Deception Lite · Deception-as-a-Service

Полная инструкция по работе с сервисом

Пошаговое руководство: вход в кабинет, настройка уведомлений, создание и размещение всех 10 типов ловушек, разбор срабатываний и решение типичных проблем.

Адрес сервиса

`http://192.168.19.58`

Личный кабинет

`http://192.168.19.58/app`

Демо-вход

Email: `demo@local` · Пароль: `demo12345`

Демо API key

`daas_devlocal000000000000000`

Версия документа

2026 · Deception Lite v1.0

Содержание

1. Что такое Deception Lite и зачем он нужен

Deception Lite — сервис цифровых ловушек (deception). Вы расставляете **приманки** в инфраструктуре компании. Когда злоумышленник, вирус или любопытный сотрудник касается приманки — вы получаете **алерт** с IP-адресом, временем и деталями.

> **Главная идея:** приманку не трогают в обычной работе. Срабатывание = почти наверняка подозрительная активность. Это дешевле и проще, чем SIEM для малого бизнеса.

Что НЕ делает сервис:

- Не ставит агенты на компьютеры
- Не лезет в вашу сеть сам — вы сами размещаете приманки
- Не блокирует атакующего — только сообщает
- Не заменяет антивирус, файрвол, EDR

2. Как устроен сервис

```
[Вы в кабинете]  →  Создаёте ловушку  →  Получаете артефакт (ссылка / файл / DNS-имя)
        ↓
[Размещение]     →  Кладёте приманку в wiki / на шару / в конфиг / в письмо
        ↓
[Атакующий]      →  Открывает ссылку / скачивает файл / резолвит DNS
        ↓
[Deception Lite] →  Фиксирует событие  →  Кабинет + Telegram/Email
```

Компоненты на сервере (у вас: `192.168.19.58`):

Компонент	Назначение	URL / порт
Главная страница	Лендинг, описание продукта	`/`
Личный кабинет	Управление ловушками	`/app`
API	Программный доступ	`/api/v1/...`
Ловушки HTTP	Ссылки, файлы, decoy	`/t/` `/f/` `/d/` `/p/`
DNS Canary	DNS-запросы к приманке	UDP порт `5353`
Healthcheck	Проверка что сервис жив	`/health`

3. Вход в личный кабинет

Откройте: **`http://192.168.19.58/app`**

Способ 1 — Регистрация (для постоянной работы)

1. Вкладка **«Регистрация»**
2. Email — ваш рабочий адрес
3. Пароль — минимум 8 символов
4. Организация — название компании (любое)
5. Кнопка **«Создать аккаунт»**

После регистрации вы сразу попадаете в кабинет. API key показывается один раз — сохраните его.

Способ 2 — Вход по email

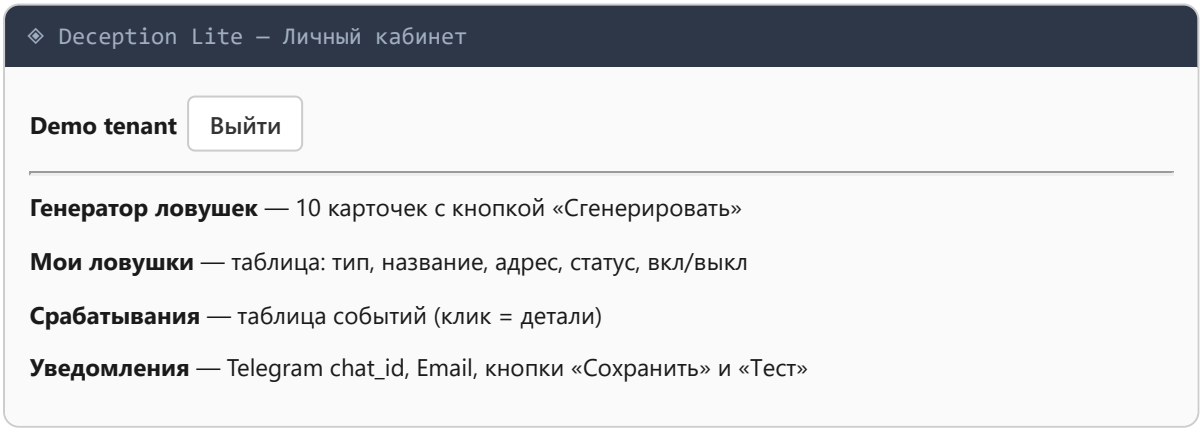
1. Вкладка **«Вход»**
2. Введите email и пароль
3. Кнопка **«Войти»**

Способ 3 — API key (быстрый тест)

1. Вкладка **«API key»**
2. Вставьте ключ целиком
3. Кнопка **«Подключиться»**

> **Демо-учётка** (уже создана на сервере):
> Email: `demo@local` · Пароль: `demo12345`
> API key: `daas_devlocal000000000000000`

