

RELATÓRIO DO SPRINT 1

Turma 2DH _ Grupo 43

1190929 _ Patrícia Barbosa

1190947 _ Pedro Fraga

1190956 _ Pedro Garcia

1190963 _ Pedro Preto

Professor:

André Moreira, ASC

Unidade Curricular:

Arquitetura de Sistemas

Data: 04/12/2021

ÍNDICE

INTRODUÇÃO	3
USER STORY 1 E 2.....	4
LINUX.....	4
WINDOWS	Erro! Marcador não definido.
USER STORY 3	6
LINUX.....	6
WINDOWS	7
USER STORY 4	8
LINUX.....	8
WINDOWS	9
USER STORY 5	10
LINUX.....	10
USER STORY 6	11
LINUX.....	11
USER STORY 7	12
LINUX.....	12
USER STORY 8	14
LINUX.....	14

INTRODUÇÃO

O presente relatório funcionará como apoio e explicação de execução das User Stories pedidas para o SPRINT 1 de ASIST. Em cada User Story, está dividido pelas tarefas realizadas em Linux e em Windows.

Nas user stories 1 e 2, não conseguimos resolver no servidor Windows, devido a um erro ao tentar configurar o dhcp.

USER STORY 1 E 2

1. Como administrador da infraestrutura quero que o servidor Windows e Linux forneçam endereços IP (na segunda placa de rede) da família **192.168.X.0/24** aos postos clientes, onde **X** é obtido por **100 + número_do_grupo** (exemplo, para o grupo 99, X=199); para o efeito devo alterar o endereço dessa placa assignado nas aulas PL
2. Como administrador da infraestrutura quero que os serviços acima referidos funcionem em *failover*, com um deles a facultar endereços de **192.168.X.50** a **192.168.X.150** e o outro de **192.168.X.151** a **192.168.X.200**

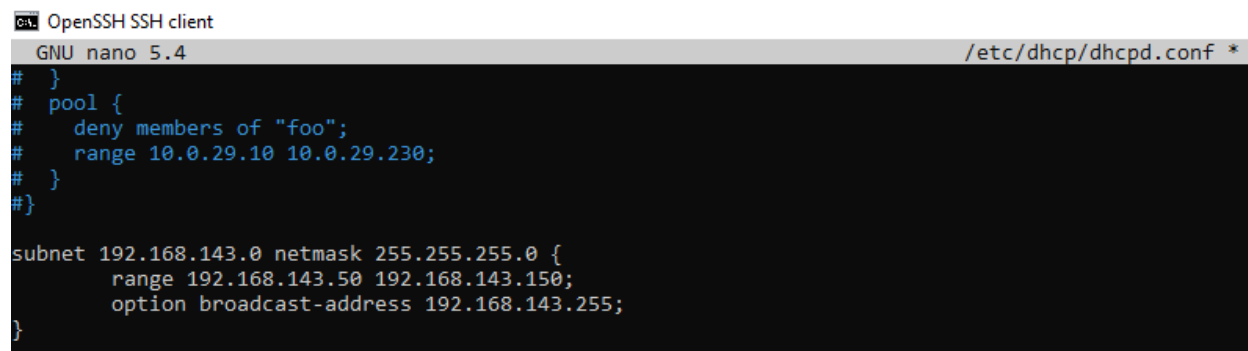
Figura 1: User stories 1 e 2

Para a realização de ambas as stories, fizemos uma configuração do DHCP, tanto em Linux, como em Windows.

LINUX

Em Linux, recorremos ao `isc-dhcp-server`, que já tinha sido previamente instalado, aquando da realização da ficha PL 5.

