МРНТИ: 81.93.29 DOI 10.48501/9935.2025.87.40.018

[1]Д.Т Тютеев, [1]К.М Сагиндыков, [1]А.К Шайханова
[1]Евразийский национальный университет им. Л.Н.Гумилева
Казахстан, Астана
(e-mail: sav3al1.l@gmail.com)

РАЗРАБОТКА ИНСТРУМЕНТА УДАЛЕННОГО ДОСТУПА ДЛЯ АВТОМАТИЗАЦИИ ПЕНТЕСТА НА GO

Аннотация. В условиях возрастающей значимости информационной безопасности автоматизация процессов пентестинга становится неотъемлемым инструментом для эффективной защиты современных цифровых систем. При проведении аудитов и тестов на проникновение предприятиям и исследовательским группам важно иметь в своем распоряжении гибкий инструмент, который обеспечит высокую степень контроля над удаленными узлами при минимальном физическом вмешательстве. Цель данного исследования — разработать и внедрить средство удаленного доступа для автоматизации тестирования безопасности на языке Go. Благодаря компактности и выразительности Go, разработчикам удалось создать решение, которое позволяет проводить всесторонние аудиты информационных систем в условиях ограниченного физического доступа и эффективно управлять процессом тестирования. В представленной статье подробно рассматриваются архитектурные решения и функциональные особенности инструмента. Ключевыми модулями являются компоненты для доступа к файловой системе, консоли и проксирования сетевых соединений, что делает инструмент универсальным и удобным на разных этапах пентестинга. Помимо этого, особое внимание уделено вопросам безопасности и устойчивости к ошибкам, поскольку удаленные операции требуют надежного шифрования и эффективного управления рисками. В ходе тестирования было проведено сравнение с существующими решениями, что позволило выявить значительные преимущества предлагаемого подхода в скорости и надежности обнаружения уязвимостей. Результаты исследования подтверждают, что разработанный инструмент способен с высокой точностью выявлять критические недостатки и точки входа, которые могут использовать злоумышленники. При этом он не только ускоряет процесс идентификации уязвимостей, но и упрощает их документирование и воспроизведение в ходе планирования мер по защите. Таким образом, предлагаемое решение значительно повышает эффективность процесса обеспечения безопасности и способствует более качественной защите ценных информационных активов, что особенно актуально в контексте постоянно усложняющегося ландшафта киберугроз.

Ключевые слова: Автоматизация пентестинга, удаленный доступ, информационная безопасность, язык программирования golang, угрозы информационной безопасности

[1]D.T. Tyuteyev, [1]K.M Sagindykov, [1]A.K Shaikhanova
[1]L.N. Gumilyov Eurasian National University
[1]Kazakhstan, Astana
(e-mail: sav3al1.l@gmail.com)

DEVELOPMENT OF A REMOTE ACCESS TOOL FOR PENTEST AUTOMATION IN GO

Abstract. In the context of growing significance of information security, the automation of penetration testing processes becomes an indispensable tool for effectively protecting modern digital systems. When conducting audits and penetration tests, it is crucial for businesses and research teams to have a flexible tool at their disposal that provides a high degree of control over remote nodes with minimal physical intervention. The goal of this study is to develop and implement a remote access tool for automating security testing in the Go programming language. Thanks to Go's compactness and expressiveness, the developers were able to create a solution that enables comprehensive audits of information systems under limited physical access conditions while efficiently managing the testing process.This article provides a detailed overview of the architectural solutions and functional features of the tool. Its key modules include components for accessing the file system, console, and proxying network connections, making the tool versatile and convenient at different stages of penetration testing. In addition, special attention is paid to security issues and fault tolerance, as remote operations require robust encryption and effective risk management. During testing, a comparison was made with existing solutions, which revealed significant advantages of the proposed approach in terms of speed and reliability in detecting vulnerabilities.The results of the study confirm that the developed tool can accurately identify critical flaws and entry points that attackers

might exploit. Moreover, it not only accelerates the vulnerability identification process but also simplifies documentation and reproduction during security planning. Consequently, the proposed solution significantly enhances the efficiency of security assurance and contributes to better protection of valuable information assets—an especially relevant outcome given the ever-evolving landscape of cyber threats.

