

# aflactf Неуправляемый мотоцикл

Были найдены пакеты, нешифрованные, которые посыпает сервер, и пакеты, которые посыпает 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	54 46920 → 30036 [ACK] Seq=38 Ack=3673 Win=65535 Len=0
17 0.765932305 10.0.2.31 46.62.129.129 TCP 69 46920 → 30036 [PSH, ACK] Seq=38 Ack=3673 Win=65535 Len=15	60 30036 → 46920 [ACK] Seq=3673 Ack=53 Win=65535 Len=0
18 1.028247123 10.0.2.31 46.62.129.129 TCP 81 30036 → 46920 [PSH, ACK] Seq=3673 Ack=53 Win=65535 Len=27	54 46920 → 30036 [ACK] Seq=53 Ack=3700 Win=65535 Len=0
19 1.028787793 10.0.2.31 46.62.129.129 TCP 69 46920 → 30036 [PSH, ACK] Seq=53 Ack=3700 Win=65535 Len=15	60 30036 → 46920 [ACK] Seq=3700 Ack=68 Win=65535 Len=0
20 1.106600041 10.0.2.31 46.62.129.129 TCP 81 30036 → 46920 [PSH, ACK] Seq=3673 Ack=53 Win=65535 Len=27	60 30036 → 46920 [ACK] Seq=53 Ack=3700 Win=65535 Len=15
21 1.106653877 10.0.2.31 46.62.129.129 TCP 54 46920 → 30036 [ACK] Seq=53 Ack=3700 Win=65535 Len=0	69 46920 → 30036 [PSH, ACK] Seq=53 Ack=3700 Win=65535 Len=15
22 1.362009077 10.0.2.31 46.62.129.129 TCP 60 30036 → 46920 [ACK] Seq=3700 Ack=68 Win=65535 Len=0	81 30036 → 46920 [PSH, ACK] Seq=3700 Ack=68 Win=65535 Len=27
23 1.362594312 10.0.2.31 46.62.129.129 TCP 69 46920 → 30036 [PSH, ACK] Seq=3700 Ack=68 Win=65535 Len=0	54 46920 → 30036 [ACK] Seq=68 Ack=3700 Win=65535 Len=27
24 1.451478933 10.0.2.31 46.62.129.129 TCP 81 30036 → 46920 [PSH, ACK] Seq=3700 Ack=68 Win=65535 Len=27	60 30036 → 46920 [ACK] Seq=68 Ack=3700 Win=65535 Len=0
25 1.451478933 10.0.2.31 46.62.129.129 TCP 54 46920 → 30036 [ACK] Seq=68 Ack=3700 Win=65535 Len=0	69 46920 → 30036 [PSH, ACK] Seq=3700 Ack=68 Win=65535 Len=27
... 16 bytes	0000 08 00 27 d1 f8 5d 52 55 00 00 02 02 08 00 45 00 ... ]RU ..... E- .C-@.>...>...P- .uT HsD .?...P- .A- lo l{7,21} kek
30 2.087693859 10.0.2.31 46.62.129.129 TCP 78 46920 → 30036 [PSH, ACK] Seq=83 Ack=3754 Win=65535 Len=16	0010 00 43 f2 fd 00 00 40 06 cb 9b 2e 3e 81 81 0a 00 .C-@.>...>...P- .uT HsD .?...P- .A- lo l{7,21} kek
31 2.088272639 46.62.129.129 10.0.2.31 TCP 60 30036 → 46920 [ACK] Seq=3754 Ack=99 Win=65535 Len=0	0020 02 1f 75 54 b7 48 73 44 d8 3f fb 94 fb fa 50 18 .A- lo l{7,21} kek
32 2.386776903 46.62.129.129 10.0.2.31 TCP 81 30036 → 46920 [PSH, ACK] Seq=3754 Ack=99 Win=65535 Len=27	0030 ff ff 41 1b 00 00 6c 6f 6c 7b 37 2c 32 31 7d 7c .A- lo l{7,21} kek
33 2.386824421 10.0.2.31 46.62.129.129 TCP 54 46920 → 30036 [ACK] Seq=99 Ack=3781 Win=65535 Len=0	0040 34 37 32 7c 57 75 41 6c 53 64 44 72 7c 6b 65 6b .A- lo l{7,21} kek
34 2.517458476 10.0.2.31 46.62.129.129 TCP 68 46920 → 30036 [PSH, ACK] Seq=99 Ack=3781 Win=65535 Len=14	0050 0a .A- lo l{7,21} kek
35 2.521686604 46.62.129.129 10.0.2.31 TCP 66 30036 → 46920 [ACK] Seq=3781 Ack=113 Win=65535 Len=0	... 16 bytes
36 2.703139678 46.62.129.129 10.0.2.31 TCP 81 30036 → 46920 [PSH, ACK] Seq=3781 Ack=113 Win=65535 Len=27	0000 52 55 0a 00 02 02 08 00 27 d1 f8 5d 00 00 45 00 RU ..... '...].E- .0@.U...>...P- .HuT 'sD P- .A- lo l{7,21} kek
37 2.703139678 10.0.2.31 46.62.129.129 TCP 54 46920 → 30036 [ACK] Seq=113 Ack=3781 Win=65535 Len=0	0010 00 38 e2 8c 40 00 40 06 9c 55 0a 00 02 1f 2e 3e .8 @.0@.U...>...P- .0@.U...>...P- .HuT 'sD P- .A- lo l{7,21} kek
38 2.703139678 10.0.2.31 46.62.129.129 TCP 0020 81 81 b7 48 75 54 fb 94 fc 27 73 44 d8 ab 50 18 .A- lo l{7,21} kek	0030 ff ff bc 08 00 00 6c 6f 6c 5b 2b 2b 2b 61 2b .A- lo l{7,21} kek
39 2.703139678 10.0.2.31 46.62.129.129 TCP 0040 2b 5d 6b 65 6b 0a .A- lo l{7,21} kek	... 16 bytes