Começámos por modificar os parâmetros do DHCP server no ficheiro “`dhcpd.conf`”, adicionando o IP da subnet, a range de endereços, a máscara e o Broadcast address no final do ficheiro:

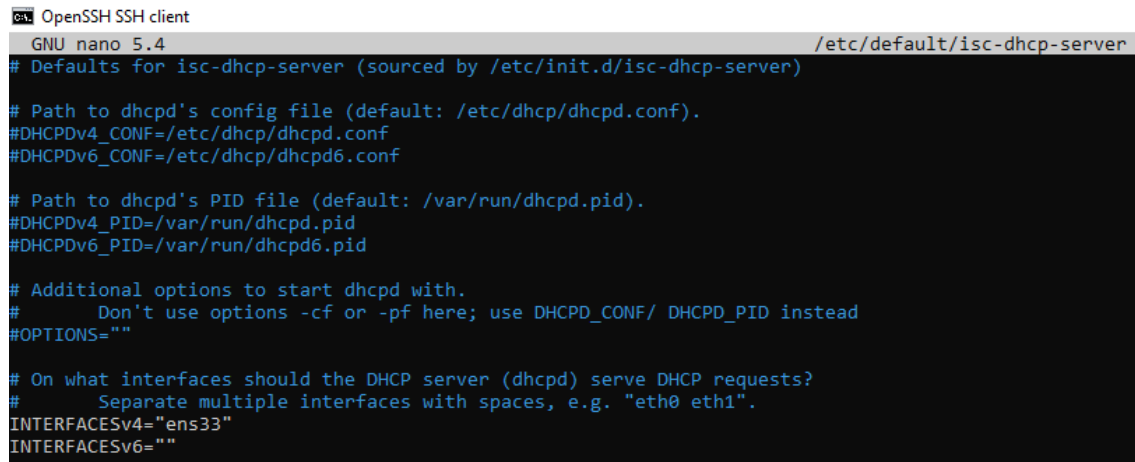


```
OpenSSH SSH client
GNU nano 5.4 /etc/dhcp/dhcpd.conf *
# }
# pool {
#     deny members of "foo";
#     range 10.0.29.10 10.0.29.230;
# }
#}

subnet 192.168.143.0 netmask 255.255.255.0 {
    range 192.168.143.50 192.168.143.150;
    option broadcast-address 192.168.143.255;
}
```

Figura 2: Alteração efetuado em `/etc/dhcp/dhcp.conf`

De seguida, definimos a NIC a ser usada pelo isc-dhcp-server. Para tal especificamos que a NIC a ser usada é a “ens33” (placa de rede secundária) em “/etc/default/isc-dhcp-server”.



```
OpenSSH SSH client
GNU nano 5.4 /etc/default/isc-dhcp-server
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

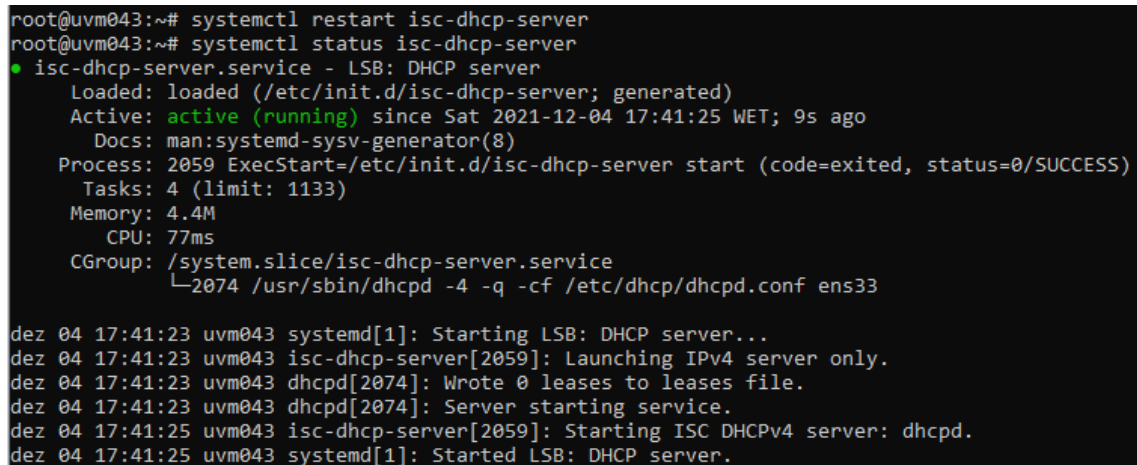
# Additional options to start dhcpd with.
# Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="ens33"
INTERFACESv6=""
```

