Sessão 3: Autenticação centralizada

Retomando o cenário apresentado na introdução da primeira sessão, num ambiente em que a virtualização é usada em larga escala, teremos diversas máquinas virtuais operando cada qual com seus serviços alocados. Imagine, hipoteticamente, que temos centenas de VMs dentro do *datacenter*. Vários desafios podem surgir à mente, mas tente responder a seguinte pergunta: com relação à gestão de contas, o que fazer quando um novo colaborar é integrado à equipe? Ou, por outro lado, quando um funcionário é desligado da empresa?

Ora, se temos centenas de VMs, é fácil supor que teremos que logar nas diferentes máquinas que novo colaborador (ou o antigo) deverá acessar e criar uma conta de usuário para ele — além disso, adicioná-lo a grupos e editar permissões relevantes. Fazer esse procedimento inúmeras vezes é claramente um processo que pode gerar erros de configuração, então poderia-se pensar em automatizá-lo, digamos, via *shell scripts*. Uma boa solução, mas não a ideal neste caso.

Sistemas de autenticação centralizados, como NIS (*Network Information Service*) ou LDAP (*Lightweight Directory Access Protocol*) são excelentes ferramentas para facilitar a gestão em cenários como o apresentado — adicionando o usuário em um único ponto, é possível distribuir essa configuração para dezenas, ou centenas, de máquinas de forma instantânea. O gerenciamento de grupos no sistema centralizado também permite atribuir permissionamento de forma fácil, ou removê-lo quando necessário.

Nesta sessão, iremos configurar um sistema de autenticação centralizado para o nosso laboratório usando LDAP, no qual gerenciaremos usuários e grupos, e faremos a integração desse sistema de autenticação com o Linux através do PAM (*Pluggable Authentication Modules*). Em lugar de fazer o controle de senhas dos usuários diretamente via `/etc/shadow` ou no LDAP, criaremos um sistema de autoridade certificadora (*Certificate Authority*) para o SSH, com o qual os usuários farão login nos servidores usando chaves assimétricas assinadas pela CA. Finalmente, para controlar ataques de força-bruta contra os servidores, usaremos o programa Fail2Ban para realizar o bloqueio automático de atacantes no firewall de host das máquinas.

1) Topologia desta sessão

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

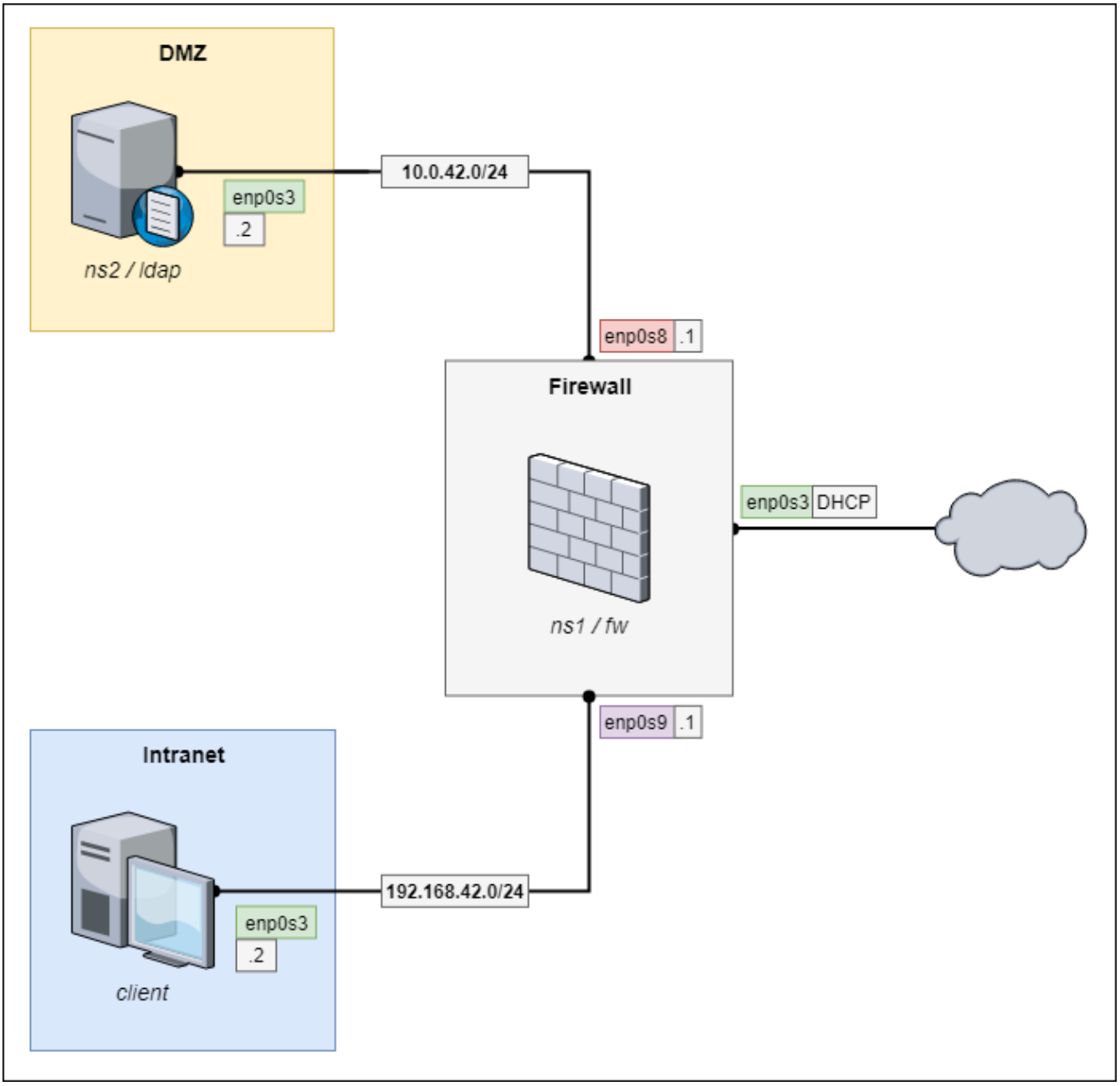

Figura 1. Topologia de rede desta sessão

Teremos três 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.
- `client`, localizada na Intranet e usada a partir de agora como ponto de partida para logins remotos nos diferentes servidores do *datacenter* simulado. Endereço IP 192.168.42.2/24.

1. Observe que a topologia acima prevê a criação de um novo *alias* de rede para a máquina `ns2`, o nome `ns2`. Antes de começar, vamos ajustar nossa configuração DNS para refletir essa situação: 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 CNAME para a máquina `ns2`, como se segue. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

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

```
# grep '^ldap ' /etc/nsd/zones/intnet.zone
ldap    IN    CNAME             ns2
```

Assine o arquivo de zonas usando o *script* criado na sessão anterior:

```
# 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 a entrada foi criada com sucesso, usando o comando `dig`:

```
# dig @127.0.0.1 ldap.intnet +noall +answer

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 ldap.intnet +noall +answer
; (1 server found)
;; global options: +cmd
ldap.intnet.            86138   IN      CNAME   ns2.intnet.
ns2.intnet.             86138   IN      A       10.0.42.2
```

Certifique-se que a transferência de zonas entre o servidor DNS primário e secundário está funcionando corretamente — repita a consulta, desta vez no servidor `ns2`:

```
# dig @10.0.42.2 ldap.intnet +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @10.0.42.2 ldap.intnet +noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48391
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
ldap.intnet.            86207   IN      CNAME   ns2.intnet.
ns2.intnet.             86207   IN      A       10.0.42.2

;; Query time: 0 msec
;; SERVER: 10.0.42.2#53(10.0.42.2)
;; WHEN: Tue Nov 13 18:28:16 -02 2018
;; MSG SIZE  rcvd: 74
```

2) Configuração do servidor LDAP

1. O servidor LDAP, como explicitado na topologia desta sessão, será a máquina `ns2`. Acesse-a como o usuário `root`:

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

   Vamos, primeiramente, instalar o servidor LDAP através dos pacotes:

   ```
   # apt-get install slapd ldap-utils ldapscripts
   ```

   Durante a instalação, será solicitada uma senha administrativa para o LDAP, que iremos redefinir em breve. Informe `rnpesr`.

2. Agora, vamos reconfigurar o servidor LDAP para definir alguns parâmetros que não foram questionados durante a instalação via `apt-get`. Execute:

   ```
   # dpkg-reconfigure -plow slapd
   ```

   Informe as seguintes opções:

   Tabela 1. Configurações do pacote slapd

Pergunta	Opção
Omitir a configuração do servidor OpenLDAP?	Não
Nome de domínio DNS	intnet
Nome da organização	seg10
Senha do administrador	rnpesr (repetir)
Backend da base de dados a ser usado	MDB
Você deseja que a base de dados seja removida quando o pacote slapd for expurgado ("purged")	Não
Move a base de dados antiga	Sim

Durante a configuração, será criada uma base LDAP (semi) vazia, apenas com a raiz `dc=intnet` e o usuário administrativo `cn=admin,dc=intnet`. Iremos popular esta base brevemente.

3. Agora, vamos instalar o `nslcd`, um *daemon* LDAP local para resolução de diretivas de autenticação. Execute:

```
# apt-get install nslcd
```

Durante a instalação do pacote, informe as seguintes opções:

Tabela 2. Configurações do pacote nslcd

Pergunta	Opção
URI do servidor LDAP	ldapi:///
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

Note que informamos que a localização do servidor LDAP é local, já que ele está instalado na máquina corrente. Além disso, iremos configurar as bases de contas (`passwd`), grupos (`group`) e senhas (`shadow`) junto ao LDAP.

4. Vamos configurar as opções padrão dos binários de linha de comando do `ldap-utils` (como os comandos `ldapsearch` e `ldapadd`, por exemplo). Edite manualmente ou use o `sed` para editar as linhas apropriadas do arquivo `/etc/ldap/ldap.conf`:

```
# sed -i 's/^#\(BASE\).*/\1 dc=intnet/' /etc/ldap/ldap.conf
```

```
# sed -i 's/^#\(URI\).*/\1 ldapi\:\/\/\//' /etc/ldap/ldap.conf
```

Verifique que os valores editados estão corretos:

```
# grep -v '^#' /etc/ldap/ldap.conf | sed '/^$/d'
BASE dc=intnet
URI ldapi:///
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

5. A seguir, vamos configurar o `ldapscripts`, um conjunto de ferramentas auxiliares que facilitam enormemente a configuração e uso de bases LDAP via linha de comando. Primeiro, edite manualmente ou use o `sed` para ajustar o arquivo `/etc/ldapscripts/ldapscripts.conf`, informando o *bind DN* do usuário administrativo na base LDAP, como se segue:

   ```
   # sed -i 's/^\(BINDDN=\).*/\1\"cn=admin,dc=intnet\"/'
   /etc/ldapscripts/ldapscripts.conf
   ```

   Depois, informe a senha do usuário administrativo no arquivo `/etc/ldapscripts/ldapscripts.passwd`. Execute:

   ```
   # echo -n "rnpesr" > /etc/ldapscripts/ldapscripts.passwd
   ```

   Como a senha do usuário administrativo do LDAP está embutida em texto claro nesse arquivo, é fundamental garantir que suas permissões estão suficientemente estritas. Verifique:

   ```
   # ls -ld /etc/ldapscripts/ldapscripts.passwd
   -rw-r----- 1 root root 6 out 29 09:16 /etc/ldapscripts/ldapscripts.passwd
   ```

6. Vamos inicializar a base LDAP usando o `ldapscripts`. Execute:

   ```
   # ldapinit -s
   ```

7. Será que funcionou? Consulte a base LDAP sem autenticação, e liste seu conteúdo:

```
# ldapsearch -x -LLL
dn: dc=intnet
objectClass: top
objectClass: dcObject
objectClass: organization
o: seg10
dc: intnet

dn: cn=admin,dc=intnet
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

dn: ou=People,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Groups

dn: ou=Hosts,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Hosts

dn: ou=Idmap,dc=intnet
objectClass: organizationalUnit
ou: Idmap
```

Perfeito! Temos configurada a raiz `dc=intnet` e usuário administrativo `cn=admin,dc=intnet` como anteriormente, e o comando `ldapinit` se encarregou de criar DNs para armazenar usuários, grupos, *hosts* e mapeamentos de identidade na base LDAP. Podemos prosseguir.

3) Habilitando logs do LDAP

Por padrão, o *daemon* `slapd` envia logs para a *facility* `local4` do sistema. Porém, após a instalação via `apt-get`, seu *log level* (nível de criticidade dos eventos registrados) é configurado para `none` — ou seja, nenhum evento é registrado. Evidentemente, essa situação não é interessante ao configurar o servidor, pois será muito difícil identificar as fontes de problemas se não tivermos nenhum log para consultar. Vamos corrigir isso.

1. Primeiro, crie um diretório específico para o armazenamento de LDIFs. LDIFs são uma sigla para LDAP *Data Interchange Format*, arquivos de texto plano que podem ser usados para representar informações em uma base LDAP. Algo equivalente a arquivos *dump* de bases SQL,

em comparação livre.

```
# mkdir /root/ldif
```

2. Crie neste diretório um LDIF de nome `/root/ldif/slapdlog.ldif`, com o seguinte conteúdo:

```
1 dn: cn=config
2 changeType: modify
3 replace: olcLogLevel
4 olcLogLevel: stats
```

O LDIF acima irá alterar o valor do parâmetro `olcLogLevel`, que define o *log level* do `slapd`, para `stats`. Para aplicar essa configuração, execute:

```
# ldapmodify -Y external -H ldapi:/// -f /root/ldif/slapdlog.ldif
```

3. O próximo passo é informar ao `rsyslog` que as mensagens enviadas para a *facility* `local4` devem ser enviadas para um arquivo específico. Crie o arquivo novo `/etc/rsyslog.d/slapd.conf` com o seguinte conteúdo:

```
1 $template slapdtmpl,"[%$DAY%-%$MONTH%-%$YEAR% %timegenerated:12:19:date-rfc3339%]
%app-name% %syslogseverity-text% %msg%\n"
2 local4.*    /var/log/slapd.log;slapdtmpl
```

4. Em seguida, reinicie ambos os *daemons* do `rsyslog` e do `slapd`:

```
# systemctl restart rsyslog.service
```

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

5. Verifique que os registros de eventos do `slapd` estão sendo, de fato, enviados para o arquivo `/var/log/slapd.log`:

```
tail /var/log/slapd.log
[13-11-2018 18:35:18] slapd debug  daemon: shutdown requested and initiated.
[13-11-2018 18:35:18] slapd debug  slapd shutdown: waiting for 0 operations/tasks
to finish
[13-11-2018 18:35:18] slapd debug  slapd stopped.
[13-11-2018 18:35:18] slapd debug  @(#) $OpenLDAP: slapd  (May 23 2018 04:25:19)
$#012#011Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>
[13-11-2018 18:35:18] slapd debug  slapd starting
```

6. A rotação de logs deve ser habilitada, caso contrário os arquivos de log poderão ficar excessivamente grandes. Crie o arquivo novo `/etc/logrotate.d/slapd`, com o seguinte conteúdo:

```
/var/log/slapd.log {
  missingok
  notifempty
  compress
  daily
  rotate 30
  sharedscripts
  postrotate
    systemctl restart rsyslog.service
  endscript
}
```

Como o `logrotate` é invocado via `cron`, não é necessário reiniciar nenhum serviço neste caso.

4) Edição de índices e permissões no LDAP

1. Para maior performance durante as consultas ao LDAP, é recomendável aumentar os parâmetros de indexação de alguns atributos. Crie o arquivo `/root/ldif/olcDbIndex.ldif`, com o seguinte conteúdo:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
replace: olcDbIndex
olcDbIndex: objectClass eq
olcDbIndex: cn pres,sub,eq
olcDbIndex: sn pres,sub,eq
olcDbIndex: uid pres,sub,eq
olcDbIndex: displayName pres,sub,eq
olcDbIndex: default sub
olcDbIndex: uidNumber eq
olcDbIndex: gidNumber eq
olcDbIndex: mail,givenName eq,subinitial
olcDbIndex: dc eq
```

O LDIF acima irá alterar o valor do parâmetro `olcDbIndex`, que define quais parâmetros serão indexados pelo `slapd` e quais tipos de busca serão suportados. Para aplicar essa configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcDbIndex.ldif
```

2. É interessante permitir que usuários possam alterar seus parâmetros `loginShell` e entrada `gecos` (informações de conta do usuário) via comandos `chsh` e `chfn`. Para fazer isso, crie o arquivo novo `/root/ldif/olcAccess.ldif` com o seguinte conteúdo:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
add: olcAccess
olcAccess: {1}to attrs=loginShell,gecos
  by dn="cn=admin,dc=intnet" write
  by self write
  by * read
```

Para aplicar a configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcAccess.ldif
```

5) Adição de grupos e usuários no LDAP

Agora sim, vamos começar a criar usuários e grupos em nossa base LDAP. Imagine que iremos criar um grupo de nome `sysadm`, no qual estarão todos os administradores de sistema que têm permissão para acessar servidores não-críticos. Nesse grupo, iremos criar o usuário `luke`, com senha `seg10luke`. Como proceder?

