

Uma imagem com alimentação

Descrição gerada automaticamente

**Relatório do Sprint 2**

**Turma 2DH \_ Grupo 43**

1190929 \_ Patrícia Barbosa

1190947 \_ Pedro Fraga

1190956 \_ Pedro Garcia

1190963 \_ Pedro Preto

**Professor:**

André Moreira, ASC

**Unidade Curricular:**

Administração de Sistemas

**Data: 09/01/2021**

Índice

[INTRODUÇÃO 3](#_Toc92751153)

[USER STORY 1 4](#_Toc92751154)

[ISCSI TARGET LINUX / ISCSI INITIATOR WINDOWS 4](#_Toc92751155)

[ISCSI TARGET WINDOWS / ISCSI INITIATOR LINUX 6](#_Toc92751156)

[USER STORY 2 10](#_Toc92751157)

[INITIATOR WINDOWS 10](#_Toc92751158)

[INITIATOR LINUX 11](#_Toc92751159)

[USER STORY 3 12](#_Toc92751160)

[LINUX 12](#_Toc92751161)

[USER STORY 4 16](#_Toc92751162)

[WINDOWS 16](#_Toc92751163)

[WEBGRAFIA 20](#_Toc92751164)

# INTRODUÇÃO

O presente relatório funcionará como apoio e explicação de execução das User Stories pedidas para o SPRINT 2 de ASIST.

# USER STORY 1

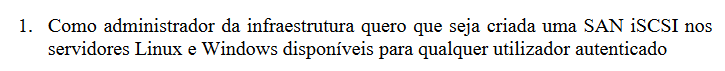


Figura : User story 1

Para a realização de ambas as *stories*, procedemos à criação de uma SAN (rede independente que permite trocar informação entre vários servidores), tanto em Linux, como em Windows.

## ISCSI TARGET LINUX / ISCSI INITIATOR WINDOWS

Para criar o ISCSI target em Linux, recorremos ao serviço tgt, instalado através do comando:

* **# apt install tgt**

O serviço *tgt* permite-nos criar um ISCSI *target* na máquina Linux, que servirá como armazenamento remoto, onde posteriormente, os ISCSI *initiators* se irão conectar. Para a configuração do mesmo, é necessário recorrer a um disco lógico. Para tal, optamos por criar a imagem de um disco, com tamanho 2G, e de um diretório para o guardar, através dos comandos:

* **#** [**mkdir**](https://www.server-world.info/en/command/html/mkdir.html) **/var/lib/iscsi\_disks**
* **# dd if=/dev/zero of=/var/lib/iscsi\_disks/disk01.img count=0 bs=1 seek=2G**

De seguida, criámos um ficheiro de configuração em /etc/tgt/conf.d, com o nome target01.conf (o *target* de Windows terá o nome target02), através do comando:

* **# nano /etc/tgt/conf.d/target01.conf**

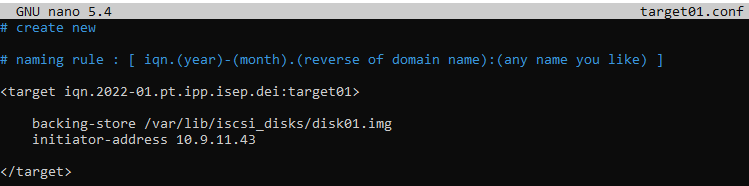


Figura : Configuração do target em /etc/tgt/conf.d/target01.conf

Começamos por definir o nome do target (IQN), que segue o seguinte formato:

* **iqn.(ano)-(mês).(inverso do domínio):target01**

De seguida, definimos qual o disco a usar, no caso a imagem do disco que criámos acima, fornecendo o caminho até ao mesmo. Por fim, escolhemos restringir o acesso e especificar que a máquina que vai servir de *initiator* será o servidor Windows (10.9.11.43).

