Sessão 7: Hardening de sistemas web

Nesta sessão, iremos configurar um website usando o CMS Wordpress seguindo as melhores práticas de segurança, bem como tornando-o altamente disponível através do balanceamento de carga em múltiplos nodos de *frontend* web.

1) Topologia desta sessão

Criaremos quatro novas máquinas nesta sessão, a saber:

- `www1`, primeiro nodo de atendimento web, instalado manualmente. Endereço IP 10.0.42.5/24.
- `www2`, segundo nodo de atendimento web, instalado de forma automática via Ansible. Endereço IP 10.0.42.6/24.
- `db`, servidor de banco de dados para as máquinas `www1` e `www2`. Endereço IP 10.0.42.7/24.
- `lb`, balanceador de carga HTTP/HTTPS que redirecionará requisições para as VMs `www1` e `www2`. Endereço IP 10.0.42.8/24.

1. Temos que criar quatro novos registros DNS diretos e reversos. 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 'www1\|www2\|db\|lb' /etc/nsd/zones/intnet.zone
www1    IN    A                10.0.42.5
www2    IN    A                10.0.42.6
db      IN    A                10.0.42.7
lb      IN    A                10.0.42.8
```

Faça o mesmo para o arquivo de zona reversa:

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

```
# grep 'www1\|www2\|db\|lb' /etc/nsd/zones/10.0.42.zone
5       IN   PTR               www1.intnet.
6       IN   PTR               www2.intnet.
7       IN   PTR               db.intnet.
8       IN   PTR               lb.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 a criação das entradas usando o comando `dig`:

```
# for host in www1 www2 db lb; do echo -n "$host: "; dig ${host}.intnet +short;
done
www1: 10.0.42.5
www2: 10.0.42.6
db: 10.0.42.7
lb: 10.0.42.8
```

```
# for ip in 5 6 7 8; do echo -n "10.0.42.${ip}: "; dig -x 10.0.42.${ip} +short;
done
10.0.42.5: www1.intnet.
10.0.42.6: www2.intnet.
10.0.42.7: db.intnet.
10.0.42.8: lb.intnet.
```

2) Configuração do servidor de banco de dados

1. Vamos começar criando a máquina que atuará como servidor de banco de dados em nossa topologia. Clone a máquina `debian-template` para uma nova, de nome `db`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.7/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, use o script `/root/scripts/changehost.sh` para fazer a configuração automática:

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

```
# bash ~/scripts/changehost.sh -h db -i 10.0.42.7 -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
10.0.42.7/24 enp0s3
db
root
```

2. Vamos aplicar à máquina `db` o *baseline* de segurança que configuramos no Ansible: `sudo`, OpenNTPD, Snoopy e Rsyslog centralizado. Acesse a máquina `client` como o usuário `ansible`:

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

Insira a máquina `db` no inventário gerenciado pelo Ansible:

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

```
$ cat ~/ansible/hosts
[srv]
db
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log

[web_server]
log
```

Execute o *playbook* `/home/ansible/ansible/srv.yml`, limitando o escopo à máquina `db` e alterando

a escala de privilégio para o `su`:

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

PLAY [srv]
*********************************************************************************
***********************


TASK [Gathering Facts]
*********************************************************************************
***********

ok: [db]

TASK [sudoers : Propagate sudoers configuration]
*******************************************************************

changed: [db]

TASK [sudoers : Sets root account as expired]
*******************************************************************

changed: [db]

TASK [ntp : Install OpenNTPD]
*********************************************************************************
****

fatal: [db]: FAILED! => {"changed": false, "module_stderr": "Shared connection to
db closed.\r\n", "module_stdout": "\r\nSua conta expirou; entre em contato com o
administrador do sistema\r\nsu: Falha de autenticação\r\n", "msg": "MODULE
FAILURE\nSee stdout/stderr for the exact error", "rc": 1}
        to retry, use: --limit @/home/ansible/ansible/srv.retry

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

db                         : ok=3    changed=2    unreachable=0    failed=1
```

A *role* irá falhar no passo de instalação do OpenNTPD, pois neste momento o `sudo` acaba de ser configurado e a conta do usuário `root`, expirada. Execute o *playbook* novamente, desta vez sem customizar o método de escalação de privilégio:

```
$ ansible-playbook -i ~/ansible/hosts -l db ~/ansible/srv.yml
```

```
PLAY [srv]
*************************************************************************************
***********************


TASK [Gathering Facts]
*************************************************************************************
***********

ok: [db]

TASK [sudoers : Propagate sudoers configuration]
*********************************************************************

ok: [db]

TASK [sudoers : Sets root account as expired]
*********************************************************************

changed: [db]

TASK [ntp : Install OpenNTPD]
*************************************************************************************
****

changed: [db]

TASK [ntp : Copy OpenNTPD configuration]
*********************************************************************************

changed: [db]

TASK [snoopy : Install Snoopy]
*************************************************************************************
***

changed: [db]

TASK [snoopy : Configure ld.so.preload]
*************************************************************************************

changed: [db]

TASK [syslog : Configure centralized syslog]
*********************************************************************

changed: [db]

RUNNING HANDLER [ntp : Restart OpenNTPD]
*************************************************************************************
```

```
changed: [db]

RUNNING HANDLER [syslog : Restart Rsyslog]
********************************************************************

changed: [db]

PLAY [web_server]
*****************************************************************************
****************

skipping: no hosts matched

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

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

Perfeito, a máquina está configurada para operar em nosso *datacenter*.

3. Agora, acesse a máquina `db` como o usuário `root`. Vamos instalar o SGBD MariaDB:

```
# usermod -e -1 root ; apt-get install -y --no-install-recommends mariadb-server ;
usermod -e 0 root
```

4. Vamos criar uma base de dados para nosso website, com o nome `seg10web`. Para acessar a base, criaremos um usuário `seg10user`, com senha `seg10pass`. Execute os comandos a seguir:

```
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

```
MariaDB [(none)]> create database seg10web;
Query OK, 1 row affected (0.00 sec)
```

```
MariaDB [(none)]> grant all on seg10web.* to seg10user@localhost identified by
'seg10pass';
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> grant all on seg10web.* to seg10user@'10.0.42.%' identified by
'seg10pass';
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

```
MariaDB [(none)]> quit
Bye
```

Note que demos permissão para login a partir de *localhost*, e também da rede 10.0.42.0/24 — logins oriundos de qualquer outro IP serão negados.

5. O MariaDB escuta apenas a conexões vindas de *localhost*, por padrão. Vamos corrigir isso:

```
# sed -i 's/^\(bind-address[\t ]*\=\).*/\1 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-
server.cnf
```

```
# systemctl restart mariadb
```

Note que o SGBD está escutando em todas as interfaces:

```
# ss -tnlp | grep 3306
LISTEN     0      80           *:3306                     *:*
users:(("mysqld",pid=3579,fd=17))
```

Essa é uma limitação do MariaDB/MySQL — apenas uma interface pode ser especificada na diretiva `bind-address` do arquivo de configuração. Se for necessário escutar em mais de uma interface de rede, e necessário especificar o *catch-all* 0.0.0.0 (referência: https://mariadb.com/kb/en/library/configuring-mariadb-for-remote-client-access/).

