

Лабораторная работа №4-5
Базовая настройка безопасности веб-сервера.

Задание

1. Ознакомиться с теоретическим материалом по основным типам шифрования (симметричное и асимметричное), а также по протоколам SSL и HTTPS.
2. Сгенерировать SSL-сертификат и настроить защищенное соединение (HTTPS) с веб-сервером.
3. Настроить доступа к терминалу сервера по ключу безопасности.
4. Настроить правила встроенного в ОС Linux Firewall.
5. Обеспечить автоматическое конфигурирование настроенных правил Firewall в случае перезапуска системы.

Ход выполнения работы:

Результатом выполнения задания являются:

Отчёт, содержащий следующую информацию:

- 1) Конспект, содержащий дополнительную информацию по ключевым темам теоретического материала.
- 2) Пошаговое описание практической части.
- 3) Отчет по лабораторной работе, содержащий ход выполнения работы с описанием и скриншотами выполнения, результаты выполнения лабораторной работы.

Для успешной защиты лабораторной работы студенты должны предоставить проект и отчет к нему.

Требования к оформлению отчета:

Способ выполнения текста должен быть единым для всей работы. **Шрифт – Times New Roman**, кегль 14, **межстрочный интервал – 1,5**, **размеры полей:** левое – 30 мм; правое – 10 мм, верхнее – 20 мм; нижнее – 20 мм. Сокращения слов в тексте допускаются только общепринятые.

Абзацный отступ (1,25) должен быть одинаковым во всей работе. **Нумерация страниц** основного текста должна быть сквозной. Номер страницы на титульном листе не указывается, задание на производственную практику является второй страницей. Сам номер располагается внизу по центру страницы или справа.

1. Необходимое программное обеспечение

Клиент SSH - **Putty** (для Windows):

<https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe>

Генератор ключей для доступа по SSH (**PuTTYgen**):

<https://puttygen.com/download.php?val=49>

2. Настройка защищенного соединения для веб-сервера

2.1 Создание ключа безопасности. Стандартным способом защищённого сетевого взаимодействия с веб-сервером является криптографический протокол **SSL (Secure Sockets Layer)** – протокол безопасности, обеспечивающий зашифрованное соединение между веб-сервером и веб-браузером.

Устанавливаем свободное программное обеспечение (криптографическую библиотеку), реализующее протокол SSL – **OpenSSL**:

```
apt install openssl
```

OpenSSL также имеет возможность генерирования ключей безопасности для шифрования данных.

Ключ шифрования – обычно представлен в виде файла, который содержит закодированные в символьном виде данные (чаще всего используется формат кодирования BASE64). Эти данные используются алгоритмом для шифрования и расшифровки информации.

Надёжность ключа зависит от его длины в битах. В технологии SSL используют шифры от 2048 бит для корневого сертификата и 128–256 бит для клиентских. Такая длина считается достаточной для безопасной передачи данных.

Протокол SSL использует два типа шифрования: асимметричное шифрование (шифрование с открытым ключом) только для установки соединения (по сути, для идентификации клиента на сервере), а передаваемые данные шифруются с помощью симметричного шифрования. Это связано с тем, что асимметричное шифрование требует значительно больше вычислительных ресурсов, но позволяет обеспечить высокий уровень защиты.

При асимметричном шифровании используются 2 ключа: **открытый** и **закрытый**. Оба формируются при запросе SSL-сертификата, а сам SSL-сертификат обычно выпускается специализированными центрами сертификации (CA), например: GlobalSign, Comodo, Symantec, TrustWave, GeoTrust. Эти центры сертификации используют «именные» корневые сертификаты, которые распознаются большинством

современных браузеров. Выдавая корневой сертификат, каждый такой центр гарантирует, что пользователи или организации, запросившие SSL-сертификат, верифицированы и действия с доменом легальны.

Открытый (публичный ключ) доступен всем. Используется для шифрования данных при обращении браузера к серверу.

Закрытый (приватный ключ) известен только администратору веб-сервера. Используется для расшифровки данных, отправленных браузером.

