МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
**«МИРЭА - Российский технологический университет»**
**РТУ МИРЭА**

Институт искусственного интеллекта
Кафедра информационной безопасности

# КУРСОВАЯ РАБОТА

по дисциплине
«Криптографические методы защиты информации»

Тема курсовой работы
**«Анализ криптографического протокола обмена сообщениями клиент-серверной архитектуры в программе Minecraft»**

Студенты группы ККСО-01-19: Плотников А.Е. _____

Работа представлена к защите «__» _____ 2023 г.

Допущен к защите «__» _____ 2023 г.

Москва 2023

# ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ ..... 3

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ ..... 4

Описание протокола ..... 4

Анализ стойкости протокола ..... 11

ПРАКТИЧЕСКАЯ ЧАСТЬ ..... 14

Демонстрация атаки на протокол ..... 14

Оценка опасности уязвимости ..... 17

ЗАКЛЮЧЕНИЕ ..... 20

СПИСОК ЛИТЕРАТУРЫ ..... 21

# ВВЕДЕНИЕ

Криптография – это наука о методах построения шифров, т.е. методах преобразования исходной информации в форму, недоступную для понимания противником [1]. Развитие криптографии и её методов привело к их широчайшей распространенности. Средства криптографии используются для контроля целостности сообщений, механизмов идентификации и аутентификации пользователей в сети, цифровой подписи сообщений и многих других задач современного цифрового мира.

Так, при передаче данных по сети, а также в мессенджерах, весь трафик шифруется. Для получения доступа к сайту может потребоваться пароль, обработка которого осуществляется с помощью криптографических средств. Устройства интернета вещей передают данные в зашифрованном виде. Группа беспилотных летательных объектов опознает друг друга по протоколу «Свой-чужой» с использованием криптографии. Все это происходит незаметно для пользователя. Мы даже не замечаем, как данные преобразуются, а затем восстанавливаются. Но без этого они были бы доступны злоумышленникам, а значит уязвимы.

В данной курсовой работе проводится анализ криптографического протокола обмена сообщениями клиент-серверной архитектуры в программе Minecraft. Minecraft – это компьютерная инди-игра в жанре песочницы, выпущенная студией Mojang AB. По данным за 2023 год является самой продаваемой игрой в истории, а количество игроков, запускавших игру хотя бы раз в месяц составило 172 миллиона [2].

Целями курсовой работы являются описание протокола, его алгоритмов защиты, анализ их стойкости, а также реализация возможной атаки на данный протокол и оценка ее опасности.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## ОПИСАНИЕ ПРОТОКОЛА

Перед подключением к серверу клиент проходит процедуру авторизации. Действующая система для подтверждения лицензии и доступа пользователя носит название Yggdrasil. Запросы отправляются на сервер https://authserver.mojang.com [3].

Клиент передает полезную нагрузку в JSON-формате:

```
{
    "agent": {
        "name": "Minecraft",
        "version": 1

    },
    "username": "mojang account name",
    "password": "mojang account password",
    "clientToken": "client identifier",
    "requestUser": true
}
```

Приведем краткое описание каждого поля. Значение ключа “agent”, как правило, не изменяется и используется для Minecraft по умолчанию. Логин (или e-mail) и пароль клиента передаются как “username” и “password” соответственно.

Поле “clientToken” содержит сформированный случайным образом уникальный токен пользователя. Как правило, это происходит при первом запуске, а значение токена сохраняется для каждого следующего запроса.

Если “requestUser” передается как true, то в ответе сервера будет добавлено поле “user” [3].

В качестве ответа поступает сообщение следующего вида:

```
{
    "user": {
        "username": "user@email.example",
        "properties": [
            {
                "name": "preferredLanguage",
                "value": "en-us"
            },
            {
                "name": "registrationCountry",
```

```
                "value": "country"
            }
        ],
        "id": "hexadecimal string"
    },
    "clientToken": "client identifier",
    "accessToken": "random access token",
    "availableProfiles": [
        {
            "name": "player username",
            "id": "hexadecimal string"
        }
    ],
    "selectedProfile": {
        "name": "player username",
        "id": "hexadecimal string"
    }
}
```

В следующей таблице приведено описание каждого поля (Таблица 1):

Таблица 1 – Описание полей из ответа сервера

| Поле | Значение |
|---|---|
| “user” | E-mail пользователя<br>Язык<br>Страна |
| “clientToken” | Токен, полученный от клиента |
| “accessToken” | Уникальный ключ, сформированный сервером при авторизации |
| “availableProfiles” | Профили, доступные пользователю |
| “selectedProfiles” | Выбранный профиль пользователя |

Соединение сервера Minecraft с клиентом осуществляется по протоколу TCP, а данные передаются при помощи пакетов. Содержимое пакета зависит от двух параметров: его идентификатора и текущего состояния соединения. У протокола четыре состояния: “Рукопожатие” (Handshake), “Статус” (Status), “Вход в игру” (Login) и “Игра” (Play) [4].

На следующем рисунке изображены все этапы, которые происходят при попытке подключения клиента к серверу (Рисунок 1).

