

Sessão 4: Controles de segurança

Estando configurado nosso sistema de autenticação centralizado, quais seriam os próximos passos para realizar o *hardening* do ambiente? Nesta sessão, iremos tratar de algumas configurações mais simples, num escopo particular, mas que somadas tornarão o *datacenter* muito mais resiliente contra ataques, além de mais funcional. Iremos verificar se as senhas escolhidas pelos usuários são de fato seguras, implementar *quotas* de disco em um servidor de arquivos Linux, permitir controle mais granular de permissões de arquivos através de ACLs (*Access Control Lists*) e realizar um controle refinado de autorizações administrativas usando o comando `sudo`.

Vamos ao trabalho?

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

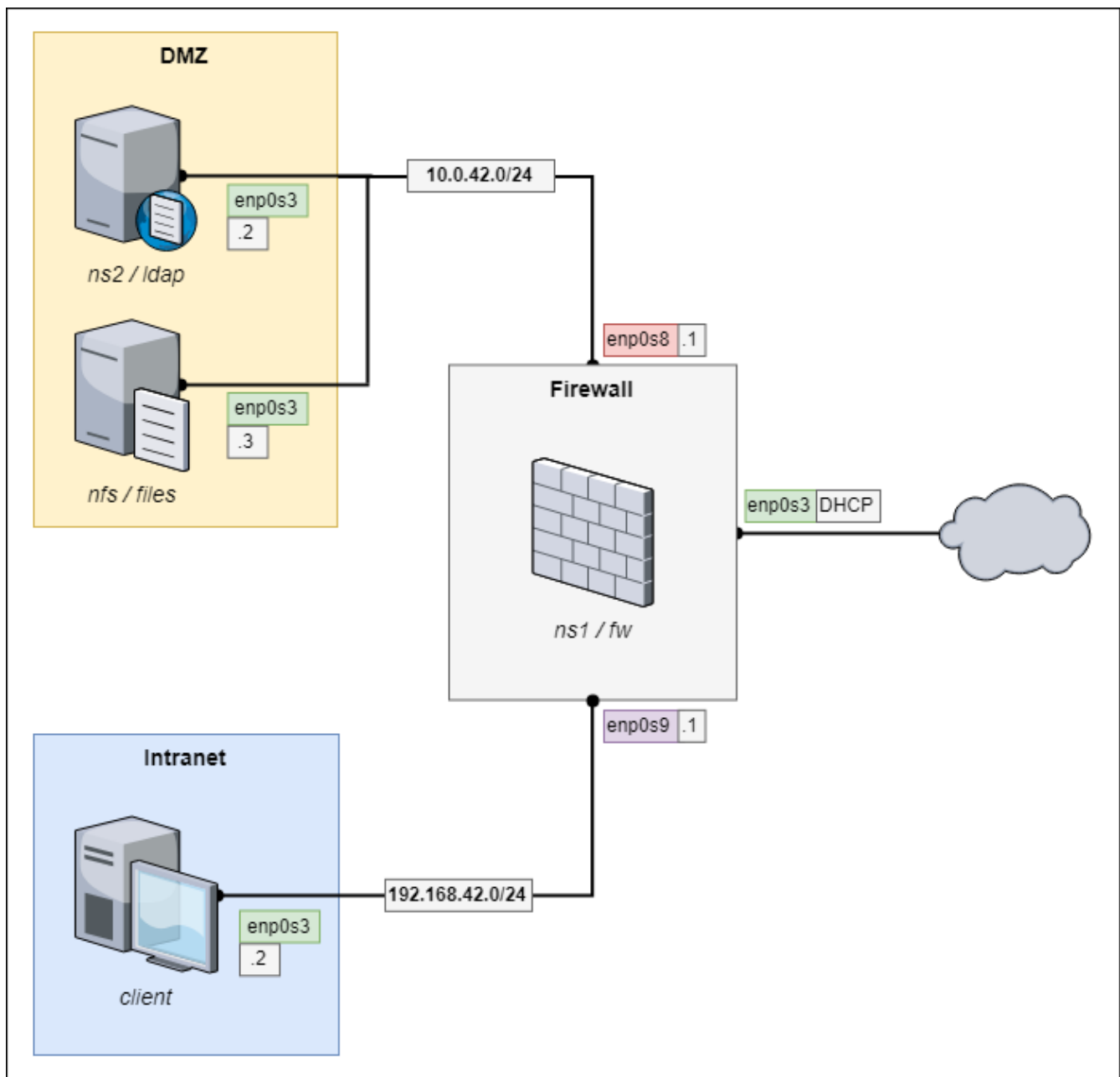


Figura 1. Topologia de rede desta sessão

Teremos agora quatro máquinas:

- **ns1**, com *alias* **fw**, atuando como firewall de rede e DNS primário. Endereços IP via DHCP (interface *bridge*), 10.0.42.1/24 (interface DMZ) e 192.168.42.1/24 (interface Intranet).
- **ns2**, com *alias* **ldap**, localizada na DMZ e atuando como DNS secundário, servidor LDAP de autenticação centralizada e repositório de chaves SSH-CA. Endereço IP 10.0.42.2/24.
- **nfs**, com *alias* **files**, localizada na DMZ e atuando como servidor de arquivos NFS. Endereço IP 10.0.42.3/24.
- **client**, localizada na Intranet e usada como ponto de partida para logins remotos nos diferentes servidores do *datacenter* simulado. Endereço IP 192.168.42.2/24.

1. Na topologia há a previsão de criação de dois novos registros DNS, um mapeamento direto para o nome **nfs** e um *alias* dessa mesma máquina para o nome **files**. Vamos ajustar o DNS: acesse a máquina **ns1** como o usuário **root**:

```
# hostname ; whoami
ns1
root
```

Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo uma entrada A e outra CNAME para a máquina `nfs`, como se segue. **Não se esqueça** de incrementar o valor do **serial** no topo do arquivo!

```
# nano /etc/nsd/zones/intnet.zone
(...)
```

```
# grep nfs /etc/nsd/zones/intnet.zone
nfs      IN      A          10.0.42.3
files    IN      CNAME      nfs
```

Desta vez também será necessário alterar o arquivo de mapeamento reverso `/etc/nsd/zones/10.0.42.zone` com um registro PTR para a nova máquina. Novamente, lembre-se de incrementar o **serial** neste arquivo também.

```
# nano /etc/nsd/zones/10.0.42.zone
(...)
```

```
# grep nfs /etc/nsd/zones/10.0.42.zone
3        IN      PTR          nfs.intnet.
```

Assine o arquivo de zonas usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 0 rrsets, 0 messages and 0 key entries
```

Verifique que as entradas foram criadas com sucesso, usando o comando **dig**:

```
# dig nfs.intnet +short
10.0.42.3
```

```
# dig files.intnet +short  
nfs.intnet.  
10.0.42.3
```

```
# dig -x 10.0.42.3 +short  
nfs.intnet.
```

2) Requisitos de senha na base LDAP

Uma preocupação frequente dos analistas de segurança é quanto às senhas dos usuários: será que elas tem um tamanho apropriado, não utilizam palavras constantes em *wordlists*, contém caracteres especiais? Apesar de termos configurado o acesso aos nossos servidores usando chaves assimétricas via SSH-CA (e, no caso da máquina *ns1*, aplicado restrição de acesso exclusivamente via chaves), não é interessante que nos despreocupemos totalmente da segurança de senhas dos usuários — afinal, os logins na máquina *ns2* ainda podem usar senhas, por exemplo.

Podemos utilizar o *policy overlay* do *slapd* (documentação em <https://www.openldap.org/doc/admin24/overlays.html> ou *man 5 slapo-ppolicy*) para implementar alguns controles no diretório LDAP para exigir aspectos mínimos de qualidade das senhas dos usuários, tais como:

- **pwdInHistory**: Histórico de senhas, mantém uma lista de senhas passadas que impede que o usuário as repita. O número de senhas mantidas em histórico é configurável.
- **pwdMaxAge**: Tempo máximo de validade da senha.
- **pwdMinAge**: Tempo mínimo de validade da senha, para evitar que o usuário circule pelo histórico rapidamente e apague o registro de uma senha que queira repetir.
- **pwdMinLength**: Tamanho mínimo da senha do usuário, em caracteres.
- **pwdMaxFailure**: Número máximo de tentativas de *bind* com senha incorreta antes que a conta do usuário seja travada.
- **pwdCheckQuality**: Define uma função externa para checagem de qualidade da senha do usuário — esta é uma extensão não-padrão da política de senhas do diretório LDAP, e não iremos configurá-la. O website <http://ltb-project.org/wiki/documentation/openldap-ppolicy-check-password> disponibiliza um software customizado que pode ser usado para implementar esse tipo de política.

1. Faça login como *root* na máquina *ns2*:

```
# hostname ; whoami  
ns2  
root
```

Para habilitar esses controles em nossa base LDAP, o primeiro passo é carregar o arquivo LDIF do *schema* com as informações de políticas de senhas:

```
# ldapadd -Y external -H ldapi:/// -f /etc/ldap/schema/ppolicy.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=ppolicy,cn=schema,cn=config"
```

2. Em seguida, iremos adicionar o módulo `/usr/lib/ldap/ppolicy.la` à lista de módulos carregados pelo `slapd` em seu início. Crie o arquivo novo `/root/ldif/olcModuleLoad.ldif` com o seguinte conteúdo:

```
1 dn: cn=module{0},cn=config
2 changetype: modify
3 add: olcModuleLoad
4 olcModuleLoad: ppolicy.la
```

Para aplicar as modificações desse LDIF à base LDAP, basta executar:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcModuleLoad.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=module{0},cn=config"
```

3. A próxima etapa é configurar o *overlay* de políticas de senhas para controlar os atributos `userPassword` de nossa base `cn=intnet`. Crie o arquivo novo `/root/ldif/olcOverlayPpolicy.ldif` com o seguinte conteúdo:

```
1 dn: olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config
2 objectClass: olcOverlayConfig
3 objectClass: olcPPolicyConfig
4 olcOverlay: ppolicy
5 olcPPolicyDefault: cn=passwordDefault,ou=Policies,dc=intnet
6 olcPPolicyHashCleartext: FALSE
7 olcPPolicyUseLockout: FALSE
8 olcPPolicyForwardUpdates: FALSE
```

Note que estamos indicando que o *overlay* `ppolicy` será aplicado sobre a base `{1}mdb`, que é exatamente a base com raiz em `dc=intnet`, como podemos confirmar através do comando:

```
# ldapsearch -Y external -H ldapi:/// -LLL -b 'cn=config'
'(&(objectClass=olcDatabaseConfig)(olcSuffix=dc=intnet))' dn 2> /dev/null
dn: olcDatabase={1}mdb,cn=config
```

Caso estivéssemos fazendo esta configuração em um ambiente que possua várias bases LDAP

carregadas dentro de um mesmo *daemon* **slapd**, seria necessário determinar o número da base MDB e editar o arquivo mostrado anteriormente.

Para aplicar as modificações desse LDIF à base LDAP, execute:

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcOverlayPpolicy.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config"
```

4. Agora, vamos definir a política de senhas da base **dc=intnet**. Crie o arquivo novo **/root/ldif/passwordDefault.ldif** com o seguinte conteúdo:

```
1 dn: ou=Policies,dc=intnet
2 ou: Policies
3 objectClass: organizationalUnit
4
5 dn: cn=passwordDefault,ou=Policies,dc=intnet
6 objectClass: pwdPolicy
7 objectClass: person
8 objectClass: top
9 cn: passwordDefault
10 sn: passwordDefault
11 pwdAttribute: userPassword
12 pwdCheckQuality: 2
13 pwdMinAge: 0
14 pwdMaxAge: 2592000
15 pwdMinLength: 8
16 pwdInHistory: 5
17 pwdMaxFailure: 3
18 pwdFailureCountInterval: 0
19 pwdLockout: TRUE
20 pwdLockoutDuration: 0
21 pwdAllowUserChange: TRUE
22 pwdExpireWarning: 0
23 pwdGraceAuthNLimit: 0
24 pwdMustChange: FALSE
25 pwdSafeModify: FALSE
```

Estamos, em ordem:

- Criando uma entrada **ou=Policies,dc=intnet** para armazenar políticas da base **dc=intnet**.
- Dentro desta OU, criando o CN **cn=passwordDefault,ou=Policies,dc=intnet** que define a política de senhas da base. Configurações mais relevantes:
 - **pwdAttribute** define o atributo que será verificado, que armazena senhas de usuários.
 - **pwdCheckQuality** ativa a checagem de qualidade de senhas; como não estamos

habilitando nenhum módulo externo, apenas a checagem de comprimento será aplicada.

- `pwdMinAge` define o tempo mínimo de validade de senhas; como queremos testar o histórico de senhas, explicado a seguir, não iremos ativar essa opção.
- `pwdMaxAge` define o tempo máximo de validade da senha, em segundos; ajustamos esse valor para 30 dias.
- `pwdMinLength` define o tamanho mínimo de senha, 8 caracteres.
- `pwdInHistory` define que iremos guardar o *hash* das 5 senhas mais recentes de cada usuário, que não poderão repeti-las.
- `pwdMaxFailure` define que usuários que errarem a senha consecutivamente mais de 3 vezes terão suas contas bloqueadas.

Para aplicar o LDIF à base LDAP temos que nos autenticar na raiz `dc=intnet`, como se segue:

```
# ldapadd -D 'cn=admin,dc=intnet' -W -f ~/ldif/passwordDefault.ldif
Enter LDAP Password:
adding new entry "ou=Policies,dc=intnet"

adding new entry "cn=passwordDefault,ou=Policies,dc=intnet"
```

5. Reinicie o `slapd` para aplicar as configurações:

```
# systemctl restart slapd.service
```

6. Vamos testar nossos controles — logue na máquina `client` como o usuário `luke`:

```
$ hostname ; whoami
client
luke
```

Tente alterar a senha do usuário para uma *string* menor que o tamanho exigido, como `mar-te` por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password fails quality checking policy
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

O `slapd` nos informa que a senha não atende os requisitos mínimos de qualidade, nesse caso, o tamanho da senha.

7. Altere a senha para um valor aceitável, como `seg10luke2`, por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

Agora, tente alterar a senha para um valor já usado anteriormente, como `seg10luke`:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password is in history of old passwords
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

Somos informados que a senha consta do histórico de senhas antigas. Como o LDAP implementa isso? Acesse a máquina `ns2` como usuário `root` e pesquise pelo campo `pwdHistory` do usuário `luke`:

```
# hostname ; whoami
ns2
root
```

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' pwdHistory
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
pwdHistory: 20181114030204Z#1.3.6.1.4.1.1466.115.121.1.40#38#{SSHA}bI0fgkHR6MZ
tAavv1bUYx9PuPucZhDyY
```

Ao informarmos uma nova senha, o `slapd` compara o seu hash com um dos *hashes* guardados no histórico do usuário (nesse caso, `luke`); se encontrada, a senha é rejeitada.

8. Vamos testar o *lockout* de contas. Como teremos que fazer logins propositalmente incorretos, ainda como o usuário `root` na máquina `ns2`, pare o serviço `Fail2ban` para evitar que sejamos bloqueados pelo firewall durante o teste:

```
# systemctl stop fail2ban
```

De volta à máquina `client` como `luke`, tente logar via SSH na máquina `ns2` usando senha e erre propositalmente a combinação por 3 vezes consecutivas:


```
$ hostname ; whoami
client
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Permission denied, please try again.
luke@ns2's password:
Permission denied, please try again.
luke@ns2's password:
Permission denied (publickey,password).
```

Agora, tente logar com a senha correta — note que seu acesso será negado:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Permission denied, please try again.
```

De volta à máquina **ns2** como o usuário **root**, vamos verificar o que aconteceu:

```
# hostname ; whoami
ns2
root
```

Execute o comando **ldapsearch** abaixo para listar todos os usuários bloqueados na base **dc=intnet**:

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'pwdAccountLockedTime=*'
pwdAccountLockedTime
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
pwdAccountLockedTime: 20181114030659Z
```

Como esperado, **luke** está bloqueado. Para desbloquear um usuário específico crie um arquivo LDIF novo, **/root/ldif/unlockUser.ldif** com o seguinte conteúdo:

```
1 dn: uid=luke,ou=People,dc=intnet
2 changetype: modify
3 delete: pwdAccountLockedTime
```

Aplique as alterações do LDIF à base com:

```
# ldapmodify -D 'cn=admin,dc=intnet' -W -f ~/ldif/unlockUser.ldif
Enter LDAP Password:
modifying entry "uid=luke,ou=People,dc=intnet"
```

De volta à máquina **client** como **luke**, tente logar novamente com a senha correta:

```
$ hostname ; whoami
client
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@ns2
luke@ns2's password:
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:15:11 2018 from 192.168.42.2
luke@ns2:~$
```

```
$ hostname ; whoami
ns2
luke
```

Perfeito, nossos controles funcionaram como esperado. Na máquina **ns2**, como **root**, não se esqueça de reiniciar o Fail2ban:

```
# hostname ; whoami
ns2
root
```

```
# systemctl start fail2ban
```

3) Busca de senhas fracas

Simplesmente configurar um tamanho mínimo de senha, como fizemos na atividade anterior, não é garantia que os usuários escolherão senhas seguras para suas contas. Por exemplo, um usuário pode definir **12345678** como sua senha — essa *string* está dentro do tamanho mínimo exigido mas não pode, nem de perto, ser considerada uma senha segura. O que fazer?