Таким образом, асимметричное шифрование гарантирует сохранность информации, т.к. даже если трафик между браузером и сервером будет перехвачен третьей стороной (атака «человек посередине» или «man in the middle»), злоумышленник не сможет расшифровать его без закрытого ключа.

Подробнее можно ознакомиться здесь:

<https://encyclopedia.kaspersky.ru/glossary/asymmetric-encryption/>

HTTPS – это безопасная версия протокола HTTP (Secure), который использует SSL/TLS протокол для шифрования данных, передаваемых между веб-браузером и веб-сервером. Протокол по умолчанию использует порт 443. Общий алгоритм работы данного протокола следующий:

- Браузер пользователя просит предоставить SSL-сертификат.
- Сайт на HTTPS отправляет сертификат.
- Браузер проверяет подлинность сертификата в центре сертификации.
- Браузер и сайт «договариваются» о симметричном ключе при помощи асимметричного шифрования.
- Браузер и сайт передают зашифрованную информацию.

В рамках данной лабораторной работы с помощью OpenSSL сгенерируем собственный (самоподписанный) сертификат безопасности для настройки HTTPS-соединения на нашем веб-сервере:

```
openssl req -new -x509 -days 1461 -nodes -out cert.pem -  
keyout cert.key -subj "/C=RU/ST=RB/L=Ufa/O=Global  
Security/OU=USATU Department/CN=lab4.usatu.local/CN=lab4"
```

x509 – формат сертификата безопасности;

days <num> – срок действия сертификата;

subj – информация о владельце сертификата.

В результате будут сгенерированы 2 файла:

- **cert.pem** – сертификат, открытая (свободно распространяемая) часть ключа;

- **cert.key** — закрытая (приватная) часть ключа (хранится исключительно на сервере).

Дополнительную информацию по **OpenSSL** и ключам безопасности можно ознакомиться здесь:

<https://www.ssl.com/ru/%D1%80%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE/%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8-%D0%B8-%D0%BF%D1%80%D0%B5%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-pem-der-crt-%D0%B8-cer-x-509/>

2.2 Настройка веб-сервера. Далее создаём дополнительную директорию в папке веб-сервера Apache:

```
mkdir /etc/apache2/ssl
```

И копируем в эту папку сгенерированные файлы ключа.

Активируем дополнительный модуль Apache для работы с openssl:

```
sudo a2enmod ssl
```

Редактируем конфигурацию по умолчанию для сайта с поддержкой SSL:

```
nano /etc/apache2/sites-available/default-ssl.conf
```

Устанавливаем параметры конфигурации:

SSLCertificateFile /etc/apache2/ssl/cert.pem

SSLCertificateKeyFile /etc/apache2/ssl/cert.key

Описание основных параметров:

ServerName — домен сайта (если имеется);

DocumentRoot — расположение файлов сайта в системе;

SSLCertificateFile и **SSLCertificateKeyFile** — пути до файлов ключей, которые были сгенерированы;

SSLCertificateChainFile — при необходимости, путь до цепочки промежуточных сертификатов (если используем не самоподписанный сертификат).

Активируем конфигурацию:

```
sudo a2ensite default-ssl
```

Перезапускаем веб-сервер для применения всех настроек:

```
systemctl restart apache2
```

Перейдите на ваш сайт, используя защищенное соединение, например:

https://192.168.56.104/wp/

При первом переходе по данному адресу веб-браузер выдаст предупреждение безопасности:



Подключение не защищено

Злоумышленники могут пытаться похитить ваши данные с сайта **192.168.56.104** (например, пароли, сообщения или номера банковских карт). [Подробнее...](#)

NET::ERR_CERT_AUTHORITY_INVALID



Чтобы браузер Chrome стал максимально безопасным, [включите режим "Улучшенная защита"](#).

Дополнительные

Вернуться к безопасной странице

Рис. 2.1 – Предупреждение безопасности веб-браузера

Это связано с тем, что используемый сертификат не является заверенным в доверенном центре сертификации. Нажмите «Дополнительно» и перейдите на сайт, несмотря на предупреждение. Сделайте скриншот начальной страницы вашего сайта, открытого через протокол HTTPS.

