АКШОНОВ А. М., ЛАДАНОВА Е. О., НИКУЛИН В. В.

РАЗРАБОТКА ТЕЛЕГРАМ-БОТА ДЛЯ УПРАВЛЕНИЯ НАСТРОЙКАМИ РОУТЕРА

Аннотация. В данной статье рассматривается подход к автоматизации управления настройками роутера с использованием Телеграм-бота. Основное внимание уделено разработке и интеграции телеграм-бота, позволяющего пользователям удаленно изменять параметры роутера, такие как настройки Wi-Fi, перезагрузка устройства, управление портами и настройка приоритизации трафика. Приведены примеры использования бота для оценки качества сетевого соединения и проведения базовой диагностики сети, включая измерение скорости, пинга и потери пакетов. Особое внимание уделено вопросам безопасности и аутентификации при удаленном доступе к роутеру через бот. Экспериментальные результаты демонстрируют эффективность предложенного решения, показывая, что бот значительно упрощает процесс управления домашними и офисными сетями.

Ключевые слова: сетевое администрирование, удаленное управление, автоматизация, аутентификация, диагностика сети.

AKSHONOV A. M., LADANOVA E. O., NIKULIN V. V.

DEVELOPING A TELEGRAM BOT FOR MANAGING ROUTER SETTINGS

Abstract. This article discusses an approach to automating router settings management using a Telegram bot. The focus is on the development and integration of a Telegram bot that allows users to remotely modify router parameters such as Wi-Fi settings, device reboot, port management, and traffic prioritization. Examples of bot usage are given for evaluating network connection quality and performing basic network diagnostics, including speed measurement, ping, and packet loss. Special attention is given to security and authentication issues during remote access to the router via the bot. Experimental results demonstrate the effectiveness of the proposed solution, showing that the bot significantly simplifies the process of managing home and office networks.

Keywords: network administration, remote control, automation, authentication, network diagnostics.

Современные сети, как домашние, так и офисные, становятся все более сложными, что требует повышения уровня удобства и эффективности их управления. С ростом числа подключенных устройств и увеличением требований к качеству соединения возникает необходимость в простых и доступных инструментах для удаленного администрирования сетевых устройств, в частности роутеров. Традиционные методы настройки и мониторинга сетей зачастую требуют использования сложных интерфейсов или физического доступа к устройствам, что не всегда удобно и эффективно. Телеграм-боты, как инструменты для

автоматизации задач, приобретают все большую популярность благодаря своей простоте в использовании и возможностям интеграции с различными системами. Они могут значительно упростить управление сетевыми устройствами, предоставляя интуитивно понятный интерфейс для изменения настроек и мониторинга состояния сети через удобный мессенджер. Использование Телеграм-ботов для управления роутерами позволяет проводить такие операции, как изменение параметров Wi-Fi, настройка приоритизации трафика, перезагрузка устройства и проведение базовой диагностики сети, удаленно и в реальном времени [1–3].

Цель данного исследования – разработка и описание Телеграм-бота, предназначенного для управления настройками роутера, а также оценка его эффективности в решении задач удаленного администрирования сети. В рамках работы будет предложен метод интеграции бота с роутером, исследованы вопросы безопасности и защиты при взаимодействии с сетевыми устройствами, а также проведены эксперименты по оценке производительности данного подхода в различных сетевых условиях.

Процесс разработки Телеграм-бота, предназначенного для управления настройками роутера, представляет собой несколько последовательных этапов, каждый из которых играет важную роль в построении общей архитектуры системы. От начального проектирования, охватывающего ключевые аспекты взаимодействия компонентов, до детальной интеграции с роутером и обеспечения безопасности передачи данных – на каждом этапе были применены методы, направленные на создание надежного и функционального решения [4]. Для более глубокого понимания рассмотрим ключевые моменты, использованные в ходе разработки и последующего тестирования предложенной системы.

На языке Go, который, будучи разработанным Google, отличается своей лаконичностью и одновременно мощной поддержкой многозадачности, был создан Телеграм-бот, выступающий в роли центрального компонента системы. Go, благодаря своей способности эффективно управлять многозадачностью посредством горутин, что особенно полезно для приложений, требующих обработки большого числа запросов в реальном времени, оказался идеальным выбором. Взаимодействие бота с Telegram API было обеспечено с помощью библиотеки go-telegram-bot-api, которая существенно упростила интеграцию и дала возможность эффективно управлять обработкой команд, отправляемых пользователями. Задачи, которые ставятся перед ботом, варьируются от простейших действий, таких как изменение настроек беспроводной сети Wi-Fi, до более сложных задач вроде перезагрузки устройства или запуска процессов диагностики сети. В ответ на команду от пользователя бот формирует соответствующий запрос и передает его на сервер, который непосредственно взаимодействует с роутером. Коммуникация происходит через протоколы передачи данных, такие как HTTP(S), что позволяет обеспечивать безопасную передачу информации.