Para finalizar, reiniciámos o serviço *tgt*, e verificámos que o *target* está operacional, através de:

* **# systemctl restart tgt**
* # **tgtadm --mode target --op show**

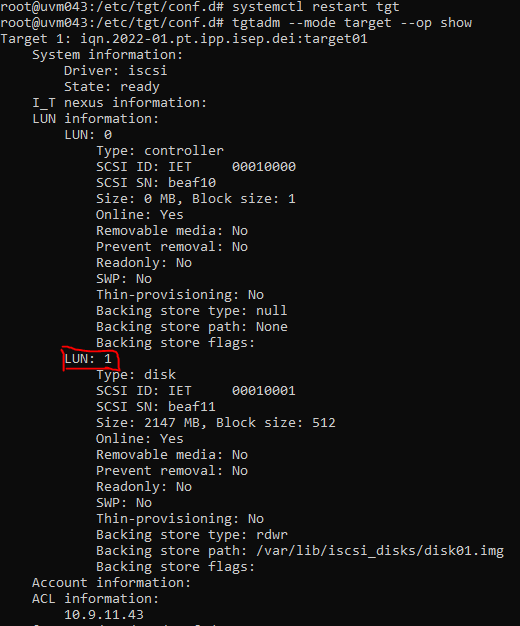


Figura : Status dos ISCSI targets

Desta forma, temos o ISCSI *target* em Linux configurado. Procedemos de seguida à configuração do *initiator* em Windows, e consequente ligação ao *target* acima criado.

Na barra de pesquisa do servidor, pesquisamos “iniciador ISCSI” e abrimos a aplicação.

Já com a aplicação aberta, escolhemos a opção “Ligação rápida” com o ip do servidor Linux. Assim, ligação será automaticamente feita.

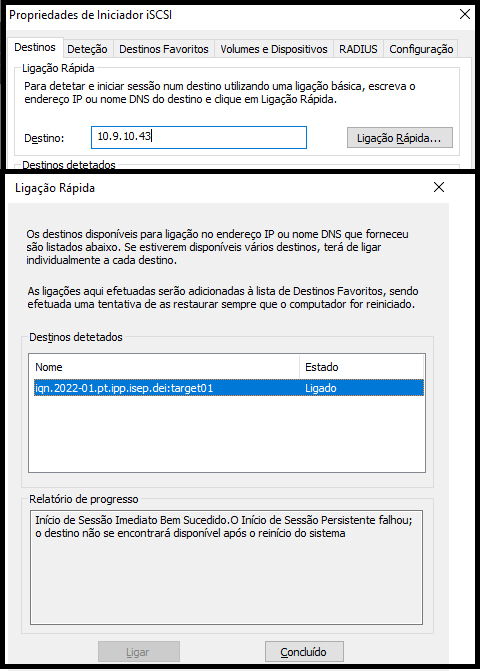


Figura 4: Ligação ao Target Linux através do Iniciador ISCSI Windows

## ISCSI TARGET WINDOWS / ISCSI INITIATOR LINUX

Para proceder à configuração do ISCSI *target* em Windows, começámos por instalar as funcionalidades necessárias:

Selecionámos:

**Gerir -> Adicionar Funções e Funcionalidades -> Instalação baseada em funções e funcionalidades**

**->Selecionar um servidor do agrupamento de servidores (o servidor Windows)**

Nas funções de servidor, escolhemos o “**Servidor de Destino ISCSI”** e “**Servidor de ficheiros”.** Nas “Funcionalidades”, não necessitámos de adicionar nenhuma, pelo que ignorámos esse passo. Para finalizar, efetuamos a instalação destes serviços.

Uma imagem com texto

Descrição gerada automaticamente

Figura 5: Instalação do Servidor de Destino ISCSI e Servidor de Ficheiros (ISCSI)

Com as funcionalidades instaladas, procedemos à criação de um disco virtual, para podermos, posteriormente, criar a SAN ISCSI.

Começamos por selecionar:

**Serviços de Ficheiros e Armazenamento -> iSCSI -> Para criar um disco virtual iSCSI inicie o Assistente de Novo Disco Virtual iSCSI**

O wizard para criação do novo disco virtual e do *target* ISCSI será iniciado. De seguida, escolhemos o servidor de localização e o nome do disco virtual (neste caso, “ISCSI”).