Podemos submeter os *hashes* de senha dos usuários a testes de segurança, como ataques de força-bruta — em que testamos combinações de caracteres exaustivamente para descobrir a senha — ou de dicionário — em que usamos uma base de senhas previamente preenchida, conhecida como *wordlist*, e verificamos se a senha do usuário se encontra nessa lista. Devido ao fato de as senhas do

LDAP serem armazenadas por padrão em formato SSHA (SHA-1 com *salt*), ataques do tipo *rainbow table*—em que comparamos o *hash* da senha do usuário com uma base de *hashes* previamente computados, buscando por similaridades—não são viáveis. Podemos verificar o *hash* utilizado para armazenar a senha do usuário **luke**, por exemplo, usando o comando:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' userPassword | grep
'^userPassword::' | awk '{print $NF}' | base64 --decode
Enter LDAP Password:
{SSHA}JK1/uM/9bm0WM/IzW1uIBM4b1Q4UEWd8
```

A ferramenta que iremos utilizar para realizar os ataques de dicionário e força-bruta mencionados anteriormente será o **hashcat** (<https://hashcat.net/hashcat/>). Uma das ferramentas mais rápidas para quebra de senhas disponíveis, é um programa *open source* multiplataforma que se utiliza da CPU ou GPUs (placas gráficas) de uma máquina para acelerar o processo de ataque sensivelmente, especialmente quando comparada com ferramentas mais tradicionais como o **john**.

Até a versão v3.00, o **hashcat** era dividido em duas versões, uma voltada para CPUs e outra para GPUs (esta, implementada via OpenCL ou CUDA). Com o lançamento da versão v3.00, as duas versões foram unificadas em uma única ferramenta, requerendo a biblioteca OpenCL (<https://www.khronos.org/opencl/>) como dependência.

É boa prática de segurança que instalemos apenas o estritamente necessário em servidores, a fim de reduzir a superfície de ataque disponível em uma eventual invasão. Por esse motivo, instalaremos o **hashcat** e as demais bibliotecas necessárias na máquina **client**, que é menos crítica que os servidores **ns2** e **ns1**.

1. Acesse a máquina **client** como o usuário **root**, e instale o **hashcat** e suas dependências:

```
# hostname ; whoami
client
root
```

```
# apt-get install --no-install-recommends hashcat libhwloc-dev ocl-icd-dev ocl-icd-
opencl-dev pocl-opencl-icd
```

2. Agora, acesse como o usuário **luke**, em seu diretório *home*.

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

Altere a senha do usuário **luke** para um valor propositalmente inseguro, como **password**:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

O primeiro passo para testarmos a segurança das senhas dos usuários é obter seus *hashes*. Vamos fazer isso, de forma remota, usando um *script* shell mostrado a seguir. Crie o arquivo novo `/home/luke/scripts/gethashes.sh` com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 TMPFILE="$( mktemp )"
4 OUTFILE="${HOME}/hashes.txt"
5
6 rm -f ${OUTFILE}
7 touch ${OUTFILE}
8
9 ldapsearch -x \
10 -LLL \
11 -H ldap://10.0.42.2 \
12 -D 'cn=admin,dc=intnet' \
13 -w 'rnpesr' \
14 -b 'dc=intnet' \
15 'userPassword=*' \
16 cn=userPassword \
17 | grep '^cn:\|^userPassword::' \
18 | awk '{print $NF}' \
19 | sed 'N;s/\n/ /' \
20 | tr ' ' ':' > ${TMPFILE}
21
22 while read l; do
23     luser="$( echo ${l} | cut -d':' -f1 )"
24     lhash="$( echo ${l} | cut -d':' -f2 )"
25
26     echo "${luser}:${( echo ${lhash} | base64 --decode )}" >> ${OUTFILE}
27 done < ${TMPFILE}
28
29 rm -f ${TMPFILE}
```

O que esse *script* faz? Vamos ver:

1. (Linhas 3-4) Criamos um arquivo temporário com o comando `mktemp`, e definimos o arquivo de saída como `~/hashes.txt`.
2. (Linhas 6-7) Se existente, removemos o arquivo de saída e criamos um novo, vazio.
3. (Linhas 9-20) Executamos um comando `ldapsearch` remoto na máquina `ns2`, executando o `bind` como o usuário `cn=admin,dc=intnet` e senha informada diretamente na linha de

comando. Buscamos todos os DNs que possuem o campo `userPassword` não-vazio, e filtramos apenas os campos `cn` e `userPassword` na saída. Finalmente, fazemos uma junção de linhas duas-a-duas usando os comandos `awk`, `sed` e inserimos um separador usando o `tr`. Essa saída é escrita no arquivo temporário criado anteriormente.

4. (Linhas 22-27) Processamos o arquivo temporário linha-a-linha. Em cada linha, extraímos o campo 1 (`cn` do usuário) e o campo 2 (`userPassword`). O campo `userPassword` está codificado em base64, então usamos `base64 --decode` para traduzir esse campo, e escrevemos o *output* em ordem no arquivo de saída.

5. (Linha 29) O arquivo temporário é removido.

3. Vamos testar o funcionamento do *script*:

```
$ bash ~/scripts/gethashes.sh
```

```
$ cat hashes.txt
admin:{SSHA}NzQZTz7uf0xNM3PYy7cp+zV6p7bKFNCy
luke:{SSHA}46Qe8Ny+QQgDsbPcps2MODqUHGtdLX41
sshca:{SSHA}+JTtQ5+XEi+sJ4sPmWK3LZXrIHSpbcbn
han:{SSHA}BE6cC89vaJQtB/g9yEJTt008HCtRabel
```

4. Vamos, primeiramente, executar um ataque de dicionário. Um ataque de dicionário, como mencionado anteriormente, é quando obtermos um arquivo com um conjunto de senhas em texto claro, calculamos seus *hashes* usando os valores de *salt* conhecidos, e comparamos os resultados com os *hashes* dos usuários.

No caso do algoritmo SSHA implementado no OpenLDAP, para extrair o *salt* devemos decodificar o *hash* original em base64 uma vez, remover o prefixo `{SSHA}`, decodificar o *hash* resultante em base64 novamente, e extrair os últimos 4 bytes; esses 4 bytes são o *salt* da senha codificada. Para ilustrar esse conceito, o *script* Perl a seguir pode ser usado para fazer a extração — crie o arquivo novo `/home/luke/scripts/getsalt.pl` com o seguinte conteúdo:

```
1 #!/usr/bin/perl -w
2
3 my $hash=$ARGV[0];
4 # The hash is encoded as base64 twice:
5 use MIME::Base64;
6 $hash = decode_base64($hash);
7 $hash=~s/{SSHA};//;
8 $hash = decode_base64($hash);
9
10 # The salt length is four (the last four bytes).
11 $salt = substr($hash, -4);
12
13 # Split the salt into an array.
14 my @bytes = split(//,$salt);
15
16 # Convert each byte from binary to a human readable hexadecimal number.
17 foreach my $byte (@bytes) {
18 $byte = uc(unpack "H*", $byte);
19 print "$byte";
20 }
```

Vamos recuperar o *hash* de senha do usuário **luke**:

```
$ ldapsearch -x -LLL -H ldap://ldap.intnet/ -D 'cn=admin,dc=intnet' -b 'dc=intnet'
-w 'rnpesr' 'uid=luke' userPassword | grep '^userPassword::' | awk '{print $NF}'
e1NTSEF9anFkZC80MW1BT3dFYVpkMFpvWUZzYk1xQTJyVlVMVlU=
```

Executando o *script* **getsalt.pl**, podemos extrair o *salt* da senha. Note que o valor de saída está em hexadecimal.

```
$ perl ~/scripts/getsalt.pl e1NTSEF9anFkZC80MW1BT3dFYVpkMFpvWUZzYk1xQTJyVlVMVlU= ;
echo
D550B554
```

5. De volta ao ataque de dicionário, vamos executá-lo usando o **hashcat**. Primeiro, temos que descobrir a qual código o *hash* SSHA do LDAP corresponde:

```
$ hashcat --help | grep SSHA
111 | nsldaps, SSHA-1(Base64), Netscape LDAP SSHA | HTTP, SMTP, LDAP
Server
1711 | SSHA-512(Base64), LDAP {SSHA512} | HTTP, SMTP, LDAP
Server
10300 | SAP CODVN H (PWDSALTEDHASH) iSSHA-1 | Enterprise Application
Software (EAS)
```

O código é, então, **111**. Quanto ao tipo de ataque:

```
$ hashcat --help | grep 'Attack Modes' -A8
- [ Attack Modes ] -

# | Mode
===+=====
0 | Straight
1 | Combination
3 | Brute-force
6 | Hybrid Wordlist + Mask
7 | Hybrid Mask + Wordlist
```

O ataque de dicionário, também conhecido como *straight mode* (https://hashcat.net/wiki/doku.php?id=dictionary_attack), possui código 0.

