

OSPAB.VPN

Разработка устойчивого сетевого протокола на базе VLESS

Автор: Сыралёв Георгий (7 "М")
Гимназия №2, Великий Новгород

ВАЖНО: ДИСКЛЕЙМЕР

⚠ Научное исследование

Данный проект является исключительно **научно-исследовательской работой**, посвященной изучению сетевых протоколов. Он не предназначен для коммерческого использования или нарушения законодательства РФ.

Цель работы

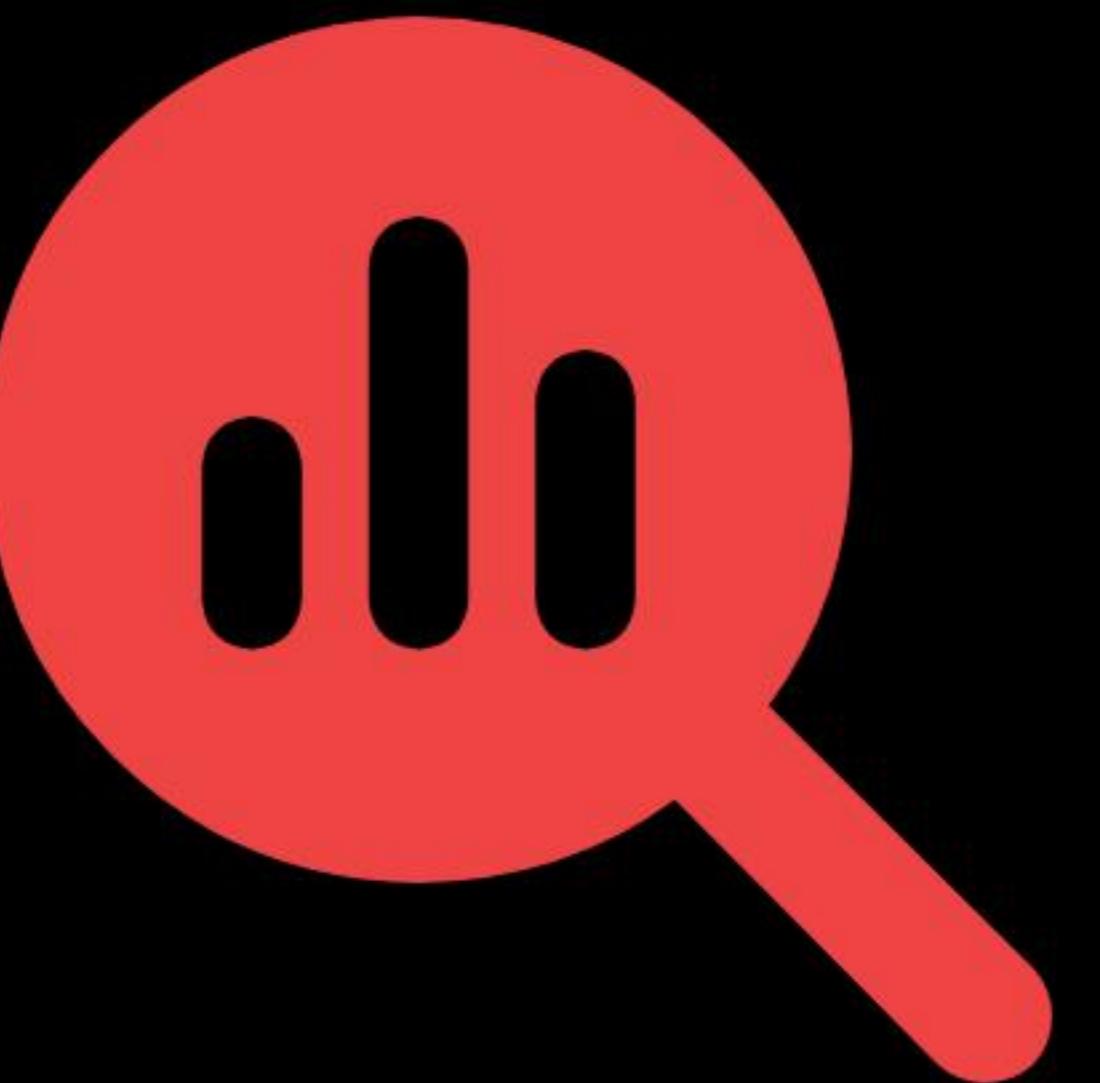
- 🧪 Изучить методы фильтрации трафика.
- 🛡 Разработать прототип с повышенной сетевой адаптивностью.
- 💻 Доказать концепцию на языке Python.

ПРОБЛЕМА: СЕТЕВАЯ ФИЛЬТРАЦИЯ

Система **DPI** (Глубокий Анализ Пакетов) отвечает за **безопасность и соответствие** сетевого трафика установленным нормам.

