

Relatório de ASIST

Sprint 2

Turma 3DI - Grupo 50

1191296 - Gabriel Gonçalves

1191369 - Tiago Leite

1201305 - Tiago Afonso

1211304 - Francisco Bogalho

Data: 26/11/2023

Índice

Índice de quadros, figuras, abreviaturas	3
Distribuição de tarefas	4
User Story 1.....	5
User Story 2.....	9
User Story 3.....	11
User Story 4.....	14
User Story 5.....	15
User Story 6.....	16
User Story 7.....	19
User Story 8.....	22

Índice de quadros, figuras, abreviaturas

Figura 1 - Script com o nome buildMDTask.sh, que foi criada para o deploy sistemático	5
Figura 2 - Configuração do acesso SSH do servidor ao repositório no Bitbucket	6
Figura 3 - Sistematização do deploy do módulo MasterData através da utilização do crontab	7
Figura 4 - Log do deploy sistemático no dia 25/11/2023 às 2h:30min.....	8
Figura 5 - ipconfig.....	9
Figura 6 - firewall.sh bash script	9
Figura 7 - Criação do ficheiro para colocação dos endereços IP.....	11
Figura 8 - Listagem das regras do filtro de pacotes IPv4 em antes da realização da US	11
Figura 9 - Permitir acesso aos IPs indicados no ficheiro ipList.txt	12
Figura 10 - Negar acesso a todas as outras ligações.....	12
Figura 11 - Listagem das regras do filtro de pacotes IPv4 depois da US	13
Figura 12 - Matriz de Risco.....	14
Figura 13 - Script para realizar cópias de segurança da base de dados em MongoDB Erro! Marcador não definido.	
Figura 14 - Sistematização das cópias de segurança diárias através da configuração do crontab.....	18
Figura 15 - Criação da pasta partilhada	19
Figura 16 - Atribuição de permissões de leitura na pasta partilhada	19
Figura 17 - Criação de um ficheiro na pasta partilhada	19
Figura 18 - Colocação de texto no ficheiro criado	19
Figura 19 - Verificação das permissões dos ficheiros da pasta partilhada	20
Figura 20 - Teste de acesso à pasta partilhada	20
Figura 21 - Teste de edição do ficheiro manual.txt	20
Figura 22 - Teste de criação de ficheiro na pasta partilhada	21
Figura 23 Bash Script que encontra os utilizadores com mais do que N acessos incorretos	22
Figura 24 Exemplo de execução do script find_users_with_excessive_failed_logins.sh	22

Distribuição de tarefas

As *User Stories* foram distribuídas pelos membros do grupo da seguinte forma:

User Story	Membro
1	Tiago Leite (1191369)
2	Tiago Afonso (1201305)
3	Gabriel Gonçalves (1191296)
4	Francisco Bogalho (1211304)
5	Francisco Bogalho (1211304)
6	Tiago Leite (1191369)
7	Gabriel Gonçalves (1191296)
8	Tiago Afonso (1201305)

User Story 1

Para esta *User Story* (US), é pedido que “como administrador do sistema quero que o *deployment* de um dos módulos do RFP numa VM do DEI seja sistemático, validando de forma agendada com o plano de testes”.

O módulo RFP escolhido para realizar esta US foi o *MasterData* (MD), ou seja, o *Backend*.

Primeiro criou-se o caminho de pastas “/root/apps/RobDroneGo-MD” (com a sequência de comandos “*cd ~; mkdir apps; cd apps; mkdir RobDroneGo-MD*”), onde irá estar o *script* a ser executado sistematicamente e a aplicação para o *deploy*. Aqui nesta localização, criou-se o script, com o comando “*touch buildMDTask.sh*”, e alterou-se as permissões do ficheiro para que apenas o *owner* (*root*), possa ler, escrever e executar o ficheiro, em que para isso se utilizou o comando “*chmod 700 buildMDTask.sh*”.

A *script* foi escrita da seguinte forma:

```
#!/bin/bash

cd /root/apps/RobDroneGo-MD/sem5pi-23-24-50/MD

echo "***** The following information refers to the run in the date $(date) *****"

echo -e '\n----- Pulling code from remote... -----'

git pull origin main

echo -e '\n----- Checking and installing new packages if they are required -----'

npm install

echo -e '\n----- Building project... -----'

npm run build

if [ $? -eq 0 ]; then
    echo -e '\n----- The application has been built successfully! -----\n'

    echo -e '\n----- Testing the application... -----'
    npm test tests/**/*.test.ts

    if [ $? -eq 0 ]; then
        echo -e '\n----- The application tests are all passing with success. Congratulations! :D -----\n'
        if [ "$(pm2 id RobDroneGo)" = "[]" ]; then
            echo '----- Starting the application service... -----'
            pm2 start npm --name "RobDroneGo" -- start
        else
            echo '----- Restarting the application service... -----'
            pm2 restart RobDroneGo
        fi
    else
        echo '----- Some tests failed. Feels bad man! :( -----'
    fi
else
    echo '----- The application could not be built. Something bad happened. -----'
fi
```

Figura 1 - Script com o nome *buildMDTask.sh*, que foi criada para o deploy sistemático

Na imagem acima, é possível verificar os diferentes passos realizados pela *script*. Primeiramente, entramos na pasta onde o módulo RFP correspondente ao *Backend* se encontra, utilizando o comando “*cd*”.