6. Para consertar essa limitação, vamos usar o firewall local. Insira uma regra que permite que as máquinas *localhost*, `www1` e `www2` conectem-se ao SGBD, e nenhuma outra:

```
# iptables -A INPUT -i lo -p tcp -m tcp --dport 3306 -j ACCEPT
```

```
# iptables -A INPUT -s 10.0.42.5,10.0.42.6 -p tcp -m tcp --dport 3306 -j ACCEPT
```

```
# iptables -A INPUT -p tcp --dport 3306 -j DROP
```

7. Para manter as regras entre *reboots*, instale o pacote `iptables-persistent`:

```
# apt-get install -y iptables-persistent
```

Na instalação do pacote, quando perguntado, responda:

Tabela 1. Configurações do iptables-persistent

Pergunta	Resposta
Salvar as regras IPv4 atuais?	Sim
Salvar as regras IPv6 atuais?	Sim

4) Configuração do servidor web www1

1. Agora, vamos para o primeiro servidor web. Clone a máquina `debian-template` para uma de nome `www1`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.5/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, 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 www1 -i 10.0.42.5 -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. Para aplicar o *baseline* de segurança via Ansible à máquina `www1`, repita o que fizemos no passo (2) da atividade (2) desta sessão:

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

```
$ sed -i '/\[srv\]/a www1' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l www1 -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l www1 ~/ansible/srv.yml
SUDO password:

(...)

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

www1                       : ok=10   changed=8    unreachable=0    failed=0
```

3. Agora, acesse a máquina `www1` como o usuário `root`:

```
# hostname ; whoami
www1
root
```

4. Vamos instalar as dependências para o correto funcionamento do CMS Wordpress em nosso servidor:

```
# apt-get install -y nginx php-fpm php-mysql
```

5. Primeiro, vamos configurar o servidor web Nginx. Apague a configuração-padrão de websites publicados do Nginx:

```
# rm /etc/nginx/sites-enabled/default
```

Crie o arquivo novo `/etc/nginx/sites-available/seg10` com o seguinte conteúdo:

```
upstream php {
        server unix:/run/php/php7.0-fpm.sock;
}

server {
        server_name seg10web.intnet;
        root /var/www/seg10;
        index index.php;

        location = /favicon.ico {
                log_not_found off;
                access_log off;
        }

        location = /robots.txt {
                allow all;
                log_not_found off;
                access_log off;
        }

        location / {
                try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
                include fastcgi.conf;
                fastcgi_intercept_errors on;
                fastcgi_pass php;
                fastcgi_buffers 16 16k;
                fastcgi_buffer_size 32k;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
                expires max;
                log_not_found off;
        }
}
```

Habilite-o na configuração do Nginx:

```
# p=$PWD ; cd /etc/nginx/sites-enabled/ ; ln -s ../sites-available/seg10 . ; cd $p
; unset p
```

Reinicie o Nginx:

```
# systemctl restart nginx.service
```

6. O Wordpress recomenda uma ligeira alteração na configuração do PHP-FPM, como se segue:

```
# sed -i 's/^;\(cgi\.fix\_pathinfo=\).*/\10/' /etc/php/7.0/fpm/php.ini
```

Reinicie o *daemon*:

```
# systemctl restart php7.0-fpm.service
```

7. Vamos fazer o download do Wordpress, desempacotá-lo e renomear o diretório de acordo com a configuração do Nginx:

```
# cd /var/www/ ; \
  wget -q https://wordpress.org/latest.tar.gz ; \
  tar zxf latest.tar.gz ; \
  rm latest.tar.gz ; \
  mv wordpress seg10 ; \
  chown -R www-data. /var/www
```

```
# ls -1 /var/www/
html
seg10
```

8. Para conseguir acessar a máquina `www1` através do IP público do firewall, teremos que fazer alguns ajustes nas regras atuais. Acesse a máquina `ns1` como `root`:

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

Para que consigamos atingir a máquina `www1` será necessário criar uma regra de DNAT na tabela `nat`, *chain* PREROUTING, além de uma regra na tabela `filter`, *chain* FORWARD, correspondente. Mapearemos a porta externa 80/TCP para a porta interna 80/TCP, sem alterações.

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.5
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.5/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

9. Perfeito! Em sua máquina física, abra o navegador e aponte-o para o IP da interface `enp0s3` da máquina `ns1`, o IP público do nosso *datacenter* simulado. Você deverá ver a tela de instalação inicial do Wordpress, como se segue:

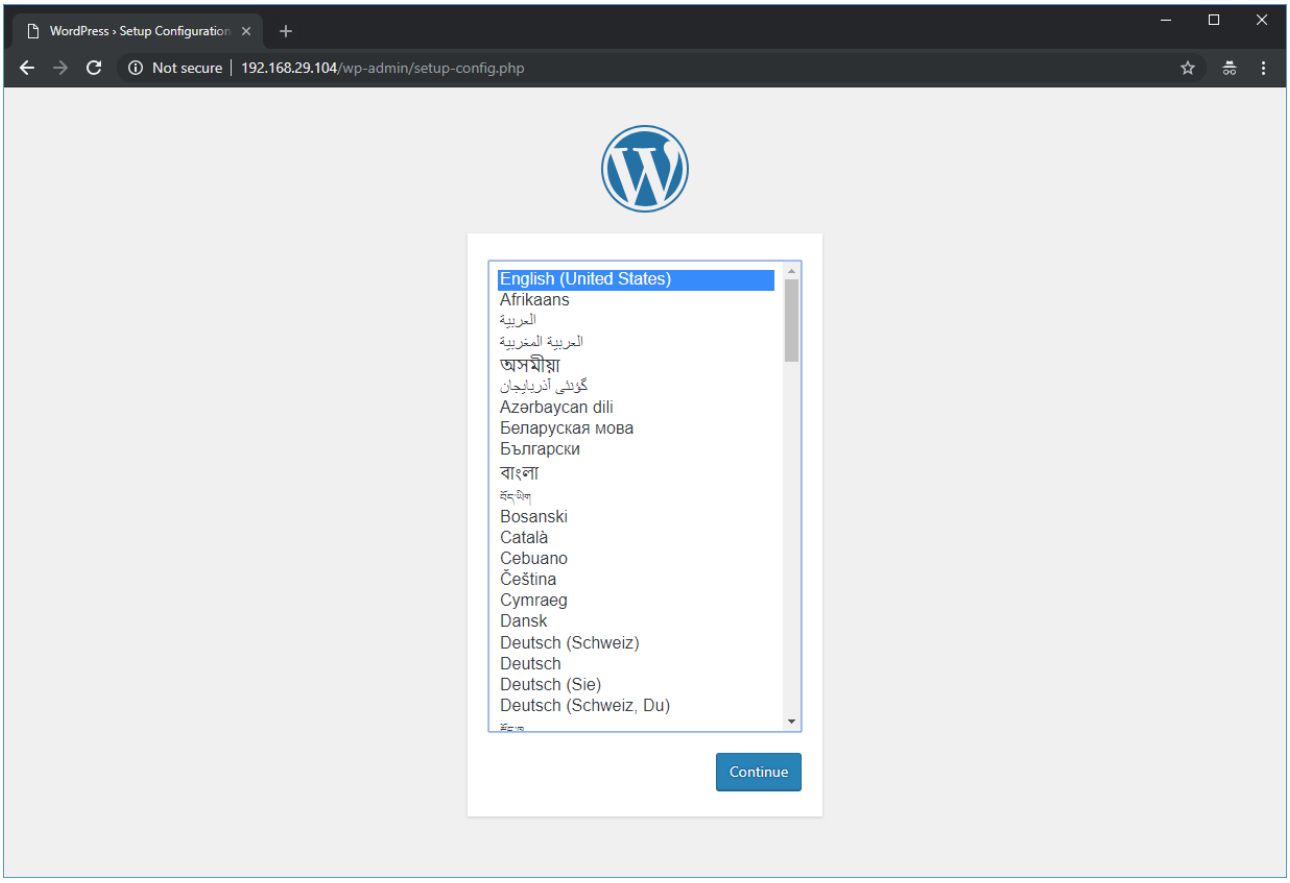

Figura 1. Tela inicial de instalação do Wordpress

Escolha o idioma *Português do Brasil*, e clique em *Continuar*. Na tela seguinte, clique em *Vamos lá!*.