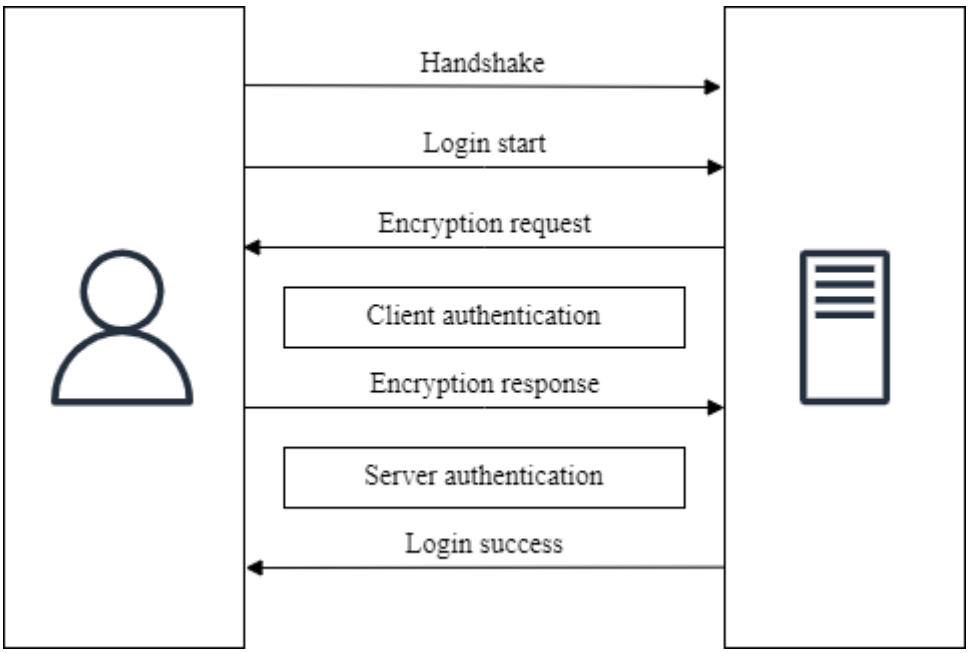


Рисунок 1 – Основные этапы протокола

Перед тем, как перейти к разбору каждого этапа, необходимо привести описание некоторых используемых в протоколе типов данных (Таблица 2) [5].

Таблица 2 – Используемые типы данных

| **Тип данных** | **Размер в байтах** | **Диапазон значений** | **Замечание** |
|---|---|---|---|
| Boolean | 1 | true или false | true кодируется как 0x01<br>false кодируется как 0x02 |
| Unsigned Short | 2 | Целое число от 0 до 65535 | Целое 16-битное число без знака |
| String (n) | $\geq 1$<br>$\leq (n \cdot 4) + 3$ | Строка в кодировке UTF-8 | Максимальная длина составляет 32767, зависит от контекста |
| Optional String (n) | 0 или<br>$\geq 1$<br>$\leq (n \cdot 4) + 3$ | Пустое значение, либо строка в кодировке UTF-8 | Наличие данного поля зависит от контекста |
| VarInt | $\geq 1$<br>$\leq 5$ | Целое число между 2147483648 и 2147483647 | Данные переменной длины, целое 32-битное число со знаком |
| VarInt Enum | $\geq 1$<br>$\leq 5$ | Конкретное значение из списка | Список возможных значений типа VarInt должен быть известен из контекста |
| UUID | 16 | Универсальный уникальный идентификатор | 128-разрядное целое число без знака |
| Optional UUID | 0 или 16 | Пустое поле или UUID | Наличие данного поля зависит от контекста |
| Array of X (n) | $\geq 0$<br>$< n \cdot size(X)$ | Ноль или более полей типа X | Размер должен быть известен из контекста |
| Byte | Размер | Зависит от контекста | Последовательность из нуля или |

| | | | |
|---|---|---|---|
| Array | является переменным | | более байт. Ее значение и длина должны быть известны из описания пакета |

Теперь разберем каждый этап подключения клиента к серверу. На стадии рукопожатия клиент отправляет серверу пакет с версией протокола, адресом и портом сервера, а также специальным номером, определяющим следующее состояние (Таблица 3).

Таблица 3 - Handshake

| ID пакета | Состояние | Получатель | Название поля | Тип поля | Замечание |
|---|---|---|---|---|---|
| 0x00 | Handshaking | Сервер | Версия протокола | VarInt | По умолчанию равно 762 |
| | | | Адрес сервера | String (255) | Имя хоста или IP-адрес, использующийся для подключения |
| | | | Порт сервера | Unsigned Short | По умолчанию равно 25565 |
| | | | Следующее состояние | VarInt Enum | По умолчанию равно 2 |

Сразу после рукопожатия клиент передает свои имя и, при наличии, уникальный идентификатор (Таблица 4).

Таблица 4 – Login Start

| ID пакета | Состояние | Получатель | Название поля | Тип поля | Замечание |
|---|---|---|---|---|---|
| 0x00 | Login | Сервер | Имя клиента | String (16) | Имя пользователя |
| | | | Наличие UUID у клиента | Boolean | Значение данного поля определяет, следует ли отправлять следующее поле |
| | | | UUID клиента | Optional UUID | Уникальный ID пользователя, вошедшего в систему |

Получив два предыдущих пакета, сервер вырабатывает пару 1024-битных ключей в кодировке DER, используя алгоритм RSA, и отправляет открытый ключ клиенту. Вместе с ним сервер передает свой ID и токен проверки [6].

В предыдущих версиях протокола ID сервера представляло собой случайно сформированную строку символов, на текущий момент — это пустая строка (Таблица 5).

