Sessão 10: Módulos de segurança do kernel

Nesta sessão iremos trabalhar com o módulo de segurança AppArmor, que implementa um modeloe de segurança MAC (*Mandatory Access Controls*) para o kernel Linux.

1) Topologia desta sessão

Criaremos apenas uma nova máquina nesta sessão, a saber:

- `lsm`, um ambiente para testes de módulos de segurança do kernel, os *Linux Security Modules*. Endereço IP 10.0.42.12/24.

1. Como de costume, vamos à criação dos registros DNS. Acesse a máquina `ns1` como o usuário `root`:

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

   Edite o arquivo de zonas `/etc/nsd/zones/intnet.zone`, inserindo entradas A para a máquinas indicadas no começo desta atividade. **Não se esqueça** de incrementar o valor do serial no topo do arquivo!

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

   ```
   # grep lsm /etc/nsd/zones/intnet.zone
   lsm     IN    A                10.0.42.12
   ```

   Faça o mesmo para o arquivo de zona reversa:

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

   ```
   # grep lsm /etc/nsd/zones/10.0.42.zone
   12      IN   PTR               lsm.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 11 rrsets, 10 messages and 0 key entries
```

Verifique a criação das entradas usando o comando `dig`:

```
# dig lsm.intnet +short
10.0.42.12
```

```
# dig -x 10.0.42.12 +short
lsm.intnet.
```

2) Criação do ambiente de segurança

1. Vamos criar a VM `lsm` e utilizá-la para testes de módulos de segurança do kernel. Clone a máquina `debian-template` para uma de nome `lsm`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.12/24.

   Concluída a clonagem, ligue a VM e logue como `root`. Use o script `/root/scripts/changehost.sh` para fazer a configuração automática, como de costume.

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

   ```
   # bash ~/scripts/changehost.sh -h lsm -i 10.0.42.12 -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!
   ```

2. Aplique o *baseline* de segurança à máquina `lsm`, repetindo o que fizemos no passo (2), atividade (2) da sessão 7:

   ```
   $ hostname ; whoami
   client
   ansible
   ```

```
$ sed -i '/\[srv\]/a lsm' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l lsm -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l lsm ~/ansible/srv.yml
SUDO password:

(...)

PLAY RECAP
*********************************************************************************
***********************

lsm                        : ok=10   changed=8    unreachable=0    failed=0
```

3) Instalação do AppArmor

1. Vamos agora instalar o AppArmor, um sistema MAC que atua como módulo de segurança do kernel Linux com o objetivo de confinar programas a um conjunto limitado de recursos.

   Acesse a máquina `lsm` como o usuário `root`.

   ```
   # hostname ; whoami
   lsm
   root
   ```

2. Instale os pacotes:

   ```
   # apt-get install -y apparmor apparmor-utils
   ```

3. Habilite o AppArmor durante o *boot* do kernel — para isso, basta alterar a linha de *boot* padrão do GRUB usando os comandos a seguir:

   ```
   # mkdir -p /etc/default/grub.d
   ```

   ```
   # echo 'GRUB_CMDLINE_LINUX_DEFAULT="$GRUB_CMDLINE_LINUX_DEFAULT apparmor=1
   security=apparmor"' \
       > /etc/default/grub.d/apparmor.cfg
   ```

   Reconstrua a configuração do GRUB com o comando:

   ```
   # update-grub
   ```

Em seguida, reinicie a máquina para que as configurações realizadas sejam carregadas durante o próximo *boot*:

```
# reboot
```

4. Após o *reboot*, logue novamente como `root` e verifique o estado de execução do AppArmor:

```
# aa-status
apparmor module is loaded.
0 profiles are loaded.
0 profiles are in enforce mode.
0 profiles are in complain mode.
0 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
0 processes are unconfined but have a profile defined.
```

4) Criação de um perfil AppArmor para o servidor web Nginx

Em que o AppArmor pode incrementar a segurança do sistema? Vamos fazer um caso de teste com o servidor web Nginx: iremos criar um perfil de segurança para essa aplicação, definindo com precisão o que ela está ou não autorizada a fazer no sistema.

1. Primeiro, instale o Nginx:

```
# apt-get install -y nginx
```

Em sua máquina física, aponte o navegador para o IP da máquina `lsm`, 10.0.42.12. Voce deverá ver a página a seguir, comprovando que o Nginx foi instalado com sucesso:

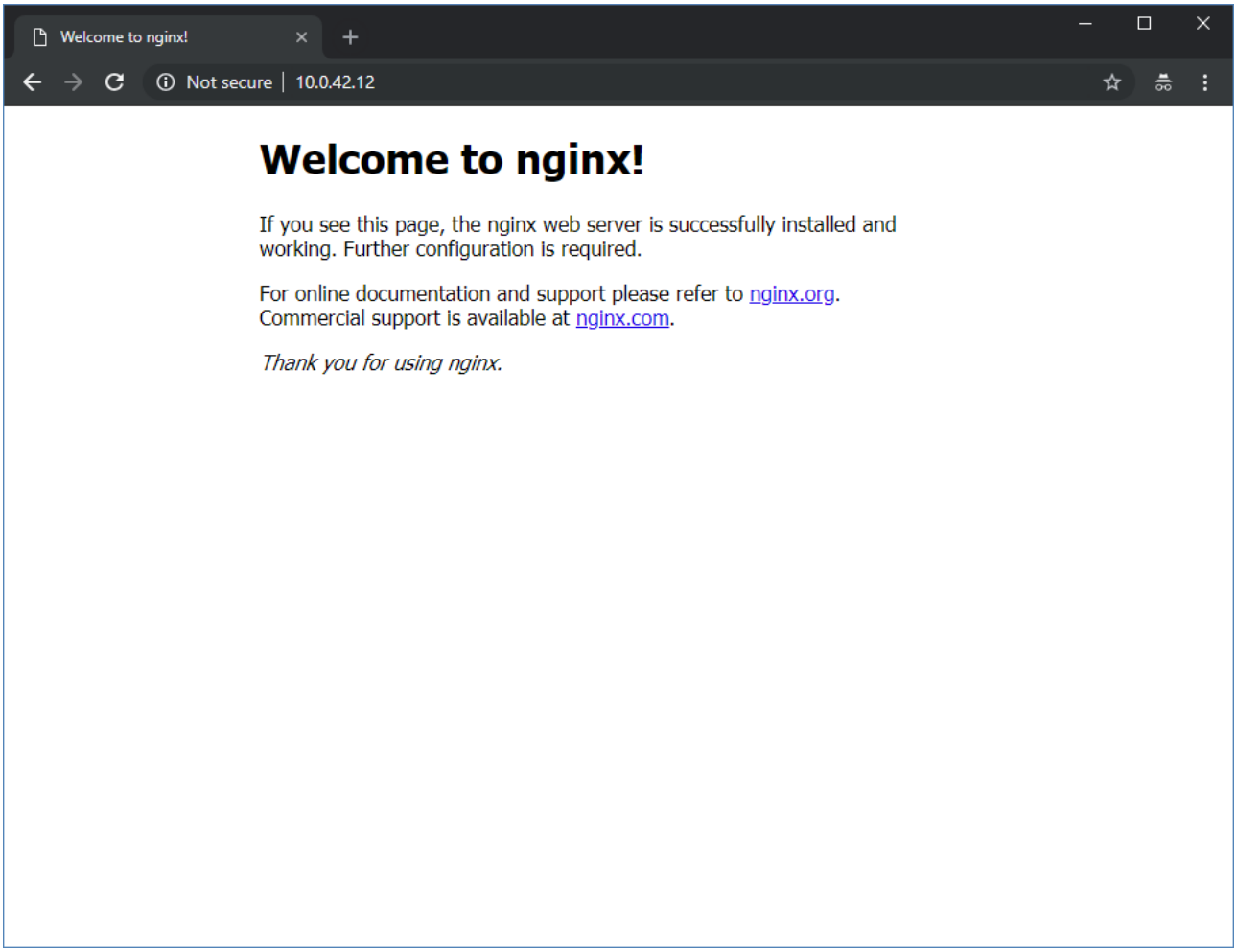

Figura 1. Nginx instalado com sucesso

2. Vamos criar duas pastas: `/data/www/safe`, à qual o Nginx deve ter acesso, e `/data/www/unsafe`, cujo acesso deve ser negado ao Nginx pelo AppArmor.

```
# mkdir -p /data/www/safe
```

```
# mkdir -p /data/www/unsafe
```

Em cada uma das pastas, crie um arquivo `index.html` que indique de forma clara no navegador que estamos, de fato, navegando no local pretendido.

```
# cat << EOF >> /data/www/safe/index.html
<html>
    <b>Oi! Acessar este arquivo e permitido.</b>
</html>
EOF
```

```
# cat << EOF >> /data/www/unsafe/index.html
<html>
    <b>Oi! Acessar este arquivo NAO e permitido.</b>
</html>
EOF
```

3. Altere a configuração do Nginx para servir esses arquivos na porta 8080/TCP, como se segue:

```
# cat << EOF >> /etc/nginx/conf.d/apparmor.conf
server {
    listen 8080;
    location / {
            root /data/www;
    }
}
EOF
```

Recarregue a configuração do Nginx:

```
# systemctl reload nginx
```

Teste o acesso à URL http://10.0.42.12:8080/safe/:

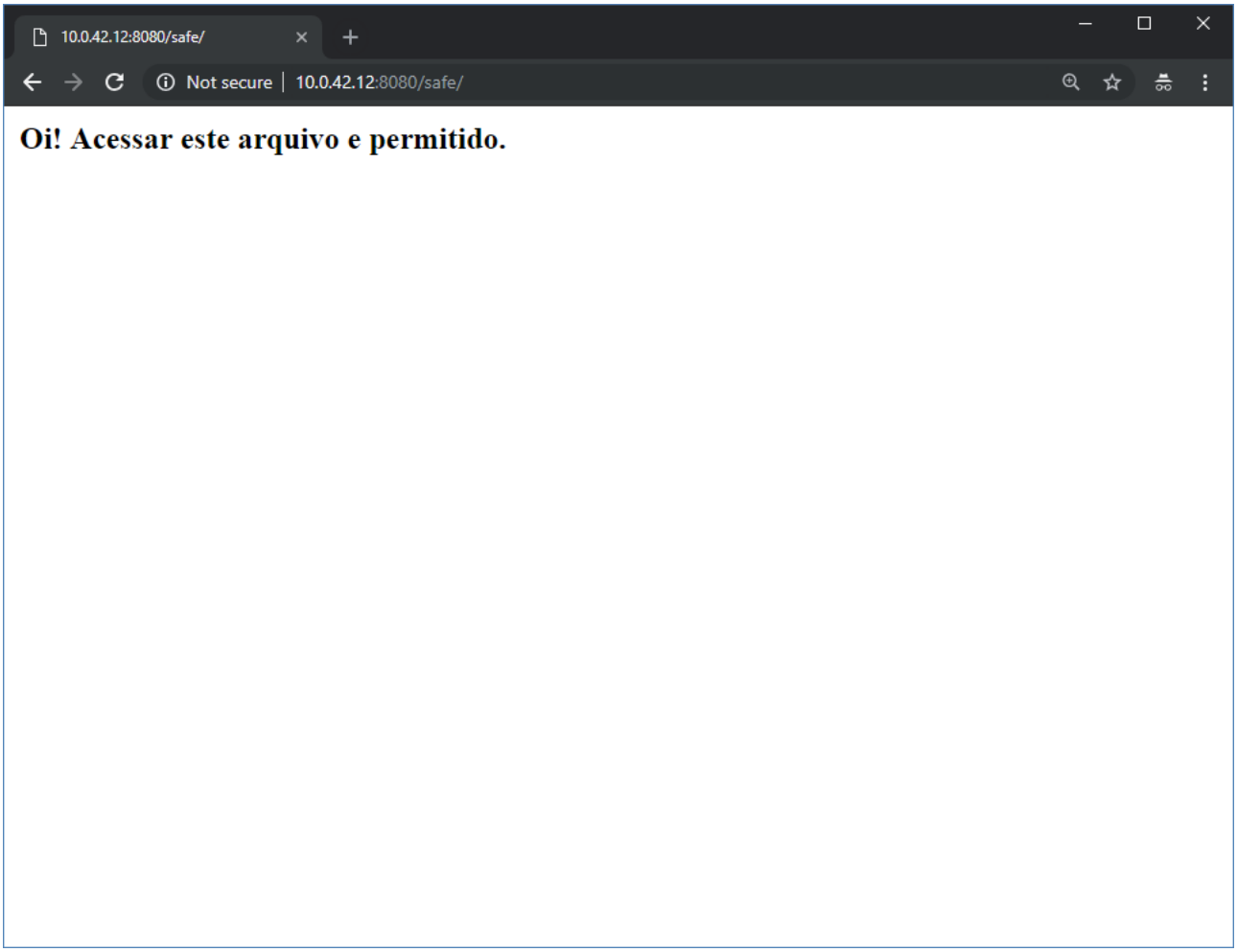

Figura 2. Acesso permitido à pasta SAFE

O acesso é permitido, como esperado. Evidentemente, o acesso a `http://10.0.42.12:8080/unsafe/` também é autorizado, já que ainda não configuramos o AppArmor.