Для настройки автоматической переадресации на https протокол при переходе на ваш сайт по стандартному протоколу http, добавьте в конфигурацию VirtualHost http-сайта следующую строку:

Redirect / https://<адрес_сайта> /

В качестве адреса сайта может быть или доменное имя или IP-адрес вашего сервера.

3. Настройка доступа к терминалу сервера по ключу безопасности

Технология идентификации пользователя по ключу безопасности также часто применяется для доступа к серверу по протоколу SSH. Для такой идентификации также используется шифрование с помощью открытого и приватного ключа.

Запустите утилиту **PuTTYgen**, выберите алгоритм шифрования RSA и нажмите кнопку «**Generate**». Далее необходимо произвольно двигать указатель мыши в области **Key** в главном окне программы (рис. 3.1):

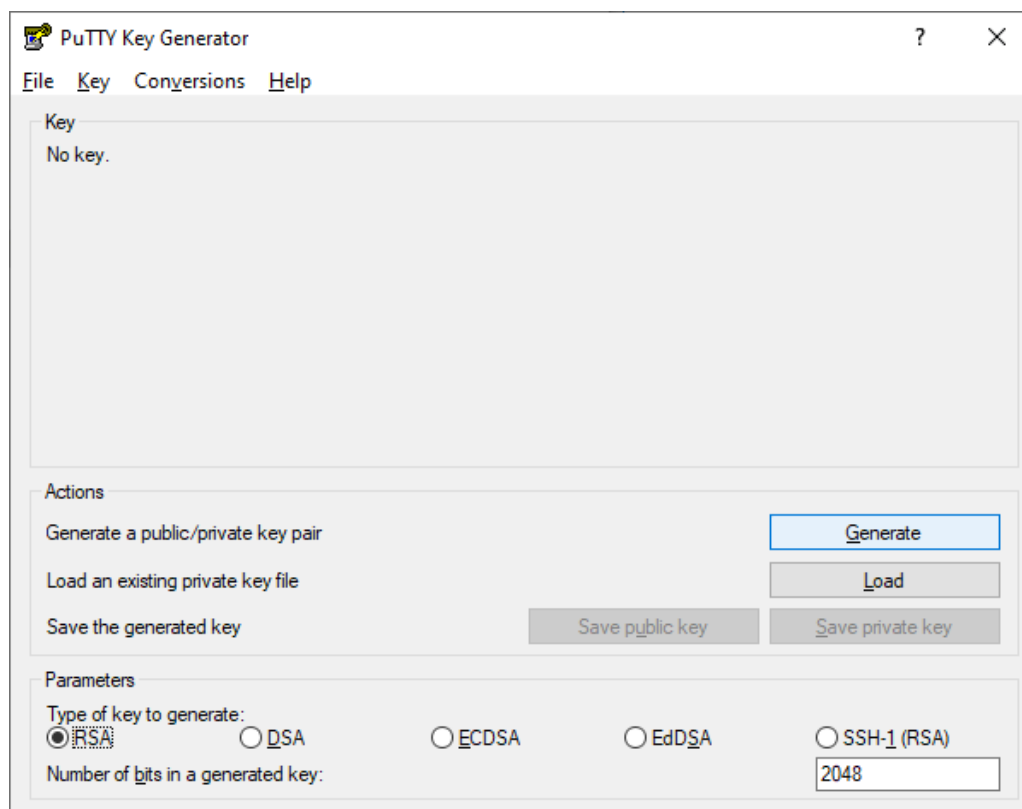


Рис. 3.1 – Генерирование ключа безопасности для SSH-доступа

После завершения генерирования ключей, содержание публичного ключа появится в поле Key в главном окне программы (рис 3.2). Сохраните файл приватного ключа (в окне предупреждения о пароле нажмите «ОК»), нажав кнопку «**Save private key**». Далее залогиньтесь на сервере через протокол SSH (с помощью PuTTY) под обычным пользователем (HE root) и скопируйте содержимое публичного ключа (из главного окна программы) в файл **authorized_keys** в каталоге **.ssh** в домашней папке данного пользователя, например:

```
nano /home/ais/.ssh/authorized_keys
```