Серверная часть системы выполняет роль посредника между Телеграм-ботом и роутером. Получив запрос от бота, сервер анализирует поступившую команду и инициирует выполнение необходимых действий. В зависимости от команды могут быть задействованы различные методы управления устройством. Например, если роутер поддерживает API, то сервер отправляет HTTP(S)-запросы, обращаясь к веб-интерфейсу устройства. В случаях, когда API отсутствует, для выполнения команд используется SSH-соединение, что позволяет отправлять команды напрямую через CLI (интерфейс командной строки), тем самым позволяя управлять роутером удаленно. После выполнения команды, будь то изменение параметров сети или запуск тестов диагностики, роутер возвращает ответ, который сервер обрабатывает и направляет обратно в Телеграм-бот. Бот, в свою очередь, формирует пользовательский ответ и передает его обратно в чат, информируя о результатах выполнения запроса. Роутер, являясь конечной точкой в системе, реализует физическое управление сетевыми параметрами. Благодаря поддержке различных методов удаленного управления, включая API, CLI и протокол SNMP (Simple Network Management Protocol), возможно не только изменение базовых настроек беспроводной сети (SSID, пароли и т.д.), но и выполнение различных диагностических операций, таких как проверка скорости соединения или мониторинг уровня потерь пакетов. В случае необходимости роутер может также передавать данные о текущем состоянии (например, нагрузка на сеть или информация о подключении) на сервер для последующего анализа и отображения в пользовательском интерфейсе Телеграм-бота.

Использование языка Go в разработке данного решения было продиктовано его преимуществами в плане производительности, а также гибкостью работы с многозадачностью, что в сочетании с библиотекой go-telegram-bot-api позволило достичь высокой скорости обработки запросов.

Одной из основных функций разработанного Telegram-бота является возможность измерения скорости интернет-соединения. Данная функция крайне полезна для мониторинга качества сети, так как многие современные онлайн-сервисы зависят от стабильного и скоростного соединения с интернетом. Регулярная проверка скорости позволяет своевременно выявить и устранить возможные проблемы, влияющие на качество пользовательского опыта. Для автоматического измерения скорости интернет-соединения в боте задействована утилита speedtest-cli, которая запускается на сервере и возвращает результаты в виде текста. Эта утилита, являясь кроссплатформенной и широко используемой, позволяет получать такие показатели, как скорость загрузки и отдачи, а также задержку (пинг). Ниже приведен пример реализации данной функции на языке программирования Go:

```
func checkSpeed(bot *tgbotapi.BotAPI, chatID int64) string {
    action := tgbotapi.NewChatAction(chatID, tgbotapi.ChatTyping)
    bot.Send(action)
```

```
    out, err := exec.Command("speedtest-cli").CombinedOutput()
    if err != nil {
        return "Ошибка при проверке скорости: " + err.Error() + "\n" + string(out)
    }

    return string(out)
}
```

В данной функции используется команда exec.Command, которая инициирует запуск утилиты speedtest-cli через системные команды. Если в процессе выполнения возникает ошибка, то функция возвращает сообщение с описанием ошибки и выводом команды. В противном случае возвращаются результаты измерений в текстовом формате. Чтобы бот выглядел более интерактивно, была добавлена анимация «печати», которая появляется на экране чата, сигнализируя пользователю о том, что запрос обрабатывается. Telegram-бот вызывает данную функцию следующим образом:

```
case "check_speed":
    speedResult := checkSpeed()
    msg := tgbotapi.NewMessage(chatID, speedResult)
    bot.Send(msg)
```

После получения команды на проверку скорости интернет-соединения, бот вызывает функцию checkSpeed, которая выполняет измерение скорости и возвращает детализированный результат пользователю в виде текстового сообщения. В результате пользователь может в любой момент проверить параметры соединения, такие как скорость загрузки, скорость отдачи, а также задержку, получив данные в удобной и читаемой форме. На рисунке 1 представлен пример того, как Telegram-бот выполняет измерение скорости интернета. Данная функциональность предоставляет пользователям гибкость и удобство, что особенно важно для тех, кто регулярно сталкивается с нестабильным интернет-соединением или проводит тестирование сетевых параметров для улучшения качества работы сетевых сервисов.

Использование speedtest-cli в качестве инструмента для измерения скорости интернета внутри нашего бота оправдано с точки зрения простоты внедрения и быстроты получения результатов. Этот инструмент уже доказал свою надежность в различных сценариях использования, что делает его эффективным решением для мониторинга интернет-соединений в реальном времени. Telegram-бот, интегрированный с данной утилитой, способен быстро реагировать на запросы пользователей и предоставлять важную информацию о состоянии сети в несколько кликов.

