



RELATÓRIO ASIST

Sprint 2

3DB Grupo 9

Vicente Cardoso (1180664)

Ana Costa (1201313)

Luís Reis (1210998)

Ricardo Ribeiro (1221695)

Índice

Conteúdo

Introdução.....	2
Distribuição de tarefas	2
<i>User Stories</i>	3
<i>User Story 1</i>	3
<i>User Story 2</i>	4
<i>User Story 3</i>	12
<i>User Story 4</i>	15
<i>User Story 5</i>	19
<i>User Story 6</i>	21
<i>User Story 7</i>	25
<i>User Story 8</i>	27
Referências	30
Anexos	31

Introdução

Este relatório foi desenvolvido no contexto da Unidade Curricular de Administração de Sistemas (ASIST), como parte integrante da Licenciatura em Engenharia Informática no ano letivo 2024-2025.

Distribuição de tarefas

Este trabalho representa o *Sprint 2* e é constituído por oito *User Stories* (US), distribuídas da seguinte forma pelos diversos elementos da equipa:

ALUNO	USER STORIES
VICENTE CARDOSO (1180664)	3 e 6
ANA COSTA (1201313)	2 e 4
LUÍS REIS (1210998)	5 e 8
RICARDO RIBEIRO (1221695)	1 e 7

User Stories

Nesta secção serão apresentadas as diferentes tarefas e a solução apresentada pela equipa.

User Story 1

Descrição:

Explicação:

Procedimento:

Resultado:

User Story 2

Descrição:

“As system administrator, I only want clients on the DEI's internal network (wired or via VPN) to be able to access the solution.”

Explicação:

De forma a restringir a aplicação a clientes conectados à rede interna do DEI, foram utilizadas as seguintes sub-redes:

- *Wireless*

IP address	172.18.159.128
Network address	172.18.152.0
Usable host IP range	172.18.152.1 - 172.18.159.254

- *VPN*

IP address	10.8.199.190
Network address	10.8.0.0
Usable host IP range	10.8.0.1 – 10.8.255.254

Assim, configurou-se a firewall de modo a bloquear todo o tráfego de entrada por SSH, exceto o proveniente do DEI. Para tal, recorreu-se ao comando **iptables**, que permite filtrar os pedidos que atravessam as *chains* de *INPUT*, *FORWARD* e *OUTPUT*.

Este comando permite filtrar redes IPv4. Para IPv6 o script seria semelhante, mas seria utilizado o comando **ip6tables** e os IPs no respetivo formato.

Estas configurações foram feitas num Debian Server SSH/SFTP disponibilizado na DEI *Virtual Servers Private Cloud*, com o acesso público desativado.

Não se considerou a rede *Wired* do DEI porque não existia acesso por cabo, de modo a conseguir-se conectar e obter o IP. No entanto, para restringir o acesso a esta sub-rede, seria dado exatamente o mesmo tratamento prestado à rede *Wireless* e por *VPN*.

Procedimento:

1. Verificar que o *iptables* está instalado e visualizar a configuração *default*, na qual é permitida todo o tipo de ligações, seja na *Chain input*, *forward* ou *output*:

```
root@vs319:~# iptables --version
iptables v1.8.7 (nf_tables)
root@vs319:~#
```

Figura 1 - versão do iptables

```
root@vs319:~# iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
root@vs319:~#
```

Figura 2 - iptables default

2. De modo a garantir que as regras do *iptables* são persistidas após se reiniciar o sistema, deve-se instalar o pacote **iptables-persistent**, que guarda as regras do *iptables* no ficheiro **/etc/iptables/rules.v4** e as carrega automaticamente ao iniciar o sistema
3. Criar *script* com **nano firewall.sh** que terá as configurações da *firewall*:
 - a. Dar *drop* das configurações já existentes
 - i. Dar *flush* das regras da *chain* de *INPUT*
 - ii. Excluir as *chains* personalizadas
 - iii. Zerar os contadores de pacotes
 - b. Permitir tráfego *incoming wireless* do DEI por SSH (porta 22)
 - c. Permitir tráfego *incoming* da VPN do DEI por SSH (porta 22)
 - d. Permitir *loopback* no *localhost*
 - e. Permitir HTTP (80) e HTTPS (443)
 - f. Permitir as conexões já estabelecidas e as relacionadas (por exemplo, respostas de pedidos e pacotes adicionais que têm de ser enviados)
 - g. Bloquear tudo o resto que queira aceder por *INPUT* e *FORWARD* e aceitar o *OUTPUT*

- h. Após a instalação, guardar as regras atuais com **netfilter-persistent save**
- i. Reiniciar o serviço de persistência com **netfilter-persistent reload**
- j. Verificar as novas regras

```
GNU nano 5.4                               firewall.sh
#!/bin/bash

# clean existing rules (default, personalized, reset counters)
iptables -F
iptables -X
iptables -Z

# allow incoming from DEI wireless
iptables -A INPUT -p tcp --dport 22 -s 172.18.152.0/24 -j ACCEPT

# allow incoming from DEI VPN
iptables -A INPUT -p tcp --dport 22 -s 10.8.0.0/16 -j ACCEPT

# allow loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# allow HTTP and HTTPS
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# allow already established ou related
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# drop everything else
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# persist firewall rules after system restart
netfilter-persistent save
netfilter-persistent reload

# check rules
iptables -L -n
```