Таблица 5 – Encryption request

| ID пакета | Состояние | Получатель | Название поля | Тип поля | Замечание |
|---|---|---|---|---|---|
| 0x01 | Login | Клиент | ID сервера | String (20) | Остается пустым |
| | | | Длина открытого ключа | VarInt | Длина открытого ключа |
| | | | Открытый ключ | Byte Array | Открытый ключ сервера в байтах |
| | | | Длина токена проверки | VarInt | Длина токена проверки. Всегда равно 4 |
| | | | Токен проверки | Byte Array | Последовательность случайных байт, сформированных сервером |

Следующим этапом является аутентификация клиента на сервере сессий. Клиент вырабатывает случайный 16-байтовый общий секрет – данные, которые будут известны только двум участвующим сторонам и использоваться далее в шифровании. Затем происходит вычисление хэш-функции с помощью алгоритма SHA-1:

```
sha1 := Sha1()
sha1.update(Server ID)
sha1.update(shared secret)
sha1.update(server's encoded public key from Encryption Request)
hash := sha1.hexdigest()  # String of hex characters
```

Использование трех методов update() фактически означает вычисление хэша от конкатенации переданных в качестве параметров строк:

$$sha1(\text{Server ID} + \text{shared secret} + \text{server's encoded public key})$$

В первой строке содержится ID сервера, “shared secret” – это сформированный ранее общий секрет, а третьим значением является открытый ключ сервера.

Далее вычисленный хэш с использованием POST-запроса отправляется на сервер https://sessionserver.mojang.com/session/minecraft/join в JSON-формате:

```
{
    "accessToken": "<accessToken>",
    "selectedProfile": "<player's uuid without dashes>",
    "serverId": "<serverHash>"
}
```

Значения “accessToken” и UUID клиент получает во время аутентификации.

Если у клиента включен режим мультиплеера, а также нет блокировки аккаунта, то этап аутентификации на сервере сессий для клиента завершается успешно [6].

Далее серверу отправляется пакет с общим секретом и токеном проверки. Предварительно клиент шифрует указанные значения при помощи открытого ключа сервера, в итоге длина массивов байтов составляет 128 байт (Таблица 6).

Таблица 6 – Encryption response

| ID пакета | Состояние | Получатель | Название поля | Тип поля | Замечание |
|---|---|---|---|---|---|
| 0x01 | Login | Сервер | Длина общего секрета | VarInt | Длина общего секрета |
| | | | Общий секрет | Byte Array | Общий секрет, зашифрованный открытым ключом сервера |
| | | | Длина токена проверки | VarInt | Длина токена проверки |
| | | | Токен проверки | Byte Array | Токен проверки, зашифрованный открытым ключом сервера |

При получении пакета сервер, используя свой закрытый ключ, расшифровывает общий секрет и токен проверки. Для проверки корректности токена вычисляется хэш-функция аналогично тому, как это делал клиент. Полученное значение с помощью GET-запроса отправляется на сервер сессий:

https://sessionserver.mojang.com/session/minecraft/hasJoined?username=username&serverId=hash&ip=ip

Поле “username” должно совпадать с именем пользователя, “ip” содержит IP-адрес подключающегося клиента, изначально инициирующего запрос сеанса, но является необязательным.

Ответ на запрос имеет следующий вид:

```
{
    "id": "<profile identifier>",
    "name": "<player name>",
    "properties": [
        {
            "name": "textures",
            "value": "<base64 string>",
            "signature": "<base64 string; signed data using
Yggdrasil's private key>"
        }
    ]
```

```
}
```

Идентификатор профиля имеет такой формат:

"AAAAAAAABBBBCCCCDDDDEEEEEEEEEEEEEE"

Перед отправкой его обратно клиенту происходит преобразование в формат

"AAAAAAAA-BBBB-CCCC-DDDD-EEEEEEEEEEEEEE"

Далее сервер уведомляет клиента об успешном входе в систему и включает шифрование AES/CFB8 (Таблица 7). В качестве вектора инициализации IV и ключа используется выработанный ранее общий секрет. С этого момента все пакеты будут полностью зашифрованы [5].

Таблица 7 – Login success

| **ID пакета** | **Состояние** | **Получатель** | **Название поля** | | **Тип поля** | |
|---|---|---|---|---|---|---|
| 0x02 | Login | Клиент | ID клиента | | UUID | |
| | | | Имя пользователя | | String (16) | |
| | | | Число характеристик | | VarInt | |
| | | | Характеристика | Имя | Array | String (32767) |
| | | | | Значение | | String (32767) |
| | | | | Наличие подписи | | Boolean |
| | | | | Подпись | | Optional String (32767) |

Стоит отметить, что пакет успешного входа в систему также отправляется в зашифрованном виде, а шифр AES обновляется непрерывно, а не запускается заново при каждом новом пакете.

## АНАЛИЗ СТОЙКОСТИ ПРОТОКОЛА

При успешном прохождении авторизации и аутентификации клиента пакеты, передаваемые в процессе взаимодействия с сервером, шифруются с помощью алгоритма AES в режиме CFB8. Рассмотрим более подробно, как работает данный режим.

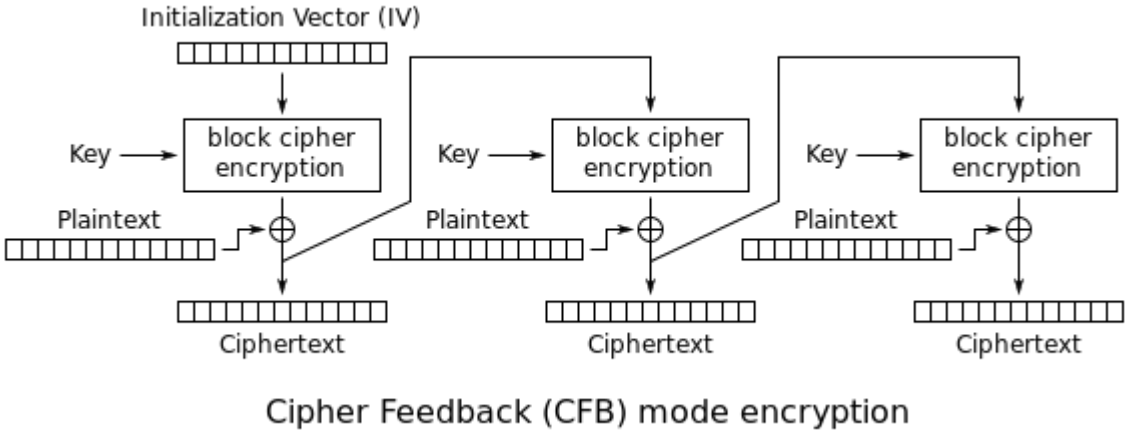


Рисунок 2 – Шифрование в режиме CFB

В режиме гаммирования с обратной связью по шифртексту (The Cipher Feedback Mode, CFB) каждый блок открытого текста складывается по модулю 2 с блоком, зашифрованным на предыдущем шаге (Рисунок 2). Начальное состояние шифра инициализируется значением синхропосылки IV [1].

