МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

По выполнению лабораторных работ по дисциплине

«Программно-аппаратные средства защиты информации»
для студентов

Специальности 10.05.03 Информационная безопасность автоматизированных систем Специализация «Анализ безопасности информационных систем»

Ставрополь
2026

Оглавление

Введение

Программно-аппаратные средства защиты информации представляют собой комплекс технических решений, сочетающих специализированное оборудование и программное обеспечение для обеспечения конфиденциальности, целостности и доступности информации. Их применение особенно критично в современных условиях цифровой трансформации, когда традиционные программные методы защиты уже не могут в полной мере противостоять сложным киберугрозам.

Исторически развитие этих средств связано с эволюцией вычислительных систем. Первые аппаратные решения появились в 1960-х годах для защиты мейнфреймов, а значительный прорыв произошел в 1980-х с созданием первых криптографических процессоров. Современный этап характеризуется широким внедрением таких технологий как Trusted Platform Module (TPM), Hardware Security Modules (HSM) и аппаратно реализованных механизмов доверенной загрузки.

Ключевые особенности программно-аппаратных средств защиты:

1. Физическая изоляция критически важных процессов
2. Аппаратная реализация криптографических алгоритмов
3. Механизмы контроля целостности на уровне микропрограмм
4. Защита от низкоуровневых атак (DMA, side-channel)

Курс лабораторных работ состоит из шестнадцати лабораторных работ, списка литературы и оглавления. Предлагаемое учебное пособие предназначено для студентов, обучающихся по специальности: 10.05.03 «Информационная безопасность автоматизированных систем».

Лабораторная работа №1 Исследование методов криптографической защиты данных (шифрование AES, RSA). Практическое использование OpenSSL для шифрования/дешифрования.

Цель и содержание работы

Цель: исследовать методы криптографической защиты данных на основе алгоритмов AES и RSA, а также освоить практическое применение утилиты OpenSSL для шифрования и дешифрования информации.

Содержание работы:

1. Изучение теоретических основ симметричного (AES) и асимметричного (RSA) шифрования.
2. Практическое применение OpenSSL для генерации ключей, шифрования и дешифрования данных.
3. Анализ особенностей и сфер применения криптографических алгоритмов.

Теоретическое обоснование

1. Криптографическая защита данных

Криптография — наука о методах защиты информации путем ее преобразования в нечитаемый вид. Основные задачи:

- **Конфиденциальность** — защита от несанкционированного доступа.
- **Целостность** — обеспечение неизменности данных.
- **Аутентификация** — подтверждение подлинности отправителя.

2. Алгоритм AES (Advanced Encryption Standard)

- **Тип:** Симметричное шифрование (один ключ для шифрования и дешифрования).
- **Длина ключа:** 128, 192 или 256 бит.
- **Принцип работы:** Блочный шифр, обрабатывающий данные блоками по 128 бит.
- **Сферы применения:** Защита данных в сетях (SSL/TLS), шифрование файлов, VPN.

3. Алгоритм RSA (Rivest-Shamir-Adleman)

- **Тип:** Асимметричное шифрование (публичный и приватный ключи).
- **Основа:** Математическая сложность факторизации больших чисел.
- **Применение:** Цифровые подписи, шифрование сеансовых ключей, SSL/TLS.

4. OpenSSL

OpenSSL — библиотека криптографических функций, поддерживающая:

- Генерацию ключей.
- Шифрование/дешифрование.
- Создание и проверку цифровых подписей.
- Работу с сертификатами.

Аппаратура и материалы

1. Компьютер с ОС Windows/Linux.
2. Установленная утилита OpenSSL.
3. Текстовый файл для тестирования шифрования.

Методика и порядок выполнения работы

1. Установка OpenSSL

- **Linux:** Уже предустановлен или устанавливается командой:

```
sudo apt-get install openssl
```

- **Windows:** Скачать с официального сайта (https://www.openssl.org/) и добавить в PATH.

2. Генерация ключей

Для AES:

```
openssl rand -hex 32 > aes_key.txt  # Генерация 256-битного ключа
```

Для RSA:

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048  # Приватный ключ
openssl rsa -pubout -in private_key.pem -out public_key.pem  # Публичный ключ
```

3. Шифрование и дешифрование AES

Шифрование:

```
openssl enc -aes-256-cbc -salt -in plaintext.txt -out encrypted.aes -pass file:aes_key.txt
```

Дешифрование:

```
openssl enc -d -aes-256-cbc -in encrypted.aes -out decrypted.txt -pass file:aes_key.txt
```

4. Шифрование и дешифрование RSA

Шифрование публичным ключом:

```
openssl pkeyutl -encrypt -in plaintext.txt -pubin -inkey public_key.pem -out encrypted.rsa
```

Дешифрование приватным ключом:

```
openssl pkeyutl -decrypt -in encrypted.rsa -inkey private_key.pem -out decrypted.t
xt
```

5. Проверка целостности данных

Сравнение исходного и дешифрованного файла:

```
diff plaintext.txt decrypted.txt
```

Содержание отчета

1. **Титульный лист**
2. **Теоретическая часть:**
   - Описание AES и RSA.
   - Сравнение симметричного и асимметричного шифрования.
3. **Практическая часть:**
   - Примеры команд OpenSSL.
   - Скриншоты выполнения операций.
4. **Выводы:**
   - Эффективность AES и RSA.
   - Преимущества и недостатки OpenSSL.

Вопросы для защиты работы

1. В чем разница между AES и RSA?
2. Почему RSA медленнее AES?
3. Как обеспечивается безопасность передачи ключей в симметричном шифровании?
4. Какие параметры ключей считаются криптостойкими в 2024 году?
5. Какие уязвимости есть у OpenSSL?

Лабораторная работа №2 Анализ стойкости парольной защиты (перебор хешей с помощью John the Ripper). Исследование методов атак на хеши MD5, SHA-1, bcrypt.

Цель и содержание работы

Цель: Исследовать стойкость парольной защиты путем анализа уязвимостей хеш-функций (MD5, SHA-1, bcrypt) и освоить практическое применение инструмента John the Ripper для проведения атак методом перебора.

Содержание работы:

1. Изучение принципов работы хеш-функций и методов их взлома.
2. Практическое применение John the Ripper для атаки на хеши.
3. Сравнительный анализ уязвимостей MD5, SHA-1 и bcrypt.

Теоретическое обоснование
1. Хеширование паролей

Хеш-функция — алгоритм, преобразующий произвольные данные в строку фиксированной длины (хеш). Основные свойства:

- **Детерминированность:** Одинаковые входные данные дают одинаковый хеш.
- **Необратимость:** Невозможно восстановить исходные данные по хешу.
- **Устойчивость к коллизиям:** Сложность найти два разных входа с одинаковым хешем.

2. Алгоритмы хеширования

- **MD5 (Message Digest Algorithm 5):**
  - Длина хеша: 128 бит.
  - Уязвимости: Высокая вероятность коллизий, быстрое вычисление.
  - Применение: Устаревший, не рекомендуется для защиты паролей.
- **SHA-1 (Secure Hash Algorithm 1):**
  - Длина хеша: 160 бит.
  - Уязвимости: Теоретические атаки на коллизии, быстрее взламывается, чем современные алгоритмы.
- **bcrypt:**
  - Основан на алгоритме Blowfish.
  - Использует "соль" (salt) и адаптивную сложность.
  - Устойчив к перебору благодаря медленному вычислению.

3. Методы атак на хеши

- **Brute-force (полный перебор):** Проверка всех возможных комбинаций символов.
- **Dictionary attack (атака по словарю):** Использование готовых списков популярных паролей.
- **Rainbow tables (радужные таблицы):** Предварительно вычисленные хеши для ускорения взлома.

4. John the Ripper

John the Ripper (JtR) — инструмент для тестирования стойкости паролей, поддерживающий:

- Атаки перебором и по словарю.
- Работу с различными алгоритмами хеширования.
- Оптимизацию для многоядерных процессоров и GPU.

Аппаратура и материалы

1. Компьютер с ОС Linux (Kali Linux рекомендуется).
2. Установленный John the Ripper (sudo apt install john).
3. Текстовый файл с хешами для анализа.

Методика и порядок выполнения работы

1. Подготовка тестовых хешей

Создадим файл hashes.txt с примерами хешей:

```
admin:5f4dcc3b5aa765d61d8327deb882cf99  # MD5 (пароль: password)
user:7c6a180b36896a0a8c02787eeafb0e4c  # SHA-1 (пароль: hello123)
root:$2a$12$N9qo8uLOickgx2ZMRZoMy.Mrq4W7.S.2lZ0CwAdZj0l8FKAZ.TmWq  # bcrypt (пароль: root)
```

2. Запуск John the Ripper

Атака по словарю:

```
john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
```

Атака перебором (маска "a?l?l?l?l" — 4 буквы):

```
john --mask=?a?l?l?l?l hashes.txt
```

3. Анализ результатов

Просмотр взломанных паролей:

```
john --show hashes.txt
```

Пример вывода:

```
admin:password
user:hello123
root:root
```

4. Сравнение времени взлома

- **MD5:** Взламывается за секунды даже без GPU.
- **SHA-1:** Требует больше времени, но уязвим к оптимизированным атакам.
- **bcrypt:** Значительно медленнее из-за адаптивной сложности.

Содержание отчета

1. **Титульный лист**
2. **Теоретическая часть:**
   - Описание MD5, SHA-1, bcrypt.
   - Принципы работы John the Ripper.
3. **Практическая часть:**

- Примеры команд для взлома хешей.
- Сравнение времени взлома для разных алгоритмов.

4. **Выводы:**
   - Рекомендации по выбору алгоритмов хеширования.
   - Методы защиты от атак перебора.

Вопросы для защиты работы

1. Чем отличается хеш от шифрования?
2. Почему MD5 считается ненадежным?
3. Как "соль" (salt) повышает стойкость парольной защиты?
4. В чем преимущество bcrypt перед SHA-1?
5. Какие методы защиты от John the Ripper вы знаете?

Лабораторная работа №3 Настройка и тестирование межсетевого экрана (iptables/firewalld). Создание правил фильтрации трафика, защита от DDoS.

Цель и содержание работы

Цель: Освоить настройку межсетевых экранов iptables и firewalld для фильтрации сетевого трафика и защиты от DDoS-атак.

Содержание работы:

1. Изучение принципов работы межсетевых экранов.
2. Настройка базовых правил фильтрации трафика.
3. Конфигурация защиты от DDoS-атак.
4. Тестирование работоспособности правил.

Теоретическое обоснование

1. Межсетевые экраны (фаерволы)

Межсетевой экран — программный или аппаратный компонент, контролирующий сетевой трафик на основе заданных правил. Основные функции:

- **Фильтрация пакетов:** Блокировка или разрешение трафика по IP-адресам, портам и протоколам.
- **Преобразование сетевых адресов (NAT):** Маскировка внутренних IP-адресов.
- **Защита от атак:** Ограничение числа соединений, блокировка подозрительного трафика.

2. iptables vs firewalld

- **iptables:**
  - Традиционный инструмент для Linux.
  - Работает с таблицами (filter, nat, mangle) и цепочками (INPUT, OUTPUT, FORWARD).
  - Требует ручного управления правилами.
- **firewalld:**
  - Современная замена iptables с динамическим управлением.
  - Использует зоны (public, home, internal) и сервисы.
  - Поддерживает runtime и permanent настройки.

3. Типы DDoS-атак и методы защиты

- **SYN-флуд:** Нагрузка на сервер полуоткрытыми соединениями.
  Защита: Ограничение числа SYN-пакетов в секунду.
- **UDP-флуд:** Перегрузка канала UDP-пакетами.
  Защита: Блокировка ненужных UDP-портов.
- **HTTP-флуд:** Множество запросов к веб-серверу.
  Защита: Rate limiting (ограничение запросов с одного IP).

Аппаратура и материалы

1. Компьютер с ОС Linux (CentOS/RHEL или Ubuntu).
2. Установленные пакеты:
   ```
   sudo apt install iptables firewalld    # Для Ubuntu/Debian
   sudo yum install iptables firewalld    # Для CentOS/RHEL
   ```
3. Доступ к терминалу с root-правами.

Методика и порядок выполнения работы

1. Настройка iptables

Базовые правила:

```
# Разрешить локальный трафик
iptables -A INPUT -i lo -j ACCEPT
# Разрешить установленные соединения
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
# Разрешить SSH (порт 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# Запретить весь остальной входящий трафик
iptables -P INPUT DROP
iptables -P FORWARD DROP
# Сохранить правила
iptables-save > /etc/iptables.rules
```

Защита от SYN-флуда:

```
iptables -N SYN_FLOOD
iptables -A INPUT -p tcp --syn -j SYN_FLOOD
iptables -A SYN_FLOOD -m limit --limit 10/s --limit-burst 20 -j RETURN
iptables -A SYN_FLOOD -j DROP
```

2. Настройка firewalld

Добавление сервисов:

```
firewall-cmd --permanent --add-service=ssh
firewall-cmd --permanent --add-service=http
firewall-cmd --reload
```

Защита от DDoS:

```
# Ограничение соединений с одного IP
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source address="192
.168.1.0/24" limit value="10/m" accept'
```

3. Тестирование правил

- Проверка блокировки ping:

  ```
  iptables -A INPUT -p icmp -j DROP
  ping <IP-адрес>  # Должен выдать "Request timeout"
  ```

- Анализ логов:

  ```
  tail -f /var/log/syslog | grep iptables
  ```

Содержание отчета

1. **Титульный лист**
2. **Теоретическая часть:**
   - Принципы работы iptables и firewalld.
   - Методы защиты от DDoS.
3. **Практическая часть:**
   - Примеры настроек правил.
   - Результаты тестирования.
4. **Выводы:**
   - Сравнение iptables и firewalld.
   - Эффективность защиты от атак.

Вопросы для защиты работы

1. Чем отличается цепочка INPUT от FORWARD в iptables?
2. Как firewalld упрощает управление правилами?
3. Какие параметры ограничивают SYN-флуд?
4. Почему важно сохранять правила iptables?
5. Как проверить активные правила firewalld?

Лабораторная работа №4 Исследование работы VPN (OpenVPN, WireGuard). Настройка защищенного туннеля, анализ трафика.

Цель и содержание работы

Цель: Исследовать принципы работы VPN-технологий на примере OpenVPN и WireGuard, освоить настройку защищенных туннелей и анализ VPN-трафика.

Содержание работы:

1. Изучение архитектуры и криптографических механизмов OpenVPN и WireGuard.
2. Практическая настройка VPN-серверов и клиентов.
3. Анализ характеристик и безопасности VPN-соединений.

Теоретическое обоснование
1. Технология VPN

Виртуальная частная сеть (VPN) создает зашифрованный туннель между устройствами через публичные сети. Основные функции:

- **Конфиденциальность:** Шифрование всего трафика.
- **Аутентичность:** Проверка подлинности участников.
- **Целостность:** Защита от подмены данных.

2. OpenVPN

- **Протокол:** Использует TLS/SSL.
- **Криптография:** Поддерживает AES-256, RSA-4096.
- **Гибкость:** Работает через TCP/UDP, обходит NAT.
- **Недостатки:** Высокая нагрузка на ЦП, сложная настройка.

3. WireGuard

- **Протокол:** Современный, на основе Noise Protocol.
- **Криптография:** ChaCha20, Poly1305, Curve25519.
- **Преимущества:** Высокая скорость, простота конфигурации.
- **Ограничения:** Фиксированный набор алгоритмов.

4. Сравнение технологий

Параметр	**OpenVPN**	**WireGuard**
Скорость	Средняя	Высокая
Нагрузка на ЦП	Высокая	Низкая
Конфигурация	Сложная	Простая
Совместимость	Кроссплатформенный	Linux/Windows

Аппаратура и материалы

1. Два компьютера с ОС Linux (Ubuntu 20.04+).
2. Установленные пакеты:

```
# Для OpenVPN
sudo apt install openvpn easy-rsa

# Для WireGuard
sudo apt install wireguard resolvconf
```

3. Сетевые интерфейсы с доступом в Интернет.

Методика и порядок выполнения работы

1. Настройка OpenVPN

Генерация сертификатов:

```
make-cadir ~/openvpn-ca
cd ~/openvpn-ca
source vars
./clean-all
./build-ca
./build-key-server server
./build-key client1
```

Конфигурация сервера (/etc/openvpn/server.conf):

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh2048.pem
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
keepalive 10 120
tls-auth ta.key 0
cipher AES-256-CBC
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
verb 3
```

Запуск сервера:

```
sudo systemctl start openvpn@server
```

2. Настройка WireGuard

Генерация ключей:

```
wg genkey | tee privatekey | wg pubkey > publickey
```

Конфигурация сервера (/etc/wireguard/wg0.conf):

```
[Interface]
Address = 10.0.0.1/24
ListenPort = 51820
PrivateKey = <СЕРВЕРНЫЙ_ПРИВАТНЫЙ_КЛЮЧ>

[Peer]
PublicKey = <КЛИЕНТСКИЙ_ПУБЛИЧНЫЙ_КЛЮЧ>
AllowedIPs = 10.0.0.2/32
```

Активация интерфейса:

```
sudo wg-quick up wg0
```

3. Анализ трафика

Захват пакетов:

```
sudo tcpdump -i tun0 -w vpn_traffic.pcap
```

Анализ в Wireshark:

1. Открыть файл vpn_traffic.pcap.
2. Проверить отсутствие открытых данных (все пакеты должны быть зашифрованы).

Содержание отчета

1. **Титульный лист** (название работы, ФИО, группа, преподаватель).
2. **Теоретическая часть:**
   - Принципы работы OpenVPN и WireGuard.
   - Сравнение алгоритмов шифрования.
3. **Практическая часть:**
   - Конфигурационные файлы серверов.
   - Результаты анализа трафика.
4. **Выводы:**
   - Преимущества и недостатки каждой технологии.
   - Рекомендации по выбору VPN-решения.

Вопросы для защиты работы

1. Чем отличается TLS-туннель OpenVPN от протокола WireGuard?
2. Почему WireGuard считается более производительным?
3. Как обеспечивается аутентификация в OpenVPN?
4. Какие ключевые файлы необходимы для работы WireGuard?
5. Как проверить состояние WireGuard-интерфейса?

Лабораторная работа №5 Защита от сниффинга (анализ сетевого трафика в Wireshark, защита с помощью SSL/TLS). Перехват и анализ незашифрованных данных.

1. Цель работы

Исследование методов перехвата и анализа сетевого трафика, изучение способов защиты информации от сниффинга с использованием современных криптографических протоколов.

2. Теоретическая часть

2.1. Понятие сниффинга

Сниффинг (от англ. "sniffing" - нюхать) - процесс перехвата и анализа сетевого трафика с целью получения конфиденциальной информации. Атаки сниффинга особенно опасны в незащищенных сетях, где данные передаются в открытом виде.

2.2. Методы перехвата трафика

1. **Пассивный перехват** - анализ трафика без воздействия на сеть
2. **Активный перехват** - использование техник типа ARP-spoofing
3. **Анализ беспроводных сетей** - перехват WiFi-трафика

2.3. Защитные механизмы

- **Шифрование данных** (SSL/TLS, IPsec)
- **Использование VPN**
- **Сегментация сети**
- **Системы обнаружения вторжений**

2.4. Протокол SSL/TLS

SSL (Secure Sockets Layer) и его преемник TLS (Transport Layer Security) - криптографические протоколы, обеспечивающие защищенную передачу данных по сети. Основные функции:

- Аутентификация сервера
- Шифрование передаваемых данных
- Обеспечение целостности информации

3. Практическая часть

3.1. Оборудование и ПО

1. Компьютер с ОС Windows/Linux
2. Программа Wireshark
3. Веб-сервер с поддержкой HTTPS
4. Утилиты: tcpdump, openssl

3.2. Ход работы

Этап 1. Перехват незащищенного трафика

1. Запустить Wireshark и начать захват трафика
2. Выполнить вход на HTTP-сайт (без SSL)
3. В фильтрах Wireshark установить:
   ```
   http.request.method == "POST"
   ```
4. Найти и проанализировать перехваченные учетные данные

Этап 2. Анализ защищенного соединения

1. Подключиться к HTTPS-сайту
2. В Wireshark применить фильтр:
   ```
   ssl.handshake
   ```
3. Просмотреть процесс установления TLS-соединения
4. Убедиться в невозможности просмотра содержимого пакетов

Этап 3. Настройка SSL/TLS на сервере

1. Сгенерировать SSL-сертификат:
   ```
   openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
   ```
2. Настроить веб-сервер (Apache/Nginx) на использование SSL

Этап 4. Тестирование защиты

1. Повторить попытку перехвата трафика
2. Проанализировать различия в результатах

4. Результаты и анализ

4.1. Таблица сравнения

Параметр	**HTTP-трафик**	**HTTPS-трафик**
Возможность перехвата данных	Да	Нет
Видимость содержимого пакетов	Полная	Только метаданные

Параметр	HTTP-трафик	HTTPS-трафик
Защита от MITM-атак	Отсутствует	Обеспечивается

4.2. Выводы

1. Незашифрованный трафик уязвим к перехвату
2. SSL/TLS эффективно защищает от сниффинга
3. Правильная настройка криптографических протоколов - необходимое условие безопасности

5. Вопросы для самопроверки

1. Какие виды сниффинга вы знаете?
2. Как работает протокол TLS?
3. Почему ARP-spoofing эффективен в локальных сетях?
4. Какие методы защиты от сниффинга наиболее эффективны?
5. Как проверить подлинность SSL-сертификата?

Лабораторная работа №6 Изучение механизмов защиты от вредоносного ПО (антивирусы, песочницы). Анализ поведения вирусов в изолированной среде.

1. Цель работы

Исследование современных методов защиты от вредоносного программного обеспечения, включая анализ работы антивирусных систем и технологий песочниц. Практическое изучение поведения вредоносных программ в изолированной среде.

2. Теоретическая часть

2.1. Вредоносное программное обеспечение

Вредоносное ПО (malware) — программы, предназначенные для несанкционированного доступа к данным, повреждения системы или нарушения ее нормальной работы. Основные типы:

- **Вирусы** — заражают файлы и распространяются через них.
- **Черви** — самореплицирующиеся программы, распространяются по сети.
- **Трояны** — маскируются под легитимное ПО для кражи данных.
- **Рекламное ПО (adware)** — навязывает рекламу.
- **Шпионское ПО (spyware)** — собирает информацию о пользователе.

2.2. Методы защиты от вредоносного ПО

2.2.1. Антивирусные системы

Антивирусы используют следующие технологии обнаружения угроз:

- **Сигнатурный анализ** — поиск известных шаблонов вредоносного кода.
- **Эвристический анализ** — выявление подозрительного поведения программ.
- **Облачные технологии** — проверка файлов через онлайн-базы.
- **Поведенческий анализ** — мониторинг активности процессов в реальном времени.

2.2.2. Песочницы (Sandbox)

Песочница — изолированная среда для запуска подозрительных программ без риска заражения основной системы. Принципы работы:

- **Виртуализация** — программа выполняется в виртуальной машине.
- **Эмуляция API** — вредоносный код не получает доступ к реальным системным вызовам.
- **Контроль ресурсов** — ограничение доступа к файлам, сети, реестру.

2.3. Современные угрозы и защита

- **Файловые вирусы → Сигнатурный анализ + эвристика**.
- **Руткиты → Поведенческий анализ + песочницы**.
- **Криптоджекинг → Мониторинг нагрузки ЦП/GPU**.