Uma imagem com texto

Descrição gerada automaticamente

Figura 6: Criação do disco virtual e Target 1/3

De seguida, escolhemos o tamanho que queremos no disco criado e o tipo de alocação. Escolhemos **expansão dinâmica** para o disco ir aumentando consoante a quantidade de informação presente.

Uma imagem com texto

Descrição gerada automaticamente

Figura 6: Criação do disco virtual e Target 2/3

Posteriormente, escolhemos o destino (*target*) ISCSI. Como não tínhamos nenhum criado, escolhemos a opção “**Novo destino iSCSI**”. O nome escolhido para o destino foi “target02”.

Uma imagem com texto

Descrição gerada automaticamente

Figura 7: Criação do disco virtual e Target 3/3

Por último, definimos quais os servidores que podem ter acesso (*initiators*), que no caso, queremos que seja o servidor Linux, pelo que definimos o endereço IP da máquina Linux (10.9.10.43).

Para nos ligarmos ao *target* Windows, precisamos de configurar o ISCSI *initiator* em Linux, e para tal, é necessário o pacote **open-iscsi**. Logo, efetuamos a sua instalação, através de:

* **apt install open-iscsi**

Com o serviço instalado, o primeiro passo é encontrar o *target*, através de:



Figura 8: Descobrir iqn do target

Tendo funcionado, descobrimos o IQN do *target*, e foi criada uma pasta com um ficheiro de configuração do mesmo, automaticamente. Este encontra-se em **/etc/iscsi/nodes/iqn.1991-05.com.microsoft\:wvm043-target02-target/10.9.11.43\,3260\,1/default**

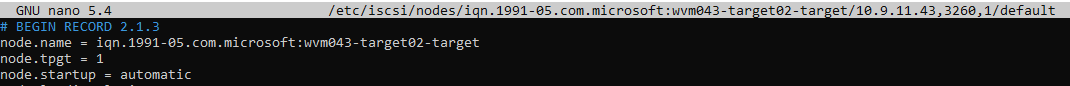


Figura 9: Alteração no ficheiro de configuração do initiator

Através do editor **nano**, abrimos o ficheiro e alteramos a configuração do **“node.startup”**, de **“manual”** para “***automatic”***. Esta alteração faz com que, ao reiniciar qualquer um dos servidores, a ligação será feita automaticamente. Para completar a ligação, reiniciamos o serviço **“open-iscsi”**. Para confirmar que esta foi efetuada, utilizamos o comando: **iscsiadm -m session**



# USER STORY 2

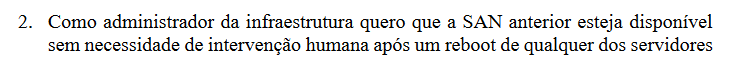


Figura 10: User story 2

## INITIATOR WINDOWS

Para realizar esta *story* no iniciador do Windows, reabrimos a aplicação “Iniciador ISCSI”.

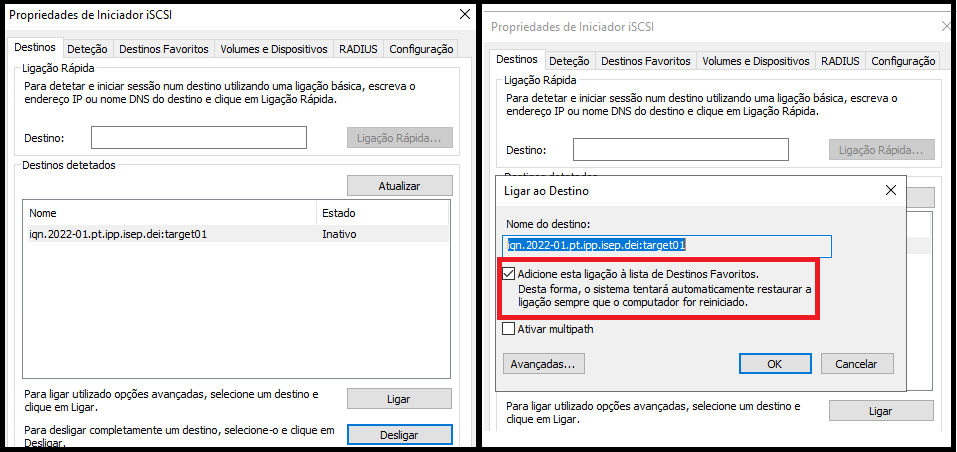


Figura 11: Ópção “ticked” para ativar a ligação automática ao Target Linux

Começamos por desligar a ligação ao target, caso ainda esteja ativa e, de seguida, clicamos na opção “Ligar”. Aparece uma janela com a seguinte opção, que vamos ativar:

* **Adicione esta ligação à lista de Destinos Favoritos**.