Na tela de configuração da conexão com o banco de dados, digite `seg10web` em *Nome do banco de dados*; para *Nome de usuário*, informe `seg10user`; em *Senha*, `seg10pass`; para *Servidor do banco de dados*, informe `db.intnet`; para o *Prefixo da tabela*, mantenha o padrão `wp_`. Sua tela deverá ficar assim:

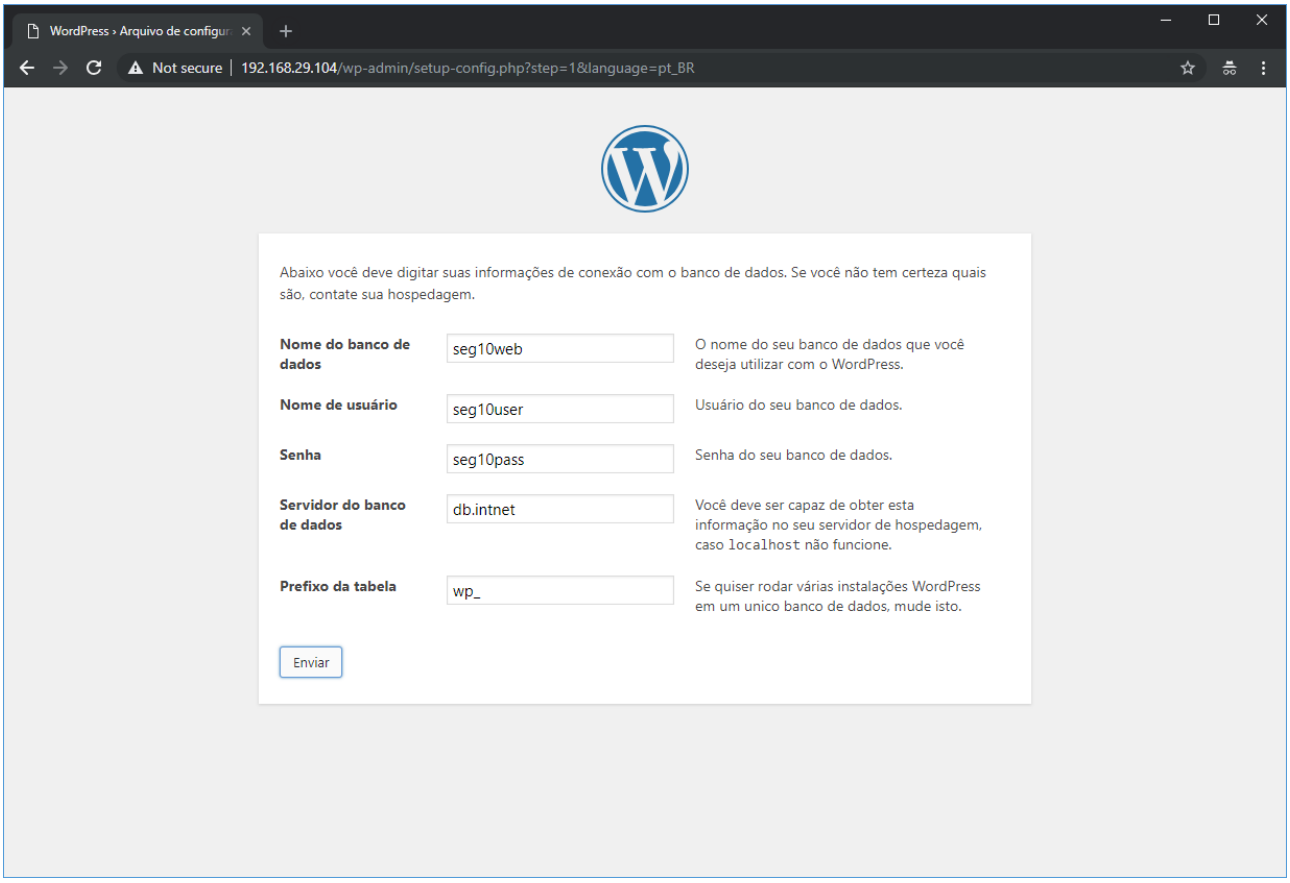

Figura 2. Configuração da conexão com o banco de dados

Clique em *Enviar*, e depois em *Instalar*.

Na tela de informações do site, digite `Seg10 Web` para o *Título do site*; em *Nome de usuário*, defina `admin`; para *Senha*, escolha `rnpesr123`; em *O seu e-mail*, informe `seg10web@int.net`; finalmente, mantenha a caixa *Visibilidade nos mecanismos de busca* desmarcada. Ao final do processo, sua tela deverá estar da seguinte forma:

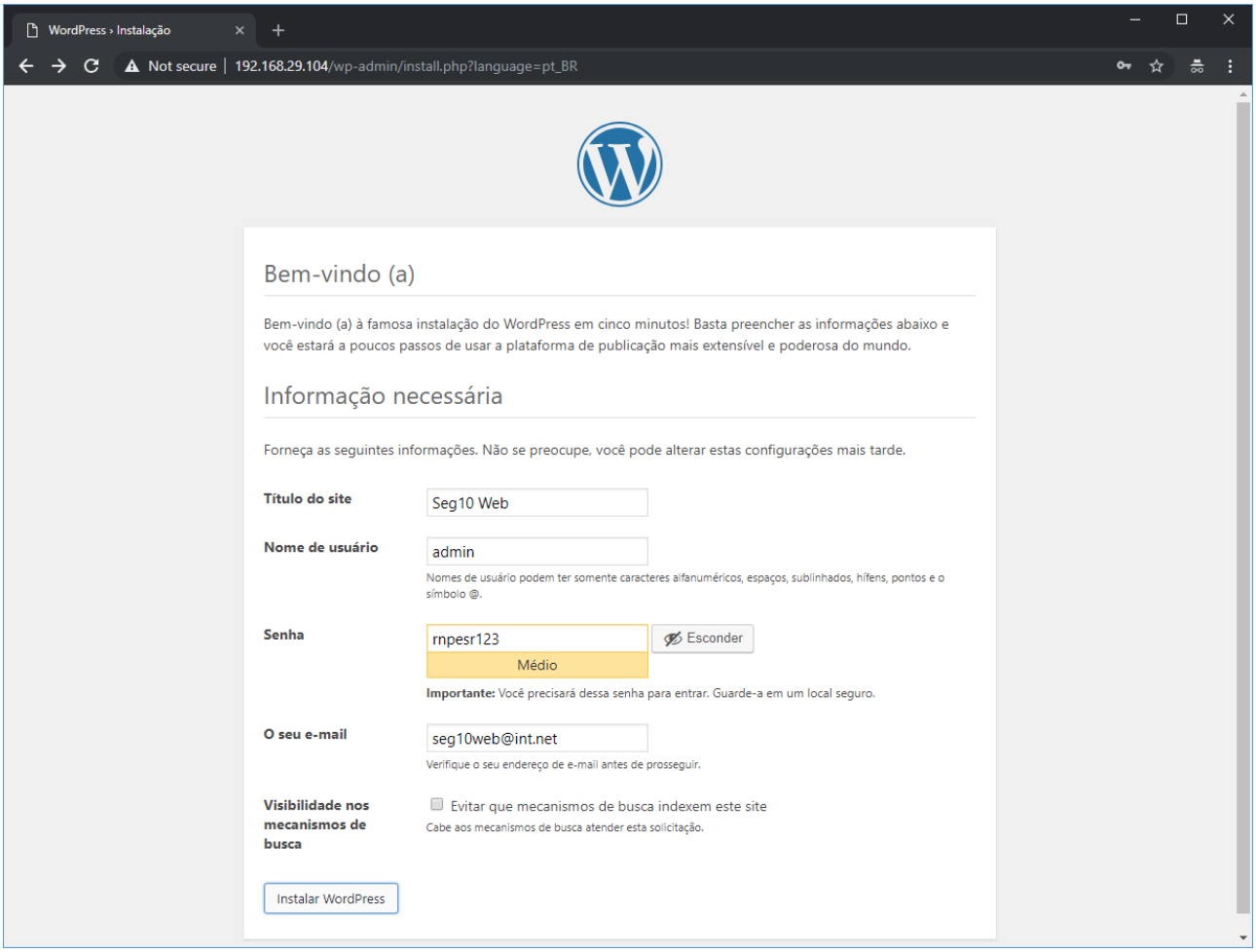

Figura 3. Configuração das informações do site

Clique em *Instalar WordPress*. Ao final do processo de instalação, clique em *Acessar* para entrar na interface administrativa do Wordpress. Alternativamente, digite o endereço IP da interface `enp0s3` da máquina `ns1` na URL do navegador para acessar a página inicial do nosso portal, como mostra a figura abaixo:

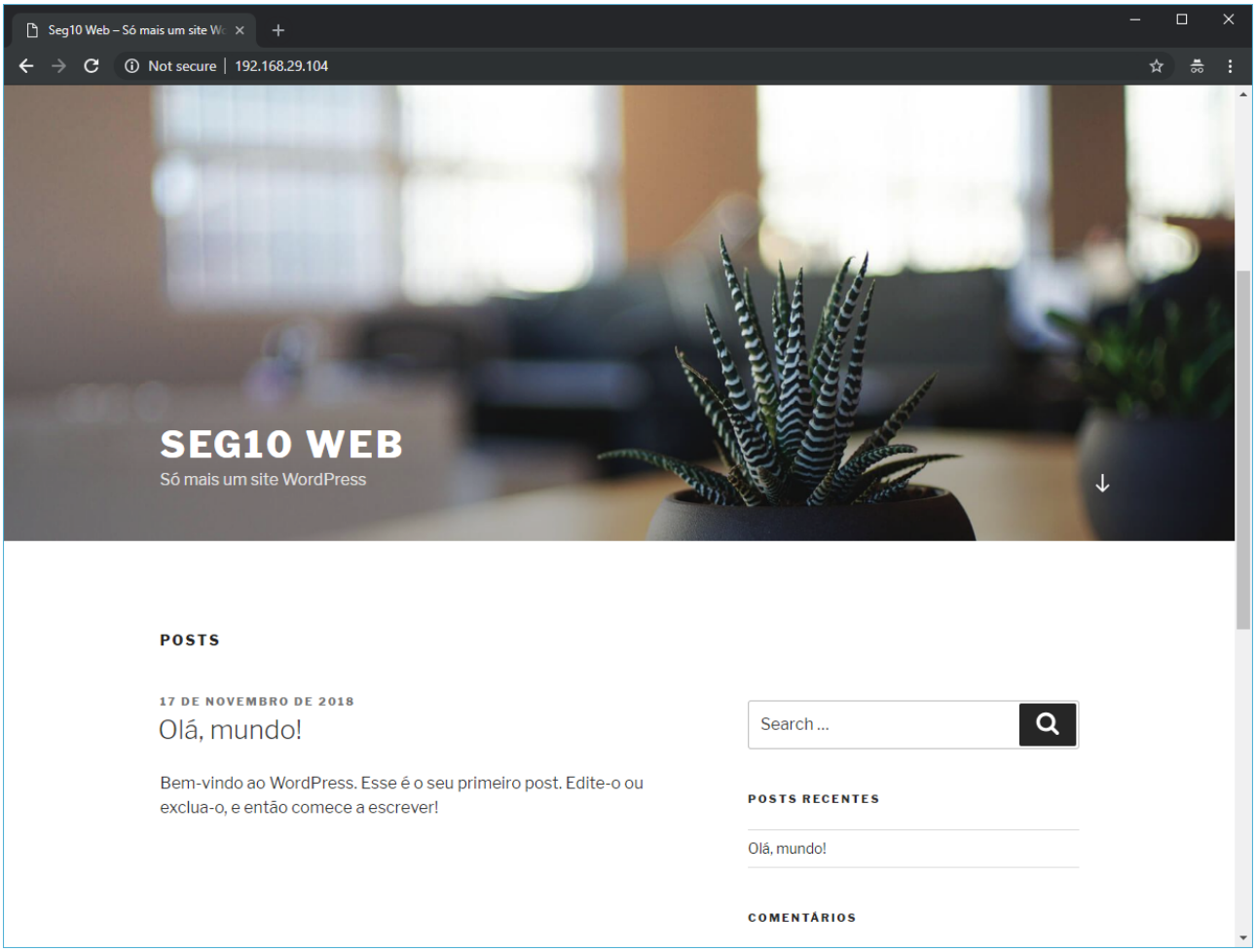

Figura 4. Página inicial do portal Seg10 Web

5) Configuração automática do servidor web www2

1. Para o segundo servidor web, a ideia é automatizar sua configuração completamente usando o Ansible. Antes de mais nada, clone a máquina `debian-template` para uma de nome `www2`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.6/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, 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 www2 -i 10.0.42.6 -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. Para aplicar o *baseline* de segurança via Ansible à máquina `www2`, repita o que fizemos no passo (2) da atividade (2) desta sessão:

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

```
$ sed -i '/\[srv\]/a www2' ~/ansible/hosts
```

```
$ ansible-playbook -i ~/ansible/hosts -l www2 -Ke ansible_become_method=su
~/ansible/srv.yml ; ansible-playbook -i ~/ansible/hosts -l www2 ~/ansible/srv.yml
SUDO password:

(...)

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

www2                       : ok=10   changed=8    unreachable=0    failed=0
```

3. Agora, ainda como o usuário `ansible` na máquina `client`, vamos criar uma *role* para automatizar totalmente a instalação e configuração do portal *Seg10 Web* na máquina `www2` (e em quaisquer outros nodos web que venhamos a adicionar no futuro):

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles seg10-web
- seg10-web was created successfully
```

4. Vamos começar adicionando o arquivo de configuração do servidor Nginx — crie o arquivo novo `/home/ansible/ansible/roles/seg10-web/files/nginx_seg10` com o seguinte conteúdo:

```
upstream php {
        server unix:/run/php/php7.0-fpm.sock;
}

server {
        server_name seg10web.intnet;
        root /var/www/seg10;
        index index.php;

        location = /favicon.ico {
                log_not_found off;
                access_log off;
        }

        location = /robots.txt {
                allow all;
                log_not_found off;
                access_log off;
        }

        location / {
                try_files $uri $uri/ /index.php?$args;
        }

        location ~ \.php$ {
                include fastcgi.conf;
                fastcgi_intercept_errors on;
                fastcgi_pass php;
                fastcgi_buffers 16 16k;
                fastcgi_buffer_size 32k;
        }

        location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
                expires max;
                log_not_found off;
        }
}
```

5. O próximo passo é o arquivo de *tasks*. Edite o arquivo `/home/ansible/ansible/roles/seg10-web/tasks/main.yml` com o seguinte conteúdo:

```
---
- name: Install nginx and deps
  apt:
    name: '{{ packages }}'
    state: present
    update_cache: true
    install_recommends: no
  vars:
```

```
    packages:
    - nginx
    - php-fpm
    - php-mysql
  notify:
    - Start nginx
    - Start php-fpm

- name: Add seg10 config
  copy:
    src: nginx_seg10
    dest: /etc/nginx/sites-available/seg10
    owner: root
    group: root

- name: Disable default site configuration
  file:
    dest: /etc/nginx/sites-enabled/default
    state: absent

- name: Enable seg10 site config
  file:
    src: /etc/nginx/sites-available/seg10
    dest: /etc/nginx/sites-enabled/seg10
    state: link