3. Практическая часть

3.1. Оборудование и ПО

1. **Физическая машина**:
   - ОС: Windows 10/11 или Linux (Ubuntu).
   - Антивирус: Kaspersky, Avast, Windows Defender.
2. **Виртуальная машина**:
   - VirtualBox/VMware с изолированной ОС.
3. **Песочницы**:
   - Sandboxie, Windows Sandbox, Cuckoo Sandbox.
4. **Тестовые образцы вредоносного ПО** (в контролируемых условиях!).

3.2. Ход работы

Этап 1. Анализ работы антивируса

1. Запустить антивирус и проверить настройки.
2. Загрузить тестовый EICAR-файл (безопасный аналог вируса):

```
text
```

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

3. Проверить реакцию антивируса (должен быть мгновенно обнаружен).
4. Отключить сигнатурный анализ и повторить тест.

Этап 2. Исследование песочницы

1. Запустить Windows Sandbox (для Windows Pro/Enterprise):

```
WindowsSandbox.exe
```

2. Скопировать в песочницу тестовый исполняемый файл (например, утилиту для мониторинга сети).
3. Проанализировать, какие системные вызовы блокируются.

Этап 3. Анализ поведения вируса в изолированной среде

1. В виртуальной машине (без сетевого доступа!) запустить образец вредоносного ПО.
2. Использовать инструменты мониторинга:
   - **Process Monitor** (Windows) — логирование действий.
   - **Wireshark** — анализ сетевой активности.
   - **Regshot** — сравнение состояния реестра до и после заражения.

4. Результаты и выводы

4.1. Таблица наблюдений

Действие	**Реакция антивируса**	**Поведение в песочнице**
Запуск EICAR-файла	Блокировка	Нет реакции (без сигнатур)
Попытка записи в системные файлы	Предупреждение	Изолированное выполнение
Сетевые соединения	Блокировка	Ограничено песочницей

4.2. Выводы

1. **Антивирусы** эффективны против известных угроз, но уязвимы к zero-day атакам.
2. **Песочницы** предотвращают повреждение системы, но требуют ресурсов.
3. **Комбинированный подход** (антивирус + песочница) обеспечивает максимальную защиту.

5. Вопросы для защиты

1. Чем отличается вирус от червя?
2. Как работает эвристический анализ?
3. Какие процессы мониторит песочница?
4. Почему криптоджекинг сложно обнаружить?
5. Как защититься от файловых вирусов?

Лабораторная работа №7 Исследование методов аутентификации (двухфакторная аутентификация, OTP, биометрия). Настройка 2FA на примере Google Authenticator.

1. Цель работы

Изучение современных методов аутентификации, включая двухфакторную аутентификацию (2FA), одноразовые пароли (OTP) и биометрические технологии. Практическое освоение настройки двухфакторной аутентификации с использованием Google Authenticator.

2. Теоретическая часть

2.1. Методы аутентификации

Аутентификация - процесс проверки подлинности пользователя или системы. Основные методы:

1. **Однофакторная аутентификация**
   - Только что-то одно: пароль, PIN-код
   - Уязвима к перехвату и подбору
2. **Двухфакторная аутентификация (2FA)**
   - Комбинация двух факторов:
     - Что-то знаете (пароль)
     - Что-то имеете (токен, телефон)
   - Повышает безопасность в 10-100 раз
3. **Многофакторная аутентификация**
   - Добавляет биометрию (отпечаток, лицо)

2.2. Технологии OTP (One-Time Password)

1. **TOTP (Time-based OTP)**
   - Пароль меняется каждые 30 сек (Google Authenticator)
   - Основан на RFC 6238
2. **HOTP (HMAC-based OTP)**
   - Пароль меняется после каждого использования
   - Описан в RFC 4226
3. **SMS/Email OTP**

- Менее безопасный вариант

2.3. Биометрические методы

1. Отпечатки пальцев
2. Распознавание лица
3. Сканирование сетчатки
4. Распознавание голоса

3. Практическая часть

3.1. Оборудование и ПО

1. Смартфон с ОС Android/iOS
2. Приложение Google Authenticator
3. Веб-сервис с поддержкой 2FA (например, GitHub, Google)
4. Компьютер с доступом в интернет

3.2. Настройка 2FA на примере GitHub

Шаг 1. Включение 2FA в настройках GitHub

1. Зайти в Settings → Security → Two-factor authentication
2. Выбрать "Configure using an app"

Шаг 2. Настройка Google Authenticator

1. Открыть приложение на смартфоне
2. Нажать "+" → "Сканировать QR-код"
3. Отсканировать код с экрана GitHub

Шаг 3. Проверка работы

1. Выйти из аккаунта GitHub
2. При повторном входе после пароля ввести код из Authenticator

3.3. Анализ механизма TOTP

1. **Секретный ключ**
   - При сканировании QR передается ключ в формате Base32
   - Пример: JBSWY3DPEHPK3PXP
2. **Генерация кода**

```
import pyotp
totp = pyotp.TOTP("JBSWY3DPEHPK3PXP")
print(totp.now()) # Выведет текущий 6-значный код
```

3. **Валидация**
   - Сервер и приложение синхронизированы по времени
   - Допуск ±1 интервал (30 сек)

4. Результаты и анализ

4.1. Сравнение методов аутентификации

Метод	Преимущества	Недостатки
Пароль	Простота	Уязвимость к фишингу
SMS OTP	Доступность	Перехват через SIM-свопинг
TOTP (Google Auth)	Оффлайн работа	Потеря устройства
Биометрия	Удобство	Ложные срабатывания

4.2. Выводы

1. 2FA значительно повышает безопасность
2. Google Authenticator - надежное решение без зависимости от сети
3. Биометрия удобна, но требует специального оборудования

5. Вопросы для защиты

1. Чем TOTP отличается от HOTP?
2. Почему SMS-коды менее безопасны?
3. Как восстановить доступ при потере телефона с Authenticator?
4. Какие альтернативы Google Authenticator существуют?
5. В чем преимущества аппаратных токенов?

Лабораторная работа №8 Анализ уязвимостей веб-приложений (SQL-инъекции, XSS, CSRF). Практическое тестирование с использованием Burp Suite

1. Цель работы

Исследование распространенных уязвимостей веб-приложений и освоение практических методов их выявления с использованием профессионального инструментария тестирования безопасности.

2. Теоретическая часть

2.1. Классификация веб-уязвимостей

1. **SQL-инъекции (SQLi)**
   - Механизм: внедрение SQL-кода через параметры запросов
   - Уровень опасности: критический (CVSS: 9.8)
   - Последствия:
     - Несанкционированный доступ к БД
     - Обход аутентификации
     - Эскалация привилегий
2. **Межсайтовый скриптинг (XSS)**
   - Типы:

        - Reflected (отраженный)
        - Stored (постоянный)
        - DOM-based
    - Векторы атак:

```
<script>alert(document.cookie)</script>
```

3. **Межсайтовая подделка запросов (CSRF)**
    - Условия эксплуатации:
        - Авторизованная сессия
        - Отсутствие CSRF-токенов

2.2. Методология тестирования

1. **Этапы:**
    - Разведка
    - Анализ поверхности атаки
    - Эксплуатация уязвимостей
    - Документирование результатов
2. **Инструменты:**
    - Burp Suite Professional
    - OWASP ZAP
    - SQLmap

3. Практическая часть

3.1. Подготовка тестовой среды

Оборудование:

1. Kali Linux 2023.3
2. Burp Suite Community/Professional
3. Уязвимый стенд DVWA (Damn Vulnerable Web App)

Настройка:

```
git clone https://github.com/digininja/DVWA
docker-compose up -d
```

3.2. Обнаружение SQL-инъекций

Шаг 1. Конфигурация Burp Suite

1. Включить прокси-сервер (127.0.0.1:8080)
2. Настроить браузер для работы через прокси

Шаг 2. Тестирование формы входа

1. Перехватить запрос:

```
POST /login.php HTTP/1.1
...
username=admin&password=test
```

2. Модифицировать параметры:

```
username=admin'-- &password=
```

Шаг 3. Верификация уязвимости

- Успешная авторизация → подтверждение SQLi

3.3. Выявление XSS-уязвимостей

Методика:

1. Ввести в поле поиска:

```
<svg/onload=alert(1)>
```

2. Анализ ответа сервера

Автоматизированное тестирование:

1. Использование Burp Scanner
2. Анализ отчета

3.4. Проверка на CSRF

Создание PoC:

```
<form action="http://victim-site.com/transfer" method="POST">
  <input type="hidden" name="amount" value="1000">
  <input type="hidden" name="account" value="attacker">
</form>
<script>document.forms[0].submit();</script>
```

4. Результаты и анализ

4.1. Сравнительная таблица

Уязвимость	**Метод обнаружения**	**Уровень риска**	**Рекомендации**
SQLi	Ручное тестирование	Critical	Использование ORM
XSS	Автосканирование	High	CSP-политики
CSRF	Анализ кода	Medium	CSRF-токены

4.2. Выводы

1. Burp Suite обеспечивает комплексный анализ безопасности
2. 78% уязвимостей связаны с недостаточной валидацией входных данных
3. Современные фреймворки снижают риски, но требуют дополнительной настройки

5. Вопросы для защиты

1. Объясните принцип слепой SQL-инъекции
2. В чем отличие Stored XSS от Reflected?
3. Почему CSRF-атаки эффективны против REST API?
4. Какие функции Burp Intruder наиболее полезны?
5. Как реализовать защиту от DOM-based XSS?

Лабораторная работа №9 Изучение систем обнаружения вторжений (Snort, Suricata). Настройка IDS и анализ атакующих пакетов

1. Цель работы

Освоение принципов работы систем обнаружения вторжений (IDS), практическая настройка Snort и Suricata для выявления сетевых атак, анализ детектируемых угроз.

2. Теоретическая часть

2.1. Системы обнаружения вторжений

Классификация IDS:

1. **По размещению:**
   - Сетевые (NIDS)
   - Хостные (HIDS)
2. **По методу обнаружения:**
   - Сигнатурные
   - Аномалийные
   - Гибридные

Сравнение Snort и Suricata:

Характеристика	**Snort**	**Suricata**
Многопоточность	Нет	Да
Поддержка IPv6	Частичная	Полная

Характеристика	Snort	Suricata
Производительность	До 1 Гбит/с	До 10 Гбит/с

2.2. Архитектура Snort

1. **Модули:**
   - Декодер пакетов
   - Препроцессоры
   - Детектор
   - Система вывода
2. **Формат правил:**

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH brute force"; flo
w:to_server; detection_filter:track by_src, count 5, seconds 60; sid:1000001;)
```

2.3. Типы анализируемых атак

1. Сканирование портов
2. DoS/DDoS
3. Эксплуатация уязвимостей
4. Аномальная сетевая активность

3. Практическая часть

3.1. Подготовка тестовой среды

Оборудование:

1. Сервер: Ubuntu Server 22.04 LTS
2. ПК для атак: Kali Linux 2023.3
3. Сетевой анализатор: Wireshark

Установка Snort:

```
sudo apt update
sudo apt install snort -y
sudo snort -V # Проверка версии
```

3.2. Базовая настройка Snort

1. **Конфигурационный файл:**

   ```
   sudo nano /etc/snort/snort.conf
   ```

   Изменить параметры:

   ```
   ipvar HOME_NET 192.168.1.0/24
   ipvar EXTERNAL_NET !$HOME_NET
   ```
2. **Запуск в режиме IDS:**

   ```
   sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
   ```

3.3. Тестирование работы

Генерация атакующего трафика:

1. Сканирование Nmap:
   ```
   nmap -sS 192.168.1.100
   ```
2. SSH-брутфорс:
   ```
   hydra -l root -P passlist.txt ssh://192.168.1.100
   ```

Анализ логов Snort:

```
sudo tail -f /var/log/snort/alert
```

Пример предупреждения:

```
[**] [1:1000001:1] SSH brute force attempt [**]
[Priority: 1] 192.168.1.50:54321 -> 192.168.1.100:22
```

3.4. Настройка Suricata

1. **Установка:**
   ```
   sudo apt install suricata -y
   ```
2. **Активация правил:**
   ```
   sudo suricata-update
   sudo systemctl enable --now suricata
   ```
3. **Анализ событий:**
   ```
   sudo tail -f /var/log/suricata/fast.log
   ```

4. Результаты и анализ

4.1. Сравнение эффективности

Тест	Snort	Suricata
Обнаружение сканирования	92%	95%
Ложные срабатывания	8%	5%
Загрузка CPU	65%	45%

4.2. Выводы

1. Suricata демонстрирует лучшую производительность
2. Snort проще в настройке для небольших сетей
3. Для enterprise-решений рекомендуется Suricata

5. Вопросы для защиты

1. Объясните архитектуру multi-thread в Suricata
2. Как работают препроцессоры в Snort?
3. В чем преимущество правил ET Open?
4. Как снизить уровень ложных срабатываний?
5. Какие методы обнаружения аномалий используются в современных IDS?

Лабораторная работа №10 Защита данных на уровне жесткого диска (шифрование BitLocker, VeraCrypt). Создание зашифрованных контейнеров и анализ устойчивости к взлому

1. Цель работы

Исследование технологий полнодискового шифрования и создание защищенных контейнеров с помощью BitLocker и VeraCrypt. Анализ криптостойкости и методов противодействия атакам на зашифрованные данные.

2. Теоретическая часть

2.1. Технологии дискового шифрования

Принципы работы:

- Прозрачное шифрование/дешифрование "на лету"
- Использование аппаратных возможностей TPM (Trusted Platform Module)
- Многофакторная аутентификация

Сравнительные характеристики:

Параметр	**BitLocker**	**VeraCrypt**
Производитель	Microsoft	Open Source
Алгоритмы	AES-128/256	AES, Serpent, Twofish
Поддержка ОС	Windows	Кроссплатформенный
Аутентификация	TPM+Пин	Пароль+ключ-файл

2.2. Криптографические алгоритмы

1. **AES (Advanced Encryption Standard)**
   - Длина ключа: 128/256 бит
   - Режимы: XTS, CBC
   - Сертификация: FIPS 140-2
2. **Каскадное шифрование** (AES-Twofish-Serpent)
   - Повышенная стойкость
   - Снижение производительности на 15-20%

2.3. Угрозы и методы защиты

1. **Атаки холодного запуска**
2. **Кейлоггеры**

3. **Эксплуатация уязвимостей загрузчика**

3. Практическая часть

3.1. Настройка BitLocker

Шаг 1. Включение функции

1. Открыть "Управление дисками"
2. ПКМ по разделу → "Включить BitLocker"
3. Выбрать метод разблокировки (пароль, смарт-карта)

Шаг 2. Настройка политик

```
manage-bde -on C: -RecoveryPassword -UsedSpaceOnly
```

Шаг 3. Проверка состояния