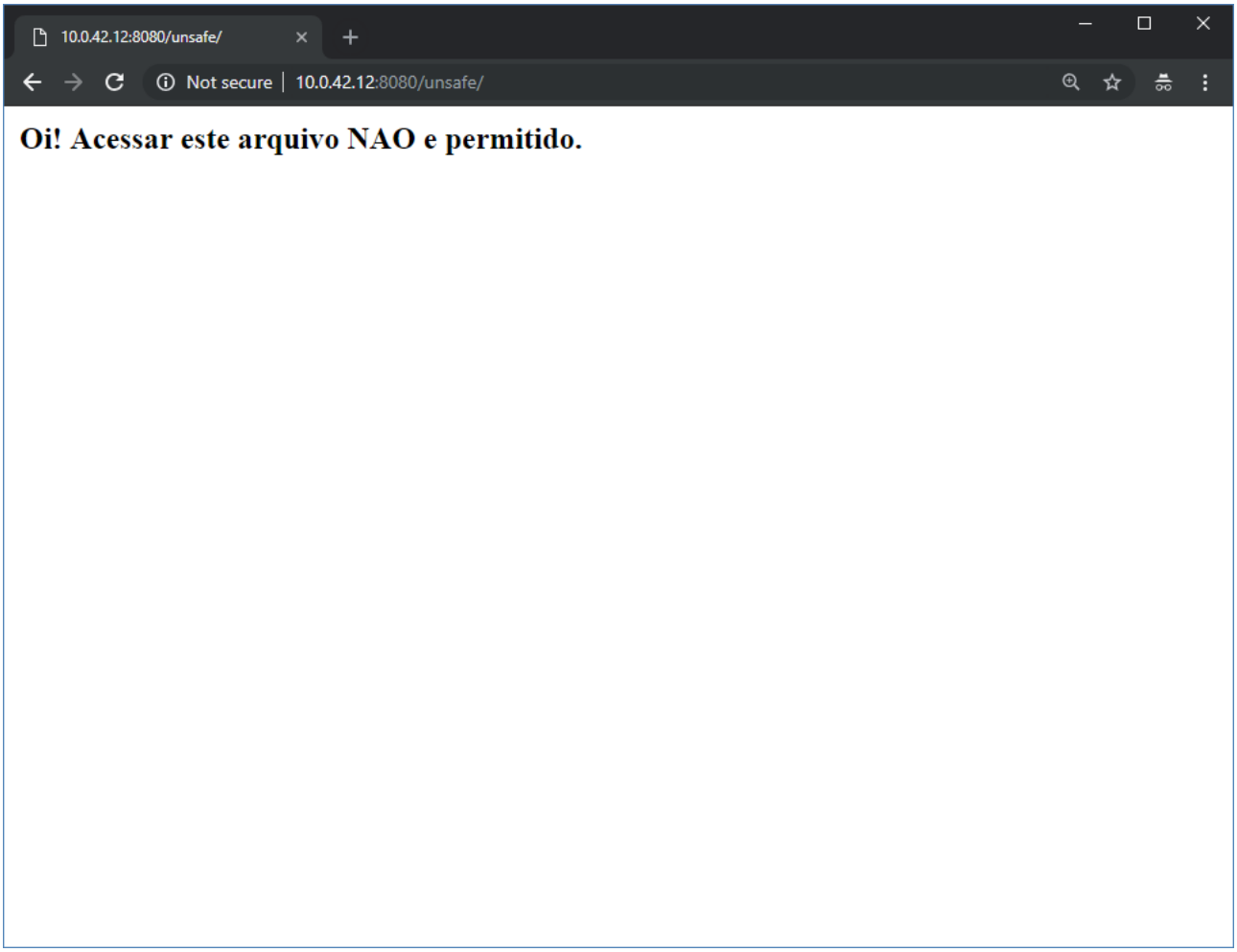

Figura 3. Acesso permitido à pasta UNSAFE

4. O Debian possui um conjunto de perfis pré-prontos para o AppArmor que irão nos auxiliar na tarefa de configuração — instale o pacote `apparmor-profiles`:

```
# apt-get install -y apparmor-profiles
```

Para verificar o estado dos perfis do AppArmor, execute o comando `apparmor_status`:

```
# apparmor_status
apparmor module is loaded.
33 profiles are loaded.
0 profiles are in enforce mode.
33 profiles are in complain mode.
   /usr/lib/dovecot/anvil
   /usr/lib/dovecot/auth
   /usr/lib/dovecot/config
   /usr/lib/dovecot/deliver
   /usr/lib/dovecot/dict
   /usr/lib/dovecot/dovecot-auth
   /usr/lib/dovecot/dovecot-lda
   /usr/lib/dovecot/dovecot-lda///usr/sbin/sendmail
   /usr/lib/dovecot/imap
   /usr/lib/dovecot/imap-login
   /usr/lib/dovecot/lmtp
   /usr/lib/dovecot/log
   /usr/lib/dovecot/managesieve
   /usr/lib/dovecot/managesieve-login
   /usr/lib/dovecot/pop3
   /usr/lib/dovecot/pop3-login
   /usr/lib/dovecot/ssl-params
   /usr/sbin/avahi-daemon
   /usr/sbin/dnsmasq
   /usr/sbin/dnsmasq//libvirt_leaseshelper
   /usr/sbin/dovecot
   /usr/sbin/identd
   /usr/sbin/mdnsd
   /usr/sbin/nmbd
   /usr/sbin/nscd
   /usr/sbin/smbd
   /usr/sbin/smbldap-useradd
   /usr/sbin/smbldap-useradd///etc/init.d/nscd
   /usr/{sbin/traceroute,bin/traceroute.db}
   klogd
   ping
   syslog-ng
   syslogd
1 processes have profiles defined.
0 processes are in enforce mode.
0 processes are in complain mode.
1 processes are unconfined but have a profile defined.
   /usr/sbin/nscd (400)
```