$$C_0 = IV$$

$$C_i = E_k(C_{i-1}, k) \oplus P_i$$

Здесь $C_i$ – блок шифртекста ($i \geq 1$), $P_i$ – открытый текст, $k$ – ключ, $E_k$ – функция шифрования [7].

Так как операция сложения по модулю 2 обратима, то процедура расшифрования в режиме CFB аналогична процедуре шифрования (Рисунок 3).

$$C_0 = IV$$

$$P_i = E_k(C_{i-1}, k) \oplus C_i$$

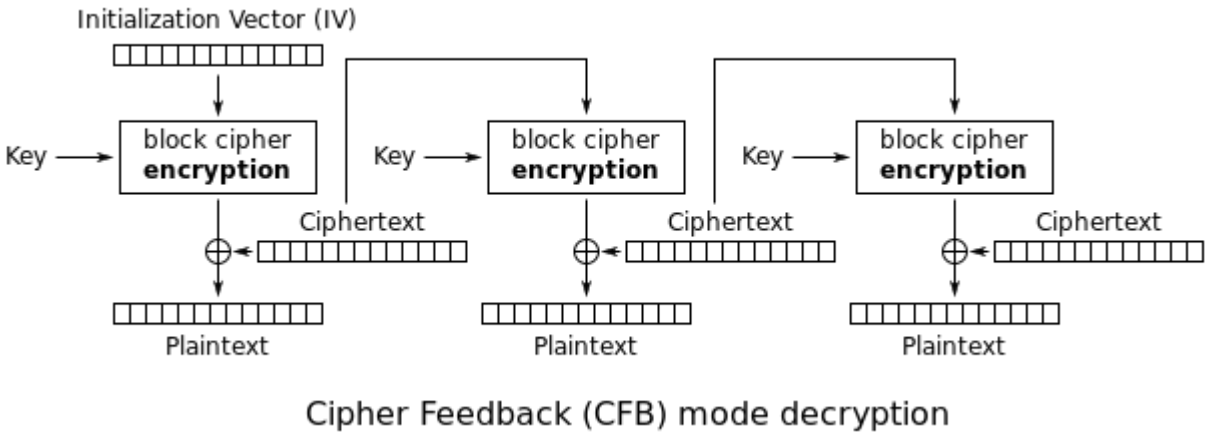


Рисунок 3 – Расшифрование в режиме CFB

Из последнего рисунка видно, что блок открытого текста получается путем сложения по модулю 2 блока шифртекста и блока, полученного после AES преобразований.

Теперь рассмотрим структуру пакета, который передается серверу при отправке пользователем сообщения из терминала (Таблица 8). Клиент передает всего одно поле – строку, которая представлена в исходном виде, то есть без какой-либо обработки.

Таблица 8 – Chat Message

| ID пакета | Состояние | Получатель | Название поля | Тип поля | Замечание |
|---|---|---|---|---|---|
| 0x03 | Play | Сервер | Message | String (256) | Клиент отправляет необрабатываемый ввод |

Из приведенных выше сведений следует, что злоумышленник, способный манипулировать закрытым текстом, также получает возможность оказывать влияние на открытый текст.

Предположим, что пользователь передает сообщение, содержащее символ “n”. Значит, после сложения по модулю 2 блока шифртекста и блока, полученного после AES преобразований, в открытом тексте будет присутствовать символ “n”. Зная это, злоумышленник может специально изменить шифртекст так, как ему необходимо.

Например, если злоумышленнику требуется заменить символ “n” (0x6e) на “x” (0x78), то нужно заменить соответствующий байт шифртекста на 0x16:

$$0x6e\ (\text{n}) \oplus 0x16 = 0x78\ (\text{x})$$

Злоумышленник может совершить приведенные действия для атаки по известному открытому тексту, используя вредоносный сервер, либо выступая в роли активного сетевого злоумышленника, то есть реализуя атаку “человек-посередине” (MITM).

Опишем примерную идею атаки:

1. Злоумышленник создает учетную запись с именем, сформированным таким образом, чтобы оно было похожим на имя администратора известного сервера. Например, имя администратора - Admin, а имя злоумышленника – Admix, то есть отличие состоит в одной букве.
2. Требуется, чтобы настоящий администратор подключился к вредоносному серверу злоумышленника, который представляет собой простой proxy, перенаправляющий все данные на реальный сервер.
3. Администратор должен выполнить команду */op Admin*, предоставляющую права администратора пользователю Admin.
4. Proxy модифицирует последний полученный байт таким образом, чтобы реальный сервер получил команду */op Admix*.
5. Злоумышленник получает права администратора на реальном сервере.