Она работает как строгий контролёр, который проверяет все данные, проходящие через сеть.

DPI идентифицирует и отфильтровывает трафик с подозрительными или несовместимыми сигнатурами.



НЕДОСТАТКИ ТРАДИЦИОННЫХ ПРОТОКОЛОВ



Классические VPN протоколы такие как OpenVPN устарели с точки зрения адаптивности:

- ✖ Имеют **уникальные сигнатуры**, которые легко обнаружить.
- ✖ Используют **устаревшие методы маскировки**.

Итог: Протокол быстро идентифицируется и ограничивается.

МОЁ РЕШЕНИЕ: МАСКИРОВКА ТРАФИКА

Мы используем метод мимикрии для **повышения адаптивности** протокола.



Традиционный VPN
Выглядит как VPN.

ФИЛЬТРУЕТСЯ

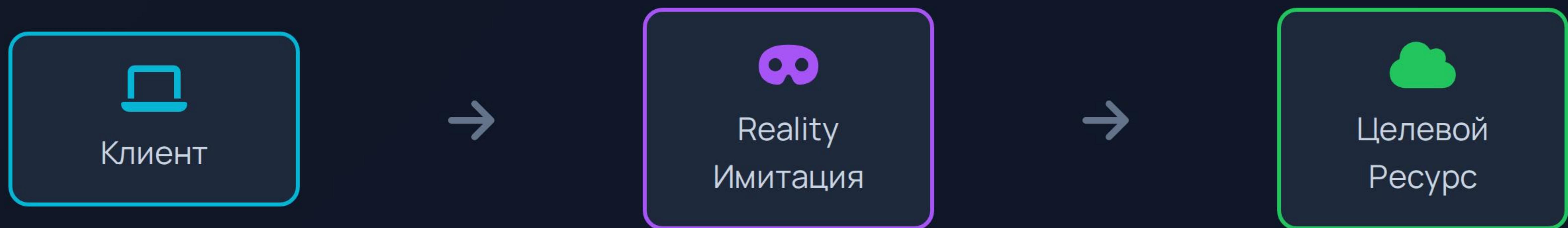


Мой проект
Выглядит как Google/Apple.

ПРОПУСКАЕТСЯ

СХЕМА РАБОТЫ ПРОТОТИПА

Трафик имитирует стандартное HTTPS-соединение к легитимному ресурсу.



✓ DPI видит стандартный HTTPS-трафик и классифицирует его как доверенный.