De seguida, é verificado se existe novas alterações no repositório, mais especificamente na *branch main*, que é onde se encontra o código que, em princípio, está 100% funcional. Para que este comando seja possível, o repositório teve de ser clonado anteriormente neste servidor. Para isso foi necessário configurar permissões no *Bitbucket*, em que foi adicionada a chave pública do servidor para este ganhar permissões de apenas leitura ao repositório via SSH. Após isto, foi utilizado no servidor o comando para dar clone do projeto.

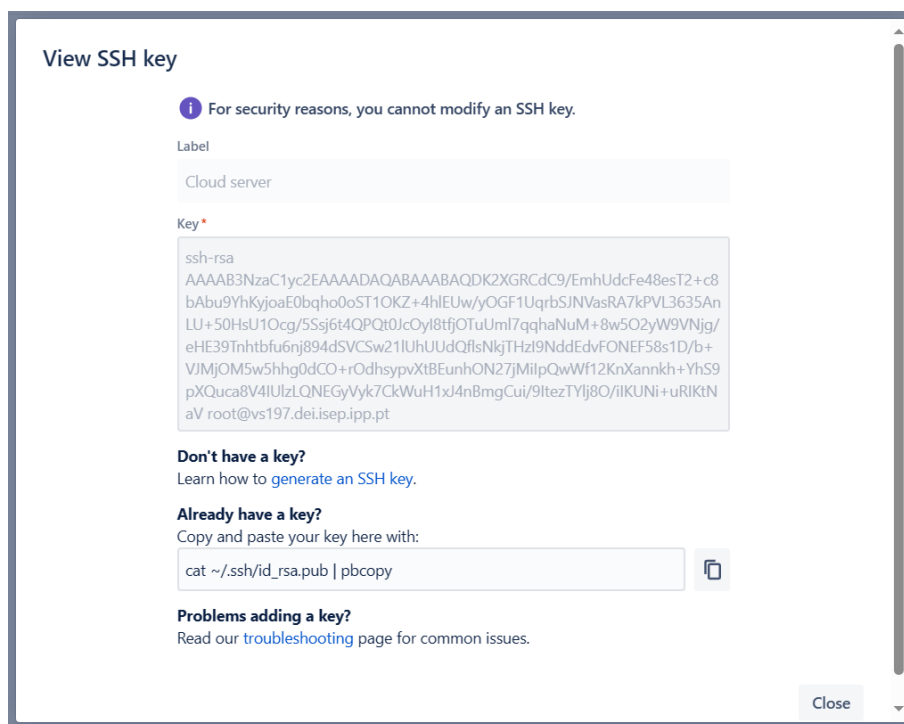


Figura 2 - Configuração do acesso SSH do servidor ao repositório no Bitbucket

Continuando a falar sobre a *script*, depois é feito o comando “*npm install*” para que os módulos em que o projeto depende sejam atualizados/importados. De seguida o projeto é compilado, e em caso de sucesso, são executados os testes, se não, é impressa uma mensagem a comunicar que a execução falhou nesta fase e ela termina aqui. Depois dos testes serem executados, se todos passarem com sucesso a aplicação é reiniciada ou iniciada, dependendo do seu estado atual. Se os testes falharem, é dada uma mensagem a avisar o sucedido e nada mais acontece. Desta forma temos a aplicação sempre funcional, e caso alguma coisa não esteja a funcionar como esperado, a versão anterior é mantida em execução. Para gerir a aplicação que se encontra em produção, é utilizado o gestor de processos *Process Manager 2* (pm2). O nome dado à aplicação em execução é *RobDroneGo*.

Após a escrita da *script*, testou-se se a sintaxe estava correta, em que para isso utilizou-se o comando “*bash -n deployApi.sh*”. Depois também se executou mesmo a *script* por forma a validar o seu *output* e se estava tudo a correr como o esperado.