**Desta forma, o sistema tentará automaticamente restaurara a ligação sempre que o computador for reiniciado**.

Por fim, clicamos na opção “Ok”. Desta forma, se algum dos servidores for reiniciado, o iniciador do Windows irá, automaticamente, tentar restaurar a ligação, como pedido.

## INITIATOR LINUX

Em Linux, esta story já foi resolvida acima, através da alteração da ópção **“node.startup = manual”**  para **“node.startup = automatic”,** no ficheiro de configuração de ligação ao *target* Windows. Desta forma, quando um dos servidores é reiniciado, o *initiator* Linux tenta repor a ligação ao *target*, automaticamente.

# USER STORY 3

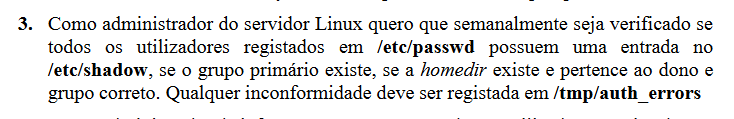


Figura 12: User story 3

De modo a realizar esta user story, foi criado um script, check\_user.sh, na pasta /scripts. De seguida foi utilizado o serviço cron para programar a execução semanal do script.

## LINUX

O script criado em /scripts/check\_user.sh foi dividido em funções, para uma leitura mais fácil e separação das diferentes fases do script, tornando também mais fácil a deteção de erros.

A função que chama todas as outras foi denominada Main, e começa por imprimir vários caracteres “=” e imprimir a data, para separar as diferentes execuções semanais do script. De seguida, cria uma variável, COUNT, que será utilizada na função seguinte. Por fim, chama as restantes funções criadas, dentro de um ciclo *for*, sendo chamadas uma vez por cada utilizador existente no ficheiro /etc/passwd, com o argumento “$USER”, que contém o *username* de um utilizador presente no ficheiro acima referido.

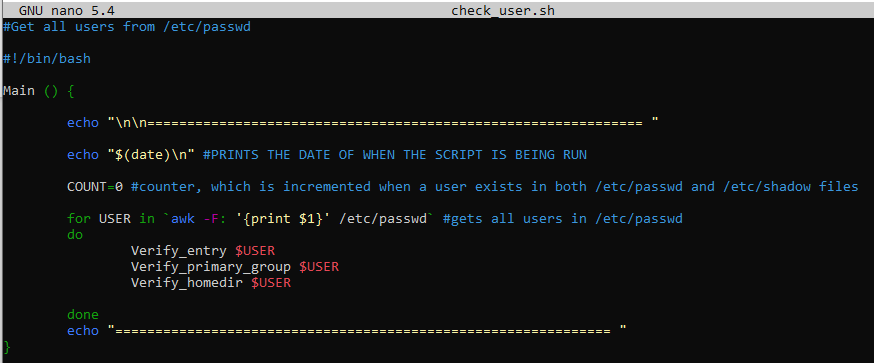


Figura 13: Primeira parte do script /scripts/check\_user.sh

A primeira função chamada pela função Main é a função Verify\_entry. O objetivo desta função é verificar se todos os utilizadores presentes no ficheiro /etc/passwd tem uma entrada no ficheiro /etc/shadow.

Para tal, começa por buscar o username de todos os utilizadores presentes em /etc/shadow, e se encontrar um username igual ao presente no argumento recebido (username presente em /etc/passwd), acrescenta 1 à variável contadora (COUNT).

