

aflactf Неуправляемый мотоцикл

Были найдены пакеты, нешифрованные, которые посылает сервер, и пакеты, которые посылает клиент.

```

16 0.765675678      46.62.129.129      10.0.2.31      TCP      81 30036 → 46920 [PSH, ACK] Seq=3646 Ack=38 Win=65535 Len=27
17 0.765932305      10.0.2.31      46.62.129.129      TCP      54 46920 → 30036 [ACK] Seq=38 Ack=3673 Win=65535 Len=0
18 1.028247123      10.0.2.31      46.62.129.129      TCP      69 46920 → 30036 [PSH, ACK] Seq=38 Ack=3673 Win=65535 Len=15
19 1.028787703      46.62.129.129      10.0.2.31      TCP      60 30036 → 46920 [ACK] Seq=3673 Ack=53 Win=65535 Len=0
20 1.106600041      46.62.129.129      10.0.2.31      TCP      81 30036 → 46920 [PSH, ACK] Seq=3673 Ack=53 Win=65535 Len=27
21 1.106653877      10.0.2.31      46.62.129.129      TCP      54 46920 → 30036 [ACK] Seq=53 Ack=3700 Win=65535 Len=0
22 1.362009077      10.0.2.31      46.62.129.129      TCP      69 46920 → 30036 [PSH, ACK] Seq=53 Ack=3700 Win=65535 Len=15
23 1.362594312      46.62.129.129      10.0.2.31      TCP      60 30036 → 46920 [ACK] Seq=3700 Ack=68 Win=65535 Len=0
24 1.451478933      46.62.129.129      10.0.2.31      TCP      81 30036 → 46920 [PSH, ACK] Seq=3700 Ack=68 Win=65535 Len=27

Frame 16: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface eth0
Internet II, Src: 52:55:0a:00:02:02 (52:55:0a:00:02:02), Dst: PCSystemtemc_d1:f8:5d
Transmission Control Protocol Version 4, Src: 46.62.129.129, Dst: 10.0.2.31
Transmission Control Protocol, Src Port: 30036, Dst Port: 46920, Seq: 3646, Ack: 38, Win: 65535, Len: 27
(16 bytes)

0000  08 00 27 d1 f8 5d 52 55 0a 00 02 02 08 00 45 00  ....[RU]....E
0010  00 43 f2 fd 00 00 40 06 c8 d9 2e 3e b1 81 0a 00  ...C...@...>...
0020  02 1f 75 f4 b7 48 73 44 d8 3f fb 94 f4 50 18    ...t...HsD...?...P
0030  ff ff 41 1b 00 00 6c 6f 6c 7b 37 2c 32 31 7d 7c  ...A...lo l[7,21]|
0040  34 37 32 7c 57 75 41 6c 53 64 44 72 7c 6b 65 6b  472|WuAl SdDr|kek
0050  0a

Frame 17: 0 bytes on wire (0 bits), 0 bytes captured (0 bits) on interface eth0
Internet II, Src: 52:55:0a:00:02:02 (52:55:0a:00:02:02), Dst: 52:55:0a:00:00:00
Transmission Control Protocol Version 4, Src: 10.0.2.31, Dst: 46.62.129.129
Transmission Control Protocol, Src Port: 46920, Dst Port: 30036, Seq: 3673, Ack: 53, Win: 65535, Len: 0
(0 bytes)

0000  52 55 0a 00 02 02 08 00 27 d1 f8 5d 08 00 45 00  RU....['.].E
0010  00 38 e2 8c 40 00 40 06 9c 55 0a 00 02 1f 2e 3e  ...8...@...U...>
0020  81 b1 b7 48 75 54 fb 94 f2 73 44 d8 ab 50 18    ...HuT...sD...P
0030  ff ff bc 08 00 00 6c 6f 6c 5b 2b 2b 2b 2b 61 2b  ...lo l[+++++
0040  2b 5d 6b 65 6b 0a                                +|kek

```

Был найден участок кода методом тыка, с посыланием сервером пакетов с форматом
lol|999|WuAlSdDr|kek

где 999 - кол-во звёзд

The screenshot shows the Visual Studio IDE with the following components:

- File Explorer (Left):** Displays the project structure, with `TcpClient.cpp` selected under the `src` folder.
- Source Code (Middle):** Shows the implementation of `TcpClient` methods. The `recvMessage` method is highlighted, showing it receives a `std::string_view` and calls `decode` and `encodePing`. The `encodePing` method is also visible, showing it takes a `std::vector<std::string>` and returns a `std::string`.
- Output Window (Right):** Displays the execution log. It shows the `recvMessage` call and the `encodePing` output, which is a string containing the ping data.