5. Vamos criar um perfil de acesso customizado para o Nginx. Entre na pasta `/etc/apparmor.d` e use o comando `aa-autodep nginx` para criar um perfil em branco:

```
# cd /etc/apparmor.d
```

```
# aa-autodep nginx
Writing updated profile for /usr/sbin/nginx.
```

Uma vez criado o perfil, use o comando `aa-complain nginx` para colocá-lo no modo *complain* — nesse perfil, violações serão autorizadas, e apenas um alerta será gerado:

```
# aa-complain nginx
Setting /usr/sbin/nginx to complain mode.
```

Finalmente, reinicie o Nginx:

```
# systemctl restart nginx
```

6. Em seu navegador, acesse a URL `http://10.0.42.12:8080/safe/` uma vez, e **apenas** essa URL (i.e. não acesse a área *unsafe*). Essa requisição gerará um evento no log de acessos do Nginx, o qual processaremos a seguir.
7. Feito isso, use o comando `aa-logprof` para processar os eventos observados no log do Nginx e gerar um perfil de acesso para a aplicação.

   O comando acima irá processar os logs do Nginx e atualizar o perfil de acesso do programa. Para cada tipo de acesso identificado, você deverá responder se deseja autorizar ou negar uma capacidade à aplicação. Assumindo que o sistema não está sob ataque, é razoável supor que todos os acessos são legítimos, portanto autorize todos com o atalho `A` (para *Allow*).

   Ao final do processo, o programa irá perguntar se você deseja salvar as informações no perfil do Nginx. Confirme com o atalho `S` (para *Save Changes*).

   Temos abaixo uma execução típica do `aa-logprof` para esse cenário:

```
# aa-logprof
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:    /usr/sbin/nginx
Capability: dac_override
Severity:   9

 [1 - capability dac_override,]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
Adding capability dac_override, to profile.

Profile:  /usr/sbin/nginx
Path:     /data/www/safe/index.html
New Mode: r
Severity: unknown
```

```
 [1 - /data/www/safe/index.html r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /data/www/safe/index.html r, to profile.

Profile:  /usr/sbin/nginx
Path:     /etc/ssl/openssl.cnf
New Mode: r
Severity: 2

 [1 - #include <abstractions/openssl>]
  2 - #include <abstractions/ssl_keys>
  3 - /etc/ssl/openssl.cnf r,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding #include <abstractions/openssl> to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-dav-ext.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-dav-ext.conf r, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/lib/nginx/modules/ngx_http_auth_pam_module.so
Old Mode: r
New Mode: mr
Severity: unknown

 [1 - #include <abstractions/ubuntu-browsers.d/plugins-common>]
  2 - /{usr/,}lib{,32,64}/** mr,
  3 - /usr/lib/nginx/modules/ngx_http_auth_pam_module.so mr,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding #include <abstractions/ubuntu-browsers.d/plugins-common> to profile.

Profile:  /usr/sbin/nginx
Path:     /etc/nginx/nginx.conf
New Mode: r
Severity: unknown

 [1 - /etc/nginx/nginx.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /etc/nginx/nginx.conf r, to profile.
```

```
Profile:  /usr/sbin/nginx
Path:     /etc/nginx/modules-enabled/
New Mode: r
Severity: unknown

 [1 - /etc/nginx/modules-enabled/ r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /etc/nginx/modules-enabled/ r, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-auth-pam.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-auth-pam.conf r, to profile.

Profile:  /usr/sbin/nginx
Path:     /var/log/nginx/access.log
New Mode: w
Severity: 8

 [1 - /var/log/nginx/access.log w,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /var/log/nginx/access.log w, to profile.

Profile:  /usr/sbin/nginx
Path:     /var/log/nginx/error.log
New Mode: w
Severity: 8

 [1 - /var/log/nginx/error.log w,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /var/log/nginx/error.log w, to profile.

Profile:  /usr/sbin/nginx
Path:     /usr/share/nginx/modules-available/mod-http-echo.conf
New Mode: r
Severity: unknown

 [1 - /usr/share/nginx/modules-available/mod-http-echo.conf r,]
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) /
Abo(r)t / (F)inish
Adding /usr/share/nginx/modules-available/mod-http-echo.conf r, to profile.

= Changed Local Profiles =
```

```
The following local profiles were changed. Would you like to save them?

 [1 - /usr/sbin/nginx]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w
(C)lean profiles / Abo(r)t
Writing updated profile for /usr/sbin/nginx.
```