Figura 3 - script para configurar iptables

Explicação das flags utilizadas:

- **-F**: dá flush / remove as regras das chains default
- **-X**: exclui as regras das chains personalizadas
- **-Z**: zera os contadores de pacotes
- **-A INPUT**: adiciona a regra à *chain INPUT*
- **-p**: especifica o protocolo
- **--dport**: especifica a porta

- **-s:** a origem do pedido
- **-j:** *jump* ou destino dado ao pacote
- **-i:** incoming
- **-o:** outgoing
- **-m conntrack:** permite usar o modulo que rastreia outras conexões
- **--ctstate:** verifica o estado das conexões
- **-P:** define a política padrão
- **-L:** lista as regras
- **-n:** mostra os IPs e portas, em vez dos nomes

4. Dar permissões de execução ao *script* com **chmod +x firewall.sh**

5. Executar o *script* com **./firewall.sh**

6. Por fim, verificar a *firewall* através do comando **iptables -L -n**

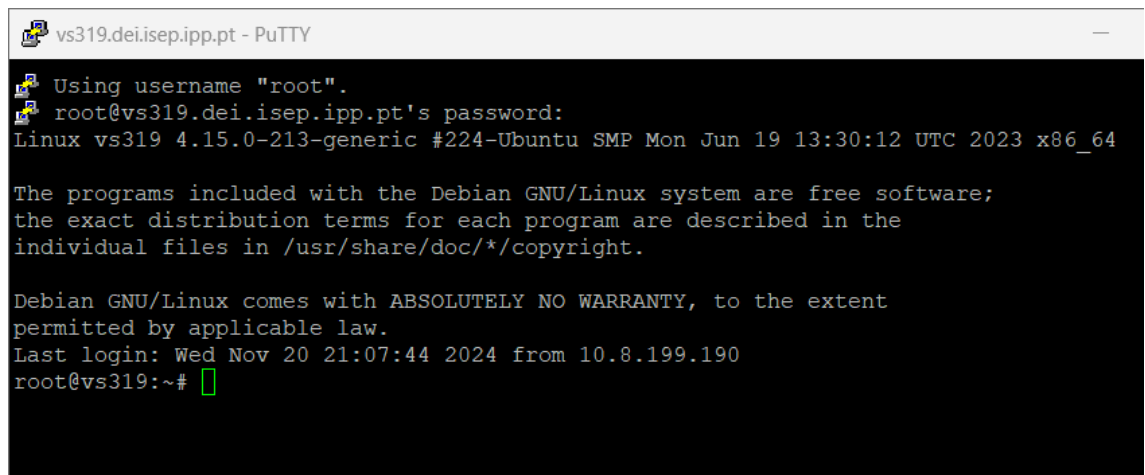
```
root@vs319:~# iptables -L -n
Chain INPUT (policy DROP)
target    prot opt source                destination            tcp dpt:22
ACCEPT    tcp  --  172.18.152.0/24        0.0.0.0/0              tcp dpt:22
ACCEPT    tcp  --  10.8.0.0/16           0.0.0.0/0              tcp dpt:22
ACCEPT    all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT    tcp  --  0.0.0.0/0             0.0.0.0/0              tcp dpt:80
ACCEPT    tcp  --  0.0.0.0/0             0.0.0.0/0              tcp dpt:443
ACCEPT    all  --  0.0.0.0/0             0.0.0.0/0              ctstate RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target    prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target    prot opt source                destination
ACCEPT    all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT    all  --  0.0.0.0/0             0.0.0.0/0              ctstate RELATED,ESTABLISHED
root@vs319:~#
```

Figura 4 - regras do iptables

7. Após novo acesso através de uma sub-rede do DEI, verifica-se que o servidor se encontra disponível:



```
vs319.dei.isep.ipp.pt - PuTTY
Using username "root".
root@vs319.dei.isep.ipp.pt's password:
Linux vs319 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 20 21:07:44 2024 from 10.8.199.190
root@vs319:~#
```

Figura 5 - login com sucesso

8. Para simular um pedido de uma rede que não pertença ao DEI e, portanto, que iria ter acesso bloqueado, tentou-se desligar a VPN. No entanto, o servidor não tem qualquer tipo de acesso se a VPN não estiver ativa.

