SSL. Кроме того, в отличие от большинства традиционных межсетевых экранов, у них есть возможность для получения будущих обновлений.

По результатам проведенного исследования были сделаны следующие выводы.

Использование технологии защищённой видеоконференцсвязи позволит избежать утечки конфиденциальной информации. Данная технология полностью импортонезависима, что позволяет получить сертификацию ФСТЭК и ФСБ.

Технология защищённой видеоконференцсвязи не требует ежемесячных платежей третьему лицу за предоставление услуг видеосвязи.

Межсетевые экраны нового поколения рекомендуется активно применять в защищенных системах видеоконференций по причине совмещения несколько базовых функций при организации защищенного канала связи таких как безопасность, учет и анализ данных.

Литература:

1. *Иконников С.Е., Ермакова А.Е., Антонов Д.А.* Оценка защищенности сети предприятия при помощи межсетевых экранов / Интеллектуальные транспортные системы: Материалы II Международной научно-практической конференции (г. Москва, 25 мая 2023 года). – Москва: РУТ (МИИТ), 2023. – С. 470-474.

2. *Чебан А.Г., Король С.А.* Особенности защиты информации в сети предприятия при помощи межсетевых экранов / Интеллектуальные транспортные системы: Материалы III Международной научно-практической конференции (г. Москва, 30 мая 2024 года). – Москва: РУТ (МИИТ), 2023. – С. 741-744.

Логинова Л.Н., Дроздов А.Д.

Анализ угроз информационной безопасности при использовании Telegram-ботов в бизнесе

Аннотация: В данной работе анализируются угрозы информационной безопасности при использовании

Telegram-ботов в бизнесе, предложены рекомендации по устранению выявленных уязвимостей.

Ключевые слова: информационная безопасность, мессенджер, уязвимости, токены, социальная инженерия

В современном мире возможности мессенджеров не уступают социальным сетям. Функционал стремительно развивается, а количество обрабатываемой информации постоянно растет. Применение подобных систем не ограничивается только личным общением. В последнее время мессенджеры пользуются большим спросом и в бизнесе, особенно, если для решения поставленных задач используется бот.

Авторы ставят своей целью провести анализ угроз информационной безопасности (ИБ) при использовании *Telegram*-ботов в бизнесе, рассмотреть основные уязвимости и предложить методы защиты.

Бизнес можно разделить на три основных направления: производство, предоставление услуг и ***деятельность по купли-продажи товаров и услуг с целью получения экономической выгоды (торговля)***. Бот, в зависимости от типа бизнеса, может быть использован для выполнения одной или сразу нескольких задач.

Перед оценкой непосредственно угроз ИБ следует определить, как инициируется взаимодействие с *Telegram*-ботом. *Telegram*-бот *API* содержит два основных обработчика сообщений: активация бота по команде или обработка сообщений, полученных от пользователей. Допускается и система фильтрации [1] по содержанию сообщений. Например, бот будет реагировать только на аудиофайлы или фотографии.

Имеется особенность и при использовании бота в групповых чатах. За доступ к сообщениям отвечает параметр *can_read_all_group_messages*. Бот сможет прочитать сообщения, если значение данного параметра *true*. Если значение выставлено на *false*, то бот не увидит сообщения от участников чата и не сможет отреагировать.

Разработчики начинают свою работу с *API* токеном, который необходим для взаимодействия с ботом. Для его получения необходимо обратиться к официальному боту *BotFather*.