8. O isso fez? Confira o conteúdo do arquivo `/etc/apparmor.d/usr.sbin.nginx`:

```
# cat /etc/apparmor.d/usr.sbin.nginx
# Last Modified: Sun Nov 18 02:48:57 2018
#include <tunables/global>

/usr/sbin/nginx flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,

  /data/www/safe/index.html r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,
  /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,
  /usr/share/nginx/modules-available/mod-http-echo.conf r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

   Cada um dos acessos autorizados foi adicionado em uma linha do arquivo acima, com a permissão correspondente ao final. Observe que **qualquer** tipo de acesso que não conste do arquivo acima será negado pelo AppArmor ao ativarmos o modo *enforcing* do sistema.

9. Tudo pronto... certo? Coloque o Nginx em modo *enforce*:

```
# aa-enforce nginx
Setting /usr/sbin/nginx to enforce mode.
```

   E reinicie ambos AppArmor e Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

OOPS! Temos um problema. Confira o que aconteceu verificando o log de erros do Nginx:

```
# tail /var/log/nginx/error.log -n1
2018/11/18 02:56:10 [emerg] 1850#1850: open() "/etc/nginx/modules-enabled/50-mod-
http-geoip.conf" failed (13: Permission denied) in /etc/nginx/nginx.conf:4
```

Uhm, aparentemente o Nginx precisa acessar o arquivo `/etc/nginx/modules-enabled/50-mod-http-geoip.conf`, mas essa permissão não consta do seu perfil. Note que esse arquivo é um *symlink* para o `/usr/share/nginx/modules-available/mod-http-geoip.conf`:

```
# ls -ld /etc/nginx/modules-enabled/50-mod-http-geoip.conf
lrwxrwxrwx 1 root root 54 nov 18 02:31 /etc/nginx/modules-enabled/50-mod-http-
geoip.conf -> /usr/share/nginx/modules-available/mod-http-geoip.conf
```

Sem problema, vamos adicionar essa permissão:

```
# sed -i '/\/usr\/share\/nginx\/modules-available\/mod-http-echo.conf/a\
\/usr\/share\/nginx\/modules-available\/mod-http-geoip.conf r,'
/etc/apparmor.d/usr.sbin.nginx
```

Veja como ficou o arquivo, agora:

```
# cat /etc/apparmor.d/usr.sbin.nginx
# Last Modified: Sun Nov 18 02:48:57 2018
#include <tunables/global>

/usr/sbin/nginx {
  #include <abstractions/base>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,

  /data/www/safe/index.html r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/mod-http-auth-pam.conf r,
  /usr/share/nginx/modules-available/mod-http-dav-ext.conf r,
  /usr/share/nginx/modules-available/mod-http-echo.conf r,
  /usr/share/nginx/modules-available/mod-http-geoip.conf r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

Agora sim! Recarregue o AppArmor e reinicie o Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

Oh não... o que foi dessa vez? Vamos ver:

```
# tail /var/log/nginx/error.log -n1
2018/11/18 03:03:04 [emerg] 1976#1976: open() "/etc/nginx/modules-enabled/50-mod-
http-image-filter.conf" failed (13:Permission denied) in /etc/nginx/nginx.conf:4
```

Podemos adicionar essa permissão, mas já posso lhe adiantar que teremos OUTRO erro logo a seguir. Que coisa, não?

10. Após algum tempo em tentativas e erros, chegamos a um arquivo de perfil que funciona para o nosso caso. Edite o arquivo `/etc/apparmor.d/usr.sbin.nginx` e deixe seu conteúdo **exatamente** como o que se segue:

```
# Last Modified: Sun Nov 18 03:32:21 2018
#include <tunables/global>

/usr/sbin/nginx {
  #include <abstractions/apache2-common>
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/openssl>
  #include <abstractions/ubuntu-browsers.d/plugins-common>

  capability dac_override,
  capability setgid,
  capability setuid,

  deny /data/www/unsafe/* r,

  /data/www/safe/* r,
  /etc/group r,
  /etc/nginx/conf.d/ r,
  /etc/nginx/conf.d/apparmor.conf r,
  /etc/nginx/mime.types r,
  /etc/nginx/modules-enabled/ r,
  /etc/nginx/nginx.conf r,
  /etc/nginx/sites-available/default r,
  /etc/nginx/sites-enabled/ r,
  /etc/nsswitch.conf r,
  /etc/passwd r,
  /etc/ssl/openssl.cnf r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  /run/nginx.pid rw,
  /usr/sbin/nginx mr,
  /usr/share/nginx/modules-available/* r,
  /var/log/nginx/access.log w,
  /var/log/nginx/error.log w,

}
```

Note que há um número significativo de diferenças entre o arquivo acima e o que tínhamos originalmente — você pode imaginar o tempo necessário para construir um perfil desse tipo. E observe que estamos trabalhando aqui no perfil de UMA única aplicação, o servidor web Nginx. Imagine se tivéssemos um número maior de programas, ou requerimentos de acesso mais complexos.