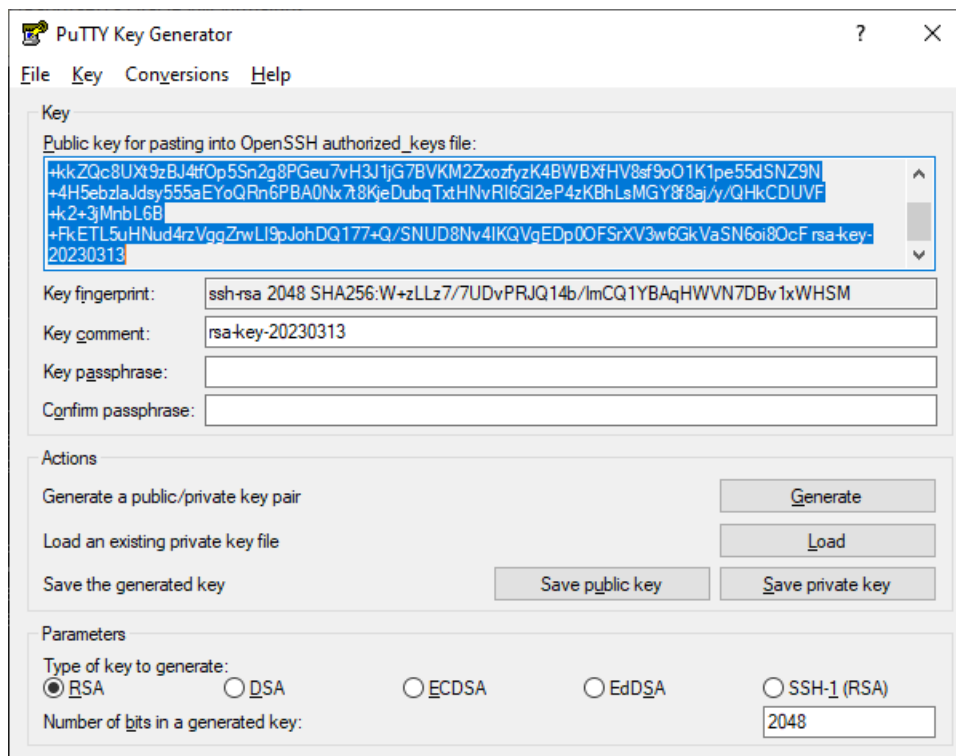


Рис. 3.2 – Сгенерированный публичный ключ

Закройте терминал и заново откройте программу PuTTY. Введите в стартовом окне программы IP-адрес и перейдите в пункт меню **Connection -> SSH -> Auth**. В поле «Private key for authentication» выберите файл приватного ключа, который был сохранён ранее (рис. 3.3):