1. Primeiro, crie o grupo usando o comando `ldapaddgroup`:

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

   Verifique que o grupo foi corretamente criado usando o `ldapsearch`:

   ```
   # ldapsearch -x -LLL 'cn=sysadm'
   dn: cn=sysadm,ou=Groups,dc=intnet
   objectClass: posixGroup
   cn: sysadm
   gidNumber: 10000
   description: Group account
   ```

2. A seguir, crie o usuário `luke`, informando seu grupo primário como `sysadm`:

   ```
   # ldapadduser luke sysadm
   Successfully added user luke to LDAP
   Successfully set password for user luke
   ```

   Novamente, consulte o `ldapsearch` para checar se o usuário foi criado com sucesso:

```
# ldapsearch -x -LLL 'cn=luke'
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
```

O comando `ldapid` também é uma boa opção neste cenário, fornecendo saída bastante similar à do comando `id`:

```
# ldapid luke
uid=10000(luke) gid=10000(sysadm) groups=10000(sysadm),10000(sysadm)
```

3. Vamos configurar a senha do usuário `luke` como `seg10luke`:

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

Para verificar, cheque que o campo `userPassword` do usuário está preenchido com um *hash* de senha. Para o comando `ldapsearch` funcionar, temos que informar um *bind DN* administrativo e senha correspondente, já que este atributo não é legível sem autenticação:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'cn=luke' userPassword
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
userPassword:: e1NTSEF9NHdUSWZRcUhGR0o5VU5jNS9tVnhoaGJzNFVvNkFzMmE=
```

O comando `ldapfinger` também é uma boa opção para consultar as informações do usuário (e seu *hash* de senha), já que faz parte do `ldapscripts` e utiliza as credenciais administrativas que configuramos anteriormente:

```
# ldapfinger luke
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
userPassword:: e1NTSEF9NEU0aFI2N3lCNGplUHdXRHNHVEZYZTBEYm5YdGxDTUg=
```

4. Apesar de termos informado o grupo `sysadm` como grupo primário do usuário `luke`, do ponto de vista do grupo esse usuário ainda não o integra. O comando `ldapaddusertogroup` faz esse trabalho:

   ```
   # ldapaddusertogroup luke sysadm
   Successfully added user luke to group cn=sysadm,ou=Groups,dc=intnet
   ```

   Para verificar o pertencimento, use o comando `ldapsearch`, consultando os atributos `memberUid`:

   ```
   # ldapsearch -x -LLL 'cn=sysadm'
   dn: cn=sysadm,ou=Groups,dc=intnet
   objectClass: posixGroup
   cn: sysadm
   gidNumber: 10000
   description: Group account
   memberUid: luke
   ```

   O `ldapgid`, parte da suíte `ldapscripts`, também é uma alternativa interessante:

   ```
   # ldapgid sysadm
   gid=10000(sysadm) users(primary)=10000(luke) users(secondary)=10000(luke)
   ```

5. Como já mencionado algumas vezes, o `ldapscripts` oferece um conjunto de programas que facilitam bastante as tarefas corriqueiras de manipulação da base LDAP via linha de comando. Apesar de termos trabalhado um bom número desses programas, listamos abaixo alguns outros que não foram utilizados. Consulte suas páginas de manual para mais informações sobre como operá-los:

   - Deleção de entradas:
     - `/usr/sbin/ldapdeletegroup`
     - `/usr/sbin/ldapdeleteuser`

    - `/usr/sbin/ldapdeleteuserfromgroup`
  - Modificação de entradas:
    - `/usr/sbin/ldapmodifygroup`
    - `/usr/sbin/ldapmodifyuser`
  - Renomear entradas:
    - `/usr/sbin/ldaprenamegroup`
    - `/usr/sbin/ldaprenameuser`
  - Configurar grupo primário de um usuário:
    - `/usr/sbin/ldapsetprimarygroup`

6) Integração e teste do sistema de autenticação com LDAP

Agora, vamos integrar o sistema de autenticação do Linux com a base LDAP que configuramos anteriormente usando o PAM (*Pluggable Authentication Modules*). Felizmente, muito do trabalho de configuração já foi feito automaticamente pelos *scripts* de instalação do `apt-get` quando instalamos o pacote `nslcd`. Faltam apenas alguns poucos passos.

1. Quando um usuário do LDAP efetua login em uma máquina pela primeira vez, seu diretório *home* não existe, naturalmente. Vamos configurar o PAM para criar esse diretório automaticamente. Crie o arquivo novo `/usr/share/pam-configs/mkhomedir` com o seguinte conteúdo:

   ```
   Name: Create home directory during login
   Default: yes
   Priority: 900
   Session-Type: Additional
   Session:
           required        pam_mkhomedir.so umask=0022 skel=/etc/skel
   ```

   Em seguida, execute:

   ```
   # pam-auth-update
   ```

   Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

   Verifique que a configuração surtiu efeito pesquisando pelo termo `mkhomedir` nos arquivos de configuração do PAM, em `/etc/pam.d`:

```
# grep -ri mkhomedir /etc/pam.d
/etc/pam.d/common-session:session       required        pam_mkhomedir.so umask=0022
skel=/etc/skel
/etc/pam.d/common-session-noninteractive:session        required
pam_mkhomedir.so umask=0022 skel=/etc/skel
```

2. Reinicie os *daemons* `nslcd` e `nscd` para que a *cache* de usuários, grupos e senhas do LDAP seja atualizada:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

3. Será que funcionou? Pesquise a lista de usuários do sistema e busque pelo usuário recém-criado `luke`:

```
# getent passwd | grep luke
luke:*:10000:10000:luke:/home/luke:/bin/bash
```

Excelente! E quando ao grupo `sysadm`?

```
# getent group | grep sysadm
sysadm:*:10000:luke
```

Finalmente, consulte se o usuário `luke` é reconhecido como membro de `sysadm` pelo sistema:

```
# groups luke
luke : sysadm
```

4. Vamos testar: tente logar via `ssh` na máquina local usando o usuário `luke`:

```
# ssh luke@localhost
The authenticity of host 'localhost (::1)' can't be established.
ECDSA key fingerprint is SHA256:mI2LM9Nxt5ltC7/VnlEgB+JBdqFbKxUj1FhhvLijcv4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
luke@localhost's password:
Creating directory '/home/luke'.
```

Perfeito. Note que o diretório `/home/luke` foi criado automaticamente, como esperado. Faça as verificações pós-login de costume:

```
$ whoami
luke
```

```
$ pwd
/home/luke
```

```
$ id
uid=10000(luke) gid=10000(sysadm) grupos=10000(sysadm)
```

5. Falta testar se o usuário consegue alterar sua senha diretamente via console, sem necessidade de edição direta à base LDAP. Primeiro, verifique o *hash* da senha atual — note que iremos usar o próprio DN do usuário `luke` como *bind DN* durante a conexão LDAP, motivo pelo qual deve-se informar a senha desse usuário, e não do administrador:

   ```
   $ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
   Enter LDAP Password:
   dn: uid=luke,ou=People,dc=intnet
   userPassword:: e1NTSEF9K29BcE55S3AwRU9XM0sreWVPeFNoZUJjdFhBbVJyVEg=
   ```

   Use o comando `passwd` para alterar a senha para um outro valor qualquer:

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

   Verifique novamente o *hash* da senha do usuário `luke` e compare os dois valores:

   ```
   $ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
   Enter LDAP Password:
   dn: uid=luke,ou=People,dc=intnet
   userPassword:: e1NTSEF9b0REb2lVbWgvR0swMm9qaGJoZWJ2ZGtNYzFKTE1kazk=
   ```

   Perfeito, os *hashes* são diferentes; ou seja, a alteração de senha via `passwd` funcionou normalmente. Retorne a senha do usuário `luke` ao valor anterior, para evitar confusões no futuro.

7) Configurando uma autoridade certificadora (CA) para o SSH

Até o momento, instalamos e configuramos uma base LDAP, e testamos sua integração com os sistemas de autenticação do sistema usando o `ssh`. Porém, a todo momento, tivemos que digitar as senhas dos usuários para nos autenticar — será que podemos fazer isso de uma forma mais segura?

Nesta atividade iremos configurar uma autoridade certificadora para o `ssh`, com a qual assinaremos pares de chaves de *hosts* e de usuários. De posse dessas chaves, não será mais necessário informar senhas durante o login, tornando o processo significativamente mais seguro (desde que se tome cuidado para não perder a chave privada, como veremos).

1. Primeiro, verifique que você está logado como usuário `root` na máquina `ns2`:

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

2. Vamos adicionar um novo usuário à base LDAP, `sshca`, que será responsável por realizar os processos de assinaturas de chaves de *hosts* e usuários. Esse usuário irá pertencer ao grupo `setup`, um grupo especial para atividades de configuração de sistemas. Sua senha será `seg10sshca`.

   Vamos por partes. Primeiro, crie o grupo:

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

   Em seguida, o usuário:

   ```
   # ldapadduser sshca setup
   Successfully added user sshca to LDAP
   Successfully set password for user sshca
   ```

   Adicione o usuário ao grupo:

   ```
   # ldapaddusertogroup sshca setup
   Successfully added user sshca to group cn=setup,ou=Groups,dc=intnet
   ```

   E, finalmente, configure a senha do usuário `sshca`:

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

3. Faça login com o usuário recém-criado no sistema local:

```
# ssh sshca@localhost
sshca@localhost's password:
Creating directory '/home/sshca'.
```

```
$ whoami
sshca
```

```
$ pwd
/home/sshca
```

4. Vamos criar dois pares de chaves para a CA (*Certificate Authority*, ou autoridade certificadora) do `ssh`: uma para assinar chaves de *hosts*, denominada `server_ca`, e outra para assinar chaves de usuários, denominada `user_ca`. Iremos criar chaves RSA de 4096 bits, e devemos escolher uma senha bastante segura para a chave privada — já que, com ela, pode-se assinar quaisquer chaves `ssh` que autorizarão máquinas a se passarem por membros do nosso *datacenter* e usuários a logarem em qualquer servidor integrado.

   Como estamos em um ambiente de laboratório, não iremos utilizar senhas para as chaves privadas de modo a agilizar a execução das atividades.

   Para criar a chave de assinatura de *hosts*, `server_ca`, execute:

```
$ ssh-keygen -f server_ca -t rsa -b 4096 -N ''
Generating public/private rsa key pair.
Your identification has been saved in server_ca.
Your public key has been saved in server_ca.pub.
The key fingerprint is:
SHA256:op2g5J5DN/1pPy8RUHxnAdh51NHP/2Pmyo3Vq5Qlvjo sshca@ns2
The key's randomart image is:
+---[RSA 4096]----+
|         o.o.+o+o|
|        . o + + o|
|         . . + ..|
|          .     o|
|  . ... S  . . ..|
| o..o+.o  . . + o|
| .o...o. . . + .o|
| ...    + oEo =+o|
|  o.   . ..=+**+.|
+----[SHA256]-----+
```

Verifique que o par de chaves foi criado com sucesso:

```
$ ls
server_ca  server_ca.pub
```

Para criar a chave de assinatura de usuários, `user_ca`, execute:

```
$ ssh-keygen -f user_ca -t rsa -b 4096 -N ''
Generating public/private rsa key pair.
Your identification has been saved in user_ca.
Your public key has been saved in user_ca.pub.
The key fingerprint is:
SHA256:T6dFFMQiJnbGCLg0rj1tzaT4mfqagBn80YQaBGuWpuM sshca@ns2
The key's randomart image is:
+---[RSA 4096]----+
|o. ... o   o+.   |
|..= . + * ...    |
|.O + o = . ..    |
|* = o .    .     |
|+= + *  S . o    |
|++= = o  o +     |
|+E = o    o      |
| . .+            |
|  ++.            |
+----[SHA256]-----+
```

Cheque que todos os quatro arquivos de chaves pública/privada foram criados:

```
$ ls
server_ca  server_ca.pub  user_ca  user_ca.pub
```

5. Dada a grande sensibilidade das chaves privadas das CAs nesse sistema de autenticação que estamos configurando, é fundamental garantir também que, além de terem uma senha forte configurada, suas permissões estejam corretamente ajustadas.

   Verifique as permissões das chaves usando o comando `ls`:

```
$ ls -l *_ca*
-rw------- 1 sshca setup 1766 out 28 08:39 server_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:39 server_ca.pub
-rw------- 1 sshca setup 1766 out 28 08:40 user_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:40 user_ca.pub
```

8) Configurando a SSH-CA no servidor LDAP

1. Vamos configurar o servidor LDAP para interoperar com a CA `ssh` que configuramos na atividade anterior. Volte ao usuário `root` na máquina `ns2`:

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

   Crie um novo arquivo, `/root/scripts/sshsign.sh` com o seguinte conteúdo:

```
#!/bin/bash

CA_USER="sshca"
CA_ADDR="10.0.42.2"
CA_USER_PASS="seg10sshca"
SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"


function cleanup() {
  sed -i '/^HostCertificate/d' /etc/ssh/sshd_config
  sed -i '/^TrustedUserCAKeys/d' /etc/ssh/sshd_config

  rm -f ~/.ssh/known_hosts       \
    /etc/ssh/ssh_host_*      \
    /etc/ssh/ssh_known_hosts \
    /etc/ssh/user_ca.pub 2> /dev/null
  dpkg-reconfigure openssh-server &> /dev/null

  systemctl restart sshd.service
```

```
}


# resetar sistema para estado conhecido
cleanup

# escanear chave do host ssh-ca
[ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
if ! ssh-keygen -F ${CA_ADDR} 2>/dev/null 1>/dev/null; then
  ssh-keyscan -t rsa -T 10 ${CA_ADDR} 2> /dev/null >> ~/.ssh/known_hosts
fi

# iterar em todas as pubkeys SSH
for pkeypath in /etc/ssh/ssh_host_*.pub; do
  pkeyname="$( echo "${pkeypath}" | awk -F'/' '{print $NF}' )"
  certname="$( echo "${pkeyname}" | sed 's/\(\.pub$\)/-cert\1 /' )"
  echo "Signing ${pkeyname} key..."

  # copiar pubkey
  sshpass -p "${CA_USER_PASS}" \
    scp ${SSH_OPTS} ${pkeypath} ${CA_USER}@${CA_ADDR}:~

  # assinar pubkey, validade [-5 min -> 3 anos]
  identity="$(hostname --fqdn)"
  principals="$(hostname),$(hostname --fqdn),$(hostname -I | tr ' ' ',' | sed
's/,$//')"
  sshpass -p "${CA_USER_PASS}" \
    ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
      ssh-keygen -s server_ca \
      -I "${identity}" \
      -n "${principals}" \
      -V -5m:+1095d \
      -h \
      ${pkeyname} 2> /dev/null

  # copiar pubkey assinada de volta
  sshpass -p "${CA_USER_PASS}" \
    scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:${certname} /etc/ssh/

  # remover temporarios do diretorio remoto
  sshpass -p "${CA_USER_PASS}" \
    ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
      rm ${pkeyname} ${certname}

  # remover pubkey RSA antiga e configurar ssh para apresentar pubkey assinada
  rm -f ${pkeypath} 2> /dev/null
  echo "HostCertificate /etc/ssh/${certname}" >> /etc/ssh/sshd_config
done

# copiar pubkey da server_ca e configurar reconhecimento de chaves de host
assinadas
```

```
69 echo "Configuring host key trust..."
70 echo "@cert-authority * $(sshpass -p "${CA_USER_PASS}" ssh ${SSH_OPTS} ${
CA_USER}@${CA_ADDR} cat server_ca.pub)" > /etc/ssh/ssh_known_hosts
71
72 # copiar pubkey da user_ca e configurar reconhecimento de chaves de usuario
assinadas
73 echo "Configuring user key trust..."
74 sshpass -p "${CA_USER_PASS}" \
75   scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:~/user_ca.pub /etc/ssh/
76 echo "TrustedUserCAKeys /etc/ssh/user_ca.pub" >> /etc/ssh/sshd_config
77
78 echo "All done!"
79 systemctl restart sshd.service
```

Você deve estar se perguntando: o que esse *script* faz? Vamos ver:

1. (Linhas 3-6) Definimos o usuário `sshca` e o IP `10.0.42.2` (o servidor `ns2` local no qual estamos logados no momento) como a origem das chaves da CA que utilizaremos a seguir. Configuramos a senha do usuário `sshca` de forma *hardcoded* para agilizar o processo de execução do *script* e informamos que os logins SSH serão feitos sempre via senha, não chaves assimétricas.
2. (Linhas 9-20) Função que reseta a situação do SSH no servidor para um estado conhecido, em caso de falha ou encerramento abrupto do usuário durante a execução do *script*.
3. (Linhas 27-30) Escaneamos a chave do servidor `10.0.42.2` e armazenamos no arquivo `~/.ssh/known_hosts`, evitando que o usuário tenha que confirmar que confia no *host* remoto antes de logar.
4. (Linhas 33-35) Iteramos sobre todas as chaves públicas no diretório `/etc/ssh` (RSA, ECDSA e ED25519), extraindo *strings* para usar à frente.
5. (Linhas 39-40) Copiamos a chave pública do *host* sendo processadao pelo *loop* para o servidor da CA.
6. (Linhas 43-52) Assinamos a chave pública do *host* sendo processadao pelo *loop* usando a chave `server_ca`, com validade de 3 anos.
7. (Linhas 55-56) Copiamos a chave pública assinada sendo processadao pelo *loop* de volta para a máquina local.
8. (Linhas 59-61) Removemos chaves pública não-assinada e assinada sendo processadao pelo *loop* da pasta do usuário `sshca` no servidor `10.0.42.2`, para evitar confusão em assinaturas futuras.
9. (Linhas 64-65) Removemos a chave pública sendo processadao pelo *loop* não-assinada e mantemos apenas a assinada, configurando o servidor `ssh` para apresentá-la para clientes.
10. (Linha 70) Configuramos a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, quaisquer logins de cliente `ssh` da máquina local para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.
11. (Linhas 74-76) Copiamos a chave `user_ca.pub` e a configuramos como uma CA confiável para usuários; a partir deste momento, qualquer usuário que apresente uma chave assinada pela CA terá seu login autorizado sem necessitar digitação de senha.

12. (Linha 79) Reiniciamos o servidor `ssh` para aplicar as configurações realizadas.

2. Vamos instalar o `sshpass`, dependência para que o *script* acima funcione corretamente.