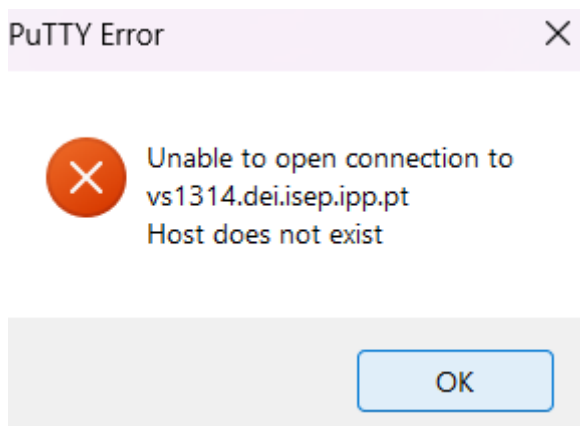


Figura 6 - erro ao conectar sem VPN

Deste modo, para testar o que aconteceria caso fosse feito um pedido de uma rede que a *firewall* não autoriza, mudou-se o *script* **firewall.sh** para apenas permitir pedidos *incoming* por SSH do IP **192.168.1.66** (um IP que não pertence à rede do DEI).

```
GNU nano 5.4 firewall.sh
#!/bin/bash

# clean existing rules (default, personalized, reset counters)
iptables -F
iptables -X
iptables -Z

# allow incoming from DEI wireless
#iptables -A INPUT -p tcp --dport 22 -s 172.18.152.0/24 -j ACCEPT

# allow incoming from DEI VPN
#iptables -A INPUT -p tcp --dport 22 -s 10.8.0.0/16 -j ACCEPT

# allow incoming from DEI VPN
iptables -A INPUT -p tcp --dport 22 -s 192.168.1.66 -j ACCEPT

# allow loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# allow HTTP and HTTPS
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# allow already established ou related
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# drop everything else
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# persist firewall rules after system restart
netfilter-persistent save
netfilter-persistent reload

# check rules
iptables -L -n
```

Figura 7 - script para apenas permitir IP fora do DEI

A *firewall* tem a seguinte configuração:

```

run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables start
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables start
Chain INPUT (policy DROP)
target     prot opt source                destination
ACCEPT     tcp  --  192.168.1.66           0.0.0.0/0           tcp dpt:22
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0             0.0.0.0/0           tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0             0.0.0.0/0           tcp dpt:443
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0           ctstate RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0           ctstate RELATED,ESTABLISHED
root@vs319:~#

```

Figura 8 - regras do iptables que apenas permitem IP fora do DEI

Quando se tenta a conexão após aplicar estas configurações, esta é bloqueada pela *firewall*:

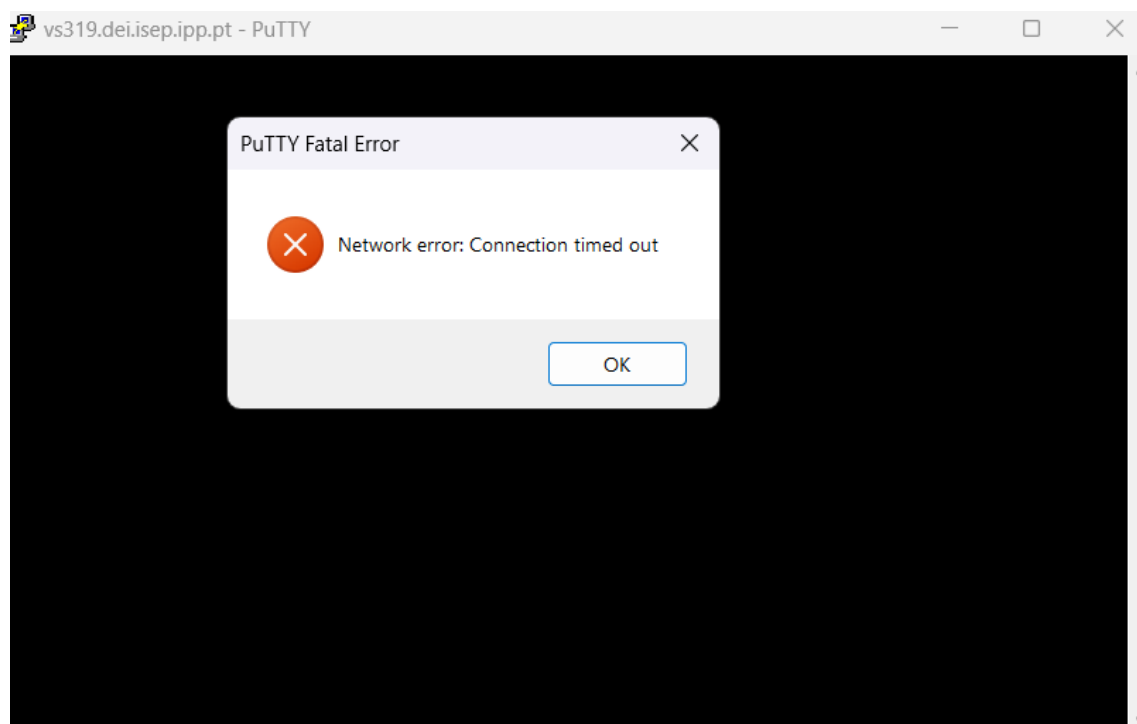


Figura 9 - acesso bloqueado

Isto bloqueou o acesso ao servidor, pelo que este foi reposto e aplicou-se as configurações da *firewall* do ponto 3, para apenas permitir tráfego *incoming* por SSH das sub-redes do DEI.