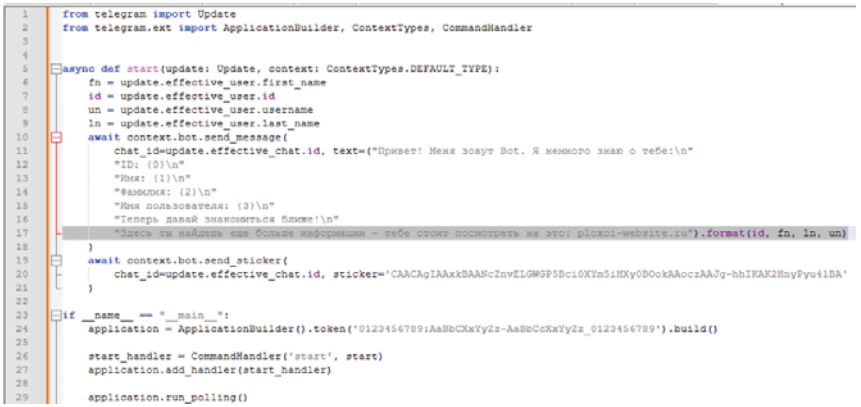
Токен имеет структуру, схожую с базовой аутентификацией в *HTTP*-протоколе: он начинается с 10 цифр, за которыми следует двоеточие и последовательность из 35 символов. Воспользовавшись методом *getMe*, разработчик может получить *json* формат с указанием идентификатора бота, что является первой частью токена. При дальнейшем анализе было определено, что обнаружение идентификаторов любого бота возможно с помощью веб-версии мессенджера *Telegram* и встроенных инструментов браузера. Вторая часть токена состоит из 35 символов, включающих 26 строчных и 26 прописных английских букв, 10 цифр, а также два специальных символа — нижнее подчеркивание и тире. В итоге получается алфавит из 64 символов, а число возможных комбинаций 64^35. Однако перебор токена с помощью метода брутфорс достаточно длительный процесс. Следовательно, основная угроза в отношении токена исходит от самого человека.

С ботом взаимодействует большое число пользователей через личное общение и групповые чаты. Для того, чтобы каждый запрос был корректно обработан и ответ вернулся обратно к тому же пользователю, используется параметр *chat.id* из объекта *API update*. Такой процесс реализуется следующим образом: запрос с *chat.id* направляется от пользователя к боту, по определенному ранее токену осуществляется обращение на сервер, выполняется обработка запроса и ответ возвращается обратно пользователю по параметру *chat.id*.

В официальной документации *Telegram Bot API* [2], когда речь идет про взаимодействие с *BotFather*, разработчиков призывают безопасно хранить и пользоваться токеном. Важно соблюдать такие требования по нескольким причинам. Наличие токена у стороннего человека приводит к беспрепятственному доступу на отправку запросов. Злоумышленник может намеренно подделать запрос так, чтобы его действия выглядели легитимными. Фрагменты кода для отправки запросов приведены на рисунках 1 и 2.

```python
import logging
from telegram import Update
from telegram.ext import ApplicationBuilder, ContextTypes, CommandHandler, MessageHandler, filters


logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    fn = update.effective_user.first_name
    id = update.effective_user.id
    un = update.effective_user.username
    ln = update.effective_user.last_name
    await context.bot.send_message(chat_id=update.effective_chat.id, text=("Привет! Меня зовут PracticeSeqBot. Я немного знаю о тебе:\n"
        "ID: {0}\n"
        "Имя: {1}\n"
        "Фамилия: {2}\n"
        "Имя пользователя: {3}\n"
        "Теперь давай знакомиться ближе!").format(id, fn, ln, un)
    )
    await context.bot.send_sticker(
        chat_id=update.effective_chat.id, sticker='CAACAgIAAxkBAANcZnvELGWGP5Bci0XYm5iHXy0BOokAAoczAAJg-hhIKAK2HnyPyu41BA'
    )

if __name__ == "__main__":
    application = ApplicationBuilder().token('0123456789:AaBbCXxYyZz-AaBbCcXxYyZz_0123456789').build()

    start_handler = CommandHandler('start', start)
    application.add_handler(start_handler)

    application.run_polling()
```

Рисунок 1 – Легитимный запрос



```python
from telegram import Update
from telegram.ext import ApplicationBuilder, ContextTypes, CommandHandler


async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    fn = update.effective_user.first_name
    id = update.effective_user.id
    un = update.effective_user.username
    ln = update.effective_user.last_name
    await context.bot.send_message(
        chat_id=update.effective_chat.id, text=("Привет! Меня зовут Bot. Я немного знаю о тебе:\n"
        "ID: {0}\n"
        "Имя: {1}\n"
        "Фамилия: {2}\n"
        "Имя пользователя: {3}\n"
        "Теперь давай знакомиться ближе!\n"
        "Здесь ты найдешь еще больше информации - тебе стоит посмотреть на это: ploxoi-website.ru").format(id, fn, ln, un)
    )
    await context.bot.send_sticker(
        chat_id=update.effective_chat.id, sticker='CAACAgIAAxkBAANcZnvELGWGP5Bci0XYm5iHXy0BOokAAoczAAJg-hhIKAK2HnyPyu41BA'
    )

if __name__ == "__main__":
    application = ApplicationBuilder().token('0123456789:AaBbCXxYyZz-AaBbCcXxYyZz_0123456789').build()

    start_handler = CommandHandler('start', start)
    application.add_handler(start_handler)

    application.run_polling()
```

Рисунок 2 – Запрос злоумышленника

Главное отличие в 17 строке кода злоумышленника, который изображен на рисунке 2. Была подложена ссылка, способная выступать в качестве промежуточного ресурса для реализации атаки.