- name: Copy web root
  synchronize:
    src: seg10
    dest: /var/www
    recursive: yes

- name: Web Root Permissions
  file:
   dest: /var/www/seg10
   mode: u=rwX,g=rX,o=rX
   state: directory
   owner: www-data
   group: www-data
   recurse: yes
  notify:
    - Restart nginx

- name: Adjust php-fpm fix_pathinfo
  replace:
    path: /etc/php/7.0/fpm/php.ini
    regexp: '^;(cgi\.fix_pathinfo).*$'
    replace: '\1=0'
  notify:
    - Restart php-fpm
```

A lista de tarefas acima irá instalar os pacotes na máquina-alvo, copiar a configuração do Nginx, desativar a configuração-padrão e criar o symlink apropriado, sincronizar usando o `rsync` os arquivos do website localizados na pasta `files` para o diretório `/var/www` no destino (copiaremos esses arquivos a seguir), configurará as permissões do *webroot* e ajustará a variável `cgi.fix_pathinfo` do PHP-FPM.

6. Temos *handlers* no arquivo acima, como visto. Edite `/home/ansible/ansible/roles/seg10-web/handlers/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Start nginx
3   service:
4     name: nginx
5     state: started
6
7 - name: Start php-fpm
8   service:
9     name: php7.0-fpm
10    state: started
11
12 - name: Restart nginx
13   service:
14    name: nginx
15    state: restarted
16
17 - name: Restart php-fpm
18   service:
19    name: php7.0-fpm
20    state: restarted
```

7. Para que a sincronização dos arquivos do website funcione, temos que obtê-los — use o `scp` para fazer isso:

```
$ scp -rpq www1:/var/www/seg10 /home/ansible/ansible/roles/seg10-web/files/
```

8. Temos que ajustar o escopo das máquinas-alvo da nossa nova *role*. Adicione o novo grupo `seg10_server` ao inventário do Ansible:

```
$ cat << EOF >> ~/ansible/hosts

[seg10_server]
www1
www2
EOF
```

Ao final, o inventário deverá ficar assim:

```
$ cat ~/ansible/hosts
[srv]
www2
www1
db
ns1
ns2
nfs
192.168.42.2
log

[ntp_server]
log

[log_server]
log

[web_server]
log

[seg10_server]
www1
www2
```

9. Vamos criar um novo *playbook* para utilizar a *role* que acabamos de criar:

```
$ cat << EOF >> ~/ansible/seg10web.yml
- hosts: seg10_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - seg10-web
EOF
```

10. Hora da verdade! Execute o *playbook*:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/seg10web.yml

PLAY [seg10_server]
*********************************************************************************
**************


TASK [Gathering Facts]
*********************************************************************************
***********
```

```
ok: [www1]
ok: [www2]

TASK [seg10-web : Install nginx and deps]
*********************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Add seg10 config]
*************************************************************************************

changed: [www2]
ok: [www1]

TASK [seg10-web : Disable default site configuration]
*************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Enable seg10 site config]
*********************************************************************

ok: [www1]
changed: [www2]

TASK [seg10-web : Copy web root]
*************************************************************************************
*

changed: [www1]
changed: [www2]

TASK [seg10-web : Web Root Permissions]
*********************************************************************

changed: [www1]
changed: [www2]

TASK [seg10-web : Adjust php-fpm fix_pathinfo]
*********************************************************************

changed: [www2]
ok: [www1]

RUNNING HANDLER [seg10-web : Start nginx]
*********************************************************************

ok: [www2]
```

```
RUNNING HANDLER [seg10-web : Start php-fpm]
*********************************************************************

ok: [www2]

RUNNING HANDLER [seg10-web : Restart nginx]
*********************************************************************

changed: [www1]
changed: [www2]

RUNNING HANDLER [seg10-web : Restart php-fpm]
*********************************************************************

changed: [www2]

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

www1                       : ok=9    changed=3    unreachable=0    failed=0
www2                       : ok=12   changed=9    unreachable=0    failed=0
```

11. Terá funcionado? Vamos alterar o firewall para redirecionar os pacotes chegando na porta 80/TCP para a máquina `www2`, ao invés da `www1`. Acesse a máquina `ns1` como `root`:

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

Remova as regras que criamos originalmente para a máquina `www1`:

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.5
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.5/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Insira as mesmas regras no firewall, mas agora para a máquina `www2`:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.6
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.6/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

12. Vamos testar. Na máquina `www2`, como o usuário `root`, monitore o log de acesso do Nginx:

```
# hostname ; whoami
www2
root
```

```
# tail -f -n0 /var/log/nginx/access.log
```

No navegador da sua máquina física, acesse novamente o endereço IP da interface `enp0s3` da máquina `ns1`... o **mesmo** website que havíamos configurado antes aparece! Volte a visualizar o log de acesso do Nginx instalado na máquina `www2`:

```
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET / HTTP/1.1" 200 20854 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64;x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.102 Safari/537.36"
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET /wp-
content/themes/twentyseventeen/style.css?ver=4.9.8 HTTP/1.1" 200 83401
"http://192.168.29.104/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36"
192.168.29.106 - - [17/Nov/2018:03:12:38 -0200] "GET /wp-
includes/js/jquery/jquery.js?ver=1.12.4 HTTP/1.1" 200 97184"http://192.168.29.104/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/70.0.3538.102 Safari/537.36"
```

De fato, os acessos estão chegando à nova máquina, e o website está funcionando como esperado. Note que não fizemos **qualquer** modificação manual na máquina `www2` — todos as dependências, arquivos de configuração e arquivos do site foram instalados de forma automática pelo Ansible. Se quiséssemos lançar agora outros dois, três ou dez nodos de atendimento web, bastaria adicionar as máquinas ao inventário do Ansible e disparar a *role*; toda a configuração é feita de forma automática.

13. Como fizemos uma nova *role* para os sistemas web, é uma boa ideia enviar nossas modificações para o repositório Git central. Acesse a máquina `client` como o usuário `ansible`:

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

Envie as alterações, adicionando uma mensagem significativa:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para configuracao
automatica de servidores web do portal seg10' ; git push

(...)

Counting objects: 1665, done.
Compressing objects: 100% (1632/1632), done.
Writing objects: 100% (1665/1665), 8.95 MiB | 8.87 MiB/s, done.
Total 1665 (delta 170), reused 0 (delta 0)
remote: Resolving deltas: 100% (170/170), completed with 1 local object.
To nfs:/home/ansible/ansible.git
   467f80d..bd3fbc2  master -> master
```

6) Configuração do balanceador de carga

1. Vamos para a instalação e configuração do balanceador de carga. Clone a máquina `debian-template` para uma de nome `lb`, com uma única interface de rede conectada à DMZ. O IP da máquina será 10.0.42.8/24.

   Concluída a clonagem, ligue a máquina e logue como `root`. Depois, 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 lb -i 10.0.42.8 -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 `lb`, repetindo o que fizemos no passo (2) da atividade (2) desta sessão:

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

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

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

(...)