De seguida, verifica se a variável contadora tem o valor de 0. Se for o caso, significa que não existe uma entrada em /etc/shadow, pelo que imprime uma mensagem de erro; se o valor for diferente de 0, significa que existe uma entrada válida.

Para finalizar, altera o valor presente na variável contadora, de modo a recomeçar a 0 na próxima iteração do ciclo *for* da função acima.

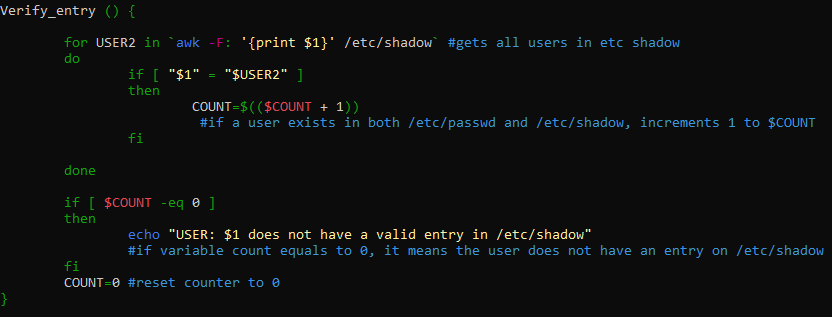


Figura 14: Segunda parte do script /scripts/check\_user.sh

A segunda função chamada pela função Main é a função Verify\_primary\_group. O objetivo desta função é verificar se o grupo primário do utilizador recebido por argumento existe.

Para isso, começa por imprimir o grupo primário do utilizador na variável GROUP.

De seguida, verifica se o grupo presente na variável GROUP existe. Se não existir, imprime uma mensagem sugestiva.

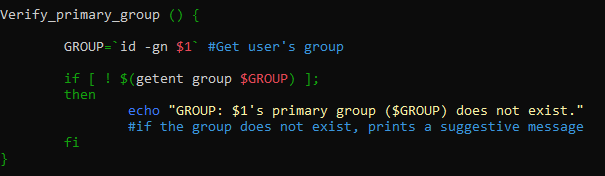


Figura 15: Terceira parte do script /scripts/check\_user.sh

A terceira e última função chamada pela função Main é a função Verify\_homedir. Esta tem como objetivo verificar se a *homedir* do utilizador recebido por parâmetro existe e, adicionalmente, verificar se pertence ao dono e grupo correto.

A função começa por imprimir a *homedir* do utilizador recebido na variável HOMEDIR.

De seguida, verifica se a *homedir* existe. Se não existir imprime uma mensagem sugestiva. Se existir, imprime o dono e o grupo da *homedir* nas variáveis HOMEDIR\_OWNER e HOMEDIR\_GROUP, respetivamente. De modo a não verificar o dono de utilizadores do sistema (que será sempre *root*), foi acrescentada uma condição para apenas verificar utilizadores cujo dono da *homedir* não é “*root*”.

Para verificar se o dono da *homedir* é o correto, verifica se o dono é o utilizador recebido por parâmetro na presente iteração. Se não for, imprime uma mensagem sugestiva.

De forma análoga, para verificar se o grupo da *homedir* é o correto, verifica se o grupo é o grupo primário do utilizador recebido por parâmetro. Se não for, imprime também uma mensagem sugestiva.