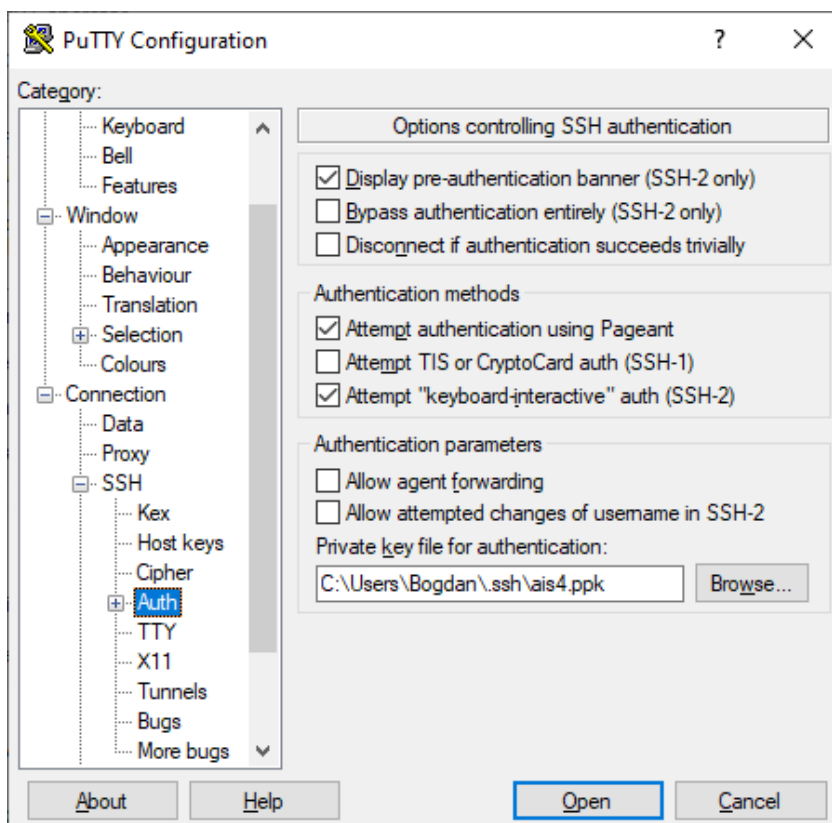
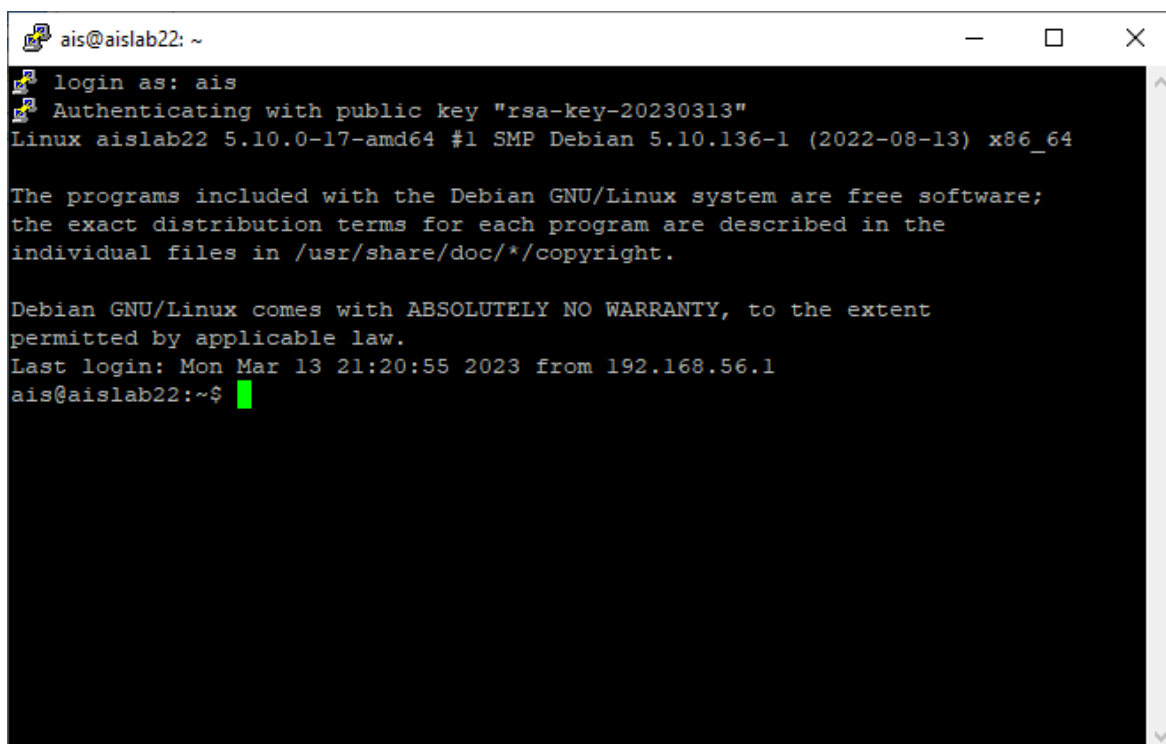