Keywords: Pentesting automation, remote access, information security, golang programming language, information security threats

[1]Д.Т Тютеев. [1]К.М. Сагиндыков. [1]А.К Шайханова
[1]Л.Н. Гумилев атындағы Еуразия ұлттық университеті
Қазақстан, Астана
(e-mail: sav3al1.l@gmail.com)

GO ЖҮЙЕСІНДЕ PENTEST АВТОМАТТАНДЫРУЫ ҮШІН ҚАШЫҚТАН ҚОЛ ЖЕТКІЗУ ҚҰРАЛЫН ӘЗІРЛЕУ

Аннотация. Ақпараттық қауіпсіздіктің маңыздылығының артуы жағдайында пентестинг процестерін автоматтандыру қазіргі заманғы цифрлық жүйелерді тиімді қорғаудың ажырамас құралына айналады. Кәсіпорындар мен зерттеу топтарына ену аудиттері мен сынақтарын жүргізу кезінде ең аз физикалық араласу кезінде қашықтағы түйіндерді бақылаудың жоғары дәрежесін қамтамасыз ететін икемді құралдың болуы маңызды. Бұл зерттеудің мақсаты-go тілінде қауіпсіздікті тексеруді автоматтандыру үшін қашықтан қол жеткізу құралын әзірлеу және енгізу. Go компамдылығы мен экспрессивтілігінің арқасында әзірлеушілер физикалық қол жетімділігі шектеулі жағдайларда ақпараттық жүйелерге жан-жақты аудит жүргізуге және тестілеу процесін тиімді басқаруға мүмкіндік беретін шешім жасай алды. Ұсынылған мақалада құралдың архитектуралық шешімдері мен функционалдық ерекшеліктері егжей-тегжейлі қарастырылады. Негізгі Модульдер файлдық жүйеге, консольге қол жеткізуге және желілік қосылымдарды прокси-серверге арналған компоненттер болып табылады, бұл құралды пентестингтің әртүрлі кезеңдерінде жан-жақты және ыңғайлы етеді. Бұдан басқа, қауіпсіздік пен қатеге төзімділік мәселелеріне ерекше назар аударылады, өйткені қашықтағы операциялар сенімді шифрлауды және тәуекелдерді тиімді басқаруды қажет етеді. Тестілеу барысында қолданыстағы шешімдермен салыстыру жүргізілді, бұл осалдықтарды анықтау жылдамдығы мен сенімділігінде ұсынылған тәсілдің айтарлықтай артықшылықтарын анықтауға мүмкіндік берді. Зерттеу нәтижелері әзірленген құрал зиянкестер пайдалана алатын маңызды кемшіліктер мен кіру нүктелерін жоғары дәлдікпен анықтай алатынын растайды. Сонымен қатар, ол осалдықтарды анықтау процесін жеделдетіп қана қоймайды, сонымен қатар оларды қорғау шараларын жоспарлау кезінде құжаттауды және көбейтуді жеңілдетеді. Осылайша, ұсынылған шешім қауіпсіздік процесінің тиімділігін айтарлықтай арттырады және құнды ақпараттық активтерді жақсырақ қорғауға ықпал етеді, бұл әсіресе үнемі күрделене түсетін киберқауіптер ландшафты контекстінде маңызды.

Түйін сөздер: Пентестингті автоматтандыру, қашықтан қол жеткізу, ақпараттық қауіпсіздік, Golang бағдарламалау тілі, ақпараттық қауіпсіздік қатерлері

Введение

В условиях стремительно развивающихся технологий и постоянно увеличивающегося числа киберугроз защита информационных систем становится приоритетом для всех организаций, занимающихся обработкой и хранением данных [1]. Одним из важнейших методов проверки защищенности инфраструктуры является пентестинг (penetration testing). Этот метод имитирует действия злоумышленника, направленные на выявление и эксплуатацию уязвимостей в системе [2]. Пентест позволяет понять, насколько защищена сеть, как легко получить доступ к важной информации и какие потенциальные риски могут возникнуть при реальной атаке [3].