Figura 3: Alteração efetuada em /etc/default/isc-dhcp-server

Por fim, reiniciámos o serviço de DHCP e verificamos o status, para confirmar que está ativo, através dos comandos:

- **systemctl restart isc-dhcp-server**
- **systemctl status isc-dhcp-server**



```
root@uvm043:~# systemctl restart isc-dhcp-server
root@uvm043:~# systemctl status isc-dhcp-server
● isc-dhcp-server.service - LSB: DHCP server
   Loaded: loaded (/etc/init.d/isc-dhcp-server; generated)
   Active: active (running) since Sat 2021-12-04 17:41:25 WET; 9s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 2059 ExecStart=/etc/init.d/isc-dhcp-server start (code=exited, status=0/SUCCESS)
    Tasks: 4 (limit: 1133)
   Memory: 4.4M
      CPU: 77ms
    CGroup: /system.slice/isc-dhcp-server.service
            └─2074 /usr/sbin/dhcpd -4 -q -cf /etc/dhcp/dhcpd.conf ens33

dez 04 17:41:23 uvm043 systemd[1]: Starting LSB: DHCP server...
dez 04 17:41:23 uvm043 isc-dhcp-server[2059]: Launching IPv4 server only.
dez 04 17:41:23 uvm043 dhcpd[2074]: Wrote 0 leases to leases file.
dez 04 17:41:23 uvm043 dhcpd[2074]: Server starting service.
dez 04 17:41:25 uvm043 isc-dhcp-server[2059]: Starting ISC DHCPv4 server: dhcpd.
dez 04 17:41:25 uvm043 systemd[1]: Started LSB: DHCP server.
```