```
manage-bde -status
```

3.2. Создание контейнера VeraCrypt

1. **Создание файла-контейнера**
   - Размер: 1 ГБ
   - Алгоритм: AES-Twofish-Serpent
   - Файловая система: NTFS
2. **Монтирование:**

   ```
   Выбрать файл → Ввести пароль → Назначить букву диска
   ```

3.3. Тестирование устойчивости

1. **Атака перебором** (John the Ripper):

   ```
   ./john --format=vcrypt --wordlist=rockyou.txt container.hc
   ```
2. **Анализ загрузочного сектора** (hex-редактор):

   ```
   xxd /dev/sda1 | head -n 50
   ```

4. Результаты и анализ

4.1. Сравнение производительности

Операция	**BitLocker**	**VeraCrypt**
Шифрование 10 ГБ	12 мин	18 мин
Скорость чтения	320 МБ/с	280 МБ/с
Загрузка ОС	+8 сек	+15 сек

4.2. Выводы

1. BitLocker лучше интегрирован в Windows
2. VeraCrypt обеспечивает повышенную безопасность
3. Для максимальной защиты рекомендуется:
   - Аппаратный TPM

- Многофакторная аутентификация
- Регулярное обновление

5. Вопросы для защиты

1. Как работает технология TPM?
2. В чем преимущество каскадного шифрования?
3. Как защититься от атак холодного запуска?
4. Почему VeraCrypt считается более безопасным?
5. Как организовать безопасное хранение ключей?

Лабораторная работа №11 Исследование методов защиты от фишинга (анализ поддельных писем, обучение пользователей). Разработка тестового фишингового письма и его детекция

1. Цель работы

Изучение современных фишинговых техник, разработка тестовых фишинговых писем для оценки уязвимости пользователей и тестирования систем защиты корпоративной почты.

2. Теоретическая часть

2.1. Фишинг: понятие и виды

Определение: Мошенническая техника, направленная на получение конфиденциальных данных через поддельные ресурсы.

Классификация атак:

1. **По каналу доставки:**
   - Email-фишинг (86% атак)
   - СМС (smishing)
   - Голосовые звонки (vishing)
2. **По цели:**
   - Кредитные данные (45%)
   - Учетные записи (32%)
   - Корпоративные данные (23%)

2.2. Технические признаки фишинга

1. **Анализ заголовков:**
   - Несоответствие Return-Path и From
   - Поддельные DKIM/SPF записи
2. **Визуальные маркеры:**
   - Некачественные логотипы
   - Ошибки в доменных именах (paypa1.com вместо paypal.com)

3. **Лингвистические особенности:**
    - Срочность ("Ваш аккаунт будет заблокирован!")
    - Ошибки в тексте

2.3. Системы защиты

1. **Технические решения:**
    - Антифишинговые фильтры (Office 365 ATP)
    - DMARC/DKIM/SPF
2. **Обучение пользователей:**
    - Проведение тестовых фишинговых атак
    - Интерактивные тренинги

3. Практическая часть

3.1. Подготовка тестовой среды

Оборудование:

1. Почтовый сервер (MailHog для тестирования)
2. Фреймворк Gophish (Open Source)
3. Аналитическая платформа Wireshark

3.2. Создание фишингового письма

Шаг 1. Разработка сценария

```
Тема: "Срочно! Проверка учетной записи"
Тело:
"Уважаемый пользователь,
Система безопасности обнаружила подозрительную активность.
Для проверки перейдите по ссылке:
<a href="http://fake-bank.ru/verify">Подтвердить данные</a>"
```

Шаг 2. Настройка Gophish

1. Импорт списка тестовых адресов
2. Настройка SMTP-сервера
3. Создание landing page

3.3. Анализ результатов

1. **Открытие писем:** 58%
2. **Переход по ссылкам:** 23%
3. **Ввод данных:** 7%

3.4. Тестирование систем защиты

1. **Проверка SPF:**

```
dig TXT fake-bank.ru
```

2. **Анализ заголовков:**

```
Received-SPF: Fail
X-Phish-Score: 8.5/10
```

4. Результаты и анализ

4.1. Эффективность методов

Метод защиты	Обнаружение	Ложные срабатывания
SPF/DKIM	92%	3%
Контент-анализ	78%	15%
Обучение пользователей	64%	-

4.2. Выводы

1. Технические методы эффективны против 85% атак
2. 15% сложных атак требуют обучения пользователей
3. Регулярные тесты снижают уязвимость на 40%

5. Вопросы для защиты

1. Как работает DMARC-политика?
2. Какие техники используют в таргетированном фишинге?
3. Как обнаружить подмену sender в письме?
4. Почему vishing опаснее email-фишинга?
5. Какие API используют антифишинговые системы?

Лабораторная работа №12 Настройка и тестирование систем контроля доступа (SELinux, AppArmor). Ограничение прав процессов, предотвращение эксплуатации уязвимостей

1. Цель работы

Исследование механизмов принудительного контроля доступа (MAC) в операционных системах Linux, практическое освоение настройки SELinux и AppArmor для защиты системных ресурсов и ограничения прав процессов.

2. Теоретическая часть

2.1. Системы принудительного контроля доступа

Принципы работы:

- Разделение субъектов (процессы) и объектов (файлы, сокеты)
- Централизованное управление политиками безопасности
- Запрет по умолчанию (default-deny)

2.2. Сравнение технологий

Характеристика	SELinux	AppArmor
Производитель	NSA	Canonical
Подход	Мандатный (MLS)	Профильный
Сложность	Высокая	Средняя
Поддержка дистрибутивов	RHEL, CentOS	Ubuntu, Debian

2.3. Ключевые концепции

Для SELinux:

- Контексты безопасности (user:role:type:level)
- Политики (targeted, strict, mls)
- Утилиты: semanage, restorecon, audit2allow

Для AppArmor:

- Профили в /etc/apparmor.d/
- Режимы (enforce, complain)
- Инструменты: aa-genprof, aa-logprof

3. Практическая часть

3.1. Подготовка среды

Оборудование:

1. Виртуальные машины с:
   - CentOS 9 (SELinux)
   - Ubuntu 22.04 (AppArmor)
2. Тестовое веб-приложение (Nginx)

3.2. Настройка SELinux

Шаг 1. Проверка состояния:

```
sestatus
getenforce
```

Шаг 2. Создание политики для веб-сервера:

```
# Разрешить доступ к нестандартному каталогу
semanage fcontext -a -t httpd_sys_content_t "/webdata(/.*)?"
```

```
restorecon -Rv /webdata
```

Шаг 3. Анализ логов:

```
ausearch -m avc -ts recent
```

3.3. Конфигурация AppArmor

Шаг 1. Генерация профиля для Nginx:

```
aa-genprof /usr/sbin/nginx
```

Шаг 2. Редактирование правил:

```
nano /etc/apparmor.d/usr.sbin.nginx
```

Добавить:

```
/webdata/** r,
```

Шаг 3. Применение изменений:

```
apparmor_parser -r /etc/apparmor.d/usr.sbin.nginx
```

3.4. Тестирование защиты

1. **Попытка доступа к запрещенным файлам:**
   ```
   sudo -u nginx touch /etc/shadow
   ```
2. **Анализ реакций систем:**
   - SELinux: записи в /var/log/audit/audit.log
   - AppArmor: отчеты в /var/log/syslog

4. Результаты и анализ

4.1. Эффективность защиты

Тест	**SELinux**	**AppArmor**
Запрет записи в /etc	Успешно	Успешно
Ограничение сетевых сокетов	Частично	Полностью
Нагрузка на систему	8-12%	3-5%

4.2. Выводы

1. SELinux обеспечивает более детальный контроль
2. AppArmor проще в настройке
3. Для максимальной защиты рекомендуется:
   - SELinux для серверов
   - AppArmor для рабочих станций

5. Вопросы для защиты

1. В чем разница между DAC и MAC?
2. Как работает трансляция контекстов в SELinux?
3. Почему AppArmor использует path-based контроль?
4. Как создать кастомный тип в SELinux?
5. Какие системные вызовы перехватывают эти системы?

Лабораторная работа №13 Изучение аппаратных средств защиты (HSM, TPM-модули, смарт-карты). Работа с аппаратными токенами для безопасного хранения ключей

1. Цель работы

Исследование аппаратных средств криптографической защиты информации, практическое освоение работы с HSM, TPM-модулями и смарт-картами для безопасного хранения и использования криптографических ключей.

2. Теоретическая часть

2.1. Аппаратные средства защиты информации

1. HSM (Hardware Security Module)

- Специализированные криптографические устройства
- Функции:
  - Генерация и хранение ключей
  - Выполнение криптоопераций
  - Физическая защита от вскрытия
- Стандарты: FIPS 140-2/3, Common Criteria

2. TPM (Trusted Platform Module)

- Криптопроцессор на материнской плате
- Основные возможности:
  - Хранение ключей в защищенной области
  - Измерение целостности ПО
  - Remote Attestation
- Спецификации: TPM 2.0 (ISO/IEC 11889)

3. Смарт-карты и токены

- Типы:
  - JavaCard
  - PKCS#11-совместимые
  - ГОСТ Р 34.10-2012
- Использование:

- Электронная подпись
- Двухфакторная аутентификация

2.2. Сравнительные характеристики

Параметр	HSM	TPM	Смарт-карты
Производительность	Высокая	Низкая	Средняя
Стоимость	$$$	$	$$
Мобильность	Нет	Нет	Да
Поддержка стандартов	FIPS, CC	TCG	PKCS#11

3. Практическая часть

3.1. Оборудование и ПО

1. Аппаратные средства:
    - HSM: Thales nShield Solo
    - TPM: Infineon SLB 9665
    - Смарт-карта: JaCarta PKI
2. Программное обеспечение:
    - OpenSC
    - tpm2-tools
    - Cryptoki

3.2. Работа с TPM-модулем

Шаг 1. Проверка наличия TPM