4. Обзор интерфейса кабинета



Раздел	Что делать
Генератор ловушек	Выбрать тип → «Сгенерировать» → скопировать артефакт из всплывающего окна
Мои ловушки	Список всех приманок. Кнопки: вкл/выкл, удалить. Адрес — URL или DNS-имя
Срабатывания	Журнал: время, действие, IP, клиент (браузер/curl/сканер), название приманки. **Клик по строке** — полные детали
Уведомления	Привязать Telegram и/или Email. Кнопка «Тест» — проверить доставку

5. Настройка Telegram-уведомлений

> Токен бота (`TELEGRAM_BOT_TOKEN`) прописывает **администратор сервера** в файле `backend/.env`. Без токена Telegram не заработает — попросите админа.

Шаг 1 — Создать бота

1. Telegram → найти **@BotFather**
2. Отправить: `/newbot`
3. Придумать имя и username (должен заканчиваться на `bot`)
4. Скопировать токен вида `123456789:AAHxxxxxxxxxxxxxxxxxxxxxxxxx`
5. Передать токен администратору сервера

Шаг 2 — Запустить бота

1. Найти своего бота в Telegram
2. Нажать **Запустить** или отправить `/start`
3. Написать любое сообщение (например «привет»)

Шаг 3 — Узнать chat_id

Откройте в браузере (подставьте свой токен):

```
https://api.telegram.org/bot<ТОКЕН>/getUpdates
```

Найдите в ответе:

```
"chat":{"id":123456789, ...}
```

Число `123456789` — это ваш **chat_id**.

Шаг 4 — Сохранить в кабинете

1. Кабинет → раздел **«Уведомления»**
2. Поле **Telegram chat_id** → вставить число

3. Кнопка **«Сохранить Telegram»**
4. В таблице каналов нажать **«Тест»**
5. В Telegram должно прийти: «✅ Тест Deception Lite»

6. Настройка Email-уведомлений

1. Кабинет → **«Уведомления»**
2. Поле **Email** → ваш адрес
3. **«Сохранить Email»**
4. **«Тест»** — проверить почту

SMTP настраивает администратор (`SMTP_HOST` , `SMTP_USER` , `SMTP_PASS` в `backend/.env`).

7. Создание ловушки — общий алгоритм

1. **Генератор ловушек** → выбрать карточку
2. Кнопка **«Сгенерировать»**
3. В модальном окне:
   - **Артефакты** — ссылка, HTML-код, файл, DNS-имя (копировать кнопкой)
   - **Куда разместить** — описание места
   - **Как разместить** — пошаговый список
   - **Проверка** — как убедиться что работает
4. Разместить приманку по инструкции
5. Выполнить проверку
6. Убедиться что событие появилось в **«Срабатывания»**

> **Формат URL ловушек на вашем сервере:**
> Ссылка: `http://192.168.19.58/t/CEKPET`
> Пиксель: `http://192.168.19.58/p/CEKPET.gif`
> Файл: `http://192.168.19.58/f/CEKPET`
> Decoy (live): `http://192.168.19.58/d/CEKPET`
> Decoy (скачать файл): `http://192.168.19.58/d/CEKPET/download`
> DNS: `секрет.192.168.19.58` (проверка: `dig @192.168.19.58 -p 5353 имя A`)

8. Все 10 ловушек — подробно

№1 · ДОКУМЕНТ

🔗 Ссылка в документе

Что ловит: открытие гиперссылки в документе, wiki, README.

Куда класть: Word/Excel, Confluence, Notion, README в git, Jira, архив на файловой шаре.

Как разместить:

1. Сгенерировать → скопировать URL
2. Word: выделить текст «backup_2024» → Вставка → Гиперссылка → вставить URL
3. Confluence/README: ссылка в разделе «Архивные конфиги» или «Legacy docs»
4. Не давать ссылку сотрудникам в обычной работе

Проверка: открыть URL в браузере → в «Срабатывания» появится `http_get`

№2 · EMAIL

📧 Web-пиксель в письме

Что ловит: загрузку 1×1 GIF — кто-то открыл HTML-письмо (часто за периметром).

Куда класть: HTML-тело письма, шаблон рассылки, HTML-отчёт на портале.

Артефакт — вставить в HTML:

```
<img src="http://192.168.19.58/p/CEKPET.gif" width="1" height="1" alt="" style="di
splay:none">
```

Проверка: вставить в test.html, открыть в браузере → событие `pixel_load`

№3 · СЕТЬ

📡 DNS Canary

Что ловит: DNS-запрос к уникальному имени — разведка сети, утечка конфига.

Куда класть: docker-compose, ansible inventory, CI/CD variables, VPN-конфиг, runbook.