Figura 4: Status do isc-dhcp-server

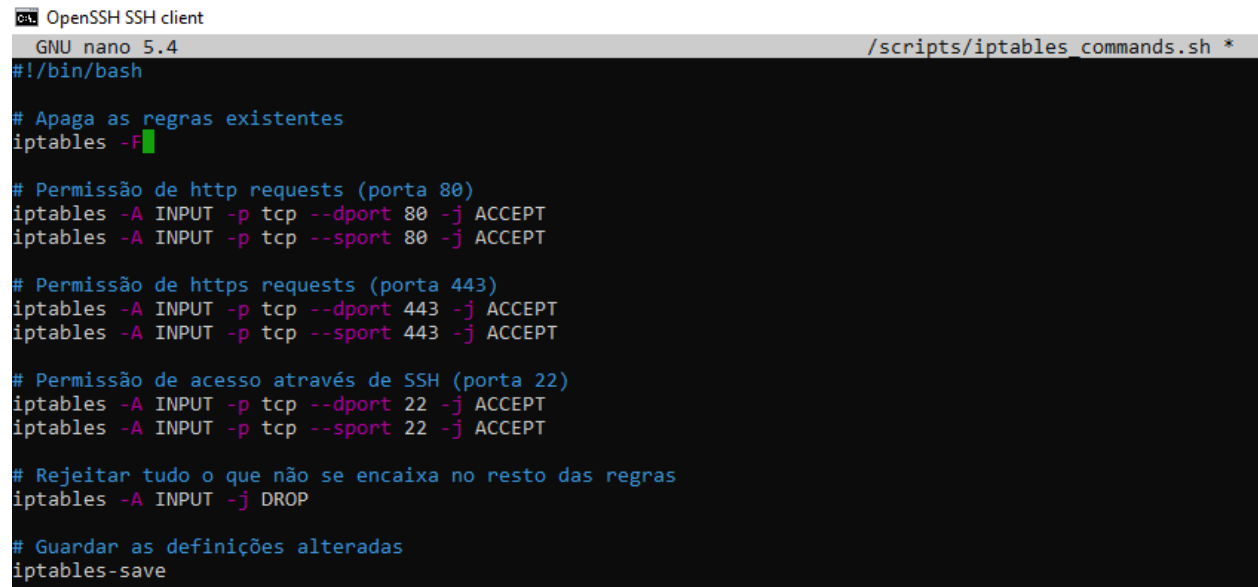
USER STORY 3

3. Como administrador da infraestrutura quero os servidores Windows e Linux estejam disponíveis apenas para pedidos **HTTP** e **HTTPS**. Tal não deve impedir o acesso por SSH ou RDP aos administradores (o grupo).

Figura 5: User story 3

LINUX

Para esta user story, recorreremos ao comando iptables para gerir o tráfego de pacotes, através da criação de regras. Contudo, estas regras não persistem após reinício do servidor se apenas utilizarmos o comando iptables. A solução encontrada pelo nosso grupo passou pela criação de um script, que contém a especificação das regras de entrada através sequencialmente. No final do script, rejeita todos os pacotes que não fizeram correspondência com nenhuma das regras, e guarda as configurações. Esse script é chamado através de um “service descriptor” (unit file), criado em “/etc/systemd/system”, que executa o script sempre que o sistema é iniciado (reboot).



```
ca. OpenSSH SSH client
GNU nano 5.4 /scripts/iptables_commands.sh *
#!/bin/bash

# Apaga as regras existentes
iptables -F

# Permissão de http requests (porta 80)
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 80 -j ACCEPT

# Permissão de https requests (porta 443)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT

# Permissão de acesso através de SSH (porta 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --sport 22 -j ACCEPT

# Rejeitar tudo o que não se encaixa no resto das regras
iptables -A INPUT -j DROP

# Guardar as definições alteradas
iptables-save
```

Figura 6: Script criado (/scripts/iptables_commands.sh)

```
OpenSSH SSH client
GNU nano 5.4 activate_script_on_boot.service
[Unit]
Description= Run script that changes iptables rules on reboot.

[Service]
Type=simple
ExecStart=/bin/bash /scripts/iptables_commands.sh

[Install]
WantedBy=multi-user.target
```

Figura 7: unit file criado em /etc/systemd/system para realizar o script a cada reboot

Por fim, alteram-se as permissões ao unit file criado (activate_script_on_boot.service) e ativa-se, através dos seguintes comandos:

- `chmod 644 /etc/systemd/system/activate_script_on_boot.service`
- `systemctl enable activate_script_on_boot.service`

WINDOWS

No Windows, a realização da user story passa pela configuração da firewall do Windows defender, onde posteriormente iremos definir as regras de entrada/saída, tal como em Linux.

Para tal, abrimos a “Firewall do Windows Defender com Segurança Avançada”, escolhemos as opções “**Regras de Entrada/Nova Regra**”. De seguida, escolhemos a opção “**/Protocolo e Portas/Porta/TCP/Portas locais específicas**” e especificámos a porta (80 para http, 443 para https ou 3389 para acesso via rdp). Por fim especificámos o nome da regra e adicionámos.