Также следует привести некоторые замечания:

- Необходимо создать такие условия, при которых настоящий администратор подключится к вредоносному серверу, например, применив методы социальной инженерии;
- Настоящий администратор должен выполнить строго определенные действия для дальнейшей деятельности злоумышленника;
- После того, как злоумышленник изменил байт в сообщении, режим CFB будет вырабатывать неправильные данные, что приведет к отключению злоумышленника от сервера;
- При наиболее корректной реализации proxy должен минимизировать число случаев ложного срабатывания обнаружения искомого пакета.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## ДЕМОНСТРАЦИЯ АТАКИ НА ПРОТОКОЛ

Для проведения атаки создан демонстрационный стенд, в составе которого входит одна виртуальная машина, функционирующая на операционной системе Ubuntu 20.04 [8]. В ходе эксплуатации уязвимости используется следующее ПО:

- Minecraft 1.18.1 (дата выхода: 10 декабря 2021 г.);
- Paper - игровой сервер Minecraft [9];
- simple-tcp-proxy - прокси сервер [10].

Приведем последовательность действий, необходимых для проведения атаки:

1. Выполняется запуск Minecraft и игрового сервера Paper (Рисунок 4):

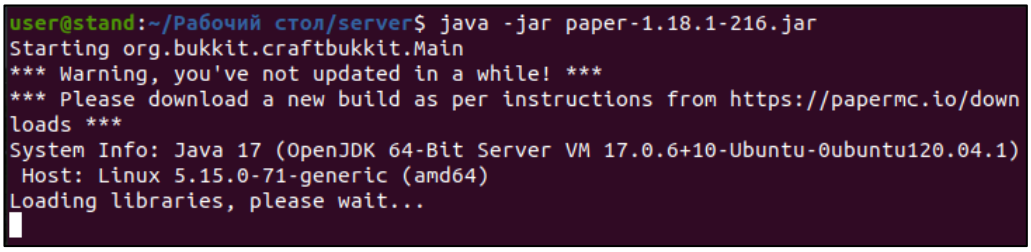


Рисунок 4 – Запуск сервера

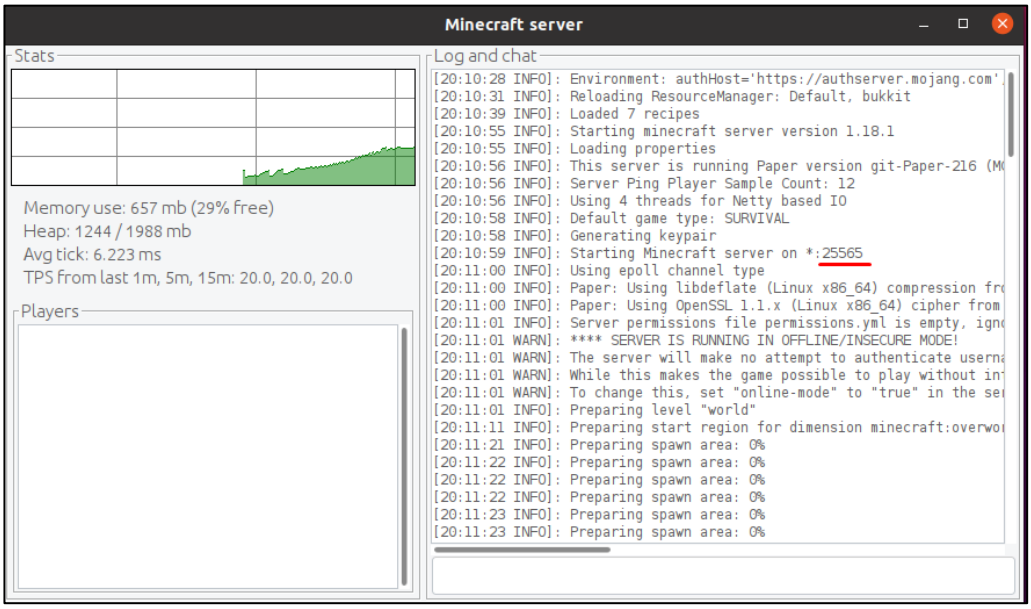


Рисунок 5 - Меню сервера

2. Для реализации подмены последнего байта в исходном коде proxy добавляется несколько следующих строк (Рисунок 6):

```
220                 FD_SET(sfd, &R);
221         to.tv_sec = 0;
222         to.tv_usec = 1000;
223         x = select(maxfd+1, &R, 0, 0, &to);
224         if (x > 0) {
225             if (FD_ISSET(cfd, &R)) {
226                 n = read(cfd, cbuf+cbo, BUF_SIZE-cbo);
227                 printf("read %d bytes from CLIENT (%d)\n", n, cfd);
228                 if(n==MESSAGE_LEN) {
229                         printf("CHAT MESSAGE DETECTED!\n");
230                         printf("encrypted chat: %s\n", (cbuf+cbo));
231                         (cbuf+cbo)[MESSAGE_LEN - 1] ^= '\x16';
232                 }
233                 syslog(LOG_INFO, "read %d bytes from CLIENT (%d)", n, cfd);
234                 if (n > 0) {
235                     cbo += n;
236                 } else {
237                     close(cfd);
238                     close(sfd);
239                     syslog(LOG_INFO, "exiting");
240                     _exit(0);
```

Рисунок 6 – Добавление функционала в proxy

Данный код проверяет размер полученного от клиента пакета и, если он равен известной заранее длине сообщения (константа MESSAGE_LEN), в консоль выводится информация об обнаружении получения требуемой строки. После этого происходит изменение последнего байта полученного сообщения.