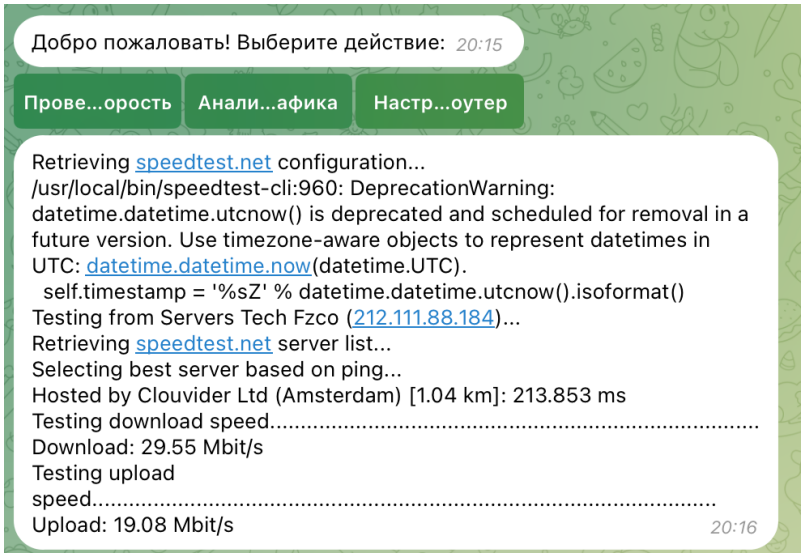

Рис. 1. Результат проверки скорости интернета.

Telegram-бота может осуществлять анализ сетевого трафика. Эта предоставляет пользователям ценную информацию о том, какой объем данных передается через активные сетевые интерфейсы. Подобный анализ важен для диагностики работы сети, позволяет выявить узкие места или подозрительную активность в реальном времени. В данном случае для выполнения анализа трафика используется библиотека gopacket, которая обеспечивает работу с сетевыми интерфейсами, предоставляя возможность захвата и анализа пакетов данных, проходящих через сеть. В основе данной функции лежит следующий код на языке Go:

```
	func analyzeTraffic() string {
		device, err := getActiveInterface()
		if err != nil {
			return "Ошибка определения активного интерфейса: " + err.Error()
		}

		handle, err := pcap.OpenLive(device, 1600, true, pcap.BlockForever)
		if err != nil {
			return "Ошибка при открытии устройства: " + err.Error()
		}
		defer handle.Close()

		packetSource := gopacket.NewPacketSource(handle, handle.LinkType())
		packetCount := 0
		for range packetSource.Packets() {
			packetCount++
			if packetCount >= 10 {
				break
			}
		}
		return fmt.Sprintf("Анализировано %d пакетов на интерфейсе %s", packetCount,
device)
	}
```

Данная функция выполняет захват сетевых пакетов с активного интерфейса, чтобы подсчитать количество переданных данных. В первую очередь функция определяет активный интерфейс с помощью getActiveInterface(). Если при определении активного интерфейса

происходит ошибка, то бот выводит соответствующее сообщение. Далее, с помощью библиотеки pcap открывается этот интерфейс для захвата пакетов в режиме реального времени. Если интерфейс был успешно открыт, производится захват данных с использованием функции gopacket.NewPacketSource, которая собирает пакеты для анализа. В цикле производится их подсчет, и при достижении заданного числа пакетов (в примере это 10) выполнение функции завершается, выводя итоговый результат. Особенность реализации заключается в возможности не только подсчета пакетов, но и потенциального расширения этой функции. Например, в дальнейшем можно интегрировать более сложный анализ пакетов – отслеживание их содержимого, IP-адресов, протоколов и других характеристик сетевого трафика, что дает возможность пользователю не просто получать общее количество переданных данных, но и видеть, какие типы данных проходят через его сеть. Telegram-бот вызывает данную функцию следующим образом:

```
case "analyze_traffic":
    trafficResult := analyzeTraffic()
    msg := tgbotapi.NewMessage(chatID, trafficResult)
    bot.Send(msg)
```

При получении команды на анализ сетевого трафика бот вызывает функцию analyzeTraffic, которая анализирует данные и возвращает отчет о количестве обработанных пакетов, переданных через активный сетевой интерфейс. Этот отчет отправляется пользователю в виде текстового сообщения, что позволяет мгновенно получить информацию о текущем состоянии сетевого трафика. На рисунке 2 представлен пример работы Telegram-бота по анализу сетевого трафика.