Однако традиционный пентест требует значительных затрат времени и ресурсов, поскольку процесс анализа, проникновения и отчётности о найденных уязвимостях зачастую выполняется вручную [4]. В этом контексте автоматизация становится неотъемлемой частью эффективного и точного проведения тестов. Автоматизированные инструменты позволяют быстро и качественно производить рутинные проверки, позволяя специалистам сосредоточиться на анализе и исследовании новых потенциальных угроз [5].

В данной статье рассмотрим возможности и подходы к разработке инструмента для автоматизации удаленного доступа в контексте пентеста на языке программирования Go. Мы обсудим ключевые аспекты автоматизации тестирования, проанализируем существующие решения на рынке, преимущества и недостатки использования Go для создания инструмента, а также основные функциональные возможности, которые должны присутствовать в таком инструменте.

Материалы и методы

Основная гипотеза исследования — разработка инструмента удаленного доступа для автоматизации пентестов, способного повысить эффективность процесса проведения тестов на проникновение за счет минимизации участия человека в рутинных операциях. Для проверки гипотезы и достижения целей исследования были использованы следующие методы:

1. **Изучение литературы и существующих инструментов**:

Анализ современных научных публикаций и материалов по теме автоматизации пентестов.

2. **Информационный поиск и тестирование**:

Исследование используемых в пентестах протоколов, подходов и архитектур.

Оценка существующих инструментов по следующим критериям:

Возможность работы в изолированных средах.

Простота развертывания и интеграции в процессы пентеста.

Надежность передачи данных.

3. **Разработка и тестирование прототипа**:

Использование языка программирования Go для создания легковесного, кроссплатформенного решения.

Тестирование прототипа в изолированных средах для проверки его функциональности.

Проведение сравнения с аналогами для оценки эффективности.

4. **Междисциплинарный подход**:

Объединение знаний в области информационной безопасности, сетевого администрирования и программирования для решения поставленных задач.

Полученные результаты

В ходе исследования был разработан и протестирован инструмент удаленного доступа для автоматизации пентестов. Для оценки эффективности разработанного решения был проведен сравнительный анализ с существующими решениями, такими как Metasploit (Meterpreter) и Cobalt Strike, по ключевым критериям (Таблица 1). В процессе исследования определены следующие параметры для оценки:

1. **Функциональные возможности**:
   - Доступ к файловой системе целевой машины.
   - Удаленное выполнение команд через терминал.
   - Прокси-соединение для работы с внутренними сетями целевой системы.
2. **Легкость развертывания**:
   - Возможность работы в изолированных средах без необходимости использования сторонних библиотек или инфраструктуры.
   - Минимальные системные требования для установки и работы.
3. **Безопасность и устойчивость к обнаружению**:
   - Шифрование всего трафика между компонентами (клиент, сервер, агент) с использованием современных криптографических алгоритмов.
   - Исключение использования контейнеризации, что снижает риск детектирования со стороны защитных систем.
4. **Интеграция в процессы пентеста**:
   - Возможность автоматизации рутинных операций, таких как удаленный доступ и управление файлами.
   - Простая настройка, минимизирующая время на подготовку к тестам.

Таблица 1 Сравнение разработанного инструмента с существующими решениями

Решение	**Преимущества**	Ограничения	Разработанное решение (сравнение)
Metasploit (Meterpreter)	-Широкий функционал, включая поддержку разнообразных payload'ов и скриптов для атаки. - Интеграция с другими инструментами.	- Высокая вероятность детектирования антивирусными системами. - Сложность в настройке для работы в изолированных средах.	- Уступает в широте возможностей. - Выигрывает в устойчивости к обнаружению. - Проще интегрируется в ограниченных (изолированных) средах.
Cobalt Strike	- Мощные средства для управления «beacon'ами» и автоматизации операций. - Поддержка сценариев для	- Высокая стоимость лицензии. - Наличие детектируемых	- Аналогичный уровень автоматизации для узкого круга задач. - Бесплатен и с

	сложных атак.	сигнатур в современных антивирусах.	минимальными признаками активности. - Позволяет сохранять скрытность и уменьшает вероятность детекта антивирусными системами.