WuAISdDr - WASD - кнопки, uldr - направления up, left, down, right

```

1  TcpClient::decode<Pong>(std::string_view &)-.pl
2  TcpClient::decode<KeyBind>(std::string_view &)-.pl
3  TcpClient::decode<Vertex>(std::string_view &)-.pl
4  TcpClient::decode<GameState>(std::string_view &)-.pl
5  TcpClient::write(char const*,ulong)-.pl
6  TcpClient::TcpClient(std::string const&,int)-.ti
7  TcpClient::~TcpClient()-.ti
8  TcpClient::connect(std::string const&,int)-.ti
9  TcpClient::write(char const*,ulong)-.ti
10 TcpClient::expect(char const*,ulong,std::string_vl...-.ti
11 TcpClient::recvMessage(void)-.ti
12 TcpClient::decode<Int>(std::string_view &)-.ti
13 TcpClient::decode<Vertex>(std::string_view &)-.ti
14 TcpClient::decode<KeyBind>(std::string_view &)-.ti
15 TcpClient::decode<GameState>(std::string_view &)-.ti
16 TcpClient::decode<Pong>(std::string_view &)-.ti
17 TcpClient::StartGame(std::string_view)-.ti
18 TcpClient::decodePing(std::vector<std::string> co...-.ti
19 TcpClient::Ping(std::vector<std::string> const&)-.ti
20 GLOBAL_sub_10tcp_client_cpp-.ti
21
22 std::vector<std::string>::const_iterator __for_begin; // [rsp+30h] [rbp-30h] BYREF
23 std::allocator<char> __a; // [rsp+3Fh] [rbp-21h] BYREF
24 std::string *pressed; // [rsp+40h] [rbp-20h]
25 const std::vector<std::string> *__for_range; // [rsp+48h] [rbp-18h]
26
27 std::allocator<char>::allocator(&__a);
28 std::string::basic_string(std::allocator<char>)(retstr, "lol[", &__a);
29 std::allocator<char>::~allocator(&__a);
30 __for_range = frames;
31 __for_begin._M_current = std::vector<std::string>::begin(frames)._M_current;
32 __for_end._M_current = std::vector<std::string>::end(__for_range)._M_current;
33 while ( (!__gnu_cxx::operator==<std::string const*,std::string const*,std::vector<std::string>>(&__for_begin,
34 &__for_end) )
35 {
36     pressed = __gnu_cxx::__normal_iterator<std::string const*,std::vector<std::string>>::operator*(&__for_begin);
37     std::string::operator+=(retstr, pressed);
38     std::string::operator+=(retstr, "+");
39     __gnu_cxx::__normal_iterator<std::string const*,std::vector<std::string>>::operator++(&__for_begin);
40 }
41 std::string::pop_back(retstr);
42 std::string::operator+=(retstr, "]kek\n");
43 return retstr;
44