```
# apt-get install sshpass
```

3. Vamos configurar a máquina `ns2` para operar com a CA do `ssh`: execute o *script* criado no passo (1).

```
# bash ~/scripts/sshsign.sh
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!
```

Note que não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele autodetecta o *hostname* e endereços da máquina local e os configura como *principals* (conjunto de endereços/*hostnames* válidos para uma determinada chave; linha 44 do *script*).

4. Vamos verificar que o *script* fez seu trabalho corretamente. Cheque as linhas finais do arquivo `/etc/ssh/sshd_config`:

```
# tail -n4 /etc/ssh/sshd_config
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
TrustedUserCAKeys /etc/ssh/user_ca.pub
```

Verifique que apenas chaves públicas assinadas existem no diretório `/etc/ssh`:

```
# ls -1 /etc/ssh/ssh_host_*.pub
/etc/ssh/ssh_host_ecdsa_key-cert.pub
/etc/ssh/ssh_host_ed25519_key-cert.pub
/etc/ssh/ssh_host_rsa_key-cert.pub
```

Compare o conteúdo dos arquivos de chave pública `/home/sshca/server_ca.pub` e de chave da CA confiável `/etc/ssh/ssh_known_hosts` — eles devem ser iguais:

```
# cat /etc/ssh/ssh_known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiun8c
K7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

```
# cat /home/sshca/server_ca.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiun8c
K7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

Observer, finalmente, que os arquivos `/home/sshca/user_ca.pub` e `/etc/ssh/user_ca.pub` também devem ser idênticos:

```
# diff /home/sshca/user_ca.pub /etc/ssh/user_ca.pub
```

Note que tanto no caso de assinatura de chaves de *host* como de chaves de usuário, futuramente, estamos fazendo o acesso no sentido **máquina remota → servidor da CA**, que não é o ideal. Em produção, o correto seria tornar o acesso ao servidor da CA o mais controlado possível, e fazer as assinaturas de chaves apenas de forma local, ou com acessos no sentido **servidor da CA → máquina remota**.

Outro aspecto relevante de segurança que sacrificamos em nosso cenário para agilizar a execução das atividades foi a configuração das chaves privadas de assinatura de *hosts* e usuários sem senha: em produção, é altamente recomendado que essas chaves sejam criadas com senhas fortes, para mitigar a possibilidade de assinatura indevida de chaves em caso de vazamento das mesmas.

9) Automatizando a assinatura de chaves SSH de usuários

1. Vamos agora fazer a segunda "perna" da configuração da CA `ssh` — assinar chaves de usuários. Na linha da atividade anterior, iremos usar um *script* para assinar chaves de usuários; porém, como o cenário de chaves de usuário é mais flexível que o de máquinas, iremos estabelecer algumas premissas para o funcionamento do *script*:

   1. Deve-se estar logado com o mesmo nome do usuário com o qual se deseja assinar a chave.
   2. Deve-se ter conectividade com o servidor `ns2`, no endereço `10.0.42.2`.
   3. O nome de chave será fixo (`~/.ssh/id_rsa`).

   Devido à primeira limitação, é conveniente que o *script* esteja localizado dentro da pasta do usuário — e, como se sabe, ao criar novos usuários o conteúdo da pasta `/etc/skel` é copiado para dentro de seu *home*. Assim, crie a pasta `/etc/skel/scripts`:

   ```
   # mkdir /etc/skel/scripts
   ```

   Dentro dela, crie o arquivo novo, `/etc/skel/scripts/sshsign_user.sh`, com o conteúdo que se segue:

   ```
    1 #!/bin/bash
    2
    3 CA_USER="sshca"
    4 CA_ADDR="10.0.42.2"
    5 CA_USER_PASS="seg10sshca"
    6 SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"
    7
    8 # testar se chave ja foi assinada
    9 if [ -f ~/.ssh/id_rsa-cert.pub ]; then
   10   echo "Key already signed, bailing."
   11   exit 1
   12 fi
   13
   14 # escanear chave do host ssh-ca, se necessario
   15 rm -f ~/.ssh/known_hosts 2> /dev/null
   16 [ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
   17 if ! ssh-keygen -F ${CA_ADDR} 2>/dev/null 1>/dev/null; then
   18   ssh-keyscan -t rsa -T 10 ${CA_ADDR} 2> /dev/null >> ~/.ssh/known_hosts
   19 fi
   20
   21 # gerar par de chaves RSA, se inexistentes
   22 [ -f ~/.ssh/id_rsa.pub ] || ssh-keygen -f ~/.ssh/id_rsa -t rsa -b 4096 -N '' &>
      /dev/null
   23
   24 # copiar pubkey RSA
   25 sshpass -p "${CA_USER_PASS}" \
   ```

```
26   scp ${SSH_OPTS} ~/.ssh/id_rsa.pub ${CA_USER}@${CA_ADDR}:~
27
28 # assinar pubkey RSA, validade [-5 min -> 1 ano]
29 echo "Signing ~/.ssh/id_rsa.pub key..."
30 user="$( whoami )"
31 sshpass -p "${CA_USER_PASS}" \
32   ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
33     ssh-keygen -s user_ca \
34     -I ${user} \
35     -n ${user} \
36     -V -5m:+1095d \
37     id_rsa.pub 2> /dev/null
38
39 # copiar pubkey assinada de volta
40 sshpass -p "${CA_USER_PASS}" \
41   scp ${SSH_OPTS} ${CA_USER}@${CA_ADDR}:~/id_rsa-cert.pub ~/.ssh/
42
43 # remover temporarios do diretorio remoto
44 sshpass -p "${CA_USER_PASS}" \
45   ssh ${SSH_OPTS} ${CA_USER}@${CA_ADDR} \
46     rm id_rsa.pub id_rsa-cert.pub
47
48 # copiar pubkey da server_ca e configurar reconhecimento de chaves de host
assinadas
49 echo "@cert-authority * $(sshpass -p "${CA_USER_PASS}" ssh ${SSH_OPTS} ${
CA_USER}@${CA_ADDR} cat server_ca.pub)" > ~/.ssh/known_hosts
50
51 # remover pubkey RSA antiga
52 rm -f ~/.ssh/id_rsa.pub 2> /dev/null
53
54 echo "All done!"
```

Como o *script* guarda grandes semelhanças com o anterior, iremos destacar apenas os pontos de divergência:

1. (Linhas 9-12) Testamos se a chave RSA do usuário já foi assinada anteriormente; se positivo, o programa se encerra.
2. (Linha 22) Caso o usuário não possua um par de chaves criado, cria-se automaticamente um par RSA de 4096 bits, sem senha.
3. (Linhas 30-37) A chave usada para assinar a chave pública do usuário desta vez é a `user_ca`. A validade é ajustada para um ano.
4. (Linha 49) De forma similar ao que foi feito anteriormente, copia-se a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, logins de cliente `ssh` deste usuário para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.

2. Vamos testar o funcionamento do *script* com o usuário `luke`. Remova o diretório dele para garantir que o conteúdo do `/etc/skel` será copiado no próximo login:

```
# rm -rf /home/luke
```

Agora, logue como o usuário `luke` usando os comandos `su` ou `ssh`:

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

```
$ ls ~/scripts
sshsign_user.sh
```

3. Execute o *script*:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

Note que, novamente, não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele utiliza o *username* do usuário informado pelo comando `whoami` como *principal* da chave.

4. Vamos verificar o funcionamento do *script*. Verifique que os arquivos de chave foram criados:

```
$ ls ~/.ssh/
id_rsa  id_rsa-cert.pub  known_hosts
```

Cheque o conteúdo do arquivo `~/.ssh/known_hosts`, que deve conter informações da CA `ssh` para chaves de *host* assinadas:

```
$ cat ~/.ssh/known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQC+ZP47AOiRPcakZiLm+szvxWF5HKNa+BauaYXt5A6XjuUUJIYXXdT
dzrq+d4rx379jxk5E+RHvPls1Y+5Cvn2/EpBgx20TXoT9+O0Nn7pemHd7qkDb2JqPzoHzjViG033L8UociI
tit13UVMlvB2+miW4RPMHhLjJpJT2BoWzvkBtJdukO/SA+3EYpIZAj8kyFfkp9+2A/A+OCREWoh5EkcjREA
QRay+pc/j3sWDYrymu65OamPwGAe3Ih9/TmflUiwdHaNfJMY0BqpJsdFyUJeS36asjRhLtZltfLNTyY1gOq
3XglLZb76YaQWOoIhzwqQ2WNSt72cY7s8p97loX5LTBSjNw6Kq0rOpKY3XmkyP2A5+L+CFxRxpl0obOkqiZ
qK/oT4cFE72EBaSg0XfFej19rd/AqqFSEHpbK4GjZzBAEID932DrgcR6ud74k1TDemQgWZ5dge0wEKMfeNV
zUXhAPuxeIyQ6E+DxuayHIrUKZ7T36ZVz5N56TXnr+iaaejqpJfuSKZxC8YT9ZmRiHeZ+edze+R6QPiV76Q
UIQqzocO+OlWPHUZ2zVINV76DZARqxzZuKWW7JzA1+kTJ3VW3zGaOs53n36lfThb39ULHplsJUPfUGiunhi
story8cK7onzIbkRibbVu/kmrNG8nr2Z4GHMGpQ6KvYyGZNFIwioGMu6Q== sshca@ns2
```

5. Agora sim, vamos testar. Tente logar na máquina local usando o endereço IP ou *hostname* (o endereço especial *localhost* não é registrado como um *principal* válido na chave de *host*).