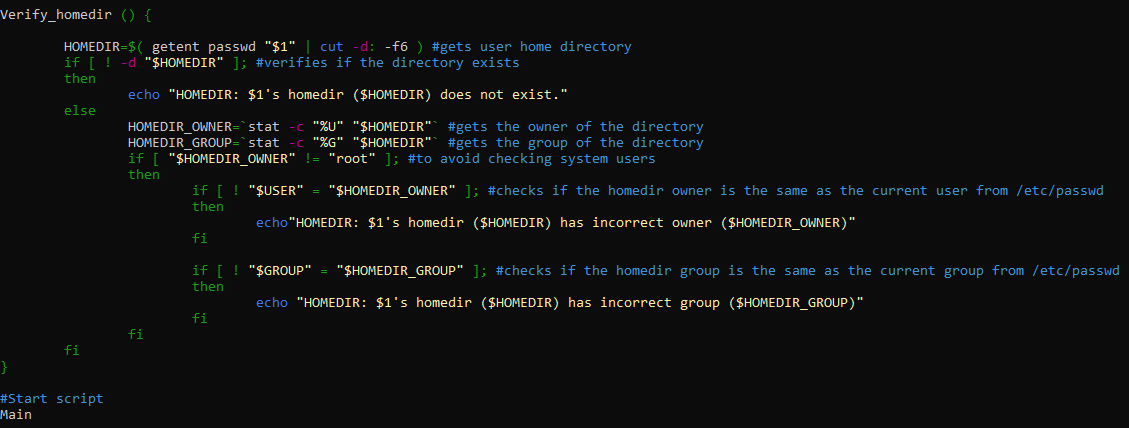


Figura 16: Quarta e última parte do script /scripts/check\_user.sh

Por fim, e já fora da função, a última linha, que contém a palavra “Main”, serve para iniciar o script, chamando a função Main que, posteriormente, chama todas as restantes criadas.



Figura 17: Linha adicionada em crontab -e

Para executar o script semanalmente, como já referido acima, foi utilizado o serviço cron.

Começamos por executar o comando “**crontab -e”**, que abre um ficheiro de texto, no qual podemos modificar os trabalhos *cron* a ser executados. Adicionámos a seguinte linha:

* **59 23 \* \* mon /scripts/check\_user.sh >> /tmp/auth\_errors**

O que esta faz é executar o script criado, imprimindo todas as mensagens sugestivas em /tmp/auth\_errors, como era pedido na user story. O “**59 23 \* \*** **mon**”, significa que vai ser executado no minuto 59, hora 23, em qualquer dia do mês (\*), em qualquer mês (\*), às segundas-feiras.

O resultado semanal da execução deste script é o seguinte:

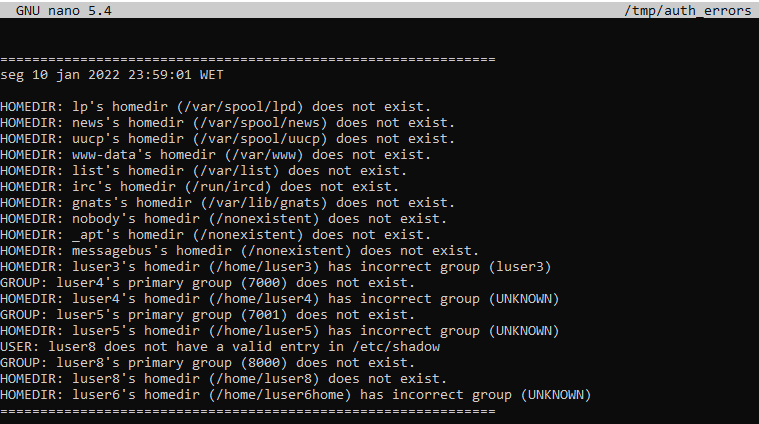


Figura 18: Resultado da execução semanal do script check\_user.sh

# USER STORY 4

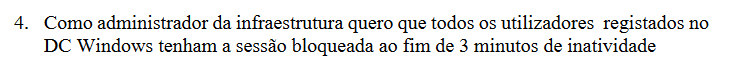


Figura 19: User story 4

## WINDOWS

Para executar o bloqueio de sessão, recorremos à criação de um novo GPO (Group Policy Object). Para tal, abrimos a aplicação “Gestor de Políticas e Grupos”, e acedemos a **“Floresta: wvdom043.dei.isep.ipp.pt -> Domínios -> wvdom043.dei.isep.ipp.pt -> Objectos de Política de grupo (clique com botão direito) -> Novo -> (inserir nome, no caso, 180 segundos bloquear utilizador) Ok “**

Uma imagem com texto

Descrição gerada automaticamente

Figura 20: Criação do novo GPO (180 segundos bloquear utilizador)

De seguida, premimos com o botão direito por cima do GPO criado, e escolhemos a ópção “Editar…”, arrancando o programa “Editor de Gestão de Políticas de Grupo”.