ПРЕИМУЩЕСТВА VLESS + REALITY

Традиционный VPN

- ☒ Низкая скорость
- ☒ Обнаруживаемая сигнатура
- ☒ Высокий расход энергии

Мой подход

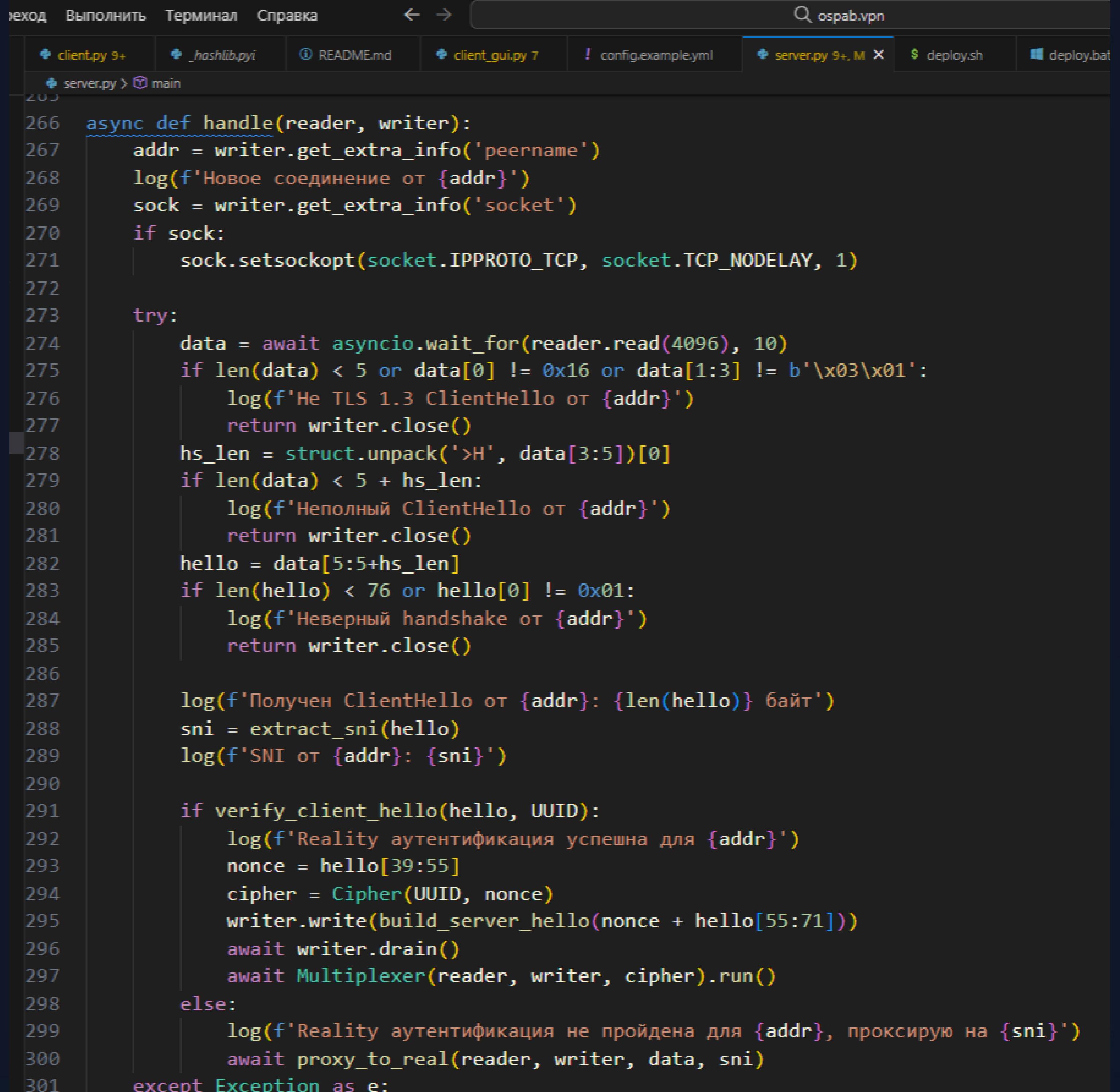
- ⚡ Высокая производительность
- ⚡ Неопределяемая сигнатура
- ⚡ Низкая нагрузка

АВТОРСКАЯ РЕАЛИЗАЦИЯ

Прототип был разработан на языке **Python**.

Это позволило детально изучить процесс создания сетевых сокетов и механизмов TLS-рукопожатия.

Я использовал архитектуру VLESS как основу для своего прототипа, реализовав всю логику работы на чистом Python.



```
вход Выполнить Терминал Справка ← → Q osrav.vpn
client.py 9+ _hashlib.pyi README.md client_gui.py 7 config.example.yml
server.py > main server.py 9+, M deploy.sh deploy.bat

266     async def handle(reader, writer):
267         addr = writer.get_extra_info('peername')
268         log(f'Новое соединение от {addr}')
269         sock = writer.get_extra_info('socket')
270         if sock:
271             sock.setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)
272
273     try:
274         data = await asyncio.wait_for(reader.read(4096), 10)
275         if len(data) < 5 or data[0] != 0x16 or data[1:3] != b'\x03\x01':
276             log(f'Не TLS 1.3 ClientHello от {addr}')
277             return writer.close()
278         hs_len = struct.unpack('>H', data[3:5])[0]
279         if len(data) < 5 + hs_len:
280             log(f'Неполный ClientHello от {addr}')
281             return writer.close()
282         hello = data[5:5+hs_len]
283         if len(hello) < 76 or hello[0] != 0x01:
284             log(f'Неверный handshake от {addr}')
285             return writer.close()
286
287         log(f'Получен ClientHello от {addr}: {len(hello)} байт')
288         sni = extract_sni(hello)
289         log(f'SNI от {addr}: {sni}')
290
291         if verify_client_hello(hello, UUID):
292             log(f'Reality аутентификация успешна для {addr}')
293             nonce = hello[39:55]
294             cipher = Cipher(UUID, nonce)
295             writer.write(build_server_hello(nonce + hello[55:71]))
296             await writer.drain()
297             await Multiplexer(reader, writer, cipher).run()
298         else:
299             log(f'Reality аутентификация не прошла для {addr}, проксирую на {sni}')
300             await proxy_to_real(reader, writer, data, sni)
301     except Exception as e:
```

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Сравнение скорости передачи данных в Мбит/с:

Прямое соединение

95 Мбит/с

Прототип (Python)

72 Мбит/с

● Наблюдаемая потеря скорости (24%) обусловлена особенностями языка Python, что является важным выводом для дальнейших разработок.

ЗАКЛЮЧЕНИЕ



Проект успешно завершен

Разработан прототип, подтверждающий высокий уровень адаптивности
сетевого трафика с использованием технологии Reality.

Спасибо за внимание!