```
sudo dmesg | grep -i tpm
sudo tpm2_getcap properties-fixed
```

Шаг 2. Создание и использование ключа

```
tpm2_createprimary -c primary.ctx
tpm2_create -C primary.ctx -u key.pub -r key.priv
tpm2_load -C primary.ctx -u key.pub -r key.priv -c key.ctx
```

3.3. Настройка HSM

Шаг 1. Инициализация устройства

```
nshield-mgr --initialize
```

Шаг 2. Генерация RSA-ключа

```
pkcs11-tool --module /usr/lib/libnfast.so --login --keygen --key-type rsa:2048 --id 1 --label "TestKey"
```

3.4. Использование смарт-карты

Шаг 1. Установка драйверов

```
sudo apt install opensc pcscd
```

Шаг 2. Работа с ключами

```
pkcs11-tool --list-slots
pkcs11-tool --login -s -p 123456 --keypairgen --key-type rsa:2048 --id 10
```

4. Результаты и анализ

4.1. Сравнение производительности

Операция	HSM	TPM	Смарт-карта
Генерация RSA-2048	0.8 с	4.2 с	2.1 с
Подпись (ECDSA)	0.3 с	1.8 с	0.9 с
Шифрование (AES-256)	0.1 с	Не поддерживается	0.4 с

4.2. Выводы

1. HSM обеспечивают максимальную производительность
2. TPM интегрированы в платформу, но ограничены функционально
3. Смарт-карты оптимальны для мобильных сценариев

5. Вопросы для защиты

1. В чем отличие TPM от HSM?
2. Как работает механизм Remote Attestation?
3. Какие алгоритмы поддерживают российские смарт-карты?
4. Как организована защита от side-channel атак в HSM?
5. Какие методы эмуляции TPM существуют?

Лабораторная работа №14 Анализ безопасности беспроводных сетей (взлом WPA2, защита WPA3). Использование Aircrack-ng для тестирования Wi-Fi

1. Цель работы

Исследование уязвимостей протоколов защиты беспроводных сетей, практическое освоение методов тестирования безопасности Wi-Fi с

использованием инструментария Aircrack-ng, сравнительный анализ стойкости WPA2 и WPA3.

2. Теоретическая часть
2.1. Протоколы защиты Wi-Fi

1. WPA2 (Wi-Fi Protected Access 2)

- Алгоритм: AES-CCMP
- Уязвимости:
  - Атака KRACK (Key Reinstallation Attacks)
  - Оффлайн-перебор PMK (Pairwise Master Key)
  - Слабые парольные фразы

2. WPA3

- Улучшения:
  - Simultaneous Authentication of Equals (SAE)
  - 192-битный режим безопасности
  - Защита от оффлайн-перебора

2.2. Методы атак

Тип атаки	**WPA2**	**WPA3**
Перехват handshake	Да	Нет
Словарная атака	Да	Ограничена
Evil Twin	Да	Да
KRACK	Да	Нет

2.3. Инструментарий Aircrack-ng

Компоненты:

- airmon-ng: мониторинг эфира
- airodump-ng: захват пакетов
- aireplay-ng: инъекция трафика
- aircrack-ng: взлом ключей

3. Практическая часть
3.1. Подготовка среды

Оборудование:

1. Адаптер Wi-Fi с поддержкой мониторинга (ALFA AWUS036NHA)
2. Kali Linux 2023.3
3. Тестовая точка доступа (TP-Link Archer C7)

3.2. Захват handshake WPA2

Шаг 1. Переход в режим мониторинга

```
sudo airmon-ng start wlan0
```

Шаг 2. Сканирование сетей

```
sudo airodump-ng wlan0mon
```

Шаг 3. Целенаправленный захват

```
sudo airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w capture wlan0mon
```

Шаг 4. Форсирование деаутентификации

```
sudo aireplay-ng -0 10 -a 00:11:22:33:44:55 -c FF:FF:FF:FF:FF:FF wlan0mon
```

3.3. Взлом ключа

Шаг 1. Создание словаря

```
crunch 8 10 1234567890 -o wordlist.txt
```

Шаг 2. Запуск перебора

```
aircrack-ng -w wordlist.txt -b 00:11:22:33:44:55 capture-01.cap
```

3.4. Тестирование WPA3

Шаг 1. Анализ поддержки SAE

```
sudo hcxdumptool -i wlan0mon --enable_status=1
```

Шаг 2. Попытка перехвата

```
sudo hcxpcapngtool -o hash.hc22000 capture.pcapng
```

4. Результаты и анализ

4.1. Сравнительная таблица

Параметр	**WPA2**	**WPA3**
Время взлома (8 символов)	4 ч	Не взломан
Уязвимость к KRACK	Да	Нет

Параметр	WPA2	WPA3
Защита от словарных атак	Нет	Да

4.2. Выводы

1. WPA2 уязвим к оффлайн-атакам при слабых паролях
2. WPA3 обеспечивает существенно лучшую защиту
3. Для критичных систем рекомендуется:
   - WPA3-Enterprise
   - Сертифицированные точки доступа

5. Вопросы для защиты

1. Как работает атака KRACK?
2. В чем преимущество Dragonfly Key Exchange?
3. Какие аппаратные требования для атак на WPA3?
4. Как защититься от Evil Twin?
5. Почему WPA3 несовместим со старыми устройствами?

Лабораторная работа № 15 Исследование методов защиты от атак типа "Человек посередине" (MITM). Перехват трафика в локальной сети и защита с помощью сертификатов

1. Цель работы

Изучение механизмов атак типа "Человек посередине" (MITM), методов перехвата трафика в локальной сети и способов защиты с использованием цифровых сертификатов.

2. Теоретическое обоснование

2.1. Атака "Человек посередине" (MITM)

MITM (Man-in-the-Middle) — это вид кибератаки, при которой злоумышленник перехватывает и, возможно, изменяет передаваемые между двумя узлами данные, оставаясь незамеченным.

Основные этапы MITM-атаки:

1. **Перехват трафика** – злоумышленник получает доступ к каналу связи (например, через ARP-spoofing, подмену DNS или использование снифферов).
2. **Дешифровка данных** (если трафик зашифрован).
3. **Модификация или анализ данных** (по необходимости).

2.2. Методы перехвата трафика в локальной сети

1. **ARP-spoofing (ARP-отравление)**

- Злоумышленник отправляет ложные ARP-ответы, связывая свой MAC-адрес с IP-адресом жертвы.
- Трафик перенаправляется через атакующего.

2. **DNS-spoofing**

- Подмена DNS-записей для перенаправления жертвы на фальшивый сервер.

3. **Сниффинг трафика**

- Использование инструментов (Wireshark, tcpdump) для пассивного перехвата незашифрованных данных.

2.3. Защита с помощью сертификатов

Цифровые сертификаты обеспечивают:

- **Аутентификацию** – подтверждение подлинности сервера/клиента.
- **Шифрование данных** – защиту от перехвата.
- **Целостность данных** – предотвращение модификации.

Протоколы, использующие сертификаты:

- **TLS/SSL** (HTTPS, FTPS, SMTPS)
- **IPSec**
- **S/MIME** (для электронной почты)

3. Аппаратура и программное обеспечение

1. **Компьютер с ОС Windows/Linux**
2. **Сетевой анализатор (Wireshark, tcpdump)**
3. **Утилиты для ARP-spoofing (Ettercap, Cain & Abel)**
4. **OpenSSL для генерации сертификатов**
5. **Веб-сервер (Apache, Nginx) для тестирования HTTPS**

4. Методика и порядок выполнения работы

4.1. Перехват трафика в локальной сети

1. **Запуск сниффера (Wireshark)**

- Открыть Wireshark, выбрать сетевой интерфейс.
- Запустить захват пакетов, фильтровать HTTP-трафик.

2. **ARP-spoofing с помощью Ettercap**
   ettercap -T -M arp:remote /192.168.1.1// /192.168.1.2//
- Где 192.168.1.1 – шлюз, 192.168.1.2 – жертва.
3. **Анализ перехваченных данных**
- Просмотр логинов, паролей в незашифрованном HTTP-трафике.

4.2. Защита трафика с помощью сертификатов

1. **Генерация самоподписанного сертификата (OpenSSL)**
   openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nod
   es
2. **Настройка HTTPS на веб-сервере (Apache)**
   apache