Обсуждение

Для создания инструмента удалённого доступа был выбран язык Go (Golang), поскольку он сочетает в себе ряд преимуществ, важных для разработки подобного решения. Во-первых, Go был разработан в компании Google с учётом современных требований к производительности и удобству разработки. Он компилируется напрямую в машинный код, что обеспечивает высокую скорость работы и снижает задержки при обработке запросов [7]. Кроме того, Go ориентирован на проекты, работающие с большим количеством параллельных задач, благодаря встроенной модели конкурентности (goroutines). Они гораздо «легче», чем классические потоки (threads), позволяя одновременное управление множеством соединений к целевым системам без значительного расхода ресурсов и ухудшения производительности (Рисунок 1).

Рисунок 1 сравнение производительности golang и python

Ещё одним важным фактором выбора Go является его простота и читаемость кода. Язык обладает лаконичным синтаксисом, что снижает риск ошибок при разработке систем, где безопасность играет ключевую роль. Для инструмента, который требует надёжности и защищённости, это особенно актуально, так как любая уязвимость в коде может поставить под угрозу весь процесс тестирования или сбора информации [8].

Функциональность, связанная с сетевым взаимодействием, также реализована в Go на высоком уровне. Широкий набор стандартных библиотек включает в себя готовые инструменты для работы с HTTP, TCP/UDP, а также средства криптографии (Рисунок 2). Это упрощает создание безопасных каналов связи и передачу конфиденциальных данных.

```go
func verify(message []byte) uint32 {
    offset := len(message) - signLen
    if offset <= 0 {
        return 0
    }
    sign := message[offset:]
    message = message[:offset]
    mac := hmac.New(sha256.New, hmacKey)
    mac.Write(message)
    expectedSign := mac.Sum(nil)
    if hmac.Equal(expectedSign, sign) {
        return uint32(offset)
    }
    return 0
}
```

Рисунок 2 функция проверки корректности HMAC-подписи

Немаловажное преимущество Go — кроссплатформенность. Скомпилированные бинарные файлы могут быть запущены в различных операционных системах (Windows, Linux, macOS) без существенных изменений, что особенно полезно при работе в гетерогенных средах (Рисунок 3).

```
(base) [redacted] % make build-alpine
CGO_ENABLED=0 GOOS=linux go build -ldflags="-w -s"   -o ./bin/binary ./cmd/app
(base) [redacted] % ls ./bin
binary
(base) [redacted] %
```

Рисунок 3. Компиляция под linux

Благодаря этому разработчики получают гибкость в выборе целевой платформы и могут без труда масштабировать систему при необходимости.

Наконец, Go обладает хорошей масштабируемостью: по мере роста числа подключаемых агентов и увеличения объёма данных система способна эффективно работать без существенного падения производительности. Это достигается за счёт эффективного управления памятью, простого параллелизма и оптимизированной стандартной библиотеки[9]. Таким образом, сочетание скорости, удобства разработки, надёжности и кроссплатформенности делает Go оптимальным решением для создания современного инструмента удаленного доступа, удовлетворя потребности как в быстродействии, так и в удобстве сопровождения и дальнейшего развития.

Наличие эффективного параллелизма и хорошей масштабируемости в Go позволило перейти от теоретических предпосылок к созданию инструмента удалённого доступа, в котором особое внимание уделяется удобству сопровождения и быстродействию. Используя встроенную модель конкурентности, Go даёт возможность обрабатывать множество одновременных подключений без существенных потерь в производительности, что особенно важно при работе с несколькими целевыми системами[10].