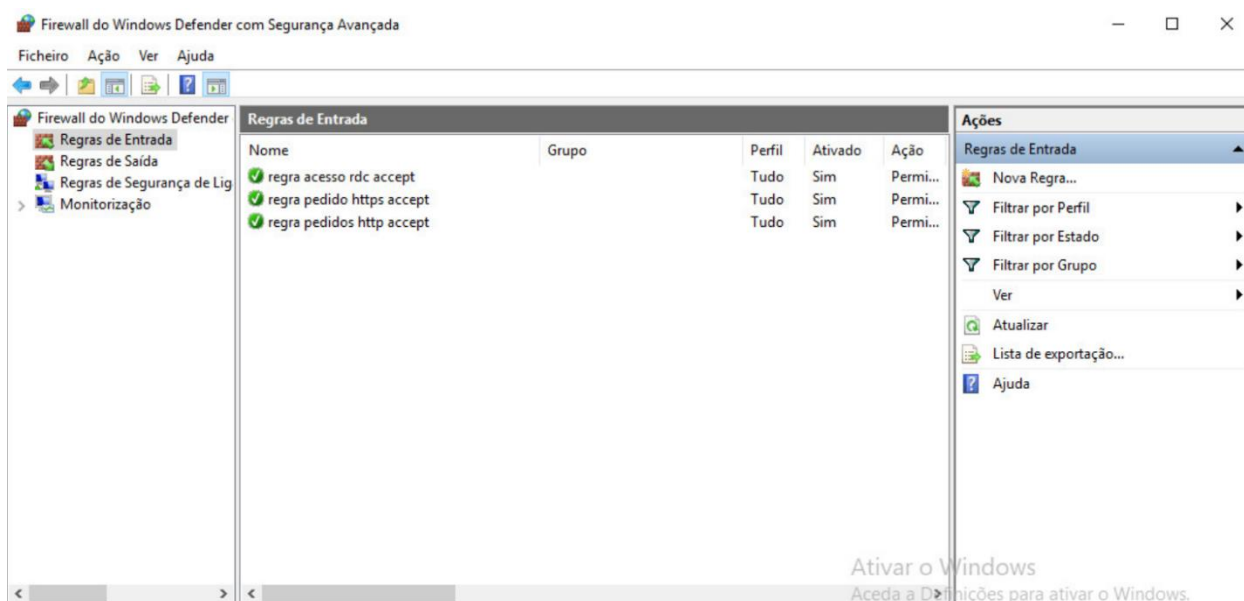


Figura 8: Regras de entrada adicionadas

USER STORY 4

4. Como administrador da infraestrutura quero impedir o **IP spoofing** na minha rede

Figura 9: User story 4

O IP spoofing é uma técnica maliciosa que ocorre quando o endereço de IP origem de pacotes é modificado para esconder a identidade de quem envia o pacote, fazendo-se passar por outro computador ou sistema.

LINUX

Em Linux, de forma semelhante à user story 3, utilizamos um comando iptables para rejeitar todo o tráfego exterior através da placa de rede primária (ens32), cujo IP de origem provém da segunda placa de rede (ens33 – 192.168.143.0/24). Esse comando foi adicionado ao script já criado também “iptables_commands.sh”.


```
Selecione OpenSSH SSH client
GNU nano 5.4 iptables_commands.sh
#!/bin/bash

# Apaga as regras existentes
iptables -F

# Permissão de http requests (porta 80)
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 80 -j ACCEPT

# Permissão de https requests (porta 443)
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT

# Permissão de acesso através de SSH (porta 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --sport 22 -j ACCEPT

# Prevenção de ip spoofing, a partir da placa de rede primária, com um host da placa de rede secundária
iptables -A INPUT -i ens32 -s 192.168.143.0/24 -j DROP

# Guardar as definições alteradas
iptables-save
```

Figura 10: Script /scripts/iptables_commands.sh com linha de prevenção de IP spoofing adicionada

Para verificar as regras adicionadas, tanto nesta user story, como na user story 3, podemos utilizar o comando iptables, com as opções -L.

```
root@uvm043:/scripts# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination           tcp dpt:http
ACCEPT     tcp  --  anywhere              anywhere              tcp spt:http
ACCEPT     tcp  --  anywhere              anywhere              tcp dpt:https
ACCEPT     tcp  --  anywhere              anywhere              tcp spt:https
ACCEPT     tcp  --  anywhere              anywhere              tcp dpt:ssh
ACCEPT     tcp  --  anywhere              anywhere              tcp spt:ssh
DROP       all  --  192.168.143.0/24      anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
root@uvm043:/scripts#
```

Figura 11: Verificação das regras adicionadas

WINDOWS

De forma semelhante à user story 3 e seguindo a mesma lógica utilizada em linux, adicionamos uma regra de entrada, neste caso que bloqueia o acesso. Colocámos o endereço remoto “192.168.143.0/24” (placa de rede secundária), e o protocolo e porta local não são especificados.