Falta apenas obter uma *wordlist* apropriada para executar o ataque. Procurando por termos como "*wordlist*", "*password*" ou "*common*" no Google, é possível encontrar uma infinidade de páginas web dedicadas ao assunto, como por exemplo <https://github.com/danielmiessler/SecLists/tree/master/Passwords>. Iremos usar uma *wordlist* que alegadamente contém as 10 milhões de senhas mais comuns, que pode ser baixada na URL anteriormente mencionada ou solicitada ao instrutor. Note que, para um arquivo que contém apenas texto puro, seu tamanho é impressionante:

```
$ wget -q https://github.com/danielmiessler/SecLists/raw/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt
```

```
$ du -sh 10-million-password-list-top-1000000.txt
8,2M    10-million-password-list-top-1000000.txt
```

Tudo pronto! Vamos executar o ataque:

```
$ hashcat --force --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-  
password-list-top-1000000.txt  
hashcat (v3.30) starting...
```

(...)

```
Session.....: hashcat  
Status.....: Exhausted  
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA  
Hash.Target.....: hashes.txt  
Time.Started.....: Wed Nov 14 01:16:49 2018 (1 sec)  
Time.Estimated...: Wed Nov 14 01:16:50 2018 (0 secs)  
Input.Base.....: File (10-million-password-list-top-1000000.txt)  
Input.Queue.....: 1/1 (100.00%)  
Speed.Dev.#1.....: 2591.6 kH/s (0.36ms)  
Recovered.....: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts  
Progress.....: 3999996/3999996 (100.00%)  
Rejected.....: 36/3999996 (0.00%)  
Restore.Point....: 999999/999999 (100.00%)  
Candidates.#1....: vjq445 -> vjht008  
HWMon.Dev.#1.....: N/A  
  
Started: Wed Nov 14 01:16:45 2018  
Stopped: Wed Nov 14 01:16:51 2018
```

Na máquina usada como exemplo (a velocidade pode variar de acordo com a velocidade da CPU/GPU disponível), o ataque aos quatro *hashes* disponíveis usando 10 milhões de senhas demorou... 9 segundos. Como visualizado em **Speed.Dev.#1**, a velocidade de tentativas foi de 2591 kilo-*hashes* por segundo ou, em outras palavras, 2591000 *hashes* por segundo. Foi descoberto um *digest*, que podemos visualizar emitindo o mesmo comando com a *flag* **--show**:

```
$ hashcat --force --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-  
password-list-top-1000000.txt --show  
luke:{SSHA}jqdd/41mA0wEaZd0ZoYFsbMqA2rVULVU:password
```

Excelente! Como era de se esperar, a senha fraca **password** do usuário **luke** foi descoberta usando o ataque de dicionário.

6. Mas, e as demais senhas? O usuário **han** e **sshca** possuem senhas relativamente mais complexas, mas sabemos que a senha do usuário **admin** é simples, **rnpesr**. Como essa *string* não consta do arquivo com 10 milhões de senhas usado no ataque anterior, ela não foi descoberta, no entanto.

Vamos executar um ataque de força-bruta contra essa senha. Para isso, alteraremos o modo de ataque do **hashcat** para **3**, e definiremos uma máscara igual a **?1?1?1?1?1** — senhas de até seis caracteres, apenas com caracteres de **[a-z]** minúsculos. Para aprender mais sobre a sintaxe de máscaras suportadas pelo **hashcat**, consulte sua página de manual ou https://hashcat.net/wiki/doku.php?id=mask_attack.

Ao trabalho:

```
$ hashcat --force --hash-type 111 --attack-mode 3 --username hashes.txt  
?l?l?l?l?l?l  
hashcat (v3.30) starting...  
  
(...)  
  
[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
```

Após a inicialização, a linha acima será mostrada. Podemos apertar os atalhos destacados entre colchetes para instruir o **hashcat** com ações durante o ataque. Apertando **s**, visualizamos o estado atual do ataque:

```
Session.....: hashcat  
Status.....: Running  
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA  
Hash.Target.....: hashes.txt  
Time.Started.....: Wed Nov 14 01:18:02 2018 (7 secs)  
Time.Estimated...: Wed Nov 14 01:18:34 2018 (25 secs)  
Input.Mask.....: ?l?l?l?l?l?l [6]  
Input.Queue.....: 1/1 (100.00%)  
Speed.Dev.#1.....: 27956.2 kH/s (6.13ms)  
Recovered.....: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts  
Progress.....: 268409856/1235663104 (21.72%)  
Rejected.....: 0/268409856 (0.00%)  
Restore.Point....: 99072/456976 (21.68%)  
Candidates.#1....: saufph -> xqoxjk  
HWMon.Dev.#1.....: N/A
```

Observando a linha **Progress**, notamos que o ataque está 22,83% concluído. Aguardamos.

```
{SSHA}AZv4/v1spKXTHw5G0K1y+vQhPrnb7CzA:rnpesr
```

Após algum tempo, a linha acima é mostrada na tela. O **hashcat** conseguiu quebrar a senha do usuário **admin**, descobrindo-a como sendo **rnpesr**. Aguardamos a conclusão do processo.

```
Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target.....: hashes.txt
Time.Started.....: Wed Nov 14 01:18:02 2018 (30 secs)
Time.Estimated...: Wed Nov 14 01:18:32 2018 (0 secs)
Input.Mask.....: ?l?l?l?l?l?l [6]
Input.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 27362.0 kH/s (6.14ms)
Recovered.....: 2/4 (50.00%) Digests, 2/4 (50.00%) Salts
Progress.....: 1235663104/1235663104 (100.00%)
Rejected.....: 0/1235663104 (0.00%)
Restore.Point....: 456976/456976 (100.00%)
Candidates.#1....: sacxqg -> xqqfqg
HWMon.Dev.#1.....: N/A

Started: Wed Nov 14 01:18:00 2018
Stopped: Wed Nov 14 01:18:33 2018
```

Depois de 33 segundos, o ataque encontra-se 100% concluído. Um novo *digest* foi descoberto, como podemos visualizar com a *flag --show*:

```
$ hashcat --force --hash-type 111 --attack-mode 3 --username hashes.txt
?l?l?l?l?l?l --show
luke:{SSHA}jqdd/41mA0wEaZd0ZoYFsbMqA2rVULVU:password
admin:{SSHA}AZv4/v1spKXTHw5GOK1y+vQhPrnb7CzA:rnpesr
```

O **hashcat** reporta não somente a senha descoberta do usuário **admin**, bem como a senha do usuário **luke** descoberta na execução anterior. Isso ocorre porque o **hashcat** mantém o histórico de ataques realizados no diretório `~/.hashcat`:

```
$ ls -l ~/.hashcat/
hashcat.dictstat
hashcat.potfile
kernels
sessions
```

E assim, concluímos nossa busca por senhas fracas, via ataques de dicionário e força-bruta. O próximo passo, naturalmente, seria alterar o valor de senha dos usuários para um valor novo (bloqueando seu acesso), informá-los da nova senha e comunicar que devem alterar sua senha para uma combinação segura assim que possível.