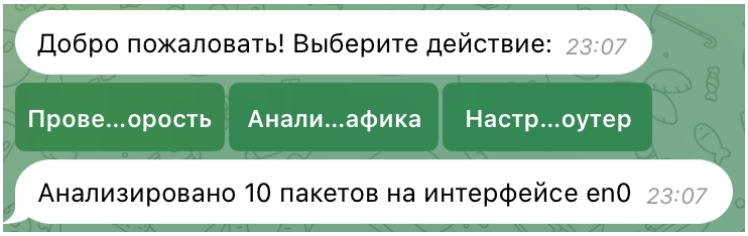

Рис. 2. Анализ сетевого трафика.

Одним из приоритетных аспектов в разработке Telegram-бота является безопасность. Для защиты конфиденциальной информации, такой как пароли, используется симметричное шифрование AES. Это позволяет хранить и передавать пароли в зашифрованном виде, минимизируя риски их утечки. Данный подход обеспечивает высокую степень защиты данных, предотвращая возможность их перехвата злоумышленниками. Пример реализации шифрования и дешифрования данных выглядит следующим образом:

```
func encrypt(text, key string) (string, error) {
	block, err := aes.NewCipher([]byte(key))
	if err != nil {
		return "", err
	}

	ciphertext := make([]byte, aes.BlockSize+len(text))
	iv := ciphertext[:aes.BlockSize]
	if _, err := io.ReadFull(rand.Reader, iv); err != nil {
		return "", err
	}

	stream := cipher.NewCFBEncrypter(block, iv)
	stream.XORKeyStream(ciphertext[aes.BlockSize:], []byte(text))

	return hex.EncodeToString(ciphertext), nil
}

func decrypt(encryptedText, key string) (string, error) {
	ciphertext, _ := hex.DecodeString(encryptedText)

	block, err := aes.NewCipher([]byte(key))
	if err != nil {
		return "", err
	}
	if len(ciphertext) < aes.BlockSize {
		return "", errors.New("шифрованный текст слишком короткий")
	}

	iv := ciphertext[:aes.BlockSize]
	ciphertext = ciphertext[aes.BlockSize:]

	stream := cipher.NewCFBDecrypter(block, iv)
	stream.XORKeyStream(ciphertext, ciphertext)

	return string(ciphertext), nil
}
```

В данном коде используются функции для шифрования и дешифрования данных с помощью алгоритма AES, что повышает безопасность Telegram-бота при передаче данных через сеть и защищает пароли от несанкционированного доступа. Безопасность является одним из важнейших аспектов работы с сетевыми устройствами, и реализация шифрования в данном Telegram-боте позволяет обеспечить необходимый уровень защиты для надежной работы в сетях различного масштаба.

Таким образом предложен и успешно реализован подход к автоматизации процесса управления настройками роутера с применением Telegram-бота. Созданный инструмент предоставляет пользователям возможность удаленно изменять ключевые параметры сети, включая настройку Wi-Fi, перезагрузку устройства, управление сетевыми портами и приоритизацию трафика. Внедрение подобного решения продемонстрировало высокую эффективность как в контексте домашних, так и офисных сетей, где системным администраторам теперь не требуется физически присутствовать рядом с оборудованием для

его настройки и управления. Дальнейшие направления исследований могут быть сосредоточены на расширении функциональных возможностей бота. В частности, перспективным является добавление поддержки более широкого спектра моделей роутеров, что позволит охватить большее количество пользователей и сценариев использования. Кроме того, одним из важнейших шагов по улучшению системы является внедрение более продвинутых инструментов диагностики, которые смогут автоматически обнаруживать и устранять неисправности. Одной из долгосрочных целей может стать интеграция бота с облачными системами управления сетями, что обеспечит значительное повышение масштабируемости решения и его применение в корпоративных средах с высоким уровнем сложности и распределённости сетевых систем.

СПИСОК ЛИТЕРАТУРЫ

1. Sucipto S., Karaman J. Integration of Legalization Information System Web-Based using Shipping API and Telegram API // JUITA: Jurnal Informatika. – 2020. – Vol. 8, No. 2. – P. 131–139.

2. Khaund T. et al. Telegram: Data Collection, Opportunities and Challenges // Information Management and Big Data: Conference Proceedings. – Springer, 2020. – P. 513–526.

3. Modrzyk N. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API. – Tokyo: Apress, 2018. – 288 p.

4. Ямашкина Е. О., Ямашкин С. А., Никулин В. В. Принципы аутентификации и распределения ролей пользователей в геопортальных системах // Вопросы информационной безопасности: Материалы VII межрегионального вебинара, Саранск – Елец – Астрахань – Волгоград, 21 февраля 2023 г. – Саранск: Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва, 2023. – С. 89–94.