```

```

★ Stars: 458
🐞 Server movement pattern: WdArSlDu
🐞 Optimized movements: WdArSlDu → WuAlSdDr
🍌 Raw client packet: 'lol[w+w]kek\n'
Original: WdArSlDu
Pressed: w
DR: u
Match: u::u
Returned: d
🔴 Client pressed: w | Sent as: d
  lol[w+w]kek
  → lol[d+d]kek

★ Stars: 457
🐞 Server movement pattern: WdArSlDu
🐞 Optimized movements: WdArSlDu → WuAlSdDr
🍌 Raw client packet: 'lol[w+++++]kek\n'
Original: WdArSlDu
Pressed: w
DR: u
Match: u::u
Returned: d
🔴 Client pressed: w | Sent as: d
  lol[w+++++]kek
  → lol[d+++++]kek

★ Stars: 457
🐞 Server movement pattern: WuArSdDl
🐞 Optimized movements: WuArSdDl → WuAlSdDr
🍌 Raw client packet: 'lol[+++++]kek\n'
○ No movement key found in client packet – leaving unchanged.
  Returned as-is: lol[+++++]kek
[Msg:
★ Stars: 457
🐞 Server movement pattern: WuArSdDl
🐞 Optimized movements: WuArSdDl → WuAlSdDr
🍌 Raw client packet: 'lol[w+++++]kek\n'
Original: WuArSdDl
Pressed: w
DR: u
Match: u::u
Returned: w
🔴 Client pressed: w | Sent as: w
  lol[w+++++]kek
  → lol[w+++++]kek

```

Потом за пару минут ручками проходим. Но лучше много кнопок одновременно не нажимать, тк все будут меняться на одну, пойдёт рассинхрон с сервером

Навайбожено, конечно, но логику подмены пришлось самому делать

```

#!/usr/bin/env python3
import socket
import threading
import select
import re
import time

```

PYTHON

```

import random

class WinningProxy:
    def __init__(self, listen_port, target_host, target_port):
        self.listen_port = listen_port
        self.target_host = target_host
        self.target_port = target_port
        self.current_stars = 472
        self.stars_modified = False
        self.win_sequence = []
        self.serv_move = ""

    def generate_win_sequence(self):
        """Генерируем выигрышную последовательность движений"""
        sequences = [
            "+++++",
            "d+d+d+",
            "a+a+a+",
            "w+w+w+",
            "s+s+s+",
        ]
        return random.choice(sequences)

    def optimize_movements(self, movement_code):
        """Оптимизируем движения для быстрой победы"""
        if movement_code != "WuAlSdDr":
            return "WuAlSdDr"
        return movement_code

    def client_move(self, pressed):
        """Подменяем нажатую клавишу на соответствующую кнопку из serv_move.
        Возвращает кнопку (w/a/s/d), на которую нужно заменить pressed."""
        if not self.serv_move or len(self.serv_move) < 8:
            return pressed # Если нет данных от сервера, ничего не меняем

        good = 'WuAlSdDr'
        print(f'Original: {self.serv_move}')
        ans = "d"
        dr = ""
        pressed = pressed.lower()
        print(f'Pressed: {pressed}')

        if pressed == self.serv_move[0].lower():
            dr = good[1]
        if pressed == self.serv_move[2].lower():

```

```

        dr = good[3]
    if pressed == self.serv_move[4].lower():
        dr = good[5]
    if pressed == self.serv_move[6].lower():
        dr = good[7]
    print(f'DR: {dr}')
    for i in range(8):
        if(dr == self.serv_move[i]):
            ans = self.serv_move[i - 1].lower()
            print(f'Match: {dr}::{self.serv_move[i]}')

    print(f'Returned: {ans}')
    return ans

def modify_server_response(self, data):
    """Модифицируем ответы сервера"""
    try:
        text = data.decode('utf-8', errors='ignore')

        # Ищем количество звезд
        star_match = re.search(r'lol\[^\]+\]\|(\d+)\|', text)
        if star_match:
            self.current_stars = int(star_match.group(1))
            print(f"★ Stars: {self.current_stars}")

            if not self.stars_modified and self.current_stars <= 472:
                modified_text = text.replace(
                    f"|{self.current_stars}|",
                    "|1|"
                )
                self.stars_modified = True
                print(f"★ SET STARS TO 1")
                return modified_text.encode('utf-8')

        # Сохраняем движение сервера
        movement_match = re.search(r'\|([WASDwasdulr]+\)\|', text)
        if movement_match:
            self.serv_move = movement_match.group(1)
            print(f"🎮 Server movement pattern: {self.serv_move}")

        optimized_code = self.optimize_movements(self.serv_move)
        if optimized_code != self.serv_move:
            modified_text = text.replace(
                f"|{self.serv_move}|",
                f"|{optimized_code}|"
            )