```
<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /path/to/cert.pem
    SSLCertificateKeyFile /path/to/key.pem
</VirtualHost>
```

3. **Проверка защиты**
   - Попытка перехватить трафик через Wireshark – данные должны быть зашифрованы.

5. Содержание отчета

1. **Титульный лист** (название работы, ФИО, группа, преподаватель).
2. **Цель работы**.
3. **Теоретическая часть** (описание MITM, методы перехвата, защита сертификатами).
4. **Практическая часть** (шаги перехвата, настройка HTTPS, скриншоты).
5. **Выводы** (эффективность сертификатов, рекомендации по защите).

6. Вопросы для защиты

1. Как работает MITM-атака?
2. Какие методы перехвата трафика существуют?
3. Как ARP-spoofing позволяет перехватывать данные?
4. Как цифровые сертификаты защищают от MITM?
5. Какие протоколы используют TLS/SSL?
6. Как проверить, что соединение защищено сертификатом?

Лабораторная работа №16 Разработка и тестирование политик информационной безопасности (на основе ISO 27001, ГОСТ Р 57580). Создание регламентов доступа, аудит защищенности системы

1. Цель работы

1. Изучение принципов разработки политик информационной безопасности в соответствии с международным стандартом ISO 27001 и национальным стандартом ГОСТ Р 57580.
2. Разработка регламентов разграничения доступа к информационным ресурсам.
3. Проведение аудита защищенности автоматизированной системы.

2. Теоретическое обоснование

2.1. Стандарты информационной безопасности

ISO 27001 - международный стандарт, описывающий требования к системе менеджмента информационной безопасности (СМИБ). Основные положения:

- Определение области применения СМИБ
- Оценка рисков информационной безопасности
- Разработка и внедрение мер защиты
- Мониторинг и непрерывное улучшение

ГОСТ Р 57580 - российский стандарт, устанавливающий требования к защите информации в информационных системах. Основные аспекты:

- Классификация информационных систем
- Требования к средствам защиты информации
- Порядок аттестации систем

2.2. Регламенты доступа

Ключевые документы:

1. **Политика разграничения доступа** - определяет:

- Категории пользователей
- Уровни доступа

- Правила назначения и изменения прав

2. **Инструкция по учету и хранению учетных данных**
3. **Регламент изменения прав доступа**

2.3. Аудит защищенности

Основные этапы:

1. **Инвентаризация активов**
2. **Анализ уязвимостей** (сканирование сети, проверка настроек)
3. **Тестирование на проникновение**
4. **Анализ соответствия требованиям стандартов**

3. Аппаратура и программное обеспечение

1. Рабочая станция с ОС Windows/Linux
2. Средства сканирования уязвимостей (Nessus, OpenVAS)
3. Инструменты аудита (Lynis, Wireshark)
4. Документация ISO 27001 и ГОСТ Р 57580

4. Методика выполнения работы

4.1. Разработка политики доступа

1. Определить категории пользователей:

- Администраторы
- Обычные пользователи
- Внешние пользователи

2. Установить уровни доступа:

- Полный доступ
- Ограниченный доступ
- Доступ только для чтения

3. Разработать матрицу доступа

4.2. Создание регламентов

1. **Регламент предоставления доступа**
   1. Заявка на доступ (форма)
   2. Согласование с руководителем
   3. Настройка прав администратором
   4. Уведомление пользователя
2. **Регламент изменения паролей**

- Минимальная длина 8 символов
- Обязательное использование спецсимволов
- Периодичность смены - 90 дней

4.3. Проведение аудита

1. **Сканирование сети** с помощью Nessus:
   ```
   nessuscli scan --target 192.168.1.0/24
   ```
2. **Проверка настроек безопасности**:

- Анализ журналов событий
- Проверка политик паролей

3. **Тестирование на проникновение** (Metasploit Framework)

5. Оформление отчета

Отчет должен содержать:

1. Титульный лист
2. Цели и задачи работы
3. Описание разработанных регламентов
4. Результаты аудита безопасности
5. Выводы и рекомендации

6. Вопросы для защиты

1. Каковы основные положения ISO 27001?
2. Как классифицируются информационные системы по ГОСТ Р 57580?
3. Какие документы входят в регламенты доступа?
4. Какие этапы включает аудит защищенности?
5. Какие инструменты используются для сканирования уязвимостей?

Список литературы

Основная литература:

1. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2025. — 312 с. — (Высшее образование). — ISBN 978-5-9916-9043-0.

2. Внуков, А. А. Основы информационной безопасности: защита информации : учебное пособие для среднего профессионального образования / А. А. Внуков. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 161 с. — (Профессиональное образование). — ISBN 978-5-534-13948-8.
3. Программно-аппаратные средства защиты информации. В 3 частях. Ч. 1 : учебное пособие / Гриднев В. А., Губсков Ю. А., Дерябин А. С. [и др.]. - Тамбовский государственный технический университет, ЭБС АСВ, 2022. - ISBN 978-5-8265-2464-0, 978-5-8265-2467-1 (ч. 1).
4. Программно-аппаратные средства защиты информации. В 3 частях. Ч. 2 : учебное пособие / Гриднев В. А., Губсков Ю. А., Дерябин А. С. [и др.]. - Тамбовский государственный технический университет, ЭБС АСВ, 2023. - ISBN 978-5-8265-2464-0, 978-5-8265-2609-5 (ч. 2).
5. Программно-аппаратные средства защиты информации. В 3 частях. Ч. 3 : учебное пособие / Гриднев В. А., Губсков Ю. А., Дерябин А. С. [и др.]. - Тамбовский государственный технический университет, ЭБС АСВ, 2024. - ISBN 978-5-8265-2795-5.
6. Казарин О. В., Забабурин А. С. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для вузов / Казарин О. В., Забабурин А. С. - М. : Юрайт, 2024. - 311 с. - (Высшее образование). - Библиогр. в конце глав. - ISBN 978-5-9916-9043-0.

Дополнительной литература:

1. Программно-аппаратные средства обеспечения информационной безопасности : учебное пособие / А. В. Душкин, О. М. Барсуков, Е. В. Кравцов, К. В. Славнов ; под редакцией А. В. Душкина. — Москва : Горячая линия-Телеком, 2018. — 248 с. — ISBN 978-5-9912-0470-5.
2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для среднего профессионального образования / О. В. Казарин, А. С. Забабурин. — Москва : Издательство Юрайт, 2025. — 312 с. — (Профессиональное образование). — ISBN 978-5-534-13221-2.
3. Лабораторный практикум по дисциплине Программно-аппаратные средства защиты информации / сост. Денисов И. А. - Московский технический университет связи и информатики, 2016.
4. Программно-аппаратные средства обеспечения информационной безопасности : учебное пособие для вузов / Душкин А. В., Барсуков О. М., Кравцов Е. В., Славнов К. В. ; ред. Душкин А. В. - М. : Горячая линия - Теклеком, 2016. - 247 с. : ил. - (Учебное пособие для вузов. Специальность). - ISBN 978-5-9912-0470-5.
5. Фомин Д. В. Защита информации: специализированные аттестованные программные и программно-аппаратные средства : практикум / Фомин Д. В. - Вузовское образование, 2021. - ISBN 978-5-4487-0795-7.