Nota:

Uma vez que este servidor pertence à rede do DEI, se o acesso público estiver ativo, independentemente do IP de origem do pedido, este é transformado num IP pertencente à rede do DEI, possivelmente através de um *proxy*.

Deste modo, qualquer pedido *incoming* para o servidor seria tratado de igual forma, sendo aceite ou recusado dependendo das regras da *firewall*.

Por exemplo, se a *firewall* apenas aceitar todos os pedidos *incoming* de uma sub-rede do DEI, qualquer pedido conseguiria entrar no servidor, mesmo que não seja proveniente desta rede.

De igual modo, se a *firewall* bloquear todos os pedidos que originam numa sub-rede do DEI, mesmo aqueles que não provêm de lá seriam bloqueados.

Assim, apenas se considerou o cenário em que o acesso público está inativo.

Nota 2:

Uma vez que o servidor onde está configurada esta *firewall* contém o módulo de planeamento (porta 8080), é preciso garantir que a *firewall* permite pedidos que pretendem comunicar com esse módulo. É preciso, também, garantir o acesso ao protocolo *Git* (porta 9418), de modo a conseguir clonar o repositório e dar *pull*, para obter o módulo de planeamento.

Assim, seria necessário adicionar a seguinte configuração ao script *firewall.sh*:

```
iptables -A INPUT -p tcp --dport 9418 -j ACCEPT
```

```
iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
```

User Story 3

Descrição:


“As system administrator, I want the clients listed in the requirement 6.4.2 to be able to be defined by simply changing a text file.”

Explicação:

Com os clientes definidos, queremos implementar uma verificação constante e automática de um ficheiro de configuração da firewall. Caso haja alterações, devem ser aplicadas no sistema.

Procedimento:

1. Criar os ficheiros “/etc/currentfirewall.rules” e “/etc/firewall.rules”. O sistema vai comparar o segundo ficheiro com o primeiro para verificar alterações. Para este exemplo, currentfirewall.rules vai estar vazio.



```
GNU nano 5.4 /etc/firewall.rules
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]

-A INPUT -i lo -j ACCEPT

-A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

-A INPUT -p tcp --dport 22 -j ACCEPT
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT

-A INPUT -p icmp -j ACCEPT

-A INPUT -p tcp --dport 3306 -j ACCEPT

COMMIT
```

2. Se não foram ainda configuradas, inicializar as iptables com o script firewall.sh.

```

GNU nano 5.4 firewall.sh *
#!/bin/bash

iptables -F
iptables -X
iptables -Z

iptables -A INPUT -p tcp --dport 22 -s 172.18.152.0/24 -j ACCEPT

iptables -A INPUT -p tcp --dport 22 -s 10.8.0.0/16 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

netfilter-persistent save
netfilter-persistent reload

iptables -L -n

```

```

root@debian:~# ./firewall.sh
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
# Warning: iptables-legacy tables present, use iptables-legacy-save to see them
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables start
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables start
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy DROP)
target     prot opt source                destination           tcp dpt:22
ACCEPT     tcp  --  172.18.152.0/24        0.0.0.0/0             tcp dpt:22
ACCEPT     tcp  --  10.8.0.0/16           0.0.0.0/0             tcp dpt:22
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0             ctstate RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0             0.0.0.0/0             ctstate RELATED,ESTABLISHED
root@debian:~# _

```

3. Criar o script “check_firewall.sh”.

```

GNU nano 5.4 check_firewall.sh
#!/bin/bash

RULES_FILE="/etc/firewall.rules"
CURRENT="/etc/currentfirewall.rules"

if [ ! -f "$CURRENT" ]; then
    cp "$RULES_FILE" "$CURRENT_RULES_FILE"
    iptables-restore < "$RULES_FILE"
    exit 0
fi

if ! cmp -s "$RULES_FILE" "$CURRENT"; then
    iptables-restore < "$RULES_FILE"
    if [ $? -eq 0 ]; then
        cp "$RULES_FILE" "$CURRENT"
    fi
fi

```

4. Correr o script. Podemos verificar que funcionou pois a conexão ‘tcp 3306’, referente a MySQL, aparece na configuração.

```

root@debian:~# ./check_firewall.sh
root@debian:~# iptables -L -v
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy DROP 39 packets, 4939 bytes)
  pkts bytes target     prot opt in     out     source destination
    0      0 ACCEPT     all  --  lo      any      anywhere anywhere
    2  1042 ACCEPT     all  --  any     any      anywhere anywhere           ctstate REL
  ATED,ESTABLISHED
    0      0 ACCEPT     tcp  --  any     any      anywhere anywhere           tcp dpt:ssh
    0      0 ACCEPT     tcp  --  any     any      anywhere anywhere           tcp dpt:htt
  p
    0      0 ACCEPT     tcp  --  any     any      anywhere anywhere           tcp dpt:htt
  ps
    0      0 ACCEPT     icmp --  any     any      anywhere anywhere
    0      0 ACCEPT     tcp  --  any     any      anywhere anywhere           tcp dpt:mys
  ql

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source destination

Chain OUTPUT (policy ACCEPT 2 packets, 140 bytes)
  pkts bytes target     prot opt in     out     source destination
root@debian:~#