Depois disto, ainda nos restou um problema a resolver, que era o *deploy* sistemático e agendado do módulo em questão. Para isso, verificou-se se havia atualizações a serem feitas ao sistema com o comando “*sudo apt-get update & sudo apt-get -y upgrade*”, e de seguida instalou-se o *crontab* com o comando “*sudo apt-get install cron*”. Após a instalação do *crontab*, editou-se o ficheiro na localização “*/etc/crontab*”. Aqui adicionou-se a informação necessária para que o *deploy* seja sistemático e de forma agendada. A configuração feita no ficheiro pode ser visualizada na seguinte imagem.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
30 2 * * tue,wed,thu,fri,sat root cd /root/apps/RobDroneGo-MD && ./buildMDTask.sh > logs/buildMDTaskLog.txt
#
```

Figura 3 - Sistematização do *deploy* do módulo *MasterData* através da utilização do *crontab*

Na imagem representada acima, a configuração inserida para a sistematização do *deploy* do MD é a última linha, logo antes do #.

Como é possível verifica, o *deploy* foi agendado para acontecer às 2h e 30min da manhã, de terça-feira a sábado. Foi escolhida esta hora, devido a que é uma boa altura para se reiniciar a aplicação, pois em princípio terá menos utilizadores ou até nenhuns (neste caso), a tentar utilizar a aplicação no *Frontend* e a requisitar dados através de pedidos ao *Backend*. Desta forma, evitamos que alguém fique “pendurado” sem poder trabalhar. A justificação para que apenas seja feito o *deploy* de terça-feira a sábado, é que ele apenas é necessário quando existe a probabilidade de alguém fazer alterações na aplicação. Visto que a semana de trabalho normalmente é de segunda-feira a sexta-feira, podemos então concluir que apenas nestes dias irão existir alterações. Desta forma, a seguir a todos os dias de trabalho, o *deploy* é feito de

forma sistemática e automática caso não existam erros acrescentados à aplicação, sendo isto validado pelos testes.

A configuração do *cron* acrescentada, utiliza a conta *root* para aceder à localização onde o projeto se encontra. De seguida executa a *script* criada, em que o seu *output* é direcionado para um ficheiro de *logs* que se encontra na pasta *logs*, que por sua vez está no mesmo local da *script*. Este ficheiro que armazena os *logs* tem o nome de *buildMDTaskLog.txt*, e armazena a informação das execuções realizadas pela *script*. Através dos *logs*, o administrador poderá verificar se tudo está em conformidade logo assim que inicie a sua atividade diária, sem precisar de ser ele a correr tudo manualmente, ou até esperar pela execução da *script*.

Agora é só aguardar pelas 2h e 30m de algum dos dias de terça-feira a sábado, e *voilà*. 😊

```
root@vs197:~/apps/RobDroneGo-MD# cat logs/buildMDTaskLog.txt
***** The following information refers to the run in the date Sat Nov 25 02:30:01 AM WET 2023 *****

----- Pulling code from remote... -----
Already up to date.

----- Checking and installing new packages if they are required -----

up to date, audited 812 packages in 14s

85 packages are looking for funding
  run `npm fund` for details

2 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.

----- Building project... -----

> bulletproof-nodejs@1.0.0 build
> tsc

----- The application has been built successfully! -----

----- Testing the application... -----

> bulletproof-nodejs@1.0.0 test
> mocha -r ts-node/register **/*.test.ts tests/**/*.test.ts
```

Figura 4 - Log do deploy sistemático no dia 25/11/2023 às 2h:30min

User Story 2

Esta US visa restringir o acesso à solução apenas aos clientes da rede interna do DEI. Por solução, entendemos a aplicação desenvolvida em Angular com a qual os clientes irão interagir diretamente, ou seja, o *frontend*.

Desenvolvimento da solução

Para isso, foram feitas as seguintes alterações na VM DEI vs484, onde está implantada a nossa aplicação (web server).

Para que somente os clientes da rede interna do DEI possam aceder à aplicação, temos de limitar o acesso à gama de endereços do DEI. Conectando à VPN do DEI e com o comando `ipconfig`, observamos que a rede interna do DEI pertence a 10.8.0.0/16

```
Windows IP Configuration

Unknown adapter VPN - VPN Client:

    Connection-specific DNS Suffix . : dei.isep.ipp.pt
    IPv6 Address. . . . . : fd1e:2bae:c6fd:1008:6391:9072:cdbb:84b9
    Temporary IPv6 Address. . . . . : fd1e:2bae:c6fd:1008:3cfa:a5d0:a309:46f2
    Link-local IPv6 Address . . . . . : fe80::9135:99d:4d5b:9786%14
    IPv4 Address. . . . . : 10.8.177.127
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : fe80::ceef:48ff:fe98:efa2%14
                                10.8.0.1
```

Figura 5 - `ipconfig`

Para isso, foi desenvolvido o seguinte script:

```
GNU nano 7.2                                firewall.sh
#!/bin/bash

iptables -P INPUT DROP                      # default policy for incoming traffic to drop
iptables -F INPUT                            # flushes all rules in INPUT chain
iptables -A INPUT -s 10.8.0.0/16 -j ACCEPT   # allow INPUT from IP range 10.8.0.0/16
iptables -A INPUT -s 10.9.20.197 -j ACCEPT   # allow INPUT from MD
iptables -A INPUT -p tcp --dport 22 -j ACCEPT # allow ssh connection
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT # allow established connections
```

Figura 6 - `firewall.sh` bash script

Primeiro, bloqueamos todo o tráfego na cadeia INPUT, ou seja, nenhuma máquina consegue se comunicar com nossa solução. Em seguida, adicionamos uma nova regra ('-A') à cadeia *INPUT* que permite o tráfego ('-j ACCEPT') na faixa de IP 10.8.0.0/16 ('-s 10.8.0.0/16'). Posteriormente, incluímos uma regra adicional na cadeia *INPUT* para permitir a comunicação com a aplicação MD, implantada em outra VM do DEI com o IP estático 10.9.20.197. Em seguida, autorizamos conexões SSH, permitindo o tráfego TCP na porta 22 (porta padrão SSH). Por fim, permitimos pacotes associados a conexões já estabelecidas (tráfego TCP) ou relacionadas a conexões já existentes (tráfego não TCP), possibilitando respostas da *default gateway* para garantir o funcionamento do SSH e a resolução de nomes DNS.

Para executar este script é necessário dar permissões de execução ao ficheiro com o comando:

```
chmod +x myfirewall.sh
```

Ao executar este script, as regras serão aplicadas e só será possível comunicar com o *webserver* através da rede interna do DEI e pela aplicação MD.

Limitações e possíveis melhorias

As configurações da iptables não são persistidas entre reinicializações, sendo necessário executar o script cada vez que a máquina é reiniciada.

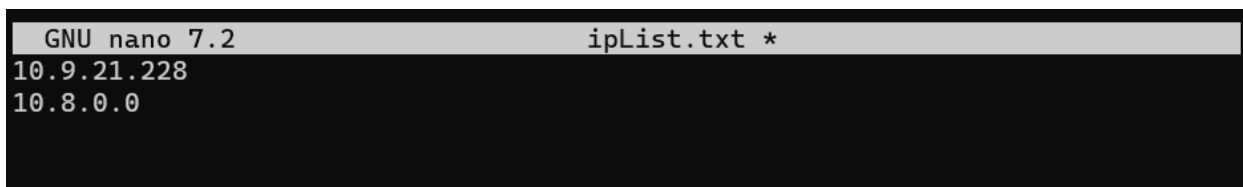
Uma possível melhoria seria a adoção da biblioteca libwrap em vez do iptables. Isso permitiria a configuração de regras de acesso mais granulares e dinâmicas, ou seja, em vez de bloquear o acesso à máquina como um todo, seria possível restringir o acesso apenas ao nosso servidor web, proporcionando maior flexibilidade. Adicionalmente, libwrap permite configuração dinâmica das regras de acesso através de ficheiros (US3), e permite a implementação de *logs*, facilitando a análise e monitorização do tráfego.

User Story 3

Na *User Story 3* pretende-se restringir o acesso à solução apenas aos clientes da rede interna do DEI (cablada ou via VPN) através da simples alteração de um ficheiro de texto.

Para a realização desta *US*, deve-se estar com a sessão iniciada na máquina virtual onde encontra-se a solução e seguidamente realiza-se os seguintes passos:

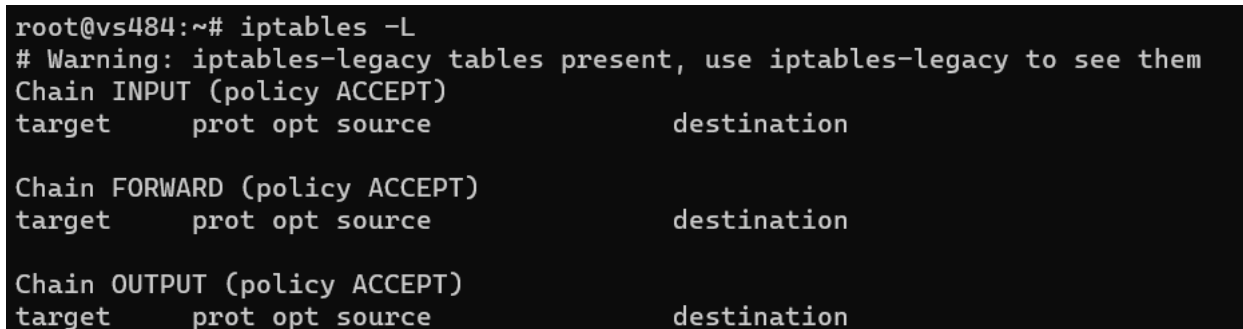
1. Criar um ficheiro através do comando “*nano ipList.txt*” para colocação dos endereços *IPs* que vão ter acesso à solução.
2. Adicionar o endereço *IP* da máquina e o *IP* referente aos clientes da rede interna do DEI.



```
GNU nano 7.2 ipList.txt *
10.9.21.228
10.8.0.0
```

Figura 7 - Criação do ficheiro para colocação dos endereços *IP*

3. Efetuar o *reset* à máquina virtual para não existir qualquer regra no filtro de pacotes *IPv4*.
4. Executar o comando “*iptables -L*” para verificar as regras existentes no filtro de pacotes *IPv4*, é expectável que neste momento não exista nenhuma.



```
root@vs484:~# iptables -L
# Warning: iptables-legacy tables present, use iptables-legacy to see them
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
```

Figura 8 - Listagem das regras do filtro de pacotes *IPv4* em antes da realização da *US*

5. Executar a instrução “*for ip in \$(cat ipList.txt); do iptables -A INPUT -p tcp --dport 22 -s \$ip/16 -j ACCEPT; done*” para permitir o acesso aos clientes indicados no ficheiro de texto *ipList.txt*. Este comando usa um ciclo *for* para percorrer a lista de endereços *IP* que constam no ficheiro *ipList.txt* e para cada endereço *IP*, adiciona uma regra ao *iptables* que permite o tráfego *TCP* na porta 22 (*SSH*) para o intervalo de endereços *IP* especificado.

De seguida, deve-se executar o comando “*sudo /sbin/iptables-save*” para que as alterações ao *iptables* sejam guardadas.