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

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

3. Acesse a máquina `lb` como o usuário `root`:

```
# hostname ; whoami
lb
root
```

4. Vamos instalar o balanceador de carga: usaremos o software HAProxy, uma solução *open-source* para balanceamento de carga e alta disponibilidade para aplicações baseadas em TCP e HTTP, reconhecido por sua excepcional performance e eficiência.

```
# apt-get install -y haproxy
```

5. Edite o arquivo de configuração do HAProxy, `/etc/haproxy/haproxy.cfg`, deixando-o exatamente com o conteúdo a seguir:

```
 1 global
 2   log /dev/log    local0
 3   log /dev/log    local1 notice
 4   chroot /var/lib/haproxy
 5   stats socket /run/haproxy/admin.sock mode 660 level admin
 6   stats timeout 30s
 7   user haproxy
 8   group haproxy
 9   daemon
10   maxconn 2048
11
12   # Default SSL material locations
13   ca-base /etc/ssl/certs
14   crt-base /etc/ssl/private
15
16   # Default ciphers to use on SSL-enabled listening sockets.
17   # For more information, see ciphers(1SSL). This list is from:
18   #  https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers/
19   # An alternative list with additional directives can be obtained from
```

```
20   #  https://mozilla.github.io/server-side-tls/ssl-config-
generator/?server=haproxy
21   ssl-default-bind-ciphers ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH
+AES128:DH+AES:RSA+AESGCM:RSA+AES:!aNULL:!MD5:!DSS
22   ssl-default-bind-options no-sslv3
23   tune.ssl.default-dh-param 2048
24
25 defaults
26   log     global
27   mode    http
28   option  httplog
29   option  dontlognull
30   option  forwardfor
31   option  http-server-close
32   timeout connect 5000
33   timeout client  50000
34   timeout server  50000
35   errorfile 400 /etc/haproxy/errors/400.http
36   errorfile 403 /etc/haproxy/errors/403.http
37   errorfile 408 /etc/haproxy/errors/408.http
38   errorfile 500 /etc/haproxy/errors/500.http
39   errorfile 502 /etc/haproxy/errors/502.http
40   errorfile 503 /etc/haproxy/errors/503.http
41   errorfile 504 /etc/haproxy/errors/504.http
42
43   stats enable
44   stats uri /stats
45   stats realm Haproxy\ Statistics
46   stats auth admin:rnpesr123
47
48 frontend www-http
49   bind 10.0.42.8:80
50   reqadd X-Forwarded-Proto:\ http
51   default_backend www-backend
52
53 frontend www-https
54   bind 10.0.42.8:443 ssl crt /etc/ssl/private/seg10web.pem
55   reqadd X-Forwarded-Proto:\ https
56   default_backend www-backend
57
58 backend www-backend
59   redirect scheme https if !{ ssl_fc }
60   server www1 www1.intnet:80 check
61   server www2 www2.intnet:80 check
```

Em linhas gerais, definimos dois *frontends* de atendimento, um para conexões HTTP e outro para HTTPS. Ambos enviam suas requisições para o mesmo *backend*, o qual é atendido pelos servidores `www1` e `www2`, escutando na porta 80/TCP.

6. Note que para o atendimento das requisições HTTPS (o chamado *SSL offloading*, ou *SSL*

termination), devemos configurar um par de chaves pública/privada auto-assinadas em formato PEM. Vamos fazer isso:

```
# openssl req -x509 -nodes -days 730 -newkey rsa:4096 -keyout
/etc/ssl/private/seg10web.key -out /etc/ssl/certs/seg10web.crt
Generating a 4096 bit RSA private key
..............................................++
.....................................................................................
.....................................................................................
.....................................................................................
.....................................++
writing new private key to '/etc/ssl/private/seg10web.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:DF
Locality Name (eg, city) []:Brasilia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:RNP
Organizational Unit Name (eg, section) []:ESR
Common Name (e.g. server FQDN or YOUR name) []:seg10web.intnet
Email Address []:seg10web@int.net
```

Concatene as duas chaves em um arquivo em formato PEM:

```
# cat /etc/ssl/certs/seg10web.crt /etc/ssl/private/seg10web.key >
/etc/ssl/private/seg10web.pem
```

```
# chmod 600 /etc/ssl/private/seg10web.pem
```

7. Agora que tudo está configurado, reinicie o HAProxy:

```
# systemctl restart haproxy.service
```

8. Temos que alterar o firewall uma última vez, desta vez para redirecionar os pacotes chegando na porta 80/TCP para a máquina `lb`. Acesse a máquina `ns1` como `root`:

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

Remova as regras que criamos para a máquina `www2`:

```
# iptables -t nat -D PREROUTING -i enp0s3 -p tcp -m tcp --dport 80 -j DNAT --to
-destination 10.0.42.6
```

```
# iptables -D FORWARD -i enp0s3 -d 10.0.42.6/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Adicione-as para o balanceador de carga — observe que como o HAProxy escuta tanto na porta 80/TCP como na 443/TCP, iremos redirecioná-las ambas:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m multiport --dports 80,443 -j
DNAT --to-destination 10.0.42.8
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.8/32 -p tcp -m multiport --dports 80,443
-j ACCEPT
```

Salve a configuração do firewall desta vez, já que ela será definitiva:

```
# /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.
```

9. Vamos ao teste! No navegador da sua máquina física, acesse novamente o endereço IP da interface `enp0s3` da máquina `ns1`. De imediato, notamos uma diferença: o acesso está sendo feito em HTTPS, já que a diretiva `redirect scheme https if !{ ssl_fc }` do arquivo de configuração do HAProxy força esse protocolo aos clientes que estão se conectando.

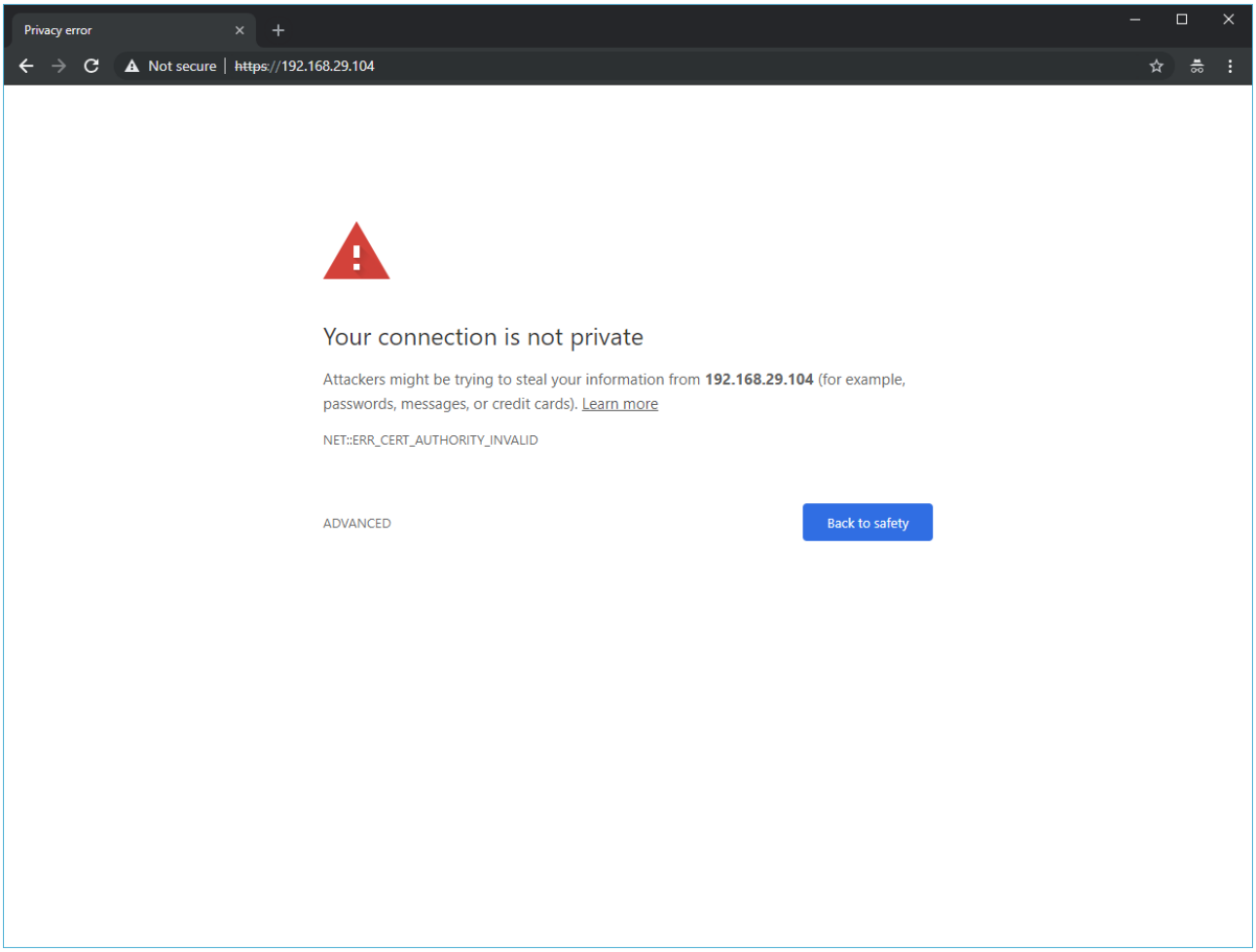

Figura 5. Acesso desviado para HTTPS pelo HAProxy

Aceite o certificado auto-assinado, e prossiga para a página do portal *Seg10 Web*. Temos um problema! O CSS da página não foi carregado corretamente:

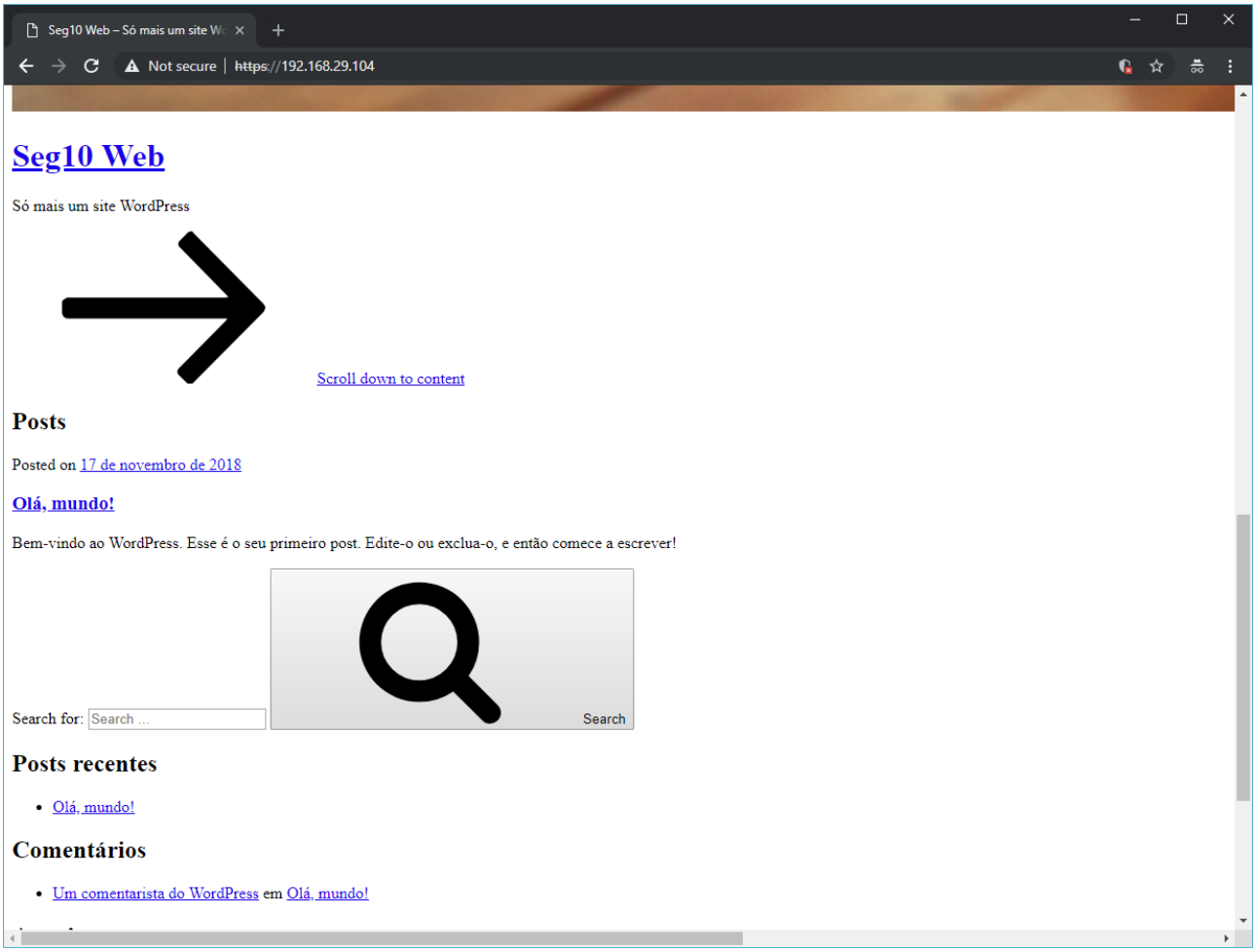

Figura 6. CSS não foi carregado na conexão HTTPS

O que fazemos? Felizmente, esse problema é facilmente solucionável. Acesse a máquina `client` como o usuário `ansible`.

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

Basta inserir a linha `if ($_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https') $_SERVER['HTTPS']='on';` antes da última linha do arquivo `/home/ansible/ansible/roles/seg10-web/files/seg10/wp-config.php`. O comando `sed` a seguir faz a alteração de forma direta:

```
$ sed -i "/wp-settings/i if (\$_SERVER['HTTP_X_FORWARDED_PROTO'] == 'https')
\$_SERVER['HTTPS']='on';" /home/ansible/ansible/roles/seg10-web/files/seg10/wp-
config.php
```

Uhm... como distribuir essas mudanças? Simples: basta re-executar nossa *role* `seg10-web` no Ansible:

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/seg10web.yml

(...)

TASK [seg10-web : Copy web root]
*********************************************************************************
*

changed: [www1]
changed: [www2]

TASK [seg10-web : Web Root Permissions]
*********************************************************************

changed: [www1]
changed: [www2]

(...)