```
$ ssh luke@ns2
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Tue Nov 13 18:40:31 2018 from ::1
luke@ns2:~$
```

Perfeito! Não tivemos que digitar a senha do usuário ou confirmar a relação de confiança com o servidor, como esperado.

6. Todos as características do sistema que queríamos testar estão funcionais. Claro que devemos levar em consideração que todos os testes foram feitos dentro da máquina `ns2` — não testamos o funcionamento de contas de usuário via LDAP na rede, ou a autenticação de login remoto usando a CA do `ssh`.

   Retorne ao usuário `root` na máquina `ns2` e remova o diretório do usuário `luke`. Vamos prosseguir com a configuração do *template* e de um cliente Linux para testar as funcionalidades.

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

   ```
   # rm -rf /home/luke
   ```

10) Configurando o template para funcionar com LDAP/SSH-CA

1. Como mencionado anteriormente, iremos configurar a VM `debian-template` para funcionar com os sistemas de autenticação do LDAP e SSH-CA. Assim, todas as máquinas que forem derivadas futuramente desse *template* estarão automaticamente integradas com o sistema de autenticação do nosso *datacenter* hipotético.

   Ligue a VM `debian-template` e faça login como o usuário `root`:

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

2. Crie o arquivo novo `/root/scripts/sshsign.sh`, e cole dentro dele o conteúdo do *script* que discutimos no passo (1) da atividade (8).

   ```
   # nano /root/scripts/sshsign.sh
   (...)
   ```

```
# ls /root/scripts/
changehost.sh  sshsign.sh  syncdirs.sh
```

3. Agora, crie o diretório `/etc/skel/scripts`, e dentro dele crie o arquivo novo `/etc/skel/scripts/sshsign_user.sh`, com conteúdo idêntico ao do *script* que foi apresentado no passo (1) da atividade (9).

```
# mkdir /etc/skel/scripts
```

```
# nano /etc/skel/scripts/sshsign_user.sh
(...)
```

```
# ls /etc/skel/scripts/
sshsign_user.sh
```

4. Instale as dependências para funcionamento dos *scripts* criados anteriormente e também para integração do sistema de autenticação PAM com o LDAP.

```
# apt-get install sshpass nslcd
```

Durante a instalação do pacote `nslcd`, informe as seguintes opções:

Tabela 3. Configurações do pacote nslcd

Pergunta	**Opção**
URI do servidor LDAP	ldap://ldap.intnet/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

5. Assim como configurado antes, informe ao PAM que diretórios *home* inexistentes de usuários do LDAP devem ser criados automaticamente. Crie o arquivo novo `/usr/share/pam-configs/mkhomedir`, e cole dentro dele o conteúdo do arquivo discutido durante o passo (1) da atividade (6).

```
# nano /usr/share/pam-configs/mkhomedir
(...)
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

6. Finalmente, vamos alterar o *script* `/root/scripts/changehost.sh`, criado durante a sessão (1) deste curso, para invocar automaticamente o *script* `/root/scripts/sshsign.sh` ao final e reiniciar os *daemons* do `nslcd` e `nscd`. Altere o conteúdo deste arquivo para:

```
#!/bin/bash


# exibir uso do script e sair
function usage() {
  echo "  Usage: $0 -h HOSTNAME -i IPADDR -g GATEWAY"
  echo "  Netmask is assumed as /24."
  exit 1
}


# testar sintaxe valida de HOSTNAME
function valid_host() {
  if [[ "$nhost" =~ [^a-z0-9-] ]]; then
    echo "  [*] HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
    usage
  elif [ ${#nhost} -gt 63 ]; then
    echo "  [*] HOSTNAME must have <63 chars"
    usage
  fi
}


# testar sintaxe valida de IPADDR/GATEWAY
function valid_ip() {
  local  ip=$1
  local  stat=1

  if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
    OIFS=$IFS
    IFS='.'
    ip=($ip)
    IFS=$OIFS
    [[ ${ip[0]} -le 255 && \
       ${ip[1]} -le 255 && \
       ${ip[2]} -le 255 && \
       ${ip[3]} -le 255 ]]
        stat=$?
  fi

  if [ $stat -ne 0 ] ; then
    echo "  [*] Invalid syntax for $2"
    usage
  fi
}

```

```
47
48 while getopts ":g:h:i:" opt; do
49   case "$opt" in
50     g)
51       ngw=${OPTARG}
52       ;;
53     h)
54       nhost=${OPTARG}
55       ;;
56     i)
57       nip=${OPTARG}
58       ;;
59     *)
60       usage
61       ;;
62   esac
63 done
64
65 # testar se parametros foram informados
66 [ -z $ngw ]   && { echo "  [*] No gateway?"; usage; }
67 [ -z $nhost ] && { echo "  [*] No hostname?"; usage; }
68 [ -z $nip ]   && { echo "  [*] No ipaddr?"; usage; }
69
70 # testar sintaxe de parametros
71 valid_ip $nip "IPADDR"
72 valid_ip $ngw "GATEWAY"
73 valid_host $nhost
74
75 # alterar endereco ip/gateway
76 iff="/etc/network/interfaces"
77 cip="$( egrep '^address ' $iff | awk -F'[ /]' '{print $2}' )"
78 cgw="$( egrep '^gateway ' $iff | awk '{print $NF}' )"
79 sed -i "s|${cip}|${nip}|g" $iff
80 sed -i "s|${cgw}|${ngw}|g" $iff
81 ip addr flush label 'enp0s*'
82
83 # alterar hostname local
84 chost="$( hostname -s )"
85 sed -i "s/${chost}/${nhost}/g" /etc/hosts
86 sed -i "s/${chost}/${nhost}/g" /etc/hostname
87
88 invoke-rc.d hostname.sh restart
89 invoke-rc.d networking restart
90 hostnamectl set-hostname $nhost
91
92 # assinar chaves SSH
93 bash /root/scripts/sshsign.sh
94
95 # reiniciar sistema de autenticacao LDAP
96 systemctl restart nslcd.service
97 systemctl restart nscd.service
```

```
 98
 99 for table in `ls -1 /var/cache/nscd` ; do
100   nscd --invalidate $table
101 done
```

Ao invocar este *script* após a criação de uma nova VM, iremos não apenas alterar seu endereço IP, *gateway* e *hostname* mas também integrá-la com o sistema de autenticação remota do LDAP e SSH-CA em um único comando, como veremos a seguir.

Findo este passo, desligue a máquina `debian-template`.

11) Ajuste das regras de firewall

1. Observando os acessos feitos pelo *script* em `/root/scripts/sshsign.sh`, note que a máquina efetuando as assinaturas de chave necessita conseguir alcançar a máquina `ns2` pela rede, e logar via SSH com o usuário `sshca`. Mais além, as autenticações via rede são feitas junto ao servidor OpenLDAP, operando com o *daemon* `slapd`. Nenhum desses requisitos foi previsto em nossas regras de firewall originais, então temos que ajustá-las para permitir que máquinas da DMZ e Intranet consigam realizar esses acessos.

   Vamos listar os requerimentos de regras:

   1. Máquinas na DMZ devem conseguir acessar a máquina `ns2` nas portas 22/TCP (SSH) e 389/TCP (LDAP). Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.
   2. Máquinas na Intranet devem conseguir acessar a máquina `ns2` nas portas 22/TCP (SSH) e 389/TCP (LDAP). Como esse tráfego passa **através do** firewall, devemos inserir regras na *chain* FORWARD.

   Logue como `root` na máquina `ns1`:

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

   Basta criar uma regra para atender o requisito (b), como se segue:

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

   Salve as regras na configuração do firewall local:

```
# /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.
```

12) Configurando um cliente Linux

Nesta atividade iremos criar uma máquina cliente Linux para utilizarmos como ponto de partida para os logins `ssh` nos diferentes servidores que configuraremos durante este curso. O nome da máquina será `client`, alocada para a Intranet com o endereço IP 192.168.42.2/24. Iremos integrá-la com os sistemas de autenticação do LDAP e SSH-CA, e testar login remoto na máquina `ns2`.