4) Criação da VM do servidor de arquivos NFS

Iremos implementar, agora, um servidor de arquivos simples para que os colaboradores da Intranet possam compartilhar arquivos e ter uma opção de backup emergencial para suas estações

de trabalho. Como o ambiente que estamos simulando é inteiramente baseado em Linux, não há a necessidade de configurar uma solução interoperável com outros sistemas operacionais, como o Samba. Por isso, utilizaremos o NFS (*Network File System*), que é significativamente mais fácil de ser implementado.

1. O primeiro passo, assim como fizemos antes, é clonar a máquina `debian-template` e criar uma nova, que chamaremos de `nfs`. Essa máquina estará conectada a uma única rede `host-only`, com o mesmo nome que foi alocado para a interface de rede da máquina virtual `ns1`, configurada durante a sessão 2, que está conectada à DMZ. O IP da máquina será 10.0.42.3/24.

Concluída a clonagem, ligue a máquina e faça login como o usuário `root`. Em seguida, use o script `/root/scripts/changehost.sh` para fazer a configuração automática:

```
# hostname ; whoami
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h nfs -i 10.0.42.3 -g 10.0.42.1
Signing ssh_host_ecdsa_key.pub key...
Signing ssh_host_ed25519_key.pub key...
Signing ssh_host_rsa_key.pub key...
Configuring host key trust...
Configuring user key trust...
All done!
```

```
$ ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.3/24 enp0s3
nfs
root
```

5) Ajuste das regras de firewall

1. Se você tentar acessar a máquina `nfs` recém-criada usando o SSH, a partir da máquina `client`, rapidamente perceberá que não é possível fazer o acesso:

```
$ hostname ; whoami
client
luke
```

```
$ ssh nfs
ssh: connect to host nfs port 22: Connection timed out
```

Qual a razão disso? Após breve reflexão, fica claro que o problema reside no firewall. Vamos revisá-lo.

2. Logue na máquina **ns1** como o usuário **root**, como de costume. Tendo em vista que o acesso que desejamos fazer **passa pelo** firewall, devemos checar a *chain* FORWARD:

```
# hostname ; whoami
ns1
root
```

```
# iptables -L FORWARD -vn
Chain FORWARD (policy DROP 171 packets, 12318 bytes)
 pkts bytes target     prot opt in     out     source        destination
31432  74M ACCEPT     all  --  *      *       0.0.0.0/0      0.0.0.0/0
state RELATED,ESTABLISHED
   3   180 ACCEPT     tcp  --  *      enp0s3  10.0.42.0/24   0.0.0.0/0
multiport dports 80,443
   3   180 ACCEPT     tcp  --  *      enp0s3  192.168.42.0/24 0.0.0.0/0
multiport dports 80,443
   2   148 ACCEPT     udp  --  *      *       192.168.42.0/24 10.0.42.2
udp dpt:53
  68  4080 ACCEPT     tcp  --  *      *       192.168.42.0/24 10.0.42.2
multiport dports 22,389
```

Observe que liberamos o acesso SSH a partir da Intranet apenas para a máquina **ns2**, e nenhuma outra. Considerando que além da máquina **ns** criaremos ainda outros servidores em nosso *datacenter* simulado, essa regra obviamente está inadequada. Vamos removê-la:

```
# iptables -D FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p tcp -m multiport
--dports 22,389 -j ACCEPT
```

Em seu lugar, vamos criar duas regras: uma que reautoriza o acesso da Intranet ao servidor LDAP (que estava embutido na regra antiga), e outra que autoriza a Intranet a realizar SSH para qualquer máquina da DMZ:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p tcp -m tcp --dport 389
-j ACCEPT
```

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.0/24 -p tcp -m tcp --dport 22
-j ACCEPT
```

3. Ainda temos que tratar do acesso da Intranet ao serviço NFS, que configuraremos na atividade a seguir. Para nossa sorte, a configuração do firewall para o NFS versão 4 (que utilizaremos neste curso) é significativamente mais fácil que as versões anteriores, bastando liberar o tráfego dos

clientes para a porta 2049/TCP do servidor remoto (em nosso caso, a máquina **nfs**). A regra a seguir irá atender este requisito:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.3/32 -p tcp -m tcp --dport 2049 -j ACCEPT
```

4. Finalmente, salve as regras de firewall atuais:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

6) Configuração do servidor de arquivos NFS e quotas de disco

1. Queremos usar o servidor NFS para armazenar arquivos de usuários da Intranet, para compartilhamento e backup. Isso imediatamente suscita uma grande preocupação: imagine um cenário em que usuários querem armazenar muitos arquivos no servidor (de forma acidental ou maliciosa)—isso pode atrapalhar o uso de outros colaboradores, ou até mesmo chegar a encher a partição raiz (/) do sistema, causando instabilidade. Para evitar esse cenário, vamos criar uma partição dedicada para arquivos de usuário no diretório **/home** do servidor **nfs**, e também aplicar *quotas* de disco aos usuários.

Desligue a máquina **nfs** e adicione a ela um novo disco de 10 GB, usando a interface do Virtualbox. A seguir, formate o disco e adicione-o ao sistema LVM, criando um novo *volume group* **vg-home** com um único volume lógico **lv-home**, de forma análoga ao que fizemos nas atividades (7) e (8) da sessão (1). Finalmente, formate esse volume em **ext4** e ative sua montagem automática durante o *boot* da máquina **nfs** no diretório **/home** com as opções **defaults,nosuid,nodev**.

Vamos ao trabalho. Após desligar a VM e adicionar o disco de 10 GB, acessamos a máquina **nfs** como o usuário **root**:

```
# hostname ; whoami
nfs
root
```

O próximo passo é descobrir sob qual nome o disco foi detectado. Nesse caso, temos a vantagem de saber que o tamanho do disco novo, 10 GB, é diferente do disco preexistente.

```
# dmesg | grep 'GiB'
[ 1.585018] sd 1:0:0:0: [sdb] 20971520 512-byte logical blocks: (10.7 GB/10.0 GiB)
[ 1.585187] sd 0:0:0:0: [sda] 16777216 512-byte logical blocks: (8.59 GB/8.00 GiB)
```

Evidentemente, o disco `/dev/sdb` é o que acabamos de adicionar, portanto. Vamos formatá-lo:

```
# fdisk /dev/sdb
```

Bem-vindo ao fdisk (util-linux 2.29.2).
As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.

A unidade não contém uma tabela de partição conhecida.
Criado um novo rótulo de disco DOS com o identificador de disco 0x3bc30929.

Comando (m para ajuda): o
Criado um novo rótulo de disco DOS com o identificador de disco 0x8a16601c.

Comando (m para ajuda): n
Tipo da partição
 p primária (0 primárias, 0 estendidas, 4 livre)
 e estendida (recipiente para partições lógicas)

Selecione (padrão p):

Usando resposta padrão p.
Número da partição (1-4, padrão 1):
Primeiro setor (2048-20971519, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-20971519, padrão 20971519):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 10 GiB.

Comando (m para ajuda): t
Selecionou a partição 1
Tipo de partição (digite L para listar todos os tipos): 8e
O tipo da partição "Linux" foi alterado para "Linux LVM".

Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.

Agora, vamos criar o volume físico:

```
# pvcreate /dev/sdb1  
Physical volume "/dev/sdb1" successfully created.
```

O grupo de volumes:

```
# vgcreate vg-home /dev/sdb1  
Volume group "vg-home" successfully created
```

E, finalmente, o volume lógico:

```
# lvcreate -l +100%FREE -n lv-home vg-home  
Logical volume "lv-home" created.
```

Vamos, agora, formatar o sistema de arquivos do LV:

```
# mkfs.ext4 /dev/mapper/vg--home-lv--home  
mke2fs 1.43.4 (31-Jan-2017)  
Creating filesystem with 2620416 4k blocks and 655360 inodes  
Filesystem UUID: be99743c-7ca0-4144-8548-d7aab33a878b  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (16384 blocks): done  
Writing superblocks and filesystem accounting information: done
```

A seguir, sincronizar os arquivos do diretório **/home** atual com o LV recém-criado:

```
# mount /dev/mapper/vg--home-lv--home /mnt/
```

```
# rsync -av /home/ /mnt/
sending incremental file list
./
aluno/
aluno/.bash_history
aluno/.bash_logout
aluno/.bashrc
aluno/.profile
aluno/.vimrc
luke/
luke/.bash_history
luke/.bash_logout
luke/.bashrc
luke/.profile
luke/scripts/
luke/scripts/sshsign_user.sh

sent 11,669 bytes  received 225 bytes  23,788.00 bytes/sec
total size is 10,787  speedup is 0.91
```

```
# umount /mnt
```

E, finalmente, configurar a montagem no `/etc/fstab` e montar o LV:

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4 defaults,nosuid,nodev 0 2
```

```
# mount -a
```

```
# df -h | sed -n '1p; /\//home/p'
Sist. Arq.                               Tam. Usado Disp. Uso% Montado em
/dev/mapper/vg--home-lv--home 9,8G  37M 9,3G  1% /home
```

2. Agora, vamos instalar os pacotes para habilitar o compartilhamento de arquivos via NFS e *quotas* de disco. Como `root`, instale os pacotes `nfs-kernel-server`, `quota` e `quotatool`:

```
# apt-get install nfs-kernel-server quota quotatool
```

3. Vamos configurar primeiro o sistema de *quotas*. Edite a entrada do diretório `/home` no arquivo

`/etc/fstab` e adicione as opções `usrquota,grpquota`, que ativam suporte a *quotas* por usuário e por grupo no sistema de arquivos.