Пример в docker-compose.yml:

```
environment:
  BACKUP_HOST: secret123.192.168.19.58
  LEGACY_DB: secret123.192.168.19.58
```

Проверка:

```
dig @192.168.19.58 -p 5353 secret123.192.168.19.58 A
```

→ событие `dns_lookup` в кабинете

№4 · ФАЙЛ

🔑 Файл passwords.txt

Что ловит: скачивание фейкового файла с паролями.

Куда класть: `\\corp\shared\IT_backup_old\`, NAS, папка passwords_archive.

1. Сгенерировать → скачать passwords.txt по trap-URL
2. Создать папку IT_backup_old на шаре
3. Положить файл — **не рядом** с реальными паролями

Проверка: `curl -O "http://192.168.19.58/f/CEKPET"` → `file_download`

№5 · ФАЙЛ

⚙️ Файл .env.production

Что ловит: скачивание фейкового конфига с секретами (DB, AWS keys).

Куда класть: zip «deploy package», staging-сервер, S3-бакет артефактов, git infra-репо.

Пример: положить в `site-backup-2024.zip` → `prod_configs/.env.production`

Проверка: `curl -O trap-URL` → `file_download`

№6 · ФАЙЛ

🗄 SQL-дамп backup.sql

Что ловит: скачивание фейкового дампа БД.

Куда класть: каталог бэкапов БД, MinIO/S3, папка DBA на шаре.

Пример пути: `\\backup\mysql\_old\backup_2024.sql`

Проверка: `wget trap-URL` → `file_download`

№7 · ФАЙЛ

🔒 VPN-профиль .ovpn

Что ловит: скачивание фейкового VPN-профиля.

Куда класть: SharePoint «VPN profiles», zip vpn_all_employees.zip, общий диск.

Проверка: скачать по trap-URL → `file_download`

№8 · DECOY

Decoy /.env

Что ловит: HTTP-запрос к фейковому .env (сканеры nikto, dirsearch) и скачивание файла-приманки.

Два артефакта после генерации:

- **URL (live endpoint):** `http://192.168.19.58/d/CEKPET` — отдаёт правдоподобный конфиг в браузере/curl.
- **Файл для размещения:** `http://192.168.19.58/d/CEKPET/download` — скачивает `.env` для копирования на шару или в zip.

Куда размещать:

- Файловая шара: `\\corp\shared\staging_backup\.env` или внутри `site-backup-2024.zip`
- Фишинговое письмо: «обнаружен конфиг staging-сервера» со ссылкой на trap-URL
- Wiki/README как «legacy config», отчёт сканера, поддельный swagger

Как разместить:

1. Сгенерировать decoy → нажать **«Скачать»** в модальном окне (файл `.env`)
2. Положить файл на шару или в архив; либо вставить live URL в письмо/документ
3. Не класть рядом с реальными секретами

Проверка:

- `curl -v trap-URL` → `http_get`, в ответе `DB_PASS=...`
- `curl -OJ trap-URL/download` → файл `.env` на диске + `file_download`

№9 · DECOY

Decoy /admin

Что ловит: запрос к фейковой панели администратора (GET, POST при брутфорсе).

Два артефакта:

- **URL:** открывается в браузере — HTML-форма входа (`http_get`)
- **Скачивание:** `trap-URL/download` → файл `admin.html` для размещения на шаре

Куда размещать:

- Шара: `\\corp\web\old_site\admin.html` или архив `backup-www-2024.zip`
- Отчёт pentest: «обнаружена открытая админка» со ссылкой trap-URL
- Письмо коллеге: «нашёл админку на старом сервере»

Как разместить:

1. Скачать `admin.html` через кнопку «Скачать» в кабинете
2. Положить в каталог «старый сайт» или zip-бэкап
3. Альтернатива: дать только live URL в отчёте сканера

Проверка: браузер → форма входа + `http_get` . `curl -OJ trap-URL/download` → `admin.html` .

№10 · DECOY

Decoy /api/users

Что ловит: запрос к фейковому REST API со списком «админов» (GET и POST).

Два артефакта:

- **Live API:** `curl trap-URL` → JSON со списком users (`http_get`)
- **Файл:** `trap-URL/download` → `api_users.json` для архива документации

Куда размещать:

- Архив `api-docs-internal.zip` на файловом сервере
- Фейковый `swagger.json` : `"users": "trap-URL"` или `servers: [{ url: trap-URL }]`
- Postman collection: GET trap-URL с названием «List users (prod)»
- Чат разработчиков: «утёкшая» коллекция API

Как разместить:

1. Скачать `api_users.json` через «Скачать»
2. Добавить в zip с API-документацией или вставить live URL в swagger/Postman

Проверка: `curl trap-URL` → JSON + `http_get` . `curl -OJ trap-URL/download` → `file_download` .