Observe ainda que estamos detalhando políticas expressas de acesso para as pastas *safe* e *unsafe*:

- Na linha `/data/www/safe/* r,`, o acesso de leitura a todos os arquivos dentro da pasta *safe* é autorizado ao Nginx.
- Já na linha `deny /data/www/unsafe/* r,`, o mesmo tipo de acesso na pasta *unsafe* é negado.

11. Enfim, reescrito o perfil acima, recarregue o AppArmor e reinicie o Nginx:

```
# systemctl reload apparmor
```

```
# systemctl restart nginx
```

Sucesso! Vamos ver se nossa proteção funcionou: retorne ao navegador e carregue a URL `http://10.0.42.12:8080/safe/`:

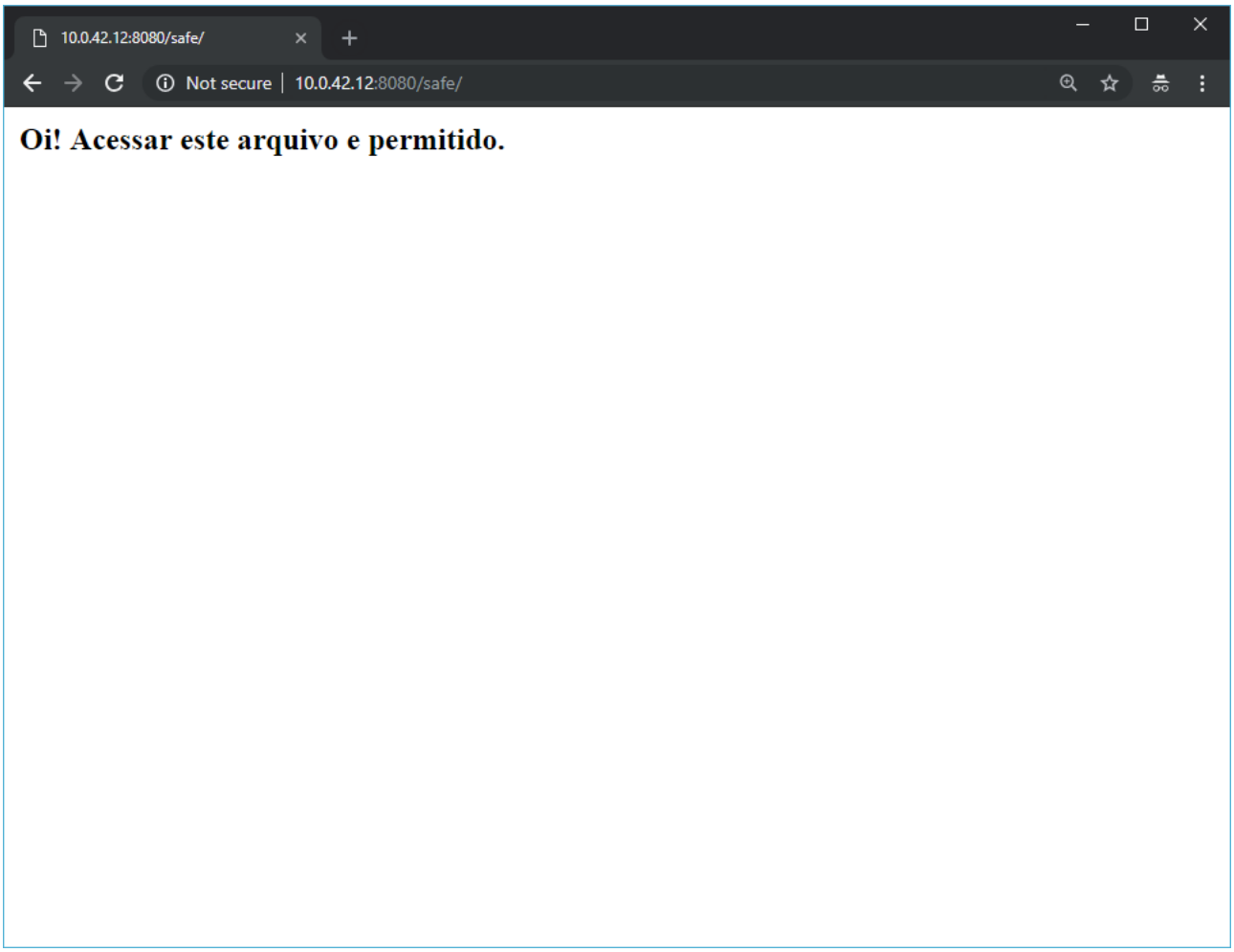

Figura 4. Acesso autorizado à área safe

Perfeito, o acesso foi autorizado. E quanto a `http://10.0.42.12:8080/unsafe/`?