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4 defaults,nosuid,nodev,usrquota,grpquota
0 2
```

Em seguida, remonte o sistema de arquivos e verifique se o sistema de *quotas* foi habilitado na partição.

```
# mount -o remount /home
```

```
# mount | grep '/home'
/dev/mapper/vg--home-lv--home on /home type ext4
(rw,nosuid,nodev,relatime,quota,usrquota,grpquota,data=ordered)
```

Crie os arquivos de configuração de *quotas* usando o comando `quotacheck`:

```
# quotacheck -ugc /home/
```

Pronto, o sistema de *quotas* está configurado. Iremos editar *quotas* de usuário e testar seu funcionamento mais à frente, após a configuração do NFS.

4. O próximo passo é configurar o serviço NFS—lembre-se que queremos disponibilizar compartilhamentos para arquivos de usuário, no diretório `/home` do sistema. Insira a linha a seguir no arquivo `/etc/exports`:

```
# echo '/home 192.168.42.0/24(rw,async,no_subtree_check,root_squash)' >>
/etc/exports
```

A linha acima irá configurar a exportação o diretório `/home` para todas as máquinas da Intranet (faixa 192.168.42.0/24), em modo leitura-escrita, assíncrono (i.e. escritas ao disco do servidor remoto não precisam ter sido efetivadas para que o cliente receba confirmação), sem checagem de sub-árvores de montagem (consultar página de manual com `man 5 exports`) e desabilitando o mapeamento de UID do `root` da máquina remota no servidor local.

Para exportar o diretório, basta executar:

```
# exportfs -a
```

Finalmente, para visualizar quais diretórios estão sendo exportados, execute:

```
# showmount -e
Export list for nfs:
/home 192.168.42.0/24
```

5. Vamos testar nosso sistema de compartilhamento de arquivos via NFS e *quotas* de disco. Acesse a máquina **client** como o usuário **root**, crie um diretório **/remote** para ser o ponto de montagem NFS e monte o diretório compartilhado.

```
# hostname ; whoami
client
root
```

```
# mkdir /remote
```

```
# mount -t nfs 10.0.42.3:/home /remote
```

```
# ls -l /remote/
total 40
drwxr-xr-x 2 aluno aluno  4096 nov 13 17:01 aluno
-rw----- 1 root  root   8192 nov 14 01:49 aquota.group
-rw----- 1 root  root   8192 nov 14 01:49 aquota.user
drwx----- 2 root  root  16384 nov 14 01:46 lost+found
drwxr-xr-x 3 luke  sysadm 4096 nov 14 01:41 luke
```

6. Um dos principais problemas em sistemas de compartilhamento de arquivos em ambientes Unix é o mapeamento de UIDs e GIDs — como garantir que os usuários de múltiplas máquinas remotas possuam os mesmos identificadores que os usuários existentes no servidor de arquivos? Felizmente, nosso sistema centralizado de autenticação usando LDAP resolve esse problema de forma transparente: todos os usuários possuem um valor de UID e GID consistente em todo o *datacenter*, já que as contas são gerenciadas em um ponto único.

Senão, vejamos: como o usuário **luke**, tente criar um arquivo novo dentro do ponto de montagem **/remote/luke**:

```
$ hostname ; whoami
client
luke
```

```
$ echo test > /remote/luke/file
```

```
$ cat /remote/luke/file  
test
```

Funcionou perfeitamente! Uma vez que os valores de UID e GID do usuário **luke** são consistentes entre a máquina **client** e o servidor de arquivos **nfs**, não temos problemas de permissão.

7. E quanto ao usuário **root**? Será que o **root** local da máquina **client** possui acesso irrestrito aos arquivos compartilhados?

```
$ su -  
Senha:  
root@client:~#
```

```
# hostname ; whoami  
client  
root
```

```
# echo test > /remote/file  
-su: /remote/file: Permissão negada
```

```
# rm /remote/aquota.user  
rm: não foi possível remover '/remote/aquota.user': Permissão negada
```

Devido à utilização da opção **root_squash** no compartilhamento configurado via arquivo **/etc/exports** da máquina **nfs**, o mapeamento de UID do usuário **root** em máquinas remotas é desativado, efetivamente impedindo-o de alterar quaisquer arquivos.

8. Vamos testar o sistema de *quotas*. Na máquina **nfs**, como o usuário **root**, edite as *quotas* do usuário **luke** usando o comando **edquota**:

```
# edquota -u luke
```

O comando **edquota** irá invocar um editor (indicado pela variável de ambiente **\$EDITOR**) para que as *quotas* sejam ajustadas. Vamos editar os campos *soft* e *hard* da seção *block* do arquivo, ajustando limites de 100 MB e 200 MB, respectivamente — note que os valores devem ser informados em kilobytes. Pode-se, opcionalmente, também setar um limite para *inodes* que o usuário pode criar.

```
Disk quotas for user luke (uid 10000):
  Filesystem          blocks      soft      hard      inodes      soft
hard
  /dev/mapper/vg--home-lv--home 32    100000    200000        8        0
0
```

Para verificar as *quotas* de um sistema de arquivos, use o comando **repquota**:

```
# repquota -u /home
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days

      Block limits              File limits
User      used  soft  hard  grace  used  soft  hard  grace
-----
root      --    20    0    0              2    0    0
aluno     --    24    0    0              6    0    0
luke      --    32 100000 200000         8    0    0
```

Para ativar as *quotas* em uma partição, utilize o comando **quotaon**:

```
# quotaon -ug /home
```

9. Acesse a máquina **client** como o usuário **luke**. Vamos tentar extrapolar o limite estabelecido pela *quota* no passo anterior.

```
$ hostname ; whoami
client
luke
```

O kernel do SO é um arquivo interessante a ser usado para esse teste, já que possui um tamanho razoável. Vamos copiá-lo sucessivas vezes para o diretório **/remote/luke** e verificar o que acontece:

```
$ du -sh /boot/vmlinuz-4.9.0-8-amd64
4,1M    /boot/vmlinuz-4.9.0-8-amd64
```

```
$ for i in {1..100}; do cp /boot/vmlinuz-4.9.0-8-amd64 /remote/luke/vmlinuz-$i;
done
cp: falha ao fechar '/remote/luke/vmlinuz-49': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-50': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-51': Disk quota exceeded
(...)
cp: falha ao fechar '/remote/luke/vmlinuz-98': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-99': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-100': Disk quota exceeded
```

Note que após 48 cópias de arquivo, o sistema reporta a *quota* de disco como excedida, e o usuário não pode mais escrever na partição. De fato, checando o estado da *quota* de disco com o comando `repquota` na máquina `nfs`, temos que:

```
# hostname ; whoami
nfs
root
```

```
# repquota -u /home/
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days
```

User	Block limits				File limits			
	used	soft	hard	grace	used	soft	hard	grace
root	-- 20	0	0		2	0	0	
aluno	-- 24	0	0		6	0	0	
luke	+- 200000	100000	200000	6days	108	0	0	

Temos, portanto, que nosso esquema de *quotas* está funcionando como esperado. Não se esqueça de apagar os diversos arquivos `vmlinuz*` que criamos, para liberar espaço no disco novamente:

```
# rm /home/luke/vmlinuz-*
```



Observe que apenas os usuários **aluno** e **luke** possuem pastas no diretório **/home** compartilhado pela máquina **nfs**. Isso se deve ao fato de que apenas esses usuários haviam feito acesso local à máquina **nfs** até aquele momento — lembre-se que o arquivo de configuração **/usr/share/pam-configs/mkhomedir** que aplicamos ao PAM cria diretórios **home** apenas quando o usuário faz acesso à máquina pela primeira vez. Como consequência, o usuário **han**, para citar um exemplo, não possui uma pasta no servidor de arquivos.

Em produção, seria interessante que a pasta compartilhada do usuário fosse criada assim que este fosse adicionado à base LDAP, juntamente com o comando **ldapadduser**, por exemplo. Um *script* shell seria ideal para resolver essa situação. Claro, é possível que nem todos os novos usuários criados na base LDAP devam ter uma pasta nesse servidor, o que pode complicar sua configuração.

7) Uso de ACLs localmente

Imagine a seguinte situação, agora: o usuário **luke** quer criar um arquivo novo, sigiloso, e dar permissão para que **han** possa visualizá-lo. Ora, com as permissões padrão disponíveis em um sistema Linux, quais são nossas opções?

Sabemos que, ao criar o arquivo, o usuário-dono será **luke** e o grupo-dono será **sysadm**. Se **luke** altera o grupo-dono do arquivo para **fwadm** e **chmod** de **640**, apesar de a permissão objetivada para **han** ser garantida, todos os outros membros do grupo **fwadm** também poderão visualizar o arquivo, que não é o que queremos. Se garante-se a permissão de **644**, não só **han** como qualquer outro usuário pode visualizar o arquivo. Finalmente, a alternativa final que seria adicionar **han** ao grupo **sysadm** pode não ser desejável ou aceitável do ponto de vista administrativo. O que fazer?

O uso de ACLs (*Access Control Lists*) é especialmente adequado para esse tipo de situação, quando precisamos configurar permissões de arquivos e diretórios de forma granular. Com o uso de ACLs, é possível definir permissões customizadas para usuários e grupos diferentes dos donos do arquivo/diretório original, solucionando problemas de permissionamento para os quais o sistema tradicional de permissões Unix é inadequado.

1. Acesse a máquina **nfs** como o usuário **root**. Para consultar e ajustar ACLs localmente, basta instalar o pacote **acl**:

```
# hostname ; whoami
nfs
root
```

```
# apt-get install acl
```

2. Vamos testar o funcionamento de ACLs localmente, usando os usuários **luke** e **han**. Acesse a máquina **nfs** como o usuário **luke**:

```
$ hostname ; whoami ; pwd
nfs
luke
/home/luke
```

Agora, crie o arquivo novo `~/teste`, com qualquer conteúdo. Em seguida, consulte suas ACLs atuais.

```
$ echo oi > teste
```

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
group::r--
other::r--
```

3. Imaginemos que o arquivo criado na atividade anterior é especialmente sigiloso, devendo ser visualizado apenas pelo usuário `han` e seu dono, `luke`. Primeiro, retire as permissões do grupo e de outros:

```
$ chmod 600 ~/teste
```

Em seguida, use ACLs para dar permissão de leitura a `han`:

```
$ setfacl -m u:han:r ~/teste
```

Verifique as permissões Unix tradicionais—observe que ao final da coluna de permissionamento do `ls` vemos o caractere `+`, que indica que o arquivo possui permissões estendidas na forma de ACLs.

```
$ ls -ld /home/luke/teste
-rw-r-----+ 1 luke sysadm 3 nov  1 18:27 /home/luke/teste
```

Consulte novamente as ACLs do arquivo, verificando que a configuração desejada foi aplicada.

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
user:han:r--
group::---
mask::r--
other::---
```

4. Terá funcionado? Vamos ver. Como o usuário **aluno**, tente visualizar o conteúdo do arquivo **/home/luke/teste**:

```
$ whoami
aluno
```

```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

E como **han**? Vamos ver:

```
$ whoami
han
```

```
$ cat /home/luke/teste
oi
```

5. Como **luke**, vamos remover a ACL de leitura do usuário **han** e testar:

```
$ whoami
luke
```

```
$ setfacl -x u:han ~/teste
```

```
$ su - han
Senha:
```

```
$ whoami
han
```



```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

Perfeito! Lembre-se que também podemos configurar ACLs para grupos através do caractere **g**, o que não foi testado nesta atividade.

8) Uso de ACLs via NFS

A atividade anterior, apesar de interessante, é pouco prática quando consideramos nossa configuração atual: se ACLs podem apenas ser manipuladas localmente mas estamos mantendo nossos arquivos compartilhados via rede com NFS, então toda vez que um usuário quiser alterar ACLs ele terá que fazer um acesso local à máquina **nfs**? Não é razoável fazermos isso. De fato, tente fazer a alteração de ACLs a partir da máquina **client** como o usuário **luke** — lembre-se: assim como fizemos na máquina **nfs**, será necessário instalar o pacote **acl** antes de realizar este teste:

```
$ hostname ; whoami
client
luke
```

```
$ setfacl -m u:han:rw /remote/luke/teste
setfacl: /remote/luke/teste: Operação não suportada
```

Com efeito, ACLs POSIX não são suportadas diretamente via **setfacl** em *mounts* NFS.

Por outro lado, *mounts* NFS versão 4 possuem suporte a ACLs—de fato, a um conjunto de permissões ainda mais granulares e expressivas que as ACLs POSIX padrão. Mas primeiro, temos que responder à pergunta: nosso compartilhamento atual está em qual versão? Vamos ver:

```
$ mount | grep '/home' | grep -o 'vers=[0-9\.]*'
vers=4.2
```

Excelente, estamos usando a versão 4.2, o que deve ser suficiente. Os comandos para visualização e edição de ACLs NFSv4 não são os mesmos que utilizamos até agora, no entanto — vamos instalá-los.

1. Acesse a máquina **client** como o usuário **root** e instale o pacote **nfs4-acl-tools**:

```
# hostname ; whoami
client
root
```

```
# apt-get install nfs4-acl-tools
```

2. Agora sim, vamos testar o funcionamento de ACLs com a pasta compartilhada via NFS. Acesse como o usuário **luke**; para tornar o uso corriqueiro dessa pasta compartilhada mais conveniente, crie um link simbólico com o nome **remote** em seu diretório *home*.

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

```
$ ln -s /remote/luke/ ~/remote
```

3. Consulte as ACLs NFSv4 do arquivo criado na atividade anterior:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

O formato de representação de permissões NFSv4 é bastante diferente do que estamos acostumados — muitas opções e controles adicionais são suportados. Nesta atividade iremos trabalhar apenas com as permissões mais usuais, **rwX**, mas a página de manual `man 5 nfs4_acl` possui uma documentação bastante completa sobre as possibilidades de uso desse sistema. Em especial, a seção *ACE PERMISSIONS* é recomendada para entender o formado do *output* acima.

Como um exemplo, vamos analisar em detalhe a ACE (*Access Control Entry*) **A::OWNER@:rwatTcCy**:

- **A**: tipo da ACE; pode ser **A** (*allow*), **D** (*deny*), **U** (*audit*, usada para configurar log de acessos) e **L** (*alarm*, para gerar alarmes de sistema em caso de acesso).
- **:::** o segundo campo, neste caso vazio, define as *flags* da ACE; pode ser utilizado para indicar ACEs aplicáveis a grupos, configurações de herança da ACE para diretórios e arquivos-filho, ou *flags* administrativas para controlar eventos de log e alarme.
- **OWNER@**: define o *principal* ao qual se aplica a ACE corrente; pode ser um usuário, grupo ou uma de três ACEs especiais, **OWNER@**, **GROUP@** e **EVERYONE@**, funcionalmente equivalentes às suas contrapartes POSIX.
- **rwatTcCy**: permissões definidas pela ACE; no caso, temos definidas:
 - **r**: permissão de leitura para arquivos, ou listagem de diretórios.
 - **w**: permissão de escrita para arquivos, ou criação de novos arquivos em diretórios.
 - **a**: *append* de dados em arquivos (escrever ao final), ou criar novos subdiretórios em diretórios.
 - **t**: ler atributos do arquivo/diretório.
 - **T**: escrever atributos do arquivo/diretório.
 - **c**: ler ACLs NFSv4 do arquivo/diretório.

- **C**: escrever ACLs NFSv4 do arquivo/diretório.
- **y**: autorizar clientes a usar I/O síncrono com o servidor.

4. Vamos configurar uma ACL NFSv4 de leitura do arquivo para o usuário **han**, assim como fizemos anteriormente.