На этой основе была разработана модульная архитектура, состоящая из трёх компонентов: **server, client** и **agent.**
Взаимодействие между ними организовано следующим образом: оператор управляет процессом через **client**, формируя команды и отправляя их на **server (Рисунок 4)**.

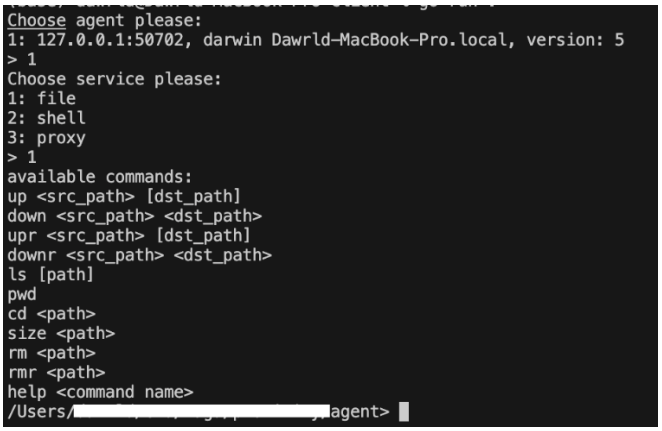

Рисунок 4 интерфейс клиента

Тот, в свою очередь выполняет роль центрального маршрутизатора: перенаправляет запросы от **client** к соответствующим **agent** и передаёт обратно ответы (Рисунок 5).

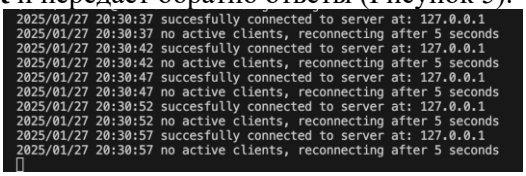

Рисунок 5 логи агента

Он отвечает за контроль доступа, шифрование и проверку подлинности (например, через HMAC или TLS) и хранит информацию о активных сессиях и взаимодействиях, что упрощает масштабирование системы и обеспечивает гибкость при подключении новых **agent** или **client:** при необходимости можно легко добавить новые агенты или масштабировать систему, не затрагивая базовую логику взаимодействия **(Рисунок 6)**.

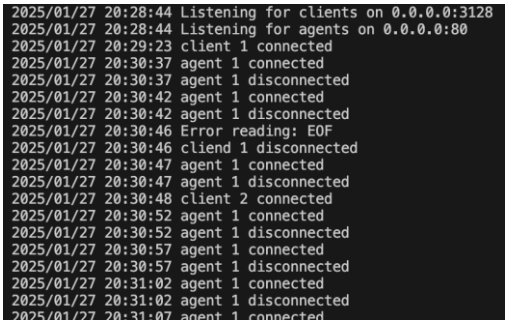

Рисунок 4 лог сервера

После написания инструмента для анализа эффективности были сопоставлены ключевые характеристики инструмента с популярными аналогами, такими как Metasploit, Cobalt Strike, и другими решениями в области информационной безопасности.

Анализ функционала разработанного инструмента

1. Поддерживаемые возможности

Разработанный инструмент предоставляет следующие функции:

- Доступ к файловой системе.
- Удаленное выполнение команд.
- Возможность прокси-соединения для работы с внутренними сетями.

Эти функции покрывают базовые потребности пентестеров и обеспечивают высокую скорость настройки и использования.

2. Преимущества в изолированных средах

Инструмент демонстрирует устойчивость к работе в ограниченных или изолированных сетях благодаря низким системным

требованиям и отсутствию зависимости от сторонних библиотек или сервисов. Это особенно полезно при тестировании систем, не подключенных к интернету.

3. Интеграция и автоматизация Разработанный REST API позволяет интегрировать инструмент в процессы пентеста и автоматизировать выполнение рутинных задач.

Сравнение с существующими решениями

Для более глубокого понимания преимуществ и ограничений разработанного решения проведено его сравнение с аналогами по ключевым критериям (таблица 2).