```

root@vs484:~# for ip in $(cat ipList.txt); do iptables -A INPUT -p tcp --dport 22 -s $ip/16 -j ACCEPT; done
root@vs484:~# sudo /sbin/iptables-save
# Generated by iptables-save v1.8.9 (nf_tables) on Thu Nov 23 22:01:04 2023
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 10.9.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
COMMIT
# Completed on Thu Nov 23 22:01:04 2023
# Generated by iptables-save v1.8.9 (nf_tables) on Thu Nov 23 22:01:04 2023
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -m tcp --dport 2222 -j REDIRECT --to-ports 22
COMMIT
# Completed on Thu Nov 23 22:01:04 2023
# Warning: iptables-legacy tables present, use iptables-legacy-save to see them

```

Figura 9 - Permitir acesso aos IPs indicados no ficheiro ipList.txt

6. Executar a instrução “**iptables -A INPUT -p tcp --dport 22 -j DROP**” para negar o acesso a todas as outras ligações. Este comando impede qualquer conexão *TCP* na porta 22 (*SSH*) na entrada da *firewall*.

De seguida, deve-se executar o comando “**sudo /sbin/iptables-save**” para que as alterações ao *iptables* sejam guardadas.

```

root@vs484:~# iptables -A INPUT -p tcp --dport 22 -j DROP
root@vs484:~# sudo /sbin/iptables-save
# Generated by iptables-save v1.8.9 (nf_tables) on Thu Nov 23 22:01:45 2023
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 10.9.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j DROP
COMMIT
# Completed on Thu Nov 23 22:01:45 2023
# Generated by iptables-save v1.8.9 (nf_tables) on Thu Nov 23 22:01:45 2023
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A PREROUTING -p tcp -m tcp --dport 2222 -j REDIRECT --to-ports 22
COMMIT
# Completed on Thu Nov 23 22:01:45 2023
# Warning: iptables-legacy tables present, use iptables-legacy-save to see them

```

Figura 10 - Negar acesso a todas as outras ligações

7. Por fim, executar o comando **“iptables -L”** para verificar as regras existentes no filtro de pacotes IPv4, é expectável que após as configurações anteriores apenas sejam aceites os clientes do *DEI* e todos os outros sejam descartados.

```
root@vs484:~# iptables -L
# Warning: iptables-legacy tables present, use iptables-legacy to see them
Chain INPUT (policy ACCEPT)
target     prot opt source                destination          tcp dpt:ssh
ACCEPT     tcp  --  10.9.0.0/16            anywhere             tcp dpt:ssh
ACCEPT     tcp  --  10.8.0.0/16            anywhere             tcp dpt:ssh
DROP       tcp  --  anywhere               anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

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

Figura 11 - Listagem das regras do filtro de pacotes IPv4 depois da US

User Story 4

A Matriz de Riscos ou Matriz de Probabilidade e Impacto é uma ferramenta de gerenciamento de riscos que permite de forma visual identificar quais são os riscos que devem receber mais atenção.



Figura 12 - Matriz de Risco

Para calcular o risco deve-se multiplicar a probabilidade e a consequência.

Tabela 1 - Ameaça, probabilidade, consequência e risco

Ameaça	Probabilidade	Consequência	Risco
Blackout nos servidores	2	2	Média - 4
Avaria nos servidores	1	4	Média - 4
Perda da base de dados	1	4	Média - 4
Acesso indevido por falta de autenticação	3	4	Alta - 12
Exposição de Dados Sensíveis	1	3	Baixo - 3
Vulnerabilidades de Software	1	4	Médio - 4

Como se pode averiguar na tabela acima a maioria das ameaças são improváveis de acontecer, mas catastróficas pelo que se deve prevenir ao máximo o seu acontecimento.

User Story 5

MBCO (Minimum Business Continuity Objective), é o nível de serviços mínimo e/ou produtos que é aceitável para a organização, para atingir os seus objetivos de negócios durante um desastre. No caso da nossa aplicação, teremos de nos focar no nível mínimo de serviços que necessitaremos para que os robôs e a aplicação funcionem.

A missão da nossa organização é gerir e configurar robôs para que eles possam executar tarefas de forma automática em áreas predefinidas e mapeadas.

Os serviços essenciais da nossa aplicação são: a criação das melhores rotas de navegação entre edifícios e dentro do mesmo edifício, criação de robôs, ligar e desligar robôs, criação de edifícios, criação de andares, criação de passagens, carregar o mapa e a disponibilidade física dos robôs.

Existindo algum problema com a base de dados do programa (ex. hack, avaria, sobreaquecimento), com a disponibilidade física dos robôs (ex. destruição, roubo, defeito) ou com a incapacidade do servidor de processar os dados, haverá vários problemas, pois sem os dados nela guardados seria impossível criar as rotas, sem os robôs seria impossível executar as funções dos mesmos e sem o servidor seria impossível calcular rotas e gerir os robôs. Mal o problema seja detetado uma equipa de resposta a incidentes será chamada, esta equipa irá ter como principais funções restaurar e avaliar os danos. Os dados de “backup” da base de dados, robôs sobresselentes e servidores alugados serão uma solução temporária ou permanente ao problema.

A equipa de resposta demorará 3 horas após ser notificada do sucedido até chegar ao local e precisará de 2 horas para diagnosticar o problema. A mesma trabalhará em turnos de 4 horas, sendo a duração total dependente da origem do problema. Para falhas na base de dados estima-se que leve 18 horas tendo em conta a infraestrutura de backup e a infraestrutura de recuperação. Para o caso do mal funcionamento da frota de robôs, estima-se que 12 horas serão o necessário para identificar se é possível reparar e se não, mobilizar e configurar uma nova frota. Para problemas oriundos do servidor responsável por controlar os robôs e pela aplicação de interação com o utilizador espera-se 16 para corrigir qualquer mal funcionamento.