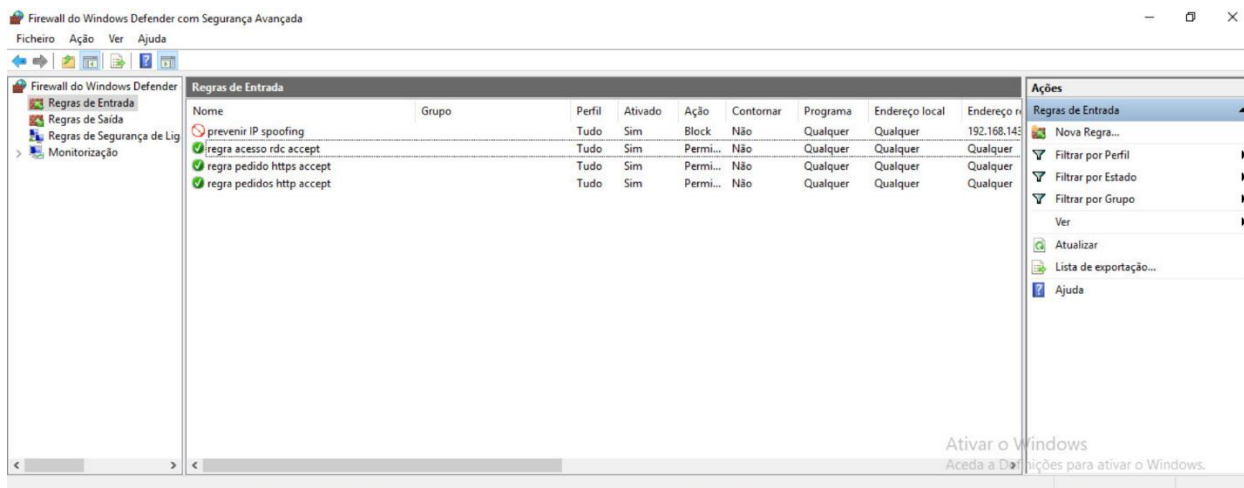


Figura 12: Regra de entrada adicionada para prevenção do ip spoofing

USER STORY 5

5. Como administrador da infraestrutura quero que os utilizadores registados noLinux com **UID** entre **6000** e **6500** só consigam aceder via **SSH** se esse acesso for a partir de uma máquina listada em **/etc/remote-hosts**

Figura 13: User story 5

LINUX

De modo a restringir o acesso aos users com id entre 6000 e 6500 via SSH, apenas se o acesso for a partir de uma máquina listada em “/etc/remote-hosts”, começámos por criar o ficheiro “/etc/remote-hosts”.



Figura 14: Ficheiro /etc/remote-hosts com os hosts dos nossos computadores

De seguida, adicionámos no topo do ficheiro “/etc/pam.d/sshd” as configurações para restringir o acesso aos utilizadores com id entre 6000 e 6500.

```
OpenSSH SSH client
GNU nano 5.4 /etc/pam.d/sshd
# PAM configuration for the Secure Shell service
auth [success=3 default=ignore] pam_succeed_if.so uid < 6000
auth [success=2 default=ignore] pam_succeed_if.so uid > 6500
auth [success=1 default=ignore] pam_listfile.so item=rhost sense=allow file=/etc/remote-hosts onerr=fail
auth requisite pam_deny.so
```

Figura 15: Configurações no ficheiro /etc/pam.d/sshd

Para testar, colocámos os nossos hosts (ver Figura 8) em “etc/remote-hosts” e tentamos conectar com o user “luser2”, que foi criado na resolução da ficha PL2 e cujo id é 6001. Tivemos sucesso no login por SSH quando o nosso host estava listado no ficheiro criado, mas quando retirámos os hosts, o acesso passa a ser negado. Para um user com id fora do intervalo restrito, o acesso é garantido sem qualquer restrição.

