходимые порты открыты
- [ ] PostgreSQL доступна только с app server
- [ ] JWT токены с коротким TTL
- [ ] Rate limiting на всех endpoints
- [ ] CORS настроен с whitelist
- [ ] Security headers установлены
- [ ] Логирование настроено, PII замаскированы

- [ ] Backup автоматизированы и протестированы
- [ ] Мониторинг и алерты настроены
- [ ] Incident response plan документирован
- [ ] Security audit проведен
- [ ] Penetration testing выполнен
- [ ] Compliance требования выполнены (152-ФЗ)

Рекомендуемые проверки

- [ ] Dependency scanning настроен
- [ ] SAST/DAST в CI/CD pipeline
- [ ] WAF (Web Application Firewall) настроен
- [ ] DDoS защита активна
- [ ] Honeypot endpoints для обнаружения сканирования
- [ ] Security training для команды
- [ ] Bug bounty программа (опционально)

Регулярные задачи безопасности

Ежедневно

- [ ] Проверка алертов безопасности
- [ ] Мониторинг failed login attempts

Еженедельно

- [ ] Проверка логов на аномалии
- [ ] Review новых CVE для используемых библиотек

Ежемесячно

- [ ] Тестирование backup/restore
- [ ] Проверка SSL сертификатов
- [ ] Review access logs
- [ ] Обновление зависимостей

Ежеквартально

- [ ] Security audit
- [ ] Penetration testing
- [ ] Disaster recovery drill
- [ ] Review и обновление security policies

Ежегодно

- [ ] Полный security assessment
- [ ] Compliance audit (152-ФЗ, GDPR)
- [ ] Обновление incident response plan
- [ ] Security training для команды

Полезные ресурсы

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- OWASP API Security Top 10 (https://owasp.org/www-project-api-security/)
- CIS Benchmarks (https://www.cisecurity.org/cis-benchmarks/)
- NIST Cybersecurity Framework (https://www.nist.gov/cyberframework)
- 152-ФЗ текст закона (http://www.consultant.ru/document/cons_doc_LAW_61801/)
- Роскомнадзор рекомендации (https://rkn.gov.ru/)

Версия: 1.0
Последнее обновление: 2025-11-30
Ответственный: Security Lead / DevOps Lead