Queremos agora configurar o GPO para bloquear a sessão do utilizador caso detete inatividade durante 3 minutos. Dentro do editor, acedemos a “**Configuração do Utilizador -> Políticas -> Modelos Administrativos: definições de política (ficheiros ADMX) obtidas a partir do computador local. -> Painel de Controlo -> Personalização”**.

Uma imagem com texto

Descrição gerada automaticamente

Figura 21: Configuração do GPO criado

Aparecem-nos agora várias definições, das quais vamos ativar:

* Ativar proteção de ecrã
* Proteger a proteção de ecrã com palavra-passe
* Tempo limite de proteção de ecrã (e definimos o “time-out” para 180 segundos (3 minutos) de inatividade)
* Forçar proteção de ecrã específica

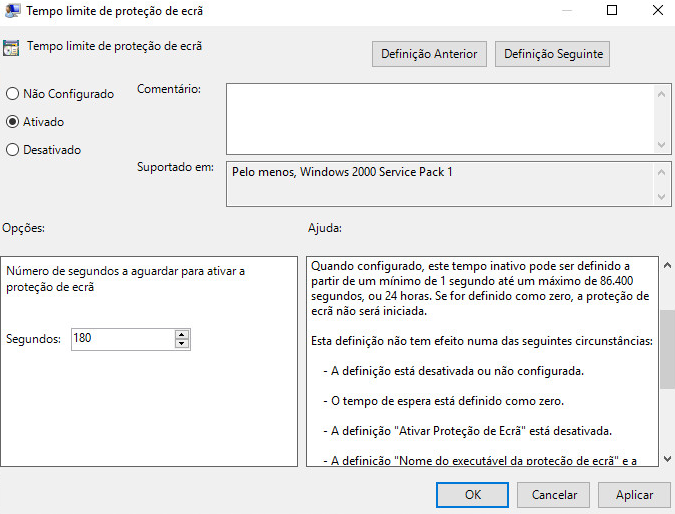


Figura 22: Alteração do tempo limite de proteção de ecrã

Estando todas as alterações aplicadas e guardadas, voltamos ao “Gestor de Políticas de Grupo” e clicamos com o botão do rato em cima de “**wvdom043.dei.isep.ipp.pt**”, e escolhemos a ópção “**Ligar um GPO Existente**”. Agora escolhemos o GPO criado e clicamos em “**Ok**”.

Uma imagem com texto

Descrição gerada automaticamente

Figura 23: Ligação do novo GPO

Para finalizar, atualizamos as políticas do sistema, através do comando “**gpupdate /force**” na “**Windows PowerShell**”.

Uma imagem com texto

Descrição gerada automaticamente

Figura 24: Comando introduzido para atualização das políticas

A partir deste momento, qualquer sessão será bloqueada ao fim de 3 minutos de inatividade.

# WEBGRAFIA

* <https://linuxize.com/post/how-to-list-users-in-linux>
* <https://unix.stackexchange.com/questions/410367/how-to-get-the-primary-group-of-a-user/410370>
* <https://stackoverflow.com/questions/10552711/how-to-make-if-not-true-conditionman>
* <https://techstop.github.io/directory-exists-bash/>
* <https://www.unix.com/man-page/linux/5/crontab/>
* <https://www.cyberciti.biz/faq/how-do-i-add-jobs-to-cron-under-linux-or-unix-oses/>
* <https://www.cyberciti.biz/faq/howto-setup-debian-ubuntu-linux-iscsi-initiator/>
* <https://www.tecmint.com/setup-iscsi-target-and-initiator-on-debian-9/>
* <https://www.server-world.info/en/note?os=Debian_10&p=iscsi&f=2>
* <https://www.sqlshack.com/installing-and-configuring-the-iscsi-target-server-on-windows-server-2016/>
* <https://www.seagate.com/pt/pt/support/kb/business-storage-nas-how-to-connect-an-iscsi-initiator-in-windows-to-the-business-storage-nas-iscsi-target-005528en/>
* <https://techexpert.tips/pt-br/windows-pt-br/diretiva-de-grupo-para-bloquear-o-computador-do-windows-apos-idle/>