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

www1                       : ok=9    changed=3    unreachable=0    failed=0
www2                       : ok=9    changed=3    unreachable=0    failed=0
```

O Ansible detecta que a cópia local está diferente da observada nas máquinas remotas, e sobrescreve o arquivo `/var/www/seg10/wp-config.php` em ambos os servidores web com as modificações locais.

Vamos verificar se isso solucionou o problema no portal. Recarregue a página web no navegador em sua máquina física:

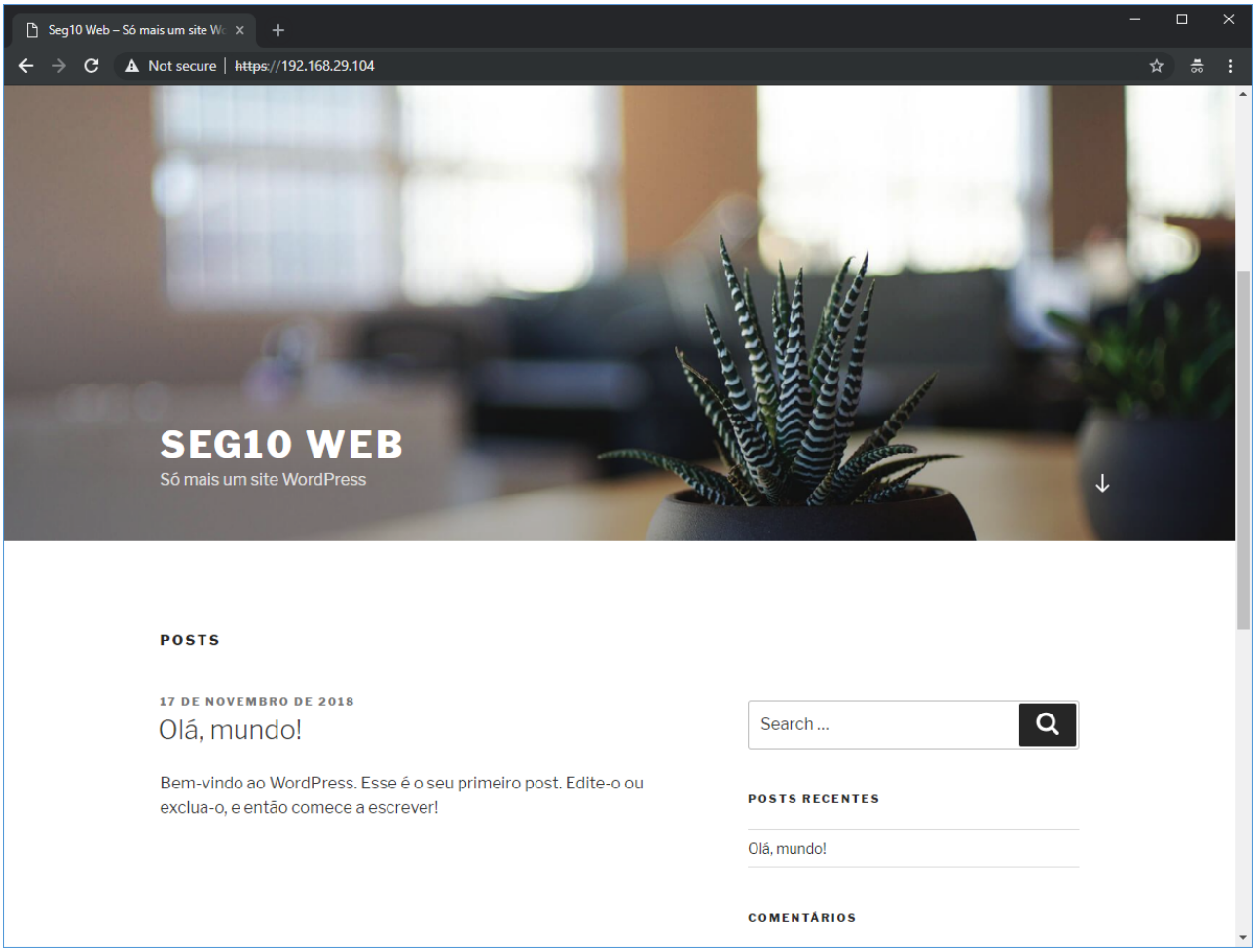

Figura 7. Correção na visualização da página via HTTPS

Excelente, tudo funcionando a contento.

10. O HAProxy possui uma página de estatísticas bastante interessante para acompanhar o funcionamento dos diferentes *frontends* e *backends* configurados. Adicione o sufixo `/stats` à URL do seu navegador, e faça login com o usuário `admin` e senha `rnpesr123` quando solicitado. Você verá a página a seguir:

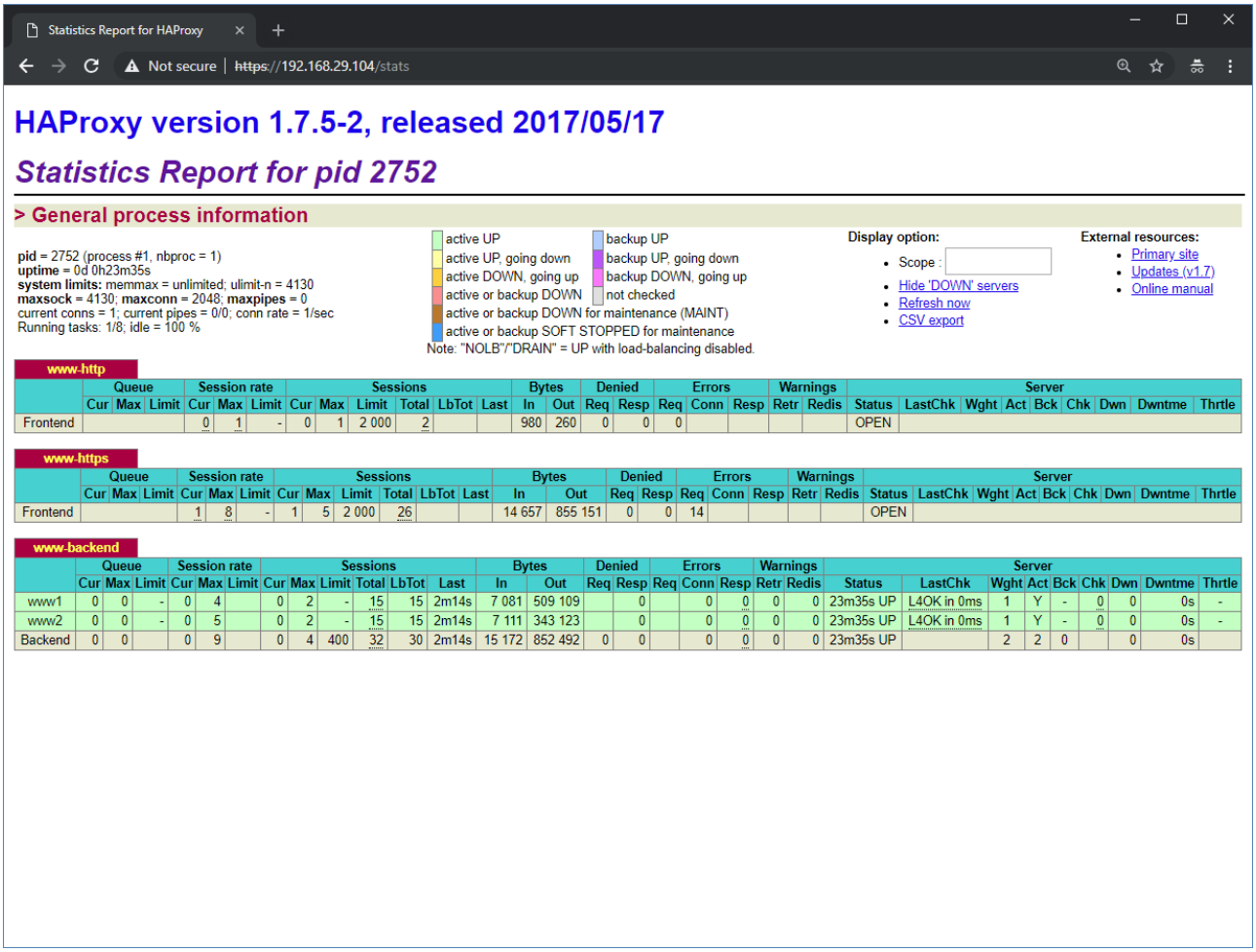

Figura 8. Página de estatísticas do HAProxy

A partir desta página, é possível verificar quais *frontends* e *backends* estão ativos, quais estão inativos ou com problemas de acesso, e quantos bytes e sessões foram trafegados para cada um dos *hosts* envolvidos.

Tente recarregar a portal *Seg10 Web* algumas vezes, e confira o número de sessões registradas para cada um dos *backends*: o HAProxy envia requisições alternadas para cada um deles, segundo um algoritmo *round robin*. Com o HAProxy, é fácil desativar seletivamente alguns *backends* para atualizações ou manutenção, e levantá-los posteriormente, mantendo a disponibilidade do serviço e minimizando o impacto para os clientes.

11. Encerradas as nossas atividades com os servidores, recomenda-se que o aluno mantenha as máquinas `www1`, `www2`, `db` e `lb` desligadas a partir desta sessão. Apesar de as máquinas individualmente não ocuparem tantos recursos de memória e processamento, o fato de não precisarmos mais usá-las e a exigência de alteração de contexto da CPU para atendê-las faz com que a liberação de recursos, nesse caso, seja recomendável.