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `client`.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM `client` e depois em *Settings*.

   Em *Network* > *Adapter 1* > *Attached to* > *Host-only Adapter*, altere o nome da rede *host-only* para o mesmo da interface da VM `ns1` que está alocada à Intranet. Seguindo o exemplo mostrado no início da sessão 2, a rede escolhida seria a `Virtualbox Host-Only Ethernet Adapter #3`.

   Clique em *OK*, e ligue a máquina `client`.

3. Logue como o usuário `root` — o primeiro login poderá demorar um pouco até que a tentativa de autenticação via rede no servidor LDAP, neste momento inatingível, incorra em *timeout*. Feito o login, use o script `/root/scripts/changehost.sh` para configurar a máquina de forma automática:

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

   ```
   # bash ~/scripts/changehost.sh -h client -i 192.168.42.2 -g 192.168.42.1
   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!
   ```

   ```
   # hostname
   client
   ```

4. Se tudo tiver funcionado corretamente, a máquina `client` já estará integrada aos sistemas de

autenticação LDAP e SSH-CA. Faça login como o usuário `luke` usando os comandos `su` ou `ssh`:

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

5. Vamos criar um par de chaves para esse usuário e assiná-las. Como o conteúdo do diretório *home* foi copiado diretamente do `/etc/skel`, temos à disposição o *script* para essa tarefa na pasta `~/scripts/sshsign_user.sh`. Execute-o:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

6. Vamos testar? Tente logar usando o usuário `luke` na máquina `ns2`:

```
$ ssh luke@ns2
Creating directory '/home/luke'.
Linux ns2 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Tue Nov 13 21:05:01 2018 from 127.0.0.1
luke@ns2:~$
```

```
$ hostname ; whoami ; pwd
ns2
luke
/home/luke
```

Excelente! Conseguimos logar sem ter que confirmar a relação de confiança com o servidor e sem digitar senha, como esperado.

13) Configurando o firewall para funcionar com LDAP/SSH-CA

Suponha, agora, que queremos integrar o firewall no sistema de autenticação LDAP/SSH-CA, mas com um maior nível de restrição. Sendo uma máquina crítica, não podemos permitir que qualquer usuário faça login nessa máquina, sendo necessário implementar controles mais estritos.

O primeiro passo, naturalmente, é fazer a integração com os sistemas de autenticação. Vamos fazer isso:

1. Logue como usuário `root` na máquina `ns1`:

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

2. Instale as dependências para funcionamento dos *scripts* e integração do sistema de autenticação.

```
# apt-get install sshpass nslcd
```

Novamente, durante a instalação do pacote `nslcd`, informe as seguintes opções:

Tabela 4. Configurações do pacote nslcd

Pergunta	Opção
URI do servidor LDAP	ldap://ldap.intnet/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

3. Configure a criação automática de diretórios, com o arquivo novo `/usr/share/pam-configs/mkhomedir`. Para maior conveniência, você pode copiar o arquivo diretamente da máquina `ns2` para o local apropriado usando o comando `scp`, como se segue:

```
# scp -o StrictHostKeyChecking=no aluno@ns2:/usr/share/pam-configs/mkhomedir
/usr/share/pam-configs/
Warning: Permanently added 'ns2,10.0.42.2' (ECDSA) to the list of known hosts.
aluno@ns2's password:
mkhomedir
100%  170   449.4KB/s   00:00
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

4. Crie o arquivo novo `/root/scripts/sshsign.sh` e cole o conteúdo do *script* de assinatura de chaves de *host* que utilizamos anteriormente:

```
# nano /root/scripts/sshsign.sh
(...)
```

```
# ls /root/scripts/
changehost.sh  signzone-intnet.sh  sshsign.sh  syncdirs.sh
```

Execute-o:

```
# bash ~/scripts/sshsign.sh
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!
```

14) Restringindo login por grupos e usuários

Agora sim, com a integração concluída, imagine o seguinte cenário: não queremos que usuários do grupo `sysadm`, do qual faz parte o usuário `luke`, possam logar no firewall. Esse permissão será dada apenas a membros do grupo `fwadm`, que criaremos a seguir. Um desses usuários é o colaborador `han`, cuja senha será `seg10han`. Como configurar esse tipo de restrição?

1. Primeiro, vamos criar o grupo e usuário. Logue na máquina `ns2` como usuário `root`:

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

2. Crie o grupo:

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

   Usuário:

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

   Adicione o usuário ao grupo:

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

   E, finalmente, configure a senha do usuário `han`:

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

3. De volta ao firewall, como usuário `root`:

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

Edite o arquivo `/etc/nslcd.conf`, configurando a opção `pam_authz_search`. Essa opção permite que sejam definidos filtros de busca para o `nslcd`, através dos quais podemos restringir que usuários e/ou grupos podem logar na máquina local. No caso, queremos que apenas membros do grupo `fwadm` possam logar, portanto adicionamos a seguinte linha ao final do arquivo:

```
# echo 'pam_authz_search
(&(objectClass=posixGroup)(cn=fwadm)(memberUid=$username))' >> /etc/nslcd.conf
```

Podemos customizar o filtro acima para incluir apenas usuários específicos, ou mesmo DNs que possuam um atributo qualquer (por exemplo, e-mails com um determinado sufixo). Tome sempre cuidado para não filtrar todos os usuários disponíveis no LDAP acidentalmente — é importante, nesses casos, sempre manter uma conta local (como `aluno` ou `root`, no nosso caso específico) com acesso ao sistema.

Reinicie os serviços do `nslcd` e `nscd`:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

Verifique que o usuário `han` é visto como membro do grupo `fwadm`:

```
# groups han
han : fwadm
```

Ocasionalmente, reiniciar o `nscd` não é suficiente para que ele detecte novas alterações na base de usuários/grupos do LDAP. Nesse caso, podemos invalidar as *caches* das tabelas do `nscd` com o comando:

```
# nscd --invalidate TABLE
```

As tabelas disponíveis podem ser consultadas na página de manual do `nscd`, ou vistas diretamente dentro da pasta `/var/cache/nscd`:

```
# ls -1 /var/cache/nscd/
group
hosts
netgroup
passwd
services
```

Para invalidar todas as *caches* do `nscd`, podemos executar por exemplo:

```
# for table in `ls -1 /var/cache/nscd` ; do nscd --invalidate $table ; done
```

4. Vamos testar a efetividade do controle aplicado. Na máquina `client`, faça login como o usuário `luke`:

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

Tente logar via `ssh` na máquina `ns1`:

```
$ ssh luke@ns1
LDAP authorisation check failed
Authentication failed.
```

Como o usuário `luke` não pertence ao grupo `fwadm`, o acesso é negado. Observando o log de *debug* do `nslcd`, podemos ver que a pesquisa com o filtro aplicado anteriormente não retorna resultados:

```
nslcd: [95f874] <authz="luke"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))"
nslcd: [95f874] <authz="luke"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))")
nslcd: [95f874] <authz="luke"> DEBUG: ldap_result(): end of results (0 total)
nslcd: [95f874] <authz="luke"> pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))" found no matches
```

5. Vamos fazer o mesmo procedimento com o usuário `han`. Logue-se como `han` na máquina `client`:

```
$ hostname ; whoami
client
han
```

Como é a primeira vez que estamos usando este usuário, gere um par de chaves assinadas para ele:

```
$ bash ~/scripts/sshsign_user.sh
Signing ~/.ssh/id_rsa.pub key...
All done!
```

Tente logar via `ssh` na máquina `ns1`:

```
$ ssh han@ns1
Creating directory '/home/han'.
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

han@ns1:~$
```

```
$ hostname ; whoami ; pwd
ns1
han
/home/han
```

Observando o log de *debug* do `nslcd`, podemos ver que a pesquisa com o filtro aplicado anteriormente retorna como resultado o grupo `cn=fwadm,ou=Groups,dc=intnet`:

```
nslcd: [138641] <authz="han"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))"
nslcd: [138641] <authz="han"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))")
nslcd: [138641] <authz="han"> DEBUG: ldap_result(): cn=fwadm,ou=Groups,dc=intnet
nslcd: [138641] <authz="han"> DEBUG: pam_authz_search found
"cn=fwadm,ou=Groups,dc=intnet"
```

15) Restringindo logins SSH apenas via chaves assimétricas

Apesar de o controle que aplicamos na atividade anterior ser interessante, ainda não resolvemos o problema completamente. Como é possível tentar login na máquina `ns1` usando senha, é possível que um atacante tente login por força-bruta, adivinhando a senha do usuário `han`, até conseguir. Vamos resolver isso.

1. Primeiro, vamos constatar o problema. Logue na máquina `client` como o usuário `han`:

```
$ hostname ; whoami
client
han
```

Para evitar que o cliente `ssh` use nossa chave assinada, passe as opções abaixo para o comando. Em seguida, digite a senha correta do usuário `han`:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns1
han@ns1's password:
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:26:13 2018 from 192.168.42.2
han@ns1:~$
```

```
$ hostname ; whoami
ns1
han
```

Note que conseguimos fazer o login usando senha normalmente, sem usar a chave assinada pela CA.