3. Запуск proxy производится следующим образом:

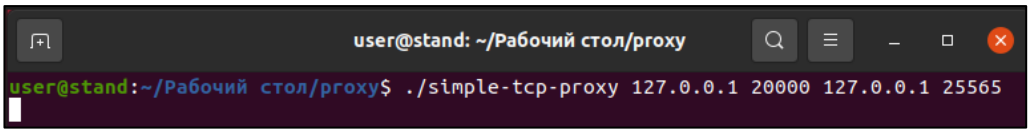


Рисунок 7 – Запуск proxy

4. После запуска всех программ нужно открыть Minecraft и подключиться к нашему proxy. Далее в терминале игры вводим команду */op Admin* (Рисунок 8):

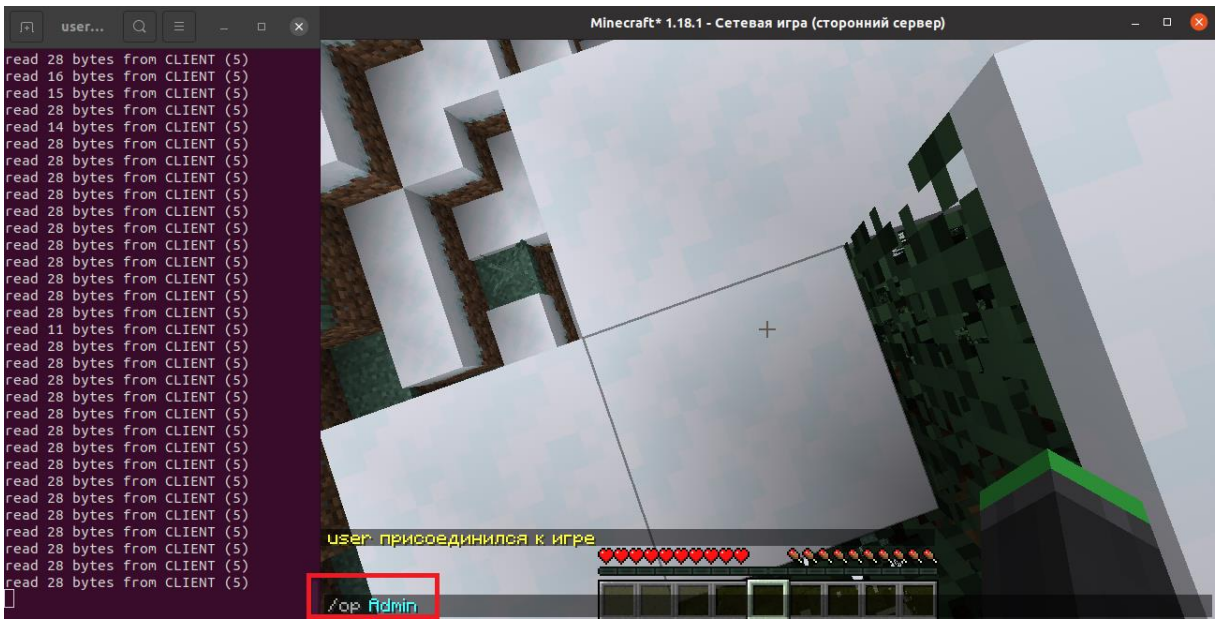


Рисунок 8 - Выдача прав администратора пользователю Admin

После отправки команды права администратора достаются пользователю с именем Admix (Рисунок 9). Как видно из информации, предоставленной proxy, нужное сообщение было найдено, после чего специальным образом преобразовано и отправлено на настоящий сервер, где произошла передача прав администратора пользователю с другим именем.

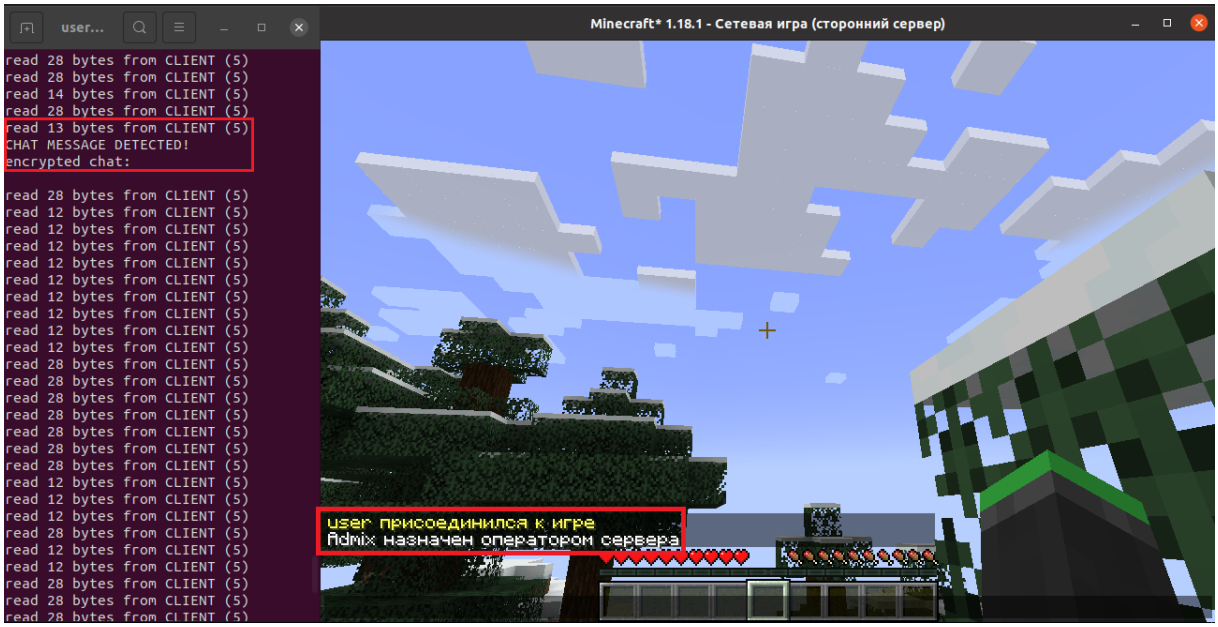


Рисунок 9 – Получение прав администратора пользователем Admix

## ОЦЕНКА ОПАСНОСТИ УЯЗВИМОСТИ

Эксплуатация уязвимостей может иметь разные последствия. Одним из подходов ранжирования, а также приоритезации обработки и устранения уязвимостей, является оценка их опасности. На сегодняшний день стандартом считается система оценок Common Vulnerability Scoring System (CVSS) [11].