Рис. 3.3 – Ввод приватного ключа авторизации в PuTTY

Далее нажмите «Open» и в терминале введите логин. Авторизация должна пройти без запроса пароля, и должно появиться сообщение об авторизации по публичному ключу (рис. 3.4):



```
ais@aislab22: ~  
login as: ais  
Authenticating with public key "rsa-key-20230313"  
Linux aislab22 5.10.0-17-amd64 #1 SMP Debian 5.10.136-1 (2022-08-13) x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon Mar 13 21:20:55 2023 from 192.168.56.1  
ais@aislab22:~$
```

Рис. 3.4 – Авторизация по публичному ключу

Если сохранить данную сессию, то в дальнейшем нет необходимости запоминать пароль, достаточно иметь на локальном компьютере приватный ключ.

Аналогичным способом (с помощью генерирования ключей безопасности) может осуществляться беспарольный доступ, например, к системам контроля версий (Git, Bitbucket), VPN-соединениям (OpenVPN) и другим сервисам.

4. Настройка сетевого экрана (Firewall) сервера

4.1 Linux Firewall. Настройка Firewall (FW) или сетевого экрана является базовым требованием для обеспечения безопасности информационной системы. В ОС Linux чаще всего используется программный пакет **iptables** для управления встроенным в ядро системы сетевым экраном (NetFilter). Устанавливаем iptables:

```
apt install iptables
```

iptables структурно состоит из таблиц (tables), в которые входят цепочки (chains), в свою очередь содержащие наборы правил (rules или chain rules).

iptables использует 5 основных цепочек (списков) правил, которые расположены в соответствии с порядком обработки входящего пакета:

1. **PREROUTING** – используется для всего входящего трафика до принятия первого решения о маршрутизации.
2. **INPUT** – применяется для входящего трафика текущего узла.
3. **FORWARD** – транзитный трафик через узел.
4. **OUTPUT** – применяется для исходящего трафика текущего узла.
5. **POSTROUTING** – используется для всего исходящего трафика после принятия всех решений о маршрутизации.

Изначально в iptables для всех цепочек правил установлено действие **ACCEPT** – трафик проходит без какой-либо обработки.

Используем команду для просмотра текущих правил iptables:

```
sudo iptables -L
```

Вывод:

```
Chain INPUT (policy ACCEPT)
target      prot opt source                destination
```

```
Chain FORWARD (policy ACCEPT)
target      prot opt source                destination
```

```
Chain OUTPUT (policy ACCEPT)
target      prot opt source                destination
```

Если сервер располагается в локальной сети и не используется для маршрутизации трафика (не является маршрутизатором, шлюзом и т.п.), т.е. сервер предназначен для работы с определенными сервисами (web-сервер, база данных, сетевое хранилище), то для базовой настройки FW на сервере такого типа достаточно ограничить доступ для входящего трафика на всех портах, кроме необходимых для соответствующих приложений. Таким образом, необходимо создать набор правил для iptables. Т.к. правила задаются командами, исполняемыми в терминале ОС Linux, то **набор таких правил может быть задан в виде набора последовательных команд, оформленных в виде скрипта для командной оболочки Linux (bash-скрипт).**

Информация по основам написания скриптов в ОС Linux можно посмотреть здесь:

<https://losst.pro/napisanie-skriptov-na-bash>

Пример скрипта с базовыми правилами iptables для web-сервера:

```
#!/bin/bash
```

```
# Сбрасываем текущие настройки iptables
```

```
sudo iptables -F
```