Таблица 2 Сравнение функционала разработанного инструмента с аналогами

Критерий	Разработанный инструмент	Metasploit (Meterpreter)	Cobalt Strike
Поддерживаемый функционал	Основной функционал	Широкий функционал	Мощные инструменты управления
Устойчивость к обнаружению	Высокая	Средняя	Низкая (обнаруживаемые сигнатуры)
Зависимость от внешних сервисов	Отсутствует	Присутствует	Присутствует
Стоимость	Бесплатно	Бесплатно	Высокая
Легкость интеграции	Высокая	Средняя	Средняя

Оценка по ключевым критериям:

1. **Функционал**

В отличие от Metasploit и Cobalt Strike, разработанный инструмент направлен на решение узкого круга задач автоматизации и не перегружен избыточным функционалом.

2. **Безопасность и маскировка**

Исключение контейнеризации и других компонентов делает инструмент менее заметным для систем обнаружения угроз. Однако, в сравнении с Cobalt Strike, ему не хватает встроенных механизмов обхода EDR.

3. **Эффективность в изолированных средах**

Разработанный инструмент превосходит аналоги, благодаря отсутствию зависимости от внешних сервисов, что делает его идеальным для тестирования систем с ограниченным доступом.

Обоснование сложности развертывания инструментов

Разработанный инструмент имеет низкую сложность развертывания. Он не требует контейнеризации и сложной конфигурации. Его можно легко интегрировать в существующую инфраструктуру пентестов благодаря минималистичному API. Автономность работы делает его удобным для изолированных сред, где доступ к интернету ограничен. Metasploit, хоть и предоставляет широкий функционал, требует значительных временных затрат на настройку, особенно в случае использования сложных payload'ов. Кроме того, работа в изолированных средах ограничена из-за необходимости подключения к внешнему серверу для получения обновлений и доступа к базам данных эксплойтов. Cobalt Strike предоставляет мощные средства автоматизации атак, но его развертывание связано с высокой сложностью. Необходима детальная настройка beacon'ов для обхода EDR/AV систем, а также поддержание постоянной связи между клиентом и сервером. Это делает инструмент малопригодным для изолированных сред (Таблица 3).

Преимущества разработанного инструмента:

1. **Легкость развертывания**

Минимальные требования к инфраструктуре позволяют быстро

установить инструмент в любой среде. Инструмент не требует подключения к сторонним сервисам, что критично для изолированных тестовых сред.

2. **Устойчивость к обнаружению** Благодаря отсутствию стандартных сигнатур и минималистичной реализации, инструмент значительно реже детектируется защитными системами.

3. **Эффективность в изолированных средах**

Возможность автономной работы делает инструмент незаменимым для тестирования изолированных сетей, где сторонние решения часто бесполезны.

Таблица 3 Аргументация сложности развертывания

Критерий	Разработанный инструмент	Metasploit (Meterpreter)	Cobalt Strike
Требования к инфраструктуре	Минимальные (только Go-исполняемые файлы[11]).	Средние (зависимость от баз данных и Metasploit серверов).	Высокие (необходим сервер и сложная настройка)
Автономность	Полная: нет зависимости от внешних сервисов.	Частичная: требует периодических обновлений.	Низкая: связь с сервером обязательна.
Настройка интеграций	Простая: базовый REST API.	Средняя: подключение модулей вручную.	Сложная: множество зависимостей.
Использование в изолированных средах	Высокое: автономная работа.	Ограниченное: сложный процесс настройки.	Низкое: серверная зависимость.

Разработанный инструмент обеспечивает основу для построения эффективного процесса автоматизации пентестов. Однако для полного соответствия современным требованиям он нуждается в доработке (Таблица 4).