2. Logue como `root` na máquina `ns1`:

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

Iremos aplicar o controle sobre a opção `PasswordAuthentication` do `sshd`, desativando-o. Assim, não será mais possível logar via senha, apenas via chaves assimétricas. Execute o comando abaixo:

```
# sed -i 's/^#\(PasswordAuthentication\).*/\1 no/' /etc/ssh/sshd_config
```

E reinicie o `sshd`:

```
# systemctl restart sshd.service
```

3. De volta à máquina `client`, como `han`:

```
$ hostname ; whoami
client
han
```

Tente novamente logar usando senha:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns1
Permission denied (publickey).
```

Note que a permissão foi negada, pois apenas o método `publickey` é aceito para autenticação. Remova as opções do `ssh` e tente novamente, desta vez usando chaves:

```
$ ssh han@ns1
Linux ns1 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64

Last login: Wed Nov 14 00:27:39 2018 from 192.168.42.2
han@ns1:~$
```

```
$ hostname ; whoami
ns1
han
```

16) Bloqueando tentativas de brute force contra o SSH

Não podemos aplicar o mesmo tipo de controle que fizemos na máquina `ns1` (o firewall da rede) no servidor `ns2` (o servidor LDAP) — nosso *script* de assinatura de chaves de *host* e de usuário utiliza login via senha com o usuário `sshca` para operar. A alteração dos *scripts* para usarem chaves e a correspondente restrição a login usando senhas teria implicações muito negativas na segurança do sistema, neste momento. Vamos implementar um controle diferente, então: proteção contra ataques de força-bruta usando a ferramenta Fail2ban.

O Fail2ban opera através da análise de eventos de log (normalmente registrados no diretório `/var/log`) e seu processamento através de expressões regulares. Caso um evento "case" (ou seja, ocorra um *match*) com uma expressão regular configurada, o Fail2ban irá adicionar uma unidade ao contador de violações de um determinado *host* remoto. Se esse *host* ultrapassar o número de violações configuradas em um dado período, o Fail2ban irá então tomar alguma ação configurada pelo administrador (registrar um evento nos logs, enviar um e-mail para o administrador ou até mesmo bloquear de forma automática o *host* no firewall local).

Várias expressões regulares já vêm pré-configuradas no Fail2ban, para as ferramentas mais populares (como o `sshd`, o servidor web Apache ou o servidor SMTP Postfix). Caso se deseje configurar expressões regulares para ferramentas customizadas, também é possível fazê-lo.

1. Logue como o usuário `root` na máquina `ns2`:

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

2. Instale a ferramenta `fail2ban`:

```
# apt-get install fail2ban
```

3. Note que, por padrão, apenas a *jail* `sshd` vem habilitada no Debian. Não teremos que fazer qualquer alteração nesse sentido, já que é justamente o serviço `ssh` que queremos proteger.

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
```

4. As opções padrão do Fail2ban ficam configuradas no arquivo `/etc/fail2ban/jail.conf`, seção `[DEFAULT]`. Em particular, temos interesse nas seguintes configurações:

   - `findtime`: Intervalo em que o Fail2ban irá registrar violações de *hosts* remotos.
   - `maxretry`: Número de violações máximo permitido dentro do período `findtime` definido acima. Caso este valor seja ultrapassado, o Fail2ban irá tomar a ação configurada pelo administrador.
   - `bantime`: Período em que o *host* remoto será afetado pela ação configurada. Caso esta ação seja, por exemplo, um bloqueio no firewall local, o *host* ficará banido pelo tempo especificado aqui.

   Os valores padrão para as variáveis acima são os que se seguem:

```
# cat /etc/fail2ban/jail.conf | sed -n -e '/^\[DEFAULT\]/,/^\[/p' | grep
'^maxretry\|^bantime\|^findtime'
bantime  = 600
findtime  = 600
maxretry = 5
```

5. Vamos configurar o seguinte cenário: caso um atacante seja detectado pelo Fail2ban com mais de 3 violações (`maxretry`) num período de dez minutos (`findtime`), então iremos bani-lo via regra no firewall local (ação `iptables-multiport`) por dez minutos (`bantime`). Como o `findtime` e o `bantime` padrão estão corretos, iremos apenas configurar as duas outras variáveis, como se segue:

```
# echo "maxretry = 3" >> /etc/fail2ban/jail.d/defaults-debian.conf
```

```
# echo "banaction = iptables-multiport" >> /etc/fail2ban/jail.d/defaults-
debian.conf
```

O arquivo `/etc/fail2ban/jail.d/defaults-debian.conf` ficou assim, portanto:

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf
[sshd]
enabled = true
maxretry = 3
banaction = iptables-multiport
```

6. Uma outra configuração necessária é comentar uma linha do arquivo `/etc/fail2ban/filter.d/sshd.conf` que contém uma expressão regular para detectar entradas no seguinte formato no arquivo `/var/log/auth.log`:

```
Oct 30 12:03:47 ldap sshd[6677]: pam_unix(sshd:auth): authentication failure;
logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.42.2  user=sshca
```

Em sistemas com autenticação LDAP, como é o nosso caso, a linha acima é inserida em tentativas de login mesmo em caso de sucesso, como reportado em https://github.com/fail2ban/fail2ban/issues/106 . Para corrigir esse falso positivo, basta executar:

```
# sed -i 's/^\(.*pam_unix.*\)/#\1/' /etc/fail2ban/filter.d/sshd.conf
```

7. Reinicie o Fail2ban para aplicar as configurações que realizamos:

```
# systemctl restart fail2ban.service
```

8. O Fail2ban criará novas *chains* no firewall para inserção de regras de banimento, quando adequado. Observe que o firewall está, até este momento, sem regras de BLOCK ou REJECT:

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 37 packets, 5021 bytes)
 pkts bytes target     prot opt in     out     source               destination
   26  1820 f2b-sshd   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
multiport dports 22

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

Chain OUTPUT (policy ACCEPT 13 packets, 1152 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
   26  1820 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Agora, vamos fazer uma simulação de ataque ao `sshd`. Monitore o arquivo `/var/log/fail2ban.log`:

```
# tail -n5 -f /var/log/fail2ban.log
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set findtime = 600
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set jail log file
encoding to UTF-8
2018-11-14 00:31:44,246 fail2ban.filter         [2099]: INFO    Set maxlines = 10
2018-11-14 00:31:44,277 fail2ban.server         [2099]: INFO    Jail sshd is not a
JournalFilter instance
2018-11-14 00:31:44,281 fail2ban.jail           [2099]: INFO    Jail 'sshd' started
```

9. Logue na máquina `client` como o usuário `han`, por exemplo:

```
$ hostname ; whoami
client
han
```

Para disparar o filtro do Fail2ban na máquina `ns2`, não poderemos usar o login via chaves assimétricas, que obterá sucesso. Faça login usando senha como mostrado no comando a seguir; digite senhas incorretas para ativar a detecção do Fail2ban:

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

Tente logar novamente:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@ns2
ssh: connect to host ns2 port 22: Connection refused
```

A máquina foi bloqueada, como esperado.

10. De volta à máquina `ns2`, como o usuário `root`:

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

Note que os eventos de senha incorreta foram registrados pelo Fail2ban, bem como o banimento:

```
2018-11-14 00:34:01,944 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:04,830 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:08,081 fail2ban.filter         [2099]: INFO    [sshd] Found
192.168.42.2
2018-11-14 00:34:08,534 fail2ban.actions        [2099]: NOTICE  [sshd] Ban
192.168.42.2
```

Observe que a regra de REJECT foi inserida automaticamente pelo Fail2ban no firewall local:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
    1    60 REJECT     all  --  *      *       192.168.42.2         0.0.0.0/0
reject-with icmp-port-unreachable
  612 70528 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Para remover o banimento de um endereço IP antes que o tempo total do `bantime` tenha transcorrido, é possível usar o comando `fail2ban-client`, como mostrado a seguir:

```
# fail2ban-client set sshd unbanip 192.168.42.2
192.168.42.2
```

Note que a regra de firewall é apagada, como esperado:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
 pkts bytes target     prot opt in     out     source               destination
  395 25992 RETURN     all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

11. De volta à máquina `client`, como `han`, podemos tentar o login via senha novamente — desta vez, digite a senha correta:

```
$ hostname ; whoami
client
han
```

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

han@ns2:~$
```

```
$ hostname ; whoami
ns2
han
```