Был найден участок кода методом тыка, с посланием сервером пакетов с форматом

lol[|999|WuAlSdDr|]kek

где 999 - кол-во звёзд

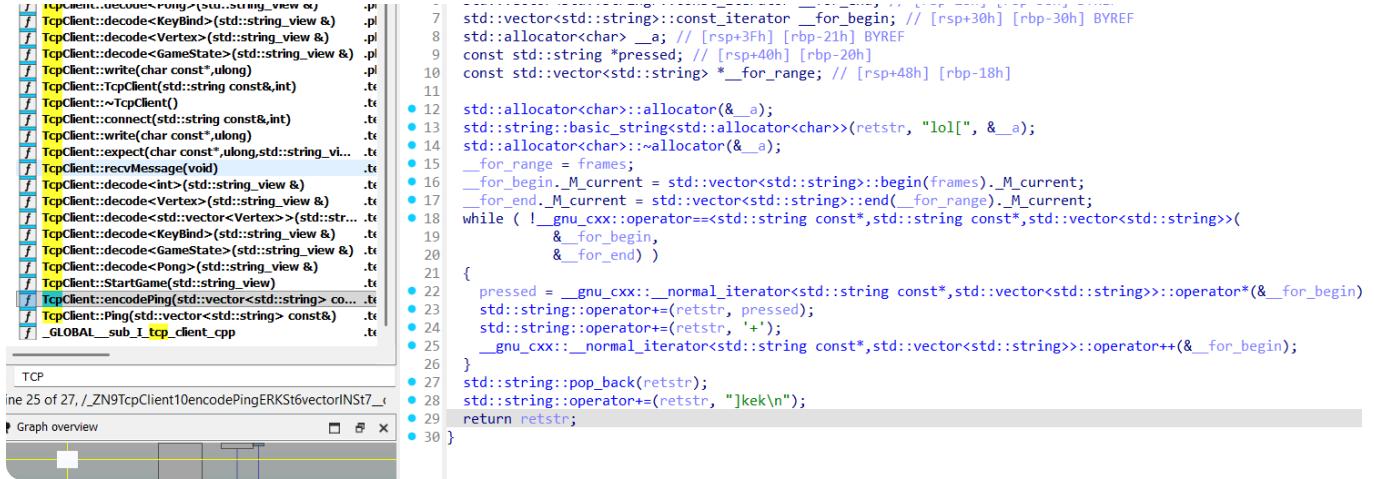
```
f TcpClient::write(char const*, ulong)          .t
f TcpClient::expect(char const*, ulong, std::string_view...) .t
f TcpClient::recvMessage(void)                  .t
f TcpClient::decode<int>(std::string_view &)    .t
f TcpClient::decode<Vertex>(std::string_view &) .t
f TcpClient::decode<std::vector<Vertex>>(std::string_view &) .t
f TcpClient::decode<KeyBind>(std::string_view &) .t
f TcpClient::decode<GameState>(std::string_view &) .t
f TcpClient::decode<Pong>(std::string_view &)   .t
f TcpClient::StartGame(std::string_view)         .t
f TcpClient::encodePing(std::vector<std::string> const&) .t
f TcpClient::Ping(std::vector<std::string> const&) .t
f _GLOBAL__sub_I_tcp_client.cpp                 .t

TCP
ne 17 of 27, /_ZN9TcpClient11recvMessageB5cxx11Ev
Graph overview
  