Ограничиваем весь входящий трафик (данная команда блокирует трафик на ВСЕХ интерфейсах, даже на локальном. **НЕ ПРИМЕНЯТЬ** отдельной командой, не имея возможности перезагрузить сервер или физический доступ к серверу!)

```
sudo iptables -P INPUT DROP
```

```
# Разрешаем весь исходящий трафик
sudo iptables -P OUTPUT ACCEPT
# Разрешаем весь транзитный трафик
sudo iptables -P FORWARD ACCEPT
# Добавляем (параметр -A) новые правила в списки.
# Разрешаем весь входящий и исходящий трафик на локальном интерфейсе
(127.0.0.1)
sudo iptables -A INPUT -i lo -j ACCEPT
sudo iptables -A OUTPUT -o lo -j ACCEPT
# Разрешаем весь входящий и исходящий icmp-трафик (позволяет делать ping к
серверу)
sudo iptables -A INPUT -p icmp -j ACCEPT
sudo iptables -A OUTPUT -p icmp -j ACCEPT
# Разрешаем весь входящий трафик на порту 80 (http для web-сервера)
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
# Разрешаем весь входящий трафик на порту 443 (https для web-сервера)
sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
# Разрешаем весь входящий трафик на порту 22 (для доступа по SSH)
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
# Открываем исходящий и входящий udp-трафик на порту 53 (необходимо для
возможности использования сервиса доменных имен – DNS)
sudo iptables -A INPUT -p udp --sport 53 -j ACCEPT
sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
sudo iptables -A OUTPUT -p udp --sport 53 -j ACCEPT
sudo iptables -A OUTPUT -p udp --dport 53 -j ACCEPT
# Разрешаем все входящие пакеты, имеющие статусы: ESTABLISHED – пакет
идёт внутри уже установленного соединения; RELATED – пакет проходит в рамках
соединения, которое связано с другим соединением, имеющим признак
ESTABLISHED.
sudo iptables -A INPUT -i enp0s3 -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

Более подробно о возможных состояниях пакета в системе ядра Linux и фильтрации с их использованием можно ознакомиться [здесь](https://www.opennet.ru/docs/RUS/iptables/#TABLE.USERLANDSTATES):

<https://www.opennet.ru/docs/RUS/iptables/#TABLE.USERLANDSTATES>

Сохраняем скрипт в файле **ipt_rules.sh**

Делаем файл исполняемым:

```
chmod +x ipt_rules.sh
```

После запуска скрипта заданные правила Firewall будут действительны до перезапуска ОС. Для автоматического применения правил после запуска ОС необходима настройка автозапуска.

4.2 Настройка автозапуска правил Firewall. Для запуска системных сервисов и служб и управления ими в процессе работы, в большинстве популярных дистрибутивов Linux используется сервис (демон) **Systemd**, использующий уже известную вам команду запуска/останова: `systemctl start/stop/restart ...`. Данный сервис использует файлы конфигурации (так называемые **unit-файлы**) из каталога `/etc/systemd/system/` для управления другими сервисами (в ранних версиях Linux Debian использовался демон `init.d`, работающий с каталогом `/etc/init.d/`).

Для начала, после загрузки системы, проверим список юнитов, которые в данный момент добавлены в systemd:

```
systemctl list-units
```

Список unit-файлов можно получить командой:

```
systemctl list-unit-files
```

Данная команда отобразит все доступные юнит-файлы (не зависимо от того, были они загружены в systemd после загрузки ОС или нет).

Чтобы вывести список активных сервисов и их состояние, выполните:

```
systemctl list-units -t service
```

Таким образом, для автозапуска скрипта с заданными правилами iptables необходимо создание соответствующего unit-файла в директории `/etc/systemd/system`. Создаем файл **ipt_rules.service** следующего содержания:

```
[Unit]
```

```
Description=iptables rules
```

```
After=network.target
```

```
[Service]
```

```
Type=simple
```

```
ExecStart=/home/ais/ipt_rules.sh
```

```
TimeoutStartSec=0
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Description – описание сервиса.

After=network.target – директива ожидания запуска юнита только после запуска всех сетевых служб.

ExecStart – путь к скрипту запуска.

TimeoutStartSec – таймаут перед запуском.

WantedBy=multi-user.target – директива запуска юнита до запуска графической оболочки ОС.

systemctl enable ipt_rules – команда добавит в автозагрузку созданный сервис ipt_rules.

Данная команда создаст символическую ссылку на копию файла, указанного в команде сервиса, в директории автозапуска systemd.

Для удаления соответствующего сервиса из автозагрузки используйте параметр disable:

systemctl disable ipt_rules

После активации автозапуска правил сетевого экрана перезапустите сервер и добавьте в отчет скриншот вывода команды:

sudo iptables -L