Таблица 4 Перспективы развития инструмента

Направление улучшений	Цели	Ожидаемые результаты
Интеграция с SIEM	Автоматизация реагирования на инциденты	Улучшение процесса анализа данных
Модули обхода EDR	Повышение устойчивости к обнаружению	Снижение риска блокировки
Расширение функционала API	Поддержка дополнительных операций	Повышение уровня автоматизации

Таким образом, разработанный инструмент продемонстрировал свою эффективность для узкоспециализированных задач и может стать основой для дальнейших разработок в области автоматизации пентестов.

Заключение

В результате анализа возможностей разработанного инструмента удаленного доступа для автоматизации пентестов можно сделать вывод, что он наиболее эффективен в изолированных средах и для выполнения базовых задач управления целевыми системами.

Его основными преимуществами являются простота развертывания, автономность и устойчивость к обнаружению средствами защиты. Эти качества делают инструмент подходящим выбором для небольших команд и специфических сценариев тестирования, где важны минимальная инфраструктура и скорость настройки.

Metasploit предоставляет широкий функционал и лучше подходит для комплексных атак, требующих использования разнообразных эксплойтов, однако его зависимость от внешних сервисов ограничивает применение в изолированных сетях. Cobalt Strike отличается

мощным инструментарием для управления атакующими объектами, но требует значительных ресурсов и сложной настройки, что затрудняет его использование в ограниченных условиях.

Разработанный инструмент, в отличие от крупных решений, ориентирован на простоту и надежность в рамках узкоспециализированных задач. Он может быть использован как дополнение к существующим инструментам пентестинга или как самостоятельное решение для оперативного тестирования систем.

Для выбора подходящего решения важно учитывать специфику задач и требования к инфраструктуре. Если приоритетом является автономная работа в изолированных средах и минимальные затраты на развертывание, разработанный инструмент станет оптимальным выбором. Однако для сложных сценариев, требующих глубокого анализа и интеграции с SIEM, необходимо рассмотреть более мощные платформы, такие как Metasploit или Cobalt Strike.

Список литературы

1. Andress, J. (2014). The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice. Syngress. [Электронный ресурс]. - Режим доступа: https://www.elsevier.com/books/the-basics-of-information-security/andress/978-0-12-800744-0 (дата обращения 29.10.2024).
2. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws. Wiley.
3. Hazlewood, J. (2021). Penetration Testing with the Bash Shell: Security Testing from Command Injection to Secure Shell. No Starch Press.
4. Russell, C., & Van Den Toorn, D. (2018). Hands-On Cybersecurity with Metasploit. Packt Publishing.
5. RFC 7454: BGP Operations and Security. (2015). The Internet Engineering Task Force (IETF). [Электронный ресурс]. - Режим доступа: https://datatracker.ietf.org/doc/html/rfc7454((дата обращения 09.11.2024).
6. Howard, M., & Lipner, S. (2006). The Security Development Lifecycle. Microsoft Press.
7. Louwers, J. (2019). Hands-On Penetration Testing with Go: Develop Pentesting Tools and Techniques Using Go. Packt Publishing.xw
8. Kalsi, H. (2016). Go Programming Blueprints. Packt Publishing.
9. Donovan, A. A., & Kernighan, B. W.). The Go Programming Language. Addison-Wesley Professional [Электронный ресурс]. - Режим доступа: https://www.gopl.io (дата обращения 11.11.2024)
10. Stuart, R., & Stamper, B. (2019). Hacking with Go: Explore the Cyber Security Capabilities of Go by Building Practical Tools. Packt Publishing.
11. The Go Programming Language Specification [Электронный ресурс]. - Режим доступа: https://golang.org/ref/spec (дата обращения 25.11.2024).