```

5. Configurar no cron a execução automática do script `check_firewall.sh`. Neste caso será feita todos os dias à meia-noite.

```

GNU nano 5.4 /tmp/crontab.kjJauM/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * /usr/local/bin/update_issue.sh
0 2 * * 0 /root/full_backup.sh
0 0 * * * /usr/local/bin/check_firewall.sh

```

Resultado:

Esta implementação permite alterar o ficheiro `firewall.rules` para atualizar a configuração da firewall. Através do cron, isto é feito de forma automática e diariamente.

User Story 4

Descrição:

“As an administrator, I want to identify and quantify the risks involved in the recommended solution.”

Explicação:

De forma a identificar e consequentemente minimizar os riscos encontrados na solução desenvolvida, será de seguida apresentada uma lista com os problemas identificados e possíveis soluções para os mitigar.

Pontos de entrada do sistema

A aplicação utiliza as seguintes portas para suportar os seus diferentes módulos:

Módulo	Porta
SPA	4200
3D	4201
Auth	7058
Backoffice	5001 (HTTPS) e 5000 (HTTP)
Planeamento	8080

Com base na análise das portas selecionadas, é possível concluir que, uma vez que usar HTTP não encripta os dados durante a sua transmissão, isto constitui-se como um risco, sendo preferível usar HTTPS.

Para além disto, a porta por onde é realizada a autenticação deve ser protegida, nomeadamente através de limitações ao número de tentativas de login e usando autenticação multifator, bem como sistemas de autenticação como o *Token Web JSON* (JWT) (JWT, n.d.).

Todas as portas devem ser protegidas, existindo regras na *firewall* para restringir os acessos, como vai ser explicado de seguida.

Origem dos pedidos REST

Embora a origem dos pedidos que querem comunicar com a nossa aplicação não seja uma falha de segurança por si só, importa garantir a autenticação, autorização e validação desses pedidos.

Para garantir que apenas utilizadores autorizados possam fazer pedidos, deve-se garantir que a autenticação utiliza um método seguro, como JWT.

Deve-se também restringir o pedido de origem, utilizando *Cross-Origin Resource Sharing (CORS)*, de forma a controlar os domínios que conseguem aceder à aplicação (Mdn web docs, n.d.).

Os pedidos a uma API devem ser através de HTTPS para garantir a encriptação dos dados a serem transmitidos, evitando ataques *man-in-the-middle* (IBM, n.d.).

Bases de dados

Uma vez que as bases de dados utilizadas, nomeadamente *SQLite* e *Microsoft SQL Server*, são externas à aplicação, verifica-se que o controlo sobre os seus acessos é menor. Assim, o risco de as informações armazenadas serem acedidas por pessoas não autorizadas está presente.

Riscos:

- *SQL injection*, quando se faz *queries* diretamente através de *strings* concatenadas, conseguindo aceder a dados armazenados (W3 schools, n.d);
- Se as credenciais da BD forem comprometidas, pode existir acessos indevidos e os dados ficam expostos;
- As configurações *default*, como portas abertas e permissões amplas, podem ser um alvo fácil para atacantes;
- Se *backups* não forem realizados com a frequência e a segurança adequadas, pode existir perda de dados ou a sua exposição;
- Pode ainda ocorrer um *Denial of Service (DoS)*, caso a BD seja sobrecarregada com pedidos, de forma a provocar a falha do serviço (CloudFlare, n.d.).

De forma a mitigar estes problemas, apresentam-se as seguintes soluções:

- Para evitar *SQL injections*, podem ser utilizadas *queries* com parâmetros ou ORM (*Object Relational Mapping*) (W3 schools, n.d);
- Encriptar dados sensíveis, como passwords e outras informações pessoais e usar SSL/TLS para encriptar a comunicação com a BD;
- Armazenar as informações de forma mais segura, por exemplo, usando variáveis de ambiente;
- Alterar as configurações *default*, tal como desativar funcionalidades que não estão a ser utilizadas e alterar as portas;
- Implementar *backups* regulares, encriptados e armazenados em local seguro
- Definir limites de pedidos e configurar o comportamento da BD em cenários de carga excessiva.

Firewall

A firewall foi configurada através de um *script* (US 6.4.2) de forma a apenas permitir pedidos SSH provenientes de dentro da rede do DEI. No entanto, caso fosse necessário alterar as redes com permissão para fazerem pedidos à aplicação, seria necessário alterar as regras da firewall, o que poderia provocar problemas, nomeadamente:

- perda de conexão ou a permissão;
- bloqueio de IPs indesejados;
- conflito de regras.