User Story 6

Para esta *User Story* (US), é pedido que “como administrador do sistema quero que seja proposta, justificada e implementada uma estratégia de cópia de segurança que minimize o *RPO* (*Recovery Point Objective*) e o *WRT* (*Work Recovery Time*)”.

Propôs-se então uma estratégia de cópia de segurança para a base de dados, apesar de esta se encontrar implementada em *cloud* no *MongoDB Atlas*. Para os repositórios também não faz sentido implementar uma cópia de segurança, pois eles encontram-se no *Bitbucket*, ou seja, em *cloud*. A estratégia que será agora proposta, visa minimizar o *RPO* e o *WRT* em conformidade com o que possa ser melhor para o sistema em questão.

Analisando o problema, é possível chegar a um consenso com a User Story 1, visto que a aplicação do *Backend* entra numa breve manutenção a partir das 2h e 30min da manhã. Desta forma podemos realizar o *backup* da base de dados a essa hora, fazendo quase que um 2 em 1. Contudo, se tivéssemos mais especificações acerca do sistema, poderíamos encontrar aqui um melhor equilíbrio e realizar duas cópias de segurança parciais por dia, caso houvesse um período em que a aplicação web não fosse utilizada. Apesar de estarmos a chegar a um acordo de tempo com a User Story 1, temos de diferenciar os dias em que o *backup* ocorre, pois é necessário garantir que todos os dias tenham um *backup*, independentemente de ser fim de semana ou não. Isto porque alguém poderá utilizar o sistema nesse período por alguma razão. Desta forma, iremos ter um *Recovery Point Objective* (*RPO*) de 24h, que caso houvesse mais especificações poderiam passar a ser de menos tempo, e apenas se houvesse essa necessidade, claro. Mas neste contexto que temos, 24h será o suficiente.

Já para o *Work Recovery Time* (*WRT*) iremos querer que ele não seja demasiado longo, e para isso iremos ter uma cópia de segurança incremental durante a semana, e depois ao fim de semana, mais especificamente ao domingo, teremos uma cópia de segurança completa. Isto leva a que caso uma falha aconteça por exemplo a uma segunda-feira, o *WRT* irá ser bastante baixo porque apenas teremos de fazer o *restore* a um *backup*, que é o completo. No entanto, se o sistema falhar a um sábado (depois das 2h e 30min), o *WRT* será maior porque irá fazer 1 cópia completa e 6 parciais. Contudo, ao fazermos uma cópia de segurança completa de 7 em 7 dias, limitamos o *WRT* no pior caso a 1 cópia de segurança completa e 6 parciais, e no melhor caso a apenas uma cópia de segurança.

Para fazer esta implementação, começou-se então por instalar as *tools* do *MongoDB* na máquina virtual, que não será aqui descrito o processo pois não é diretamente importante para a documentação da US.

Criou-se então uma *script* para conseguirmos fazer as cópias de segurança desejadas. Para o fazer, foi utilizado o comando “*nano /etc/mongodb-backup.sh*”.

```
#!/bin/bash
db_backup_dir=/mongo_db_backup
mongodump_dir=/root/mongodb
mkdir -p $db_backup_dir/old
day_of_week=$(date +%w)
if [ "$day_of_week" == "0" ]; then # Se hoje for domingo, mover os ficheiros de local
    echo 'omg'
    mv -f $db_backup_dir/mongodb_backup* $db_backup_dir/old
fi
mkdir -p $mongodump_dir
mongodump --out $mongodump_dir --uri mongodb+srv://admin.q3BiY7FsD0c6KZNE@robdronego.2glxkqb.mongodb.net/RobDroneGo?retryWrites=true&w=majority
chmod -R 700 $mongodump_dir
tar -c -z -p -f $db_backup_dir/mongodb_backup.$day_of_week.tgz -g $db_backup_dir/mongodb_backup.snap /root/mongodb/RobDroneGo >> $db_backup_dir/mongodb_backup.log 2>&1
chmod -R 700 $db_backup_dir
```

Figura 13 - Script para realizar cópias de segurança da base de dados em MongoDB

Na *script* desenvolvida, primeiro começasse por criar o caminho de pastas “/mongo_db_backup/old”, caso ainda não exista. De seguida, verificasse se o dia atual da semana é domingo, e se for, movemos o conteúdo da pasta que contém todos os ficheiros de *backup* da semana, para a localização “/mongo_db_backup/old”. Desta forma, o ficheiro “.snap” já não se irá encontrar naquele local com informação do que é novo ou está diferente nos ficheiros, e será então feita uma cópia de segurança total e não apenas parcial.

Após isto, criasse o caminho de pastas “/root/mongodb” caso ainda não exista, onde será guardado os ficheiros com os dados sobre a base de dados. De seguida utilizasse o comando “*mongodump*” para que os dados existentes na base de dados sejam exportados para o nosso sistema, em que no caso dizemos com a opção “--out” que os ficheiros irão para o caminho “/root/mongodb”. De seguida, mudamos as permissões do caminho de pastas criado para que apenas o *owner* (*root*) tenha todas as permissões existentes.