```

```
closed_error::closed_error(exception, &str);
std::string::~string(&str);
std::allocator<char>::~allocator(&__a);
__cxa_throw(
    exception,
    (struct type_info *)&`typeinfo for'closed_error,
    (void (*) (void *))closed_error::~closed_error);
}
std::string_view::basic_string_view(&expect, "lol");
TcpClient::expect(this, buf, n, expect);
do
{
    n = c_recv(this->m_sockfd, buf, 4096u);
    std::string::append(retstr, buf, n);
}
while ( !std::string::ends_with(retstr, "kek\n") );
v3 = std::string::size(retstr) - 4;
std::string::resize(retstr, v3);
return retstr;
}
  
```

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

Клиент же посыпал пакеты вида lol[w+a+s+d+++]kek, где wasd - нажатые клавиши



```
1 TcpClient::decodePong(std::string_view &)
2 TcpClient::decode<KeyBind>(std::string_view &)
3 TcpClient::decode<Vertex>(std::string_view &)
4 TcpClient::decode<GameState>(std::string_view &)
5 TcpClient::write(char const*, ulong)
6 TcpClient::TcpClient(std::string const&, int)
7 TcpClient::~TcpClient()
8 TcpClient::connect(std::string const&, int)
9 TcpClient::write(char const*, ulong)
10 TcpClient::expect(char const*, ulong, std::string_view &)
11 TcpClient::recvMessage(void)
12 TcpClient::decode<int>(std::string_view &)
13 TcpClient::decode<Vertex>(std::string_view &)
14 TcpClient::decode<Vertex>(std::string_view &)
15 TcpClient::decode<GameState>(std::string_view &)
16 TcpClient::decode<Pong>(std::string_view &)
17 TcpClient::StartGame(std::string_view)
18 TcpClient::encodePing(std::vector<std::string> const&)
19 TcpClient::Ping(std::vector<std::string> const&)
20 _GLOBAL_sub_I_tcp_client.cpp
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 const 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
31 __for_range = frames;
32 __for_begin._M_current = std::vector<std::string>::begin(frames)._M_current;
33 __for_end._M_current = std::vector<std::string>::end(__for_range)._M_current;
34 while ( __gnu_cxx::operator==(<std::string const*, std::string const*, std::vector<std::string>>(
35     &__for_begin,
36     &__for_end ) )
37 {
38     pressed = __gnu_cxx::__normal_iterator<std::string const*, std::vector<std::string>>::operator*&(&__for_begin)
39     std::string::operator+=(retstr, pressed);
40     std::string::operator+=(retstr, '+');
41     __gnu_cxx::__normal_iterator<std::string const*, std::vector<std::string>>::operator++(&__for_begin);
42 }
43 std::string::pop_back(retstr);
44 std::string::operator+=(retstr, "]kek\n");
45
46 return retstr;
47
48 }
```

Вот такими темпами было решено поднять прокси сервер, который принимал пакеты сервера, редактил их на адекватное WuAlSdDr, чтобы клиент по человечески это дело воспринимал, и редактил пакеты клиента, так, чтобы посыпалась та кнопка, которая по мнению сервера отвечает за нужное движение. Например послано WrAdSrDu, клиент движется по человечески вверх, хочет послать w, но это дело ловим, сопоставляем, что чтобы идти вверх, нужно нажать d, заменяем в пакете w на d.

```
★ 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

Навайбкожено, конечно, но логику подмены пришлось самому делать

```
#!/usr/bin/env python3

import socket
import threading
import select
import re
import time
```

```
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"

    # Логируем 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 логируем 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("🚀 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()
```