De modo a reduzir estes riscos, poder-se-ia aplicar as seguintes medidas:

- manter uma segunda sessão ativa, nomeadamente por SSH, para caso a conexão principal seja fechada não se perca o acesso;
- realizar backups das regras do *iptables*, antes de aplicar mudanças;
- aplicar as novas regras sequencialmente e pela ordem adequada, para garantir que não há interferências entre elas;
- documentar as regras e consultar os *logs* (*/var/log/syslog*).

Ficheiro com os IPs dos clientes

A existência de um ficheiro com os IPs dos clientes que conseguem aceder à firewall (US

6.4.3) constitui um risco acrescido, pois:

- o IP é considerado um dado pessoal no âmbito do Regulamento Geral de Proteção de Dados (RGPD), sendo um dado privado que não deve ser acedido (Your Europe, n.d.);
- ficheiros de texto estão suscetíveis a acessos não autorizados e podem ser expostos acidentalmente;
- se estiverem armazenados em texto simples, não estão encriptados;
- se todos os IPs estiverem num único local, ou seja, se o seu armazenamento estiver centralizado, se esse ficheiro for comprometido toda a informação é acedida.

Assim, as soluções propostas passam por:

- Encriptar os IPs antes de os armazenar e guardar a chave de encriptação de forma segura;
- Limitar o acesso ao ficheiro com restrições (leitura, escrita);
- Guardar os IPs apenas se existir essa necessidade e eliminá-los após um certo período, definindo uma política de retenção dos dados;
- Distribuir e armazenar a informação de forma segura.

User Story 5

Descrição:

Como administrador do sistema quero que seja definido o MBCO (Minimum Business Continuity Objective) a propor aos stakeholders

Explicação:

O **MBCO** é o nível mínimo de serviços ou funções essenciais que uma organização deve garantir durante ou após um incidente que comprometa a operação normal. Este objetivo serve como referência para planear as estratégias de continuidade de negócio e assegurar que os processos críticos continuam a funcionar, mesmo em situações de emergência.

No sistema em questão, existem diferentes módulos que suportam funcionalidades como autenticação (Auth), planeamento (Planeamento), backoffice (Backoffice) e SPA (Single Page Application). Estes módulos estão associados a várias portas de comunicação, e cada um desempenha um papel fundamental na operação do sistema.

O objetivo do MBCO é assegurar que, em caso de interrupção, os módulos mais críticos continuem a funcionar, minimizando o impacto nos serviços prestados.

Proposta do MBCO: Após análise dos requisitos e funcionalidades do sistema, o MBCO é definido como:

- Garantir a disponibilidade dos seguintes módulos durante uma interrupção:
 - Módulo de autenticação (Auth): Essencial para validar utilizadores e manter a segurança.
 - Planeamento: Necessário para continuar operações críticas.
- Tempo máximo de interrupção aceitável (Maximum Tolerable Downtime - MTD): 2 horas.
- Nível de performance mínimo aceitável: 70% da capacidade normal, permitindo o processamento de pedidos prioritários.
- Utilizadores prioritários: Clientes internos e utilizadores previamente autenticados.

Justificação:

- O módulo de autenticação é essencial para qualquer operação, pois sem autenticação os utilizadores não conseguem aceder aos serviços.
- O módulo de planeamento foi identificado como prioritário devido à sua relevância para manter a continuidade das operações críticas.
- O limite de 2 horas foi estabelecido com base no impacto financeiro e operacional de uma interrupção mais prolongada.

Estratégias Propostas:

1. **Redundância de sistemas:** Configurar servidores de backup para os módulos críticos (Auth e Planeamento) com sincronização em tempo real.
2. **Planos de recuperação:** Implementar um plano de recuperação rápida (Disaster Recovery Plan) para restaurar o sistema principal no prazo de 2 horas.
3. **Testes regulares:** Realizar simulações e testes periódicos para validar a eficácia do plano de continuidade.
4. **Monitorização contínua:** Configurar alertas e logs em tempo real para identificar e resolver falhas antes que estas comprometam os serviços.

Conclusão: O **MBCO** proposto garante a continuidade dos serviços essenciais (autenticação e planeamento), minimizando os impactos operacionais em caso de interrupção. Com estratégias de redundância, recuperação rápida e monitorização contínua, a solução assegura que os requisitos críticos dos stakeholders são cumpridos.

User Story 6

Descrição:

“As system administrator, I want a backup strategy to be proposed, justified and implemented that minimizes RPO (Recovery Point Objective) and WRT (Work Recovery Time).”

Explicação:

Uma estratégia de backup é uma política de salvaguarda de dados baseada na produção contínua de cópias dos dados do sistema. Para este sistema, foi indicada a priorização de RPO (Recovery Point Objective) e WRT (Work Recovery Time).