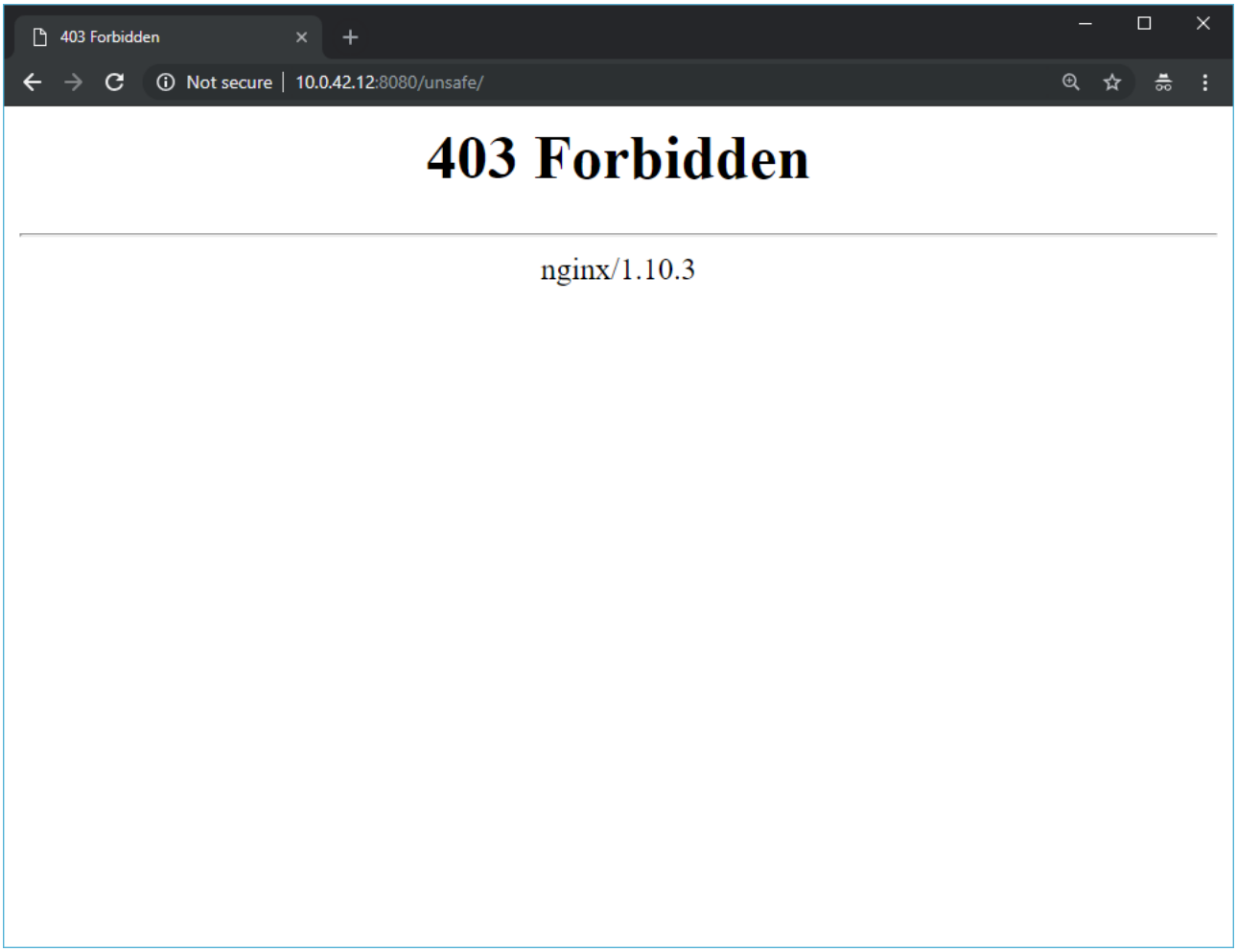

Figura 5. Acesso negado à área unsafe

Como esperado, o acesso foi negado pelo AppArmor. Observe a mensagem de erro no log do Nginx:

```
# tail -f -n0 /var/log/nginx/error.log
2018/11/18 03:36:10 [error] 4035#4035: *2 open() "/data/www/unsafe/index.html"
failed (13: Permission denied), client: 10.0.42.254, server: , request: "GET
/unsafe/ HTTP/1.1", host: "10.0.42.12:8080"
```

12. Como objetivado, conseguimos usar o AppArmor para construir um perfil de segurança específico para o servidor web Nginx, e testar esse perfil com sucesso. Fica claro, também, que a construção de perfis por meio de ferramentas MAC (como o AppArmor ou o SELinux), apesar de se apresentar como uma ferramenta muito poderosa, também traz consigo um custo administrativo bastante alto.

    Leve isso em consideração ao implantar esquemas de segurança em seus servidores. É possível que haja um ganho de segurança significativo por meio da construção de perfis de segurança, mas convém analisar a criticidade dos sistemas-alvo para determinar se o custo administrativo compensa, de fato, o risco que se incorre com a falta da proteção.