References

1. Andress, J. (2014). The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice. Syngress. [Jelektronnyj resurs]. - Rezhim dostupa: https://www.elsevier.com/books/the-basics-of-information-security/andress/978-0-12-800744-0 (data obrashhenija 29.10.2024).
2. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws. Wiley.
3. Hazlewood, J. (2021). Penetration Testing with the Bash Shell: Security Testing from Command Injection to Secure Shell. No Starch Press.
4. Russell, C., & Van Den Toorn, D. (2018). Hands-On Cybersecurity with Metasploit. Packt Publishing.
5. RFC 7454: BGP Operations and Security. (2015). The Internet Engineering Task Force (IETF). [Jelektronnyj resurs]. - Rezhim dostupa: https://datatracker.ietf.org/doc/html/rfc7454((data obrashhenija 09.11.2024).
6. Howard, M., & Lipner, S. (2006). The Security Development Lifecycle. Microsoft Press.
7. Louwers, J. (2019). Hands-On Penetration Testing with Go: Develop Pentesting Tools and Techniques Using Go. Packt Publishing.xw
8. Kalsi, H. (2016). Go Programming Blueprints. Packt Publishing.
9. Donovan, A. A., & Kernighan, B. W.). The Go Programming Language. Addison-Wesley Professional [Jelektronnyj resurs]. - Rezhim dostupa: https://www.gopl.io (data obrashhenija 11.11.2024)
10. Stuart, R., & Stamper, B. (2019). Hacking with Go: Explore the Cyber Security Capabilities of Go by Building Practical Tools. Packt Publishing.
11. The Go Programming Language Specification [Jelektronnyj resurs]. - Rezhim dostupa: https://golang.org/ref/spec (data obrashhenija 25.11.2024).

Сведения об авторах/ Авторлар туралы мәліметтер / Information about the authors

Тютеев Дамир Тасбергенулы - магистрант 1-го курса; специальность информационной безопасности; Евразийский национальный университет имени Л.Н. Гумилева; Республика Казахстан; e-mail: sav3al1.l@gmail.com, ORCID: https://orcid.org/0009-0002-5413-615X

Tyuteyev Damir - 1st year master's degree; specialty of information security; Eurasian National University named after L.N. Gumilyov; The Republic of Kazakhstan; e-mail: sav3al1.l@gmail.com, ORCID: https://orcid.org/0009-0002-5413-615X

Тютеев Дамир Тасбергенулы - 1 курс магистрант; ақпараттық қауіпсіздік мамандығы; Л.Н. Гумилёв атындағы Еуразия ұлттық университеті; Қазақстан Республикасы; e-mail: sav3al1.l@gmail.com, ORCID: https://orcid.org/0009-0002-5413-615X

Сагиндыков Каким Молдабекович - доцент кафедры информационной безопасности; Евразийский национальный университет имени Л.Н. Гумилева; Республика Казахстан; e-mail: sagindykov_km@enu.kz. ORCID: https://orcid.org/0000-0002-2651-403X

Sagindykov Kakim - Associate Professor, Department of Information Security; Eurasian National University named after L.N. Gumilyov; Republic of Kazakhstan; e-mail: sagindykov_km@enu.kz. ORCID: https://orcid.org/0000-0002-2651-403X

Сагиндыков Каким Молдабекович - ақпараттық қауіпсіздік кафедрасының доценті; Л. Н. Гумилев атындағы Еуразия ұлттық университеті; Қазақстан Республикасы; e-mail: sagindykov_km@enu.kz. ORCID: https://orcid.org/0000-0002-2651-403X

Шайханова Айгуль Кайрулаевна - профессор кафедры информационной безопасности; Евразийский национальный университет имени Л.Н. Гумилева; Республика Казахстан; e-mail: shaikhanova_ak@enu.kz. ORCID: https://orcid.org/0000-0001-6006-4813

Shaikhanova Aigul -professor of the department of Information Security; Eurasian National University named after L.N. Gumilyov; Republic of Kazakhstan; e-mail: shaikhanova_ak@enu.kz. ORCID: https://orcid.org/0000-0001-6006-4813

Шайханова Айгуль Кайрулаевна - ақпараттық қауіпсіздік кафедрасының профессор; Л.Н. Гумилёв атындағы Еуразия ұлттық университеті; Қазақстан Республикасы; e-mail: shaikhanova_ak@enu.kz, ORCID: https://orcid.org/0000-0001-6006-4813