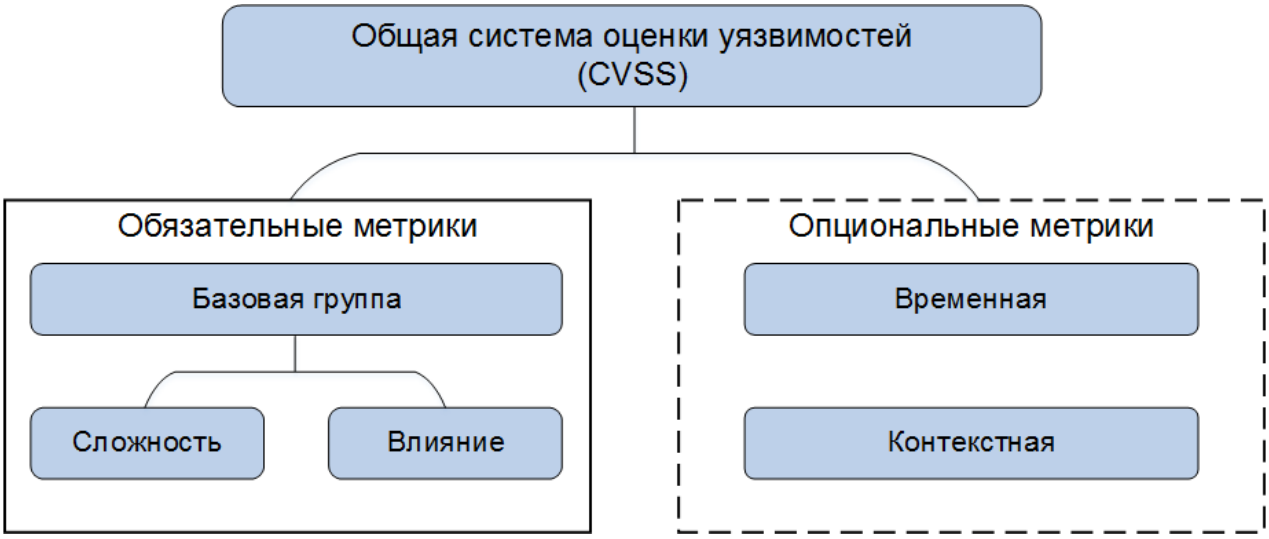


Рисунок 10 – Общая система оценки уязвимостей CVSS

Проведем оценку описанной уязвимости по системе CVSS v3.1 [12]. Она производится на основе трех метрик: базовой, временной и контекстной. С помощью группы базовых метрик описывают характеристики уязвимости, не меняющиеся с течением времени и не зависящие от среды исполнения, а также оценивается сложность эксплуатации уязвимости и потенциальный ущерб для конфиденциальности, целостности и доступности информации [13].

Временные и контекстные метрики опциональны и применяются для более точной оценки опасности, которую представляет данная уязвимость, поэтому в данной работе проведена оценка только базовых метрик.

В стандарте системы CVSS v3.1 указаны следующие метрики:

- Вектор атаки (Access Vector - AV) - критерий степени удаленности нарушителя от уязвимого объекта;
- Сложность атаки (Attack Complexity - AC) – критерий сложности получения доступа нарушителем;

- Уровень привилегий (Privileges Required - PR) – критерий требуемого уровня привилегий;
- Взаимодействие с пользователем (User Interaction - UI) – критерий необходимости взаимодействия с пользователем;
- Граница эксплуатации (Scope - S) – критерий влияния на другие компоненты системы;
- Конфиденциальность (Confidentiality - C) – критерий влияния на конфиденциальность;
- Целостность (Integrity - I) – критерий влияния на целостность;
- Доступность (Availability - A) – критерий влияния на доступность.

Каждая метрика имеет свое множество возможных значений. Рассмотрим каждое из них, одновременно выбирая значения для описанной уязвимости и обосновывая их (Таблица 9).

Таблица 9 – Список значений базовых метрик

| Название метрики | Возможные значения метрики | Значение метрики описанной уязвимости | Замечание |
|---|---|---|---|
| AV | Network (N)<br>Adjacent Network (A)<br>Local (L)<br>Physical (P) | Network (N) | Злоумышленник является посредником между клиентом и сервером |
| AC | High (H)<br>Low (L) | High (H) | Успешная атака зависит от условий, не контролируемых злоумышленником |
| PR | High (H)<br>Low (L)<br>None (N) | Low (L) | Злоумышленник должен обладать привилегиями, предоставляющими базовые возможности пользователя |
| UI | Required (R)<br>None (N) | Required (R) | Для реализации атаки администратор должен совершить действия |
| S | Changed (C)<br>Unchanged (U) | Unchanged (U) | Эксплуатируемая уязвимость не влияет на другие компоненты системы |
| C | High (H)<br>Low (L)<br>None (N) | Low (L) | Права администратора позволяют оказать влияние на конфиденциальность, целостность и доступность информации, но оно не является существенным |
| I | | Low (L) | |
| A | | Low (L) | |

Таким образом, получился следующий вектор CVSS v3.1:

AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:L

Используя значения группы базовых метрик, вычислим оценку CVSS v3.1 с помощью специального калькулятора, размещенного на сайте БДУ ФСТЭК [14].