9. Раздел «Срабатывания»

Поле	Что значит	Пример
Время	Когда произошло (МСК)	15.07.2026 11:30:45
Действие	Тип события	`http_get`, `file_download`, `dns_lookup`, `pixel_load`
IP	Откуда запрос	185.x.x.x (публичный) или 10.x.x.x (внутренний)
Клиент	User-Agent	Chrome, curl, wget, nikto, python-requests
Приманка	Какая ловушка сработала	backup-link, dns-canary-prod

Типы действий

Код	Расшифровка
`http_get`	Открыли ссылку или decoy (GET-запрос)
`http_post`	POST-запрос к decoy (брутфорс, сканер)
`pixel_load`	Загрузили web-пиксель в письме
`dns_lookup`	DNS-запрос к canary-имени
`file_download`	Скачали файл-приманку (в т.ч. decoy через /download)
`test`	Тестовое уведомление из кабинета

Клик по строке в таблице — раскрывает детали: referer, POST body, полный User-Agent, severity (high/medium/low).

10. Что делать при срабатывании

1. **Не паниковать.** Это сигнал, не подтверждённый взлом.
2. Открыть детали события — записать IP, время, User-Agent.
3. Определить источник:
   - Ваш тест → всё ок
   - Известный IP сотрудника → поговорить
   - curl/wget/python-requests/nikto → **подозрительно**
   - Неизвестный внешний IP → **эскалация в ИБ**
4. Сделать скриншот события.
5. При необходимости отключить ловушку в «Мои ловушки».
6. Проверить смежные системы (логи файрвола, SIEM если есть).

11. API — для автоматизации

Базовый URL: `http://192.168.19.58/api/v1`

Авторизация: заголовок `Authorization: Bearer <api_key>` или JWT-токен после login.

Примеры curl

```
# Каталог ловушек
curl -H "Authorization: Bearer daas_devlocal000000000000000" \
  http://192.168.19.58/api/v1/traps/catalog

# Создать ловушку (ссылка в документе)
curl -X POST -H "Authorization: Bearer КЛЮЧ" \
  -H "Content-Type: application/json" \
  -d '{"trap_kind":"document_link","name":"my-trap"}' \
  http://192.168.19.58/api/v1/traps

# Список срабатываний
curl -H "Authorization: Bearer КЛЮЧ" \
  http://192.168.19.58/api/v1/events

# Логин
curl -X POST -H "Content-Type: application/json" \
  -d '{"email":"demo@local","password":"demo12345"}' \
  http://192.168.19.58/api/v1/auth/login
```

Основные эндпоинты

Метод	Путь	Описание
GET	`/traps/catalog`	10 типов ловушек
POST	`/traps`	Создать ловушку `{"trap_kind":"..."}`
GET	`/traps`	Список ваших ловушек
GET	`/traps/:id`	Ловушка + инструкция
GET	`/events`	Срабатывания
POST	`/auth/login`	Вход, получить JWT
POST	`/auth/register`	Регистрация
POST	`/channels/:id/test`	Тест уведомления

12. Типичные проблемы

Не приходит Telegram

Причина	Решение
Не написали /start боту	Открыть бота → Запустить → написать сообщение
Неверный chat_id	Перепроверить через getUpdates. Только число, без пробелов
Нет TELEGRAM_BOT_TOKEN на сервере	Админ добавляет в backend/.env и перезапускает backend
api.telegram.org недоступен	Админ: VPN или TELEGRAM_PROXY в .env

Нет срабатываний

Причина	Решение
Приманку не разместили	Не только скопировать — реально положить/вставить
Ловушка выключена	«Мои ловушки» → включить
DNS: неверный порт	Использовать `dig @IP -p 5353`, не порт 53
Тестируете с того же IP без изменений	Попробовать curl с другой машины

Не открывается кабинет

Причина	Решение
Неверный URL	Нужен `/app` в конце: `http://192.168.19.58/app`
Сервер выключен	Админ: `docker compose -f docker-compose.prod.yml ps`
Healthcheck	`curl http://192.168.19.58/health` → `{"status":"ok"}`

13. Глоссарий

Термин	Объяснение
Ловушка / приманка	Фейковый объект (ссылка, файл, DNS, страница), касание = алерт
Артефакт	Готовый продукт ловушки: URL, файл, HTML-код, hostname
Срабатывание / событие	Факт касания приманки, запись в журнале
Decoy	Фейковый HTTP-endpoint (/.env, /admin, /api/users)
DNS Canary	Уникальное DNS-имя; lookup = разведка
Tenant	Ваша организация в системе (изолированные данные)
API key	Ключ `daas_...` для доступа к API
JWT	Токен после login, для API вместо api key
User-Agent	Строка «кто стучится»: браузер, curl, сканер
Severity	Критичность: low / medium / high / critical