Небезопасное хранение токена может привести к считыванию информации из чатов. Реализация подобного способа сложнее, поскольку атакующему необходимо подобрать такой момент, когда пользователь взаимодействует с ботом. И если его получается

подобрать, то в запущенных одновременно сессиях по одинаковому токену будут возникать ошибки.

Были выявлены еще две особенности работы *Telegram Bot API*. Во-первых, история всех сообщений стирается перед каждым запуском бота, так как по умолчанию выполняется метод *deleteWebhook*. Вторая особенность заключается в хранении сообщений: история переписки находится на сервере не более 24 часов.

Представленные угрозы реализуются только в случае компрометации уникального токена. Возможность нахождения ключа связана с распространённой ошибкой разработчиков, которая заключается в хранении кода вместе с уникальным токеном. В том случае, если репозиторий окажется публичным, любой человек сможет воспользоваться токеном.

Рассматривая уязвимости внутри системы бота, можно провести аналогию с веб-уязвимостями, поскольку, как правило, используется идентичный набор технологий. К ним относятся *sql*-инъекции, межсайтовый скриптинг, внедрение бэкдора и включение в код вредоносных фрагментов.

Ещё один выявленный вектор атаки – это применение методов социальной инженерии. Количество случаев [3-4] подтверждает распространенность таких атак. К ним, например, можно отнести фишинг или создание ботов-двойников. Реализации такого вида мошенничества сопровождается все возможными формами уловок – это и приглашения на праздники, и информация, касающаяся профессиональной деятельности или социальных вопросов. Все это может быть использовано против пользователя. Зачастую именно методы социальной инженерии используются в качестве первого этапа для реализации большой и сложной атаки.

В результате проведения анализа были четко определены следующие угрозы:

– угрозы небезопасного хранения токена;
– технические уязвимости;
– применение социальной инженерии.

К методам защиты от угроз при небезопасном хранении токена относится ведение приватного репозитория, исключение передачи токена третьим лицам, использование функции *gitignore* и системы

предотвращения конфликтов, способной отследить сессию и в случае возникновения ошибок сгенерировать новый *API* токен.

Для устранения технических уязвимостей бота, в первую очередь, необходимо валидировать входные данные от пользователей и придерживаться принципа минимальных привилегий.

Последний вектор угроз – это социальная инженерия. Главным методом защиты бота от подобных атак будет разработка система поверки подлинности бота. Компании должны быть заинтересованы в этом, чтобы исключить нежелательные последствия. Также, рекомендуемой практикой является обновление токена с определенной периодичностью для исключения его компрометации.

В заключении рассмотрим последствия вышеперечисленных угроз. Остановка бизнес-процесса может привести к потере дохода и конкурентным преимуществам. Кража конфиденциальной информации влечет за собой юридические последствия, а также, как следствие, потерю доверия клиентов любой компании. В связи с этим, рекомендуется уделять особое внимание ботам, как и веб технологиям.

Своевременное ознакомление сотрудников с возможными угрозами и понимание важности правильного хранения токена позволит взаимодействовать с *Telegram*-ботами эффективно и безопасно.

Литература:

1. Как обеспечить безопасность телеграм-бота. – URL: https://tproger.ru/articles/obespechenie-bezopasnosti-telegram-botov/ (дата обращения 16.08.2024).

2. Telegram Bot API. – URL: https://core.telegram.org/bots/api/ (дата обращения 14.08.2024).

3. Рынок фишинга в Telegram. – URL: https://securelist.ru/telegram-phishing-services/107193/ (дата обращения 17.08.2024).

4. «Яндекс» удалил свой Telegram-бот «Квиз Яндекс Плюс» – мошенники использовали его для рассылки спама. – URL: https://habr.com/ru/news/589219/ (дата обращения 21.08.2024).