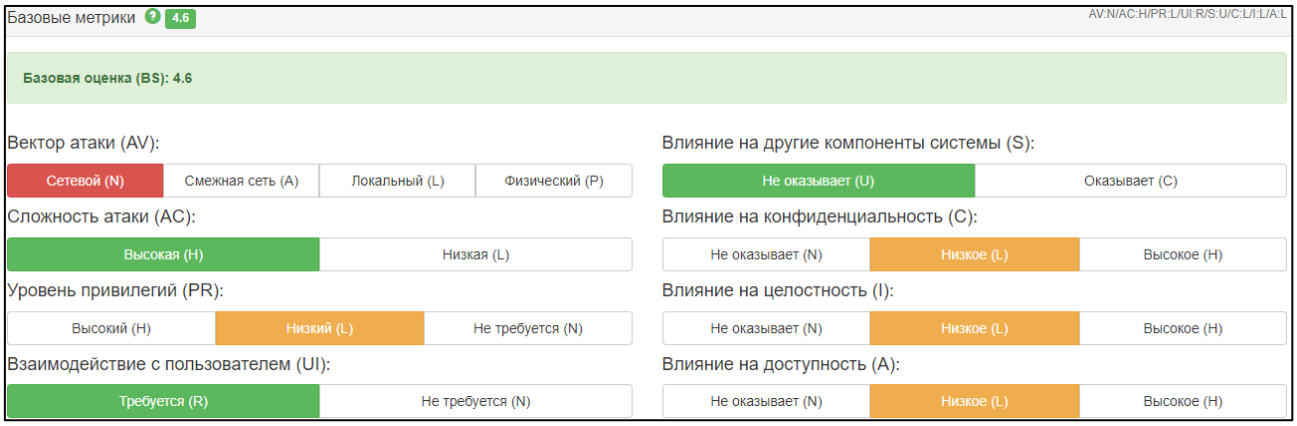


Рисунок 11 – Расчет оценки CVSS3.1

Оценка CVSS v3.1 составляет 4.6 (количественная оценка), что соответствует среднему уровню опасности (качественная оценка).

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе мы подробно рассмотрели криптографический протокол обмена сообщений клиент-серверной архитектуры в программе Minecraft, провели его анализ и нашли уязвимое место.

Для демонстрации атаки собран специальный стенд и дана инструкция, используя которые можно проэксплуатировать описанную уязвимость в протоколе.

Также мы сделали общий обзор стандарта системы оценки CVSS v3.1, с помощью которого провели оценку опасности найденной уязвимости. Полученная оценка составила 4.6, что составляет среднему уровню опасности. Идеальную систему оценки уязвимостей создать скорее всего невозможно, но использование любой версии CVSS в большинстве случаев позволяет правильно расставить приоритеты в обработке уязвимостей и оценить возможные риски.

Стоит отметить, что успешная эксплуатация описанной уязвимости зависит не только от действий злоумышленника, но и от жертвы – администратора, что значительно усложняет реализацию атаки. Также полученные в ходе эксплуатации уязвимости права имеют ограниченное влияние на конфиденциальность, целостность и доступность информации. Таким образом, приведенные факторы делают описанную атаку нецелесообразной по затратам или вообще невозможной.

# СПИСОК ЛИТЕРАТУРЫ


[1] Криптографические методы защиты информации : учебник для академического бакалавриата / А. Б. Лось, А. Ю. Нестеренко, М. И. Рожков. – 2-е изд., испр. – М.: Издательство Юрайт, 2016. – 473 с. – Серия : Бакалавр. Академический курс.

[2] Wikipedia. Minecraft. Режим доступа: https://ru.wikipedia.org/wiki/Minecraft (дата обращения: 01.04.23).

[3] Wiki.vg. Legacy Mojang Authentication. Режим доступа: https://wiki.vg/Legacy_Mojang_Authentication (дата обращения: 01.04.23).

[4] Habr. Разбор шифрования в Minecraft. Режим доступа: https://habr.com/ru/articles/718848/ (дата обращения: 01.04.23).

[5] Wiki.vg. Protocol. Режим доступа: https://wiki.vg/Protocol (дата обращения: 01.04.23).

[6] Wiki.vg. Protocol Encryption. Режим доступа: https://wiki.vg/Protocol_Encryption (дата обращения: 01.04.23).

[7] Wikipedia. Block cipher mode of operation. Режим доступа: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation (дата обращения: 01.04.23).

[8] Демонстрационный стенд. Режим доступа: https://drive.google.com/file/d/12Q-5V7gWPFrrEHhuc1JCKkHadj35wTby/view?usp=sharing

[9] Paper. Режим доступа: https://papermc.io/software/paper (дата обращения: 01.04.23).

[10] Прокси-сервер. Режим доступа: https://github.com/wessels/simple-tcp-proxy (дата обращения: 01.04.23).

[11] Safe-surf. Общий обзор системы оценки уязвимостей (CVSS 2.0/3.0). Режим доступа: https://safe-surf.ru/specialists/article/5211/596644/ (дата обращения: 01.04.23).

[12] FIRST. Common Vulnerability Scoring System v3.1: Specification Document. Режим доступа: https://www.first.org/cvss/v3.1/specification-document (дата

обращения: 01.04.23).

[13] Habr. Оценка уязвимостей CVSS 3.0. Режим доступа: https://habr.com/ru/companies/pt/articles/266485/ (дата обращения: 01.04.23).

[14] Калькулятор CVSS V3.1. Режим доступа: https://bdu.fstec.ru/calc31 (дата обращения: 01.04.23).