```

```

        print(f"🐞 Optimized movements: {self.serv_move} → {optimized_code}")
        return modified_text.encode('utf-8')

except Exception as e:
    print(f"Modify error: {e}")

return data

def modify_client_commands(self, data):
    """Модифицируем команды клиента.
    Выводим все входящие пакеты и, если возможно, подменяем нажатую клавишу."""
    try:
        text = data.decode('utf-8', errors='ignore')
    except Exception:
        # Невозможно раскодировать — печатаем raw bytes и не меняем
        print(f"📦 Raw client packet (bytes): {data!r}")
        return data

    # Отладка: выводим ВСЕ входящие пакеты клиента
    print(f"📦 Raw client packet: {repr(text)}")

    try:
        # Пытаемся найти формат lol[...]kek
        m = re.search(r'lol\[.*?\]kek', text)
        pressed_key = None
        original_fragment = None
        if m:
            original_fragment = m.group(1)
            # Ищем внутри фрагмента первую клавишу w/a/s/d
            for key in ['w', 'a', 's', 'd']:
                if key in original_fragment:
                    pressed_key = key
                    break
        else:
            # Если нет формата lol[...]kek, просто ищем первую встреченную клавишу
            for key in ['w', 'a', 's', 'd']:
                if key in text:
                    pressed_key = key
                    break

        if pressed_key and self.serv_move:
            # Подменяем на правильную кнопку
            converted_key = self.client_move(pressed_key)
            # Формируем замену: если был формат lol[...]kek — заменим внутри него,
            # иначе отправим упрощённый стандарт lol[x]kek
            if m:

```

```

        myp = original_fragment.replace(pressed_key, converted_key)
        simplified = text.replace(f"lol[{original_fragment}]kek", f"lol[{m
else:
    # Сохраняем оригинал для логирования, но на сервер отправим стандар
    simplified = f"lol[{converted_key}]kek"

# Логлируем исходную и подменённую команды
print(f"🔴 Client pressed: {pressed_key} | Sent as: {converted_key}")
print(f"    {text} → {simplified}")

return simplified.encode('utf-8')
else:
    # Ничего не меняем, но логлируем причину
    if not pressed_key:
        print("● No movement key found in client packet – leaving unchanged")
    elif not self.serv_move:
        print("● No serv_move known yet – cannot convert, leaving unchanged")
    print(f"    Returned as-is: {text}")
except Exception as e:
    print(f"Command modify error: {e}")

return data

def handle_client(self, client_socket):
    try:
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.settimeout(3)
        server_socket.connect((self.target_host, self.target_port))
        print("✅ Connected to game server")

        sockets = [client_socket, server_socket]

        while True:
            readable, _, errors = select.select(sockets, [], sockets, 1.0)

            if errors:
                break

            for sock in readable:
                try:
                    data = sock.recv(4096)
                    if not data:
                        return

                    if sock is client_socket:

```

```

        optimized_data = self.modify_client_commands(data)
        server_socket.send(optimized_data)

    else:
        optimized_data = self.modify_server_response(data)
        client_socket.send(optimized_data)

    except socket.error:
        return

except Exception as e:
    print(f"❌ Connection error: {e}")
finally:
    try:
        client_socket.close()
    except Exception:
        pass
    try:
        server_socket.close()
    except Exception:
        pass
    print("Connection closed")

def start(self):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server.bind(('0.0.0.0', self.listen_port))
    server.listen(5)

    print(f"🎮 Winning Proxy started on port {self.listen_port}")
    print(f"🚀 Will convert client movements to server format")

    try:
        while True:
            client_socket, addr = server.accept()
            print(f"👤 New player: {addr}")

            thread = threading.Thread(target=self.handle_client, args=(client_socket,))
            thread.daemon = True
            thread.start()

    except KeyboardInterrupt:
        print("\n🛑 Shutting down...")
    finally:
        server.close()

```

```
if __name__ == "__main__":  
    proxy = WinningProxy(30037, "jetski-455wf7wv.alfactf.ru", 30036)  
    proxy.start()
```