USER STORY 6

6. Como administrador da infraestrutura quero que o acesso ao sistema seja inibido aos utilizadores listados em /etc/bad-guys

Figura 16: User story 6

LINUX

Na resolução desta user story, começamos por criar o ficheiro “/etc/bad-guys”, e adicionamos o user “luser1”, para mais tarde testar.

```
OpenSSH SSH client
GNU nano 5.4 /etc/bad-guys
luser1
```

Figura 17: Ficheiro /etc/bad-guys criado

De seguida, e à semelhança da user story 5, alterámos o ficheiro “etc/pam.d/sshd”, escrevendo no topo do ficheiro, uma configuração que inibe o acesso aos utilizadores listados em “/etc/bad-guys”.

```
OpenSSH SSH client
GNU nano 5.4 /etc/pam.d/sshd
# PAM configuration for the Secure Shell service
auth required pam_listfile.so item=user sense=deny file=/etc/bad-guys onerr=fail
```

Figura 18: Configuração efetuado no ficheiro /etc/pam.d/sshd

Por fim, tentamos dar login com o user “luser1”, mas o acesso foi negado, devido à presença do user no ficheiro criado.

USER STORY 7

7. Como administrador da infraestrutura quero que as mensagens pré-login e pós-login bem-sucedido sejam dinâmicas (por exemplo, “[Bom dia] | [Boa tarde] *username*”, etc.)

Figura 19: User story 7

LINUX

Para a mensagem de pré-login, descomentámos a linha referente ao banner, no ficheiro “/etc/ssh/sshd.conf” e adicionámos o caminho para o banner criado, no caso em “/etc/ssh/banner”.

```
OpenSSH SSH client
GNU nano 5.4 /etc/ssh/sshd_config *
# no default banner path
Banner /etc/ssh/banner
```

Figura 20: Caminho para o banner em /etc/ssh/sshd.conf

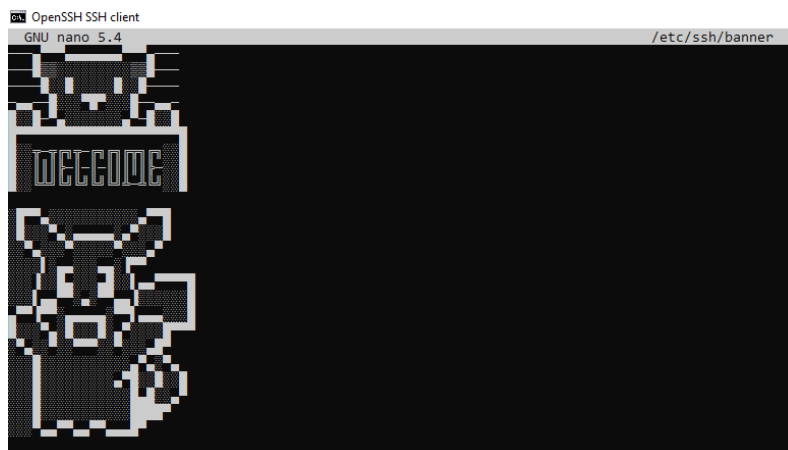


Figura 21: Banner criado

Para a mensagem de pós-login, alterámos o ficheiro “/etc/profile”, criando um script que imprime uma mensagem alterada consoante a altura do dia (manhã, tarde, noite) e o user que acedeu:

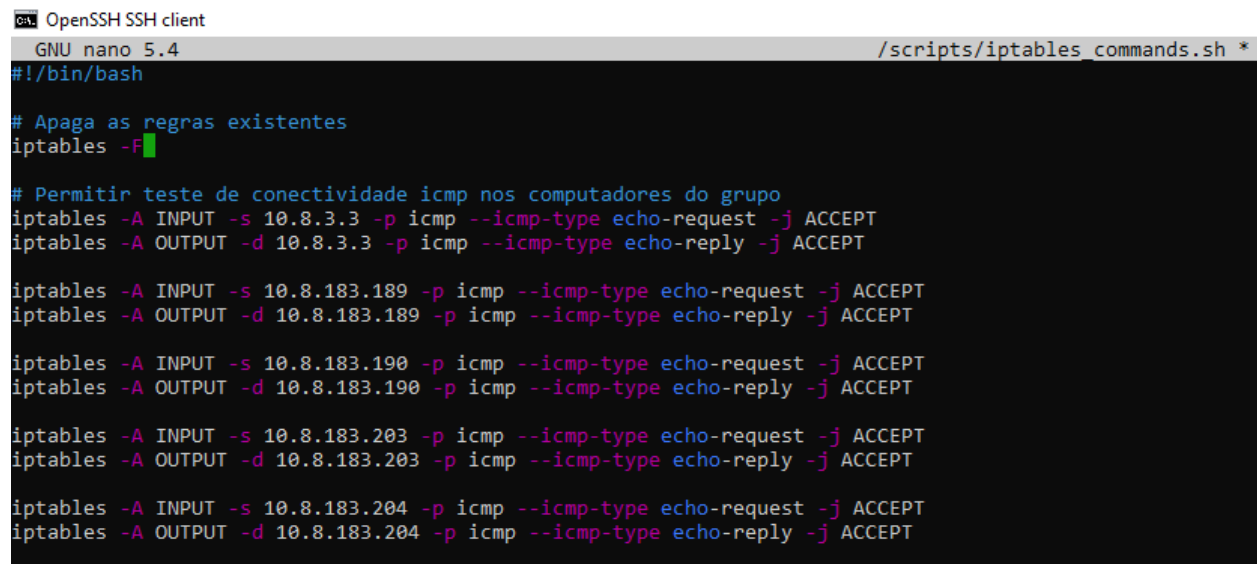
USER STORY 8

8. Como administrador da infraestrutura quero que o servidor Linux responda e envie pedidos **ICMP** para teste de conectividade apenas e só aos computadores dos elementos do grupo

Figura 24: User story 8

LINUX

Para realização desta user story, recorreremos ao script criado na user story 3, adicionando regras através do comando iptables que permitissem respostas e pedidos ICMP por parte do servidor Linux aos computadores dos elementos do grupo. De seguida, demos reboot de modo que o script fosse executado.



```
OpenSSH SSH client
GNU nano 5.4 /scripts/iptables_commands.sh *
#!/bin/bash

# Apaga as regras existentes
iptables -F

# Permitir teste de conectividade icmp nos computadores do grupo
iptables -A INPUT -s 10.8.3.3 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -d 10.8.3.3 -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A INPUT -s 10.8.183.189 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -d 10.8.183.189 -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A INPUT -s 10.8.183.190 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -d 10.8.183.190 -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A INPUT -s 10.8.183.203 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -d 10.8.183.203 -p icmp --icmp-type echo-reply -j ACCEPT

iptables -A INPUT -s 10.8.183.204 -p icmp --icmp-type echo-request -j ACCEPT
iptables -A OUTPUT -d 10.8.183.204 -p icmp --icmp-type echo-reply -j ACCEPT
```

Figura 25: Comandos inseridos no topo de /scripts/iptables_commands.sh

Contudo o endereço de IP de cada máquina não é sempre o mesmo, pelo que temos noção que esta configuração não seria sempre válida. Tentámos criar um script que adicionaria a regra, utilizando o valor atual do endereço de ip de cada máquina para as opções “-s” e “-d”, mas sem sucesso.

WEBGRAFIA

- <https://ixnfo.com/en/iptables-quick-setup-script.html>
- <https://www.baeldung.com/linux/run-script-on-startup>
- https://linux.die.net/man/8/pam_succeed_if
- <https://www.howtoforge.com/tutorial/install-and-configure-isc-dhcp-server-in-debian-9/>
- https://www.veritas.com/support/en_US/article.100027250