```
$ nfs4_setfacl -a A::han@intnet:rtcy ~/remote/teste
```

Vamos ver como ficaram as ACEs do arquivo:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::10002:rtcy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

Note que o nome de usuário **han@intnet** foi mapeado para o UID **10002** — que é consistente entre todas as máquinas do *datacenter* graças à integração com o LDAP que fizemos na sessão 3. Verifique a correspondência do UID:

```
$ getent passwd han
han*:10002:10002:han:/home/han:/bin/bash
```

5. Vamos testar? Acesse como o usuário **aluno** e tente exibir o conteúdo do arquivo **/remote/luke/teste**:

```
$ su - aluno
Senha:
```

```
$ whoami
aluno
```

```
$ cat /remote/luke/teste
cat: /remote/luke/teste: Permissão negada
```

Agora, teste com o usuário **han**:

```
$ su - han
Senha:
```

```
$ whoami  
han
```

```
$ cat /remote/luke/teste  
oi
```

Excelente, tudo funcionando a contento.

6. Como remover uma ACL NFSv4? É simples:

```
$ nfs4_setfacl -x A::han@intnet:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste  
A::OWNER@:rwatTcCy  
A::10002:rtcy  
A::GROUP@:tcy  
A::EVERYONE@:tcy
```

Ué, não funcionou. Para deletar ACEs, temos que especificá-las **exatamente** no mesmo formato da linha reportada pelo comando `nfs4_getfacl`, ou usando o índice numérico da regra. Vamos tentar novamente:

```
$ nfs4_setfacl -x A::10002:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste  
A::OWNER@:rwatTcCy  
A::GROUP@:tcy  
A::EVERYONE@:tcy
```

Agora sim, perfeito. Vamos verificar que a remoção da ACE surtiu efeito:

```
$ su - han  
Senha:
```

```
$ whoami  
han
```

```
$ cat /remote/luke/teste  
cat: /remote/luke/teste: Permissão negada
```

9) Controle granular de permissões via sudo

Para todas as ações privilegiadas que precisamos tomar até aqui, sempre usamos o comando `su` para nos tornarmos o usuário `root`, e então efetuamos a instalação de pacotes, adição de usuários ou criação de arquivos de configuração. Mas, como fica essa situação em um ambiente de *datacenter* como o que estamos simulando? Seria interessante passar a senha do usuário `root` para os usuários `luke` e `han` (e outros que viermos a criar), permitindo que tomem quaisquer ações nas máquinas?

O `sudo` (*Super User DO*) é um comando que permite que usuários comuns obtenham privilégios de outro usuário, em geral o `root`, para executar tarefas específicas dentro do sistema de maneira segura e controlável pelo administrador. Assim, podemos delimitar que um determinado usuário ou grupo pode executar apenas um pequeno conjunto de comandos dentro de um servidor específico. Como o `sudo` é compatível com *hostnames* e endereços IP, é possível utilizar o mesmo arquivo em todas as máquinas do parque, facilitando tremendamente o esforço de configuração.

Para ilustrar esse cenário, vamos solucionar dois exemplos hipotéticos:

- A colaboradora `leia` acaba de se juntar à equipe de `han`, o grupo `fwadm` em nosso sistema LDAP. Imagine que ela ficará responsável por editar regras no firewall de borda, a máquina `ns1`. Mas, por estar começando agora na empresa, `han` quer restringir o conjunto de comandos que `leia` pode executar na máquina, liberando apenas a edição do firewall via `iptables`. Sua senha será `seg10leia`. Nas demais máquinas (`ns2` e `nfs`) `leia` não deve ter qualquer acesso especial, apenas como um usuário regular.
- O colaborador `chewie` foi contratado para auxiliar na manutenção da base LDAP da empresa. Para desempenhar suas tarefas, iremos colocá-lo em um novo grupo `ldapadm`. Os membros desse grupo devem ter acesso aos principais comandos de edição do LDAP (criação, modificação e deleção de usuários e grupos) na máquina `ns2`. Sua senha será `seg10chewie`. Nas demais máquinas (`ns1` e `nfs`) `chewie` não deve ter qualquer acesso especial, apenas como um usuário regular.
- Os usuários atuais, `luke` e `han`, terão permissão para executar qualquer comando como o usuário `root`, em qualquer máquina.
- Observe que temos controles alheios ao `sudo` já aplicados que irão restringir o acesso de certos usuários — por exemplo, apenas membros do grupo `fwadm` conseguem fazer login na máquina `ns1` devido à configuração do `ns1cd` que realizamos na atividade (13) da sessão (3).

Vamos solucionar esses problemas?

1. Primeiro, devemos criar os usuários e grupos — vamos começar com `leia`. Como `root`, na máquina `ns2`:

```
# hostname ; whoami
ns2
root
```

```
# ldapadduser leia fwadm
Successfully added user leia to LDAP
Successfully set password for user leia
```

```
# ldapaddusertogroup leia fwadm
Successfully added user leia to group cn=fwadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd leia
Changing password for user uid=leia,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=leia,ou=People,dc=intnet
```

Agora, **chewie**. Lembre-se que no caso dele temos também que adicionar um novo grupo, **ldapadm**:

```
# ldapaddgroup ldapadm
Successfully added group ldapadm to LDAP
```

```
# ldapadduser chewie ldapadm
Successfully added user chewie to LDAP
Successfully set password for user chewie
```

```
# ldapaddusertogroup chewie ldapadm
Successfully added user chewie to group cn=ldapadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd chewie
Changing password for user uid=chewie,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=chewie,ou=People,dc=intnet
```

Fácil, não é mesmo?

2. Como já instalamos o **sudo** na máquina **debian-template** na sessão 1, o comando deve estar disponível no **\$PATH**:

```
# which sudo
/usr/bin/sudo
```

Vamos testar? Edite o arquivo **/etc/sudoers** e autorize o usuário **luke** a usar o comando

`/bin/grep` como o usuário `root`. Edite o arquivo com:

```
# visudo
(...)
```

Insira a linha `luke ALL=/bin/grep` abaixo da entrada do usuário `root` na seção *User privilege specification*, como se segue:

```
# grep -A2 'User privilege specification' /etc/sudoers
# User privilege specification
root    ALL=(ALL:ALL) ALL
luke    ALL=(root)    /bin/grep
```

Como o usuário `luke`, tente usar o comando `grep` com o `sudo` para visualizar um arquivo restrito, como o `/etc/shadow`:

```
# su - luke
```

```
$ whoami
luke
```

```
$ sudo grep root /etc/shadow
```

Presumimos que você recebeu as instruções de sempre do administrador de sistema local. Basicamente, resume-se a estas três coisas:

- #1) Respeite a privacidade dos outros.
- #2) Pense antes de digitar.
- #3) Com grandes poderes vêm grandes responsabilidades.

[sudo] senha para luke:

```
root:$6$s7Gt1cd.$UXQf67CVYxR7HP..h2wvh0x4n0tBT7do28R1uChYdMpZc.uLi430KdtentrWD2zSTK
v9EyB7Bdqcpwr6nAlNo.:17848:0:99999:7:::
```

Agora, tente executar um comando não-autorizado, como o `cat`:

```
$ sudo cat /etc/shadow
Sinto muito, usuário luke não tem permissão para executar "/bin/cat /etc/shadow"
como root em ns2.intnet.
```

Perfeito, nosso teste inicial funcionou com sucesso. Remova a linha referente ao usuário `luke` no arquivo `/etc/sudoers`, e vamos prosseguir.

3. Queremos controlar os comandos utilizados nos servidores do *datacenter*, que até o momento são as máquinas *ns1*, *ns2* e *nfs*.

Tecnicamente, seria possível configurar o *sudo* em cada um dos servidores — uma vez que as regras para a usuária *leia* são específica para a máquina *ns1* e as do usuário *chewie* se aplicam à máquina *ns2* — mas não faremos isso. Imagine que, ao invés de três máquinas, nosso *datacenter* tivesse centenas de VMs: seria factível controlar as regras de *sudo* localmente em cada um dos servidores? É evidente que não.

Temos algumas opções para configurar o *sudo* de forma coordenada entre múltiplos servidores. A gestão de configuração de servidores, usando ferramentas como o Puppet, Chef ou Ansible (que veremos na sessão seguinte) é um dos métodos mais modernos e eficientes de atingir esse objetivo.

Assim sendo, ao invés de desenvolver uma solução inadequada para o problema de configuração do *sudo* neste momento, iremos fazer uma breve pausa nessa tarefa. Na próxima sessão, aprenderemos as bases do Ansible e, com isso, a solução do problema ficará bem mais fácil. Nos vemos em breve!