Entre as três principais estratégias de backup (full, incremental e differential), categorizámo-las da seguinte forma:

Full: Pior RPO, Melhor WRT – A diferença de dados entre duas cópias é maior, logo há uma maior perda de dados. Contudo, como os dados estão todos centralizados numa cópia, resgatar tais dados é extremamente rápido.

Differential: Médio RPO, Médio WRT – Dado que na estratégia differential cada cópia guarda apenas as mudanças de dados comparadas com a cópia original, a perda de dados de cópia a cópia é menor. Isto implica também que a recuperação de dados seja comparada com a cópia original, levando a um WRT menor que a estratégia full.

Incremental: Melhor RPO, Pior WRT – As cópias incrementais são mais leves e podem ser feitos mais frequentemente sem grande custos de armazenamento, oferecendo um RPO forte. É importante referir, no entanto, que, caso se perca uma cópia incremental antiga, todas as cópias posteriores passam a ser inutilizáveis, algo que não acontece com a estratégia differential. Isto acontece porque é necessário comparar todas as cópias incrementais com a original, levando a um pobre WRT.

Após a avaliação de cada estratégia, deduzimos que a **differential** alcança não só um bom equilíbrio entre RPO e WRT, como é também leve nos custos de armazenamento. Infelizmente, devido à difícil execução da estratégia differential, decidimos recorrer à estratégia **full**.

Procedimento:

1. Começamos por criar os seguintes diretórios:
 - /backups/full - armazena os diferentes backups.
 - /backup_logs – armazena os logs referentes a cada processo de backup.
2. Instalar a ferramenta 'rsync' com os comandos 'apt update' e 'apt install rsync'.
3. Verificar que o processo 'cron' está instalado e a correr.
5. Criar o script full_backup.sh com nano. O script cria um novo diretório cujo nome refere-se à data de criação do backup. O comando 'rsync -av --delete' copia todos os ficheiros e diretórios do local fonte (/home neste caso) para o novo diretório. A operação é registada num ficheiro log.

```
GNU nano 5.4 /root/full_backup.sh
#!/bin/bash

SOURCE="/home"
DEST="/backups/full"
TIMESTAMP=$(date +%Y%m%d%H%M%S)
LOGFILE="/backup_logs/full/full_backup$TIMESTAMP.log"
BACKUP_DIR="$DEST/$TIMESTAMP_"

mkdir -p "$BACKUP_DIR"

rsync -av --delete "$SOURCE" "$BACKUP_DIR" >> "$LOGFILE" 2>&1
echo "Full backup completed on $(date)" >> "$LOGFILE"
```

6. Podemos testar o script para verificar a cópia criada.

```

root@debian:/home# ls
aquota.group  asist      luser1  luser3  luser5    luser7  usert
aquota.user   lost+found luser2  luser4  luser6home ricardo
root@debian:/home# cd ~
root@debian:~# ./full_backup.sh
root@debian:~# cd /backups/full
root@debian:/backups/full# ls
20241124193908
root@debian:/backups/full# cd 20241124193908
root@debian:/backups/full/20241124193908# ls
home
root@debian:/backups/full/20241124193908# cd home
root@debian:/backups/full/20241124193908/home# ls
aquota.group  asist      luser1  luser3  luser5    luser7  usert
aquota.user   lost+found luser2  luser4  luser6home ricardo
root@debian:/backups/full/20241124193908/home#

```

7. Se houver mudanças no diretório fonte, uma nova cópia irá guardá-las.

```

root@debian:/home# ls
aquota.group  asist      luser1  luser3  luser5    luser7  usert
aquota.user   lost+found luser2  luser4  luser6home ricardo
root@debian:/home# touch test.txt
root@debian:/home# ls
aquota.group  asist      luser1  luser3  luser5    luser7  test.txt
aquota.user   lost+found luser2  luser4  luser6home ricardo  usert
root@debian:/home# cd ~
root@debian:~# ./full_backup.sh
root@debian:~# cd /backup_full
-bash: cd: /backup_full: No such file or directory
root@debian:~# cd /backups/full
root@debian:/backups/full# ls
20241124193908 20241124194309
root@debian:/backups/full# cd 20241124194309/home
root@debian:/backups/full/20241124194309/home# ls
aquota.group  asist      luser1  luser3  luser5    luser7  test.txt
aquota.user   lost+found luser2  luser4  luser6home ricardo  usert
root@debian:/backups/full/20241124194309/home#

```

8. Por fim, atualizamos o processo cron para correr o script que criámos. Como estamos a fazer cópias dos dados todos, não convém fazermos backup muito regularmente, de modo a evitar esgotar o armazenamento disponível muito rapidamente. Por isso, definimos que o backup seja realizado uma vez por semana, às 2:00h de domingo com a linha '0 2 * * 0 /root/full_backup.sh' (0 e 2 representam os minutos e hora enquanto o último 0 refere ao dia de domingo).


```

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
* * * * * /usr/local/bin/update_issue.sh

0 2 * * 0 /root/full_backup.sh

```

Resultado:

Com esta implementação, temos um processo de backup automático capaz de gerar cópias de todos os dados. Embora exista uma maior perda potencial de dados com esta abordagem, a obtenção dos mesmo é quase imediata, estando cada cópia devidamente identificada consoante a data de backup.