Agora utilizasse o comando “*tar*” para realizar as cópias de segurança, comprimindo também os ficheiros extraídos da base de dados. Aqui são utilizados alguns valores:

“-c” – Cria um ficheiro novo;

“-z” – É para utilizar a compressão *gzip*;

“-p” – Preserva as permissões dos ficheiros e pastas na pasta comprimida.

“-f” – Especifica o nome da pasta comprimida a ser criada, indicando também em que caminho irá ser colocada.

“-g” – Utiliza um ficheiro para manter o registo incremental de quais ficheiros foram criados ou modificados desde o último backup (*snapshot*).

De seguida o caminho `"/root/mongodb/RobDroneGo"` indica o caminho do diretório que está a ser comprimido, fazendo a cópia de segurança. E por fim a saída padrão (`stdout`) e a saída de erros (`stderr`) é escrita num ficheiro de *logs* com o nome `"mongodb_backup.log"`, que se encontra no mesmo caminho do backup realizado.

Por fim, mudamos as permissões ao caminho de pastas `"/mongo_db_backup"` para garantir que apenas o *owner* (*root*) possa aceder, ler e manipular os ficheiros e pastas lá presentes.

Depois de criar a script, mudou-se as permissões para que apenas o *owner* (*root*) possa ler, escrever e executar a script. Para isso utilizou-se o comando `"chmod 700 /etc/mongodb-backup.sh"` na linha de comandos.

Testou-se também a script para verificar se toda a sintaxe estava correta, utilizando comando `"bash -n /etc/mongodb-backup.sh"`. Após isso, também se executou a script para verificar se tudo ocorria como esperado.

Após a criação da script, ainda nos faltava fazer com que a cópia de segurança fosse realizada de forma sistemática como foi dito mais acima. Para isso, editou-se o ficheiro `"/etc/crontab"`, e adicionou-se a última linha logo acima do `#` que podemos verificar na seguinte imagem.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
30 2 * * tue,wed,thu,fri,sat root cd /root/apps/RobDroneGo-MD && ./buildMDTask.sh > logs/buildMDTaskLog.txt
30 2 * * * root /etc/mongodb-backup.sh
#
```

Figura 14 - Sistematização das cópias de segurança diárias através da configuração do *crontab*

Como se pode visualizar na imagem acima, a script que realiza as cópias de segurança, será executada diariamente às 2h e 30min da madrugada de todos os dias.

User Story 7

Na *User Story 7* pretende-se definir uma pasta pública para todos os utilizadores registados no sistema. Após questionar o cliente foi recomendado que as permissões sejam apenas de leitura (excluindo os administradores) para poder no futuro incluir documentos (instruções, entre outros) que se tornem necessários.

Para a realização desta US, deve-se estar com a sessão iniciada como **root** na máquina virtual, seguidamente realiza-se os seguintes passos:

1. Criar uma pasta através do comando “**mkdir /shared_documents**”.

```
root@uvm050:/# mkdir /shared_documents
```

Figura 15 - Criação da pasta partilhada

2. Atribuir permissões de leitura na pasta criada para todos os utilizadores registados no sistema com o comando “**chmod -R a+r /shared_documents**”.

Neste comando o **-R** significa que esta alteração será recursivamente aplicada a todos os arquivos e subdiretórios dentro deste diretório. O **a** representa todos e o **+r** a adição de permissões de leitura, ou seja, é adicionado permissões de leitura para todos os utilizadores.

```
root@uvm050:~# chmod -R a+r /shared_documents/
```

Figura 16 - Atribuição de permissões de leitura na pasta partilhada

3. Criar um ficheiro para testar as permissões utilizando o comando “**nano /shared_documents/manual.txt**”.

```
root@uvm050:~# nano /shared_documents/manual.txt
```

Figura 17 - Criação de um ficheiro na pasta partilhada

4. Colocar algum texto no ficheiro e guardar.

```
GNU nano 5.4 /shared_documents/manual.txt *  
Manual para mostrar como utilizar a aplicação.
```

Figura 18 - Colocação de texto no ficheiro criado

Podemos verificar as permissões que se encontram neste ficheiro com o comando “**ls -l /shared_documents**”. O ficheiro `manual.txt` tem como permissões “**-rw-r--r--**”, que significa que o proprietário tem permissões de escrita e leitura e o grupo e outros utilizadores permissões de leitura.