User Story 7

Descrição: Como administrador do sistema quero definir uma pasta pública para todos os utilizadores registados no sistema, onde podem ler tudo o que lá for colocado.

Explicação:

Criar um grupo que contem todos os utilizadores normais (UID > 1000) e definir permissões de apenas leitura para esse grupo.

Procedimento:

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

```
PUBLIC_FOLDER="/publicFolder"
```

```
GROUP_NAME="ASIST64"
```

```
# Ensure the group exists
```

```
getent group "$GROUP_NAME" >/dev/null || groupadd "$GROUP_NAME"
```

```
# Create the public folder
```

```
mkdir -p "$PUBLIC_FOLDER"
```

```
chown root:"$GROUP_NAME" "$PUBLIC_FOLDER"
```

```
chmod 755 "$PUBLIC_FOLDER"
```

```
# Add all valid users to the group
```

```
for username in $(awk -F: '$3 >= 1000 || $3 == 0 {print $1}' /etc/passwd); do
```

```
    usermod -aG "$GROUP_NAME" "$username"
```

```
done
```

```
echo "Public folder setup complete. Path: $PUBLIC_FOLDER"
```

Resultado:

Criação de um directorio “/publicFolder” onde apenas o utilizador root consegue escrever e todos os outros utilizadores apenas conseguem ler.

User Story 8

Descrição:

Como administrador do sistema quero obter os utilizadores com mais do que 3 tentativas de acesso incorretas.

Procedimento:

Verificar se no ficheiro /etc/ssh/ssh_config na linha LogLevel se encontra assim

```
GNU nano 5.4          etc/ssh/sshd_config
# $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO
LogLevel VERBOSE
# Authentication:
-
#LoginGraceTime 2m
```

Adicionar VERBOSE

Reiniciar o serviço ssh

De seguida, criar o script

```
GNU nano 5.4 /usr/local/bin/failed_logins.sh *
#!/bin/bash

#Obter a data de hoje no formato dos logs
today=$(date '+%b %d')

#caminho do log de autenticação
auth_log="/var/log/auth.log"

#verificar as tentativas falhadas e contar por utilizador e verificar os que têm 3 ou mais
grep 'Failed password' "$auth_log" | awk '{print $(NF-5)}' | sort | uniq -c | awk '$1 >= 3 {print $1}
```

Criar o script na pasta /usr/local/bin

“today” vai obter a data no formato usado nos logs

O script vai ler o ficheiro /var/log/auth.log e filtra as mensagens que tenham “Failed password”

awk '{print \$(NF-5)}' - Vai buscar o nome do utilizador que é o 5º campo a partir do fim da linha

As ocorrências vão ser contabilizadas e por fim vai guardar apenas os utilizadores com pelo menos 3 tentativas falhadas

Tornar o script executável:

```
root@debian:/# chmod +x /usr/local/bin/failed_logins.sh _
```

Resultado:

Por fim para testar pode-se fazer da seguinte forma

ssh [nomeUtilizador@10.9.10.9](#)

E depois executar o script

/usr/local/bin/failed_logins.sh

E vai aparecer o número de tentativas e os utilizadores com mais de 3 tentativas falhadas de login como no exemplo

```
root@debian:/# /usr/local/bin/failed_logins.sh
root@debian:/# ssh 1210998@10.9.10.9
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
1210998@10.9.10.9: Permission denied (publickey,password).
root@debian:/# /usr/local/bin/failed_logins.sh
root@debian:/# ssh 1210998@10.9.10.9
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
1210998@10.9.10.9: Permission denied (publickey,password).
root@debian:/# ssh 1210998@10.9.10.9
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
Permission denied, please try again.
1210998@10.9.10.9's password:
1210998@10.9.10.9: Permission denied (publickey,password).
root@debian:/# /usr/local/bin/failed_logins.sh
9 1210998
```

Referências

CloudFlare. (n.d.). *What is a denial-of-service (DoS) attack?* | Cloudflare. Retrieved November 21, 2024, from <https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/>

IBM. (n.d.). *What Is a Man-in-the-Middle (MITM) Attack?* | IBM. Retrieved November 21, 2024, from <https://www.ibm.com/think/topics/man-in-the-middle>

JWT. (n.d.). *JSON Web Token Introduction* - jwt.io. Retrieved November 21, 2024, from <https://jwt.io/introduction>

Mdn web docs. (n.d.). *Cross-Origin Resource Sharing (CORS) - HTTP* | MDN. Retrieved November 21, 2024, from <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS>

W3 schools. (n.d.). *SQL Injection*. Retrieved November 21, 2024, from https://www.w3schools.com/sql/sql_injection.asp

Your Europe. (n.d.). *Proteção de dados ao abrigo do RGPD* - Your Europe. Retrieved November 21, 2024, from https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_pt.htm

Anexos