```
root@uvm050:~# ls -l /shared_documents/
total 4
-rw-r--r-- 1 root root 49 Nov  8 14:04 manual.txt
```

Figura 19 - Verificação das permissões dos ficheiros da pasta partilhada

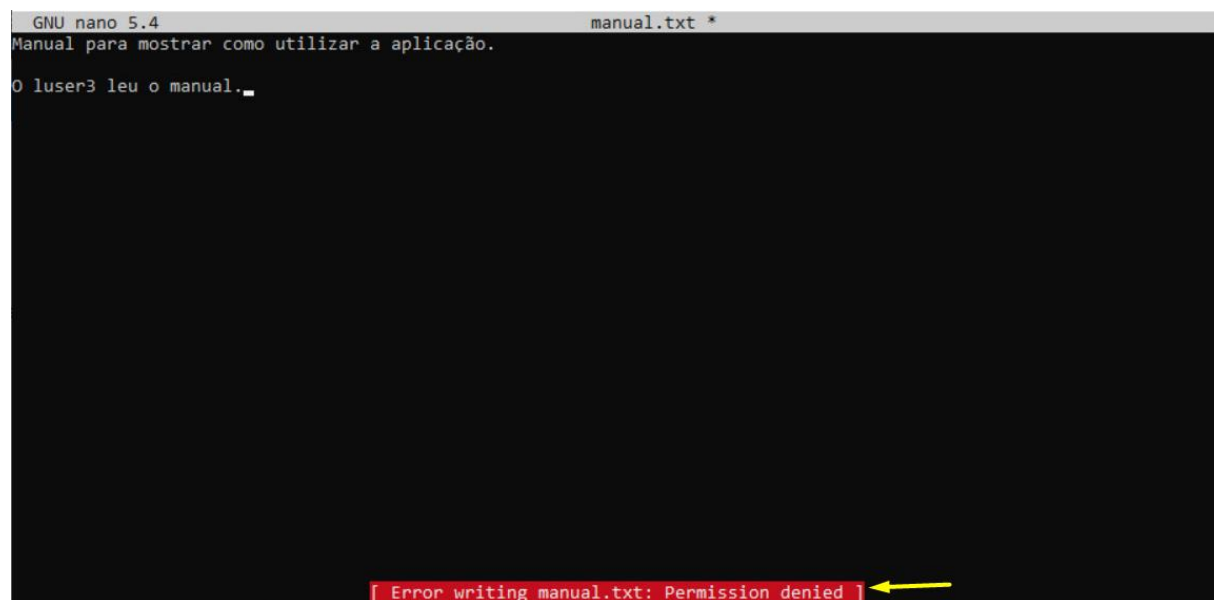
Por fim, vamos realizar o acesso **ssh** com um utilizador para testar o acesso à pasta e também as respetivas permissões.

Neste caso utilizou-se o utilizador **luser3** e entrou-se na pasta partilhada com o comando “**cd /shared_documents**”, de seguida realizou-se uma leitura ao conteúdo do ficheiro **manual.txt** com o comando “**cat manual.txt**”, que foi realizado com sucesso.

```
luser3@uvm050:~$ cd /shared_documents/
luser3@uvm050:/shared_documents$ ls -l
total 4
-rw-r--r-- 1 root root 49 Nov  8 14:04 manual.txt
luser3@uvm050:/shared_documents$ cat manual.txt
Manual para mostrar como utilizar a aplicação.
```

Figura 20 - Teste de acesso à pasta partilhada

No 2º teste realizado, tenta-se editar o conteúdo do ficheiro **manual.txt**. Como apenas possui-se permissões de leitura é expectável o insucesso da operação como verifica-se na seguinte figura.



```
GNU nano 5.4 manual.txt *
Manual para mostrar como utilizar a aplicação.
O luser3 leu o manual.
[ Error writing manual.txt: Permission denied ]
```

Figura 21 - Teste de edição do ficheiro manual.txt

No último teste, vamos criar um ficheiro na pasta partilhada, e tal como o último teste é expectável o insucesso da operação devido à inexistência de permissões de escrita na pasta partilhada.

```
luser3@uvm050:/shared_documents$ "Eu sou o luser3" > me.txt  
-bash: me.txt: Permission denied
```

Figura 22 - Teste de criação de ficheiro na pasta partilhada

User Story 8

Esta US tem como objetivo obter os utilizadores com mais de 3 acessos incorretos. Para tal, foi feito o seguinte **script** em **bash**.

```
#!/bin/bash

limit=$1

output_file="more_than_${1}_failed_logins_$(date +%Y-%m-%dT%H:%M:%S.%3NZ').txt"

awk -F ' ' ' /Failed password/ && !/invalid/ {
    users[$9]++;
}
END {
    for (user in users) {
        if (users[user] > limite) {
            print user, "had", users[user], "wrong logins" >> "$output_file";
        }
    }
}' /var/log/auth.log

echo "Saved in $output_file"
```

Figura 23 Bash Script que encontra os utilizadores com mais do que N acessos incorretos

O **limite** é definido pelo **primeiro parâmetro**, assim o utilizador pode escolher qual o limite de acessos incorretos.

Este script procura o ficheiro **/var/log/auth.log**, que contém o histórico de logins, por linhas que contenham “**Failed Password**”, e, usando uma **map**, incrementa por 1 o número de logins errados do utilizador. Se um utilizador exceder o limite de logins errados, será guardado num ficheiro com o formato ‘**more_than_N_failed_logins_YYYY-MM-DDTHH:mm:ss.SSSZ.txt**’.

```
root@uvm050:~# ./find_users_with_excessive_failed_logins.sh 3
Saved in more_than_3_failed_logins_2023-11-10T09:57:53.309Z.txt
root@uvm050:~# cat more_than_3_failed_logins_2023-11-10T09\:57\:53.309Z.txt
luser3 had 4 wrong logins
luser1 had 6 wrong logins
```

Figura 24 Exemplo de execução do script `find_users_with_excessive_failed_logins.sh`