

# Sessão 6: Registro e correlacionamento de eventos

A gestão de registros de eventos em sistemas computacionais é com certeza um dos temas mais difíceis — e importantes — de se tratar corretamente em um *datacenter* de grande porte. Imagine a seguinte situação: você quer monitorar todos os eventos acontecendo em um servidor web de alto tráfego em sua organização, buscando por anomalias; para isso, você monitora o log de acesso do servidor web com um comando como `tail -f`, e observa as mensagens voando em alta velocidade pela tela — após alguns poucos segundos fica claro que o enorme volume de mensagens é impossível de ser processado por um ser humano... são eventos demais! Agora, multiplique esse desafio por múltiplos servidores e máquinas virtuais no *datacenter*, cada qual com uma infinidade de serviços instalados: como dar conta desse trabalho?

Ferramentas SIEM (do inglês, *Security Information and Event Management*) são soluções de software que permitem que os eventos gerados por diversas aplicações de segurança (tais como firewalls, proxies, sistemas de prevenção a intrusão e antivírus) sejam coletados, normalizados, armazenados e correlacionados; essa gestão possibilita uma rápida identificação e resposta aos incidentes. Os SIEMs combinam ferramentas do tipo SIM (*Security Information Management*) e SEM (*Security Event Manager*) — enquanto ferramentas SEM oferecem monitoramento em tempo real dos eventos de segurança, coletando e agregando os dados (com resposta automática em alguns casos), ferramentas SIM oferecem análise histórica dos eventos de segurança, também coletando e correlacionando os eventos, porém não em tempo real; isso permite consultas mais complexas ao repositório.

Dentre as diversas funcionalidades desejáveis em ferramentas SIEM para auxiliar a gestão de logs corporativos, destacamos algumas:

- Acesso em tempo real, centralizado e consistente a todos os logs e eventos de segurança, independente do tipo de tecnologia e fabricante.
- Correlação de logs de tecnologias heterôgeneas, conectando atributos comuns e/ou significativos entre as fontes, de modo a transformar os dados em informação útil.
- Identificação de comportamentos, incidentes, fraudes, anomalias e quebras de *baseline* de segurança.
- Alertas e notificações que podem ser disparadas automaticamente no caso de não conformidade com as políticas de segurança e/ou normas regulatórias, ou ainda, de acordo com as regras de negócio pré-estabelecidas.
- Emissão de relatórios sofisticados sobre as condições de segurança do ambiente para equipes de SOC (*Security Operations Center*) auditoria ou resposta a incidentes.
- Retenção e indexação a longo prazo dos dados possibilitando posterior análise forense.

Nesta sessão iremos instalar e configurar o Graylog, uma das mais populares ferramentas SIEM *open-source* para armazenamento e correlação de eventos. Iremos integrá-lo com o RSyslog já instalado em nossos servidores, bem como com o sistema de autenticação centralizado via OpenLDAP. Juntamente com o Graylog, será necessário configurar um serviço NTP (*Network Time Protocol*) para manter a hora dos servidores sincronizada — a ferramenta OpenNTPD, do OpenBSD,

é uma alternativa simples e segura para solucionar esse problema. Finalmente, faremos também uso da ferramenta Snoop para registrar comandos digitados pelos usuários no terminal, permitindo análise posterior e garantia de não-repúdio nos acessos remotos via SSH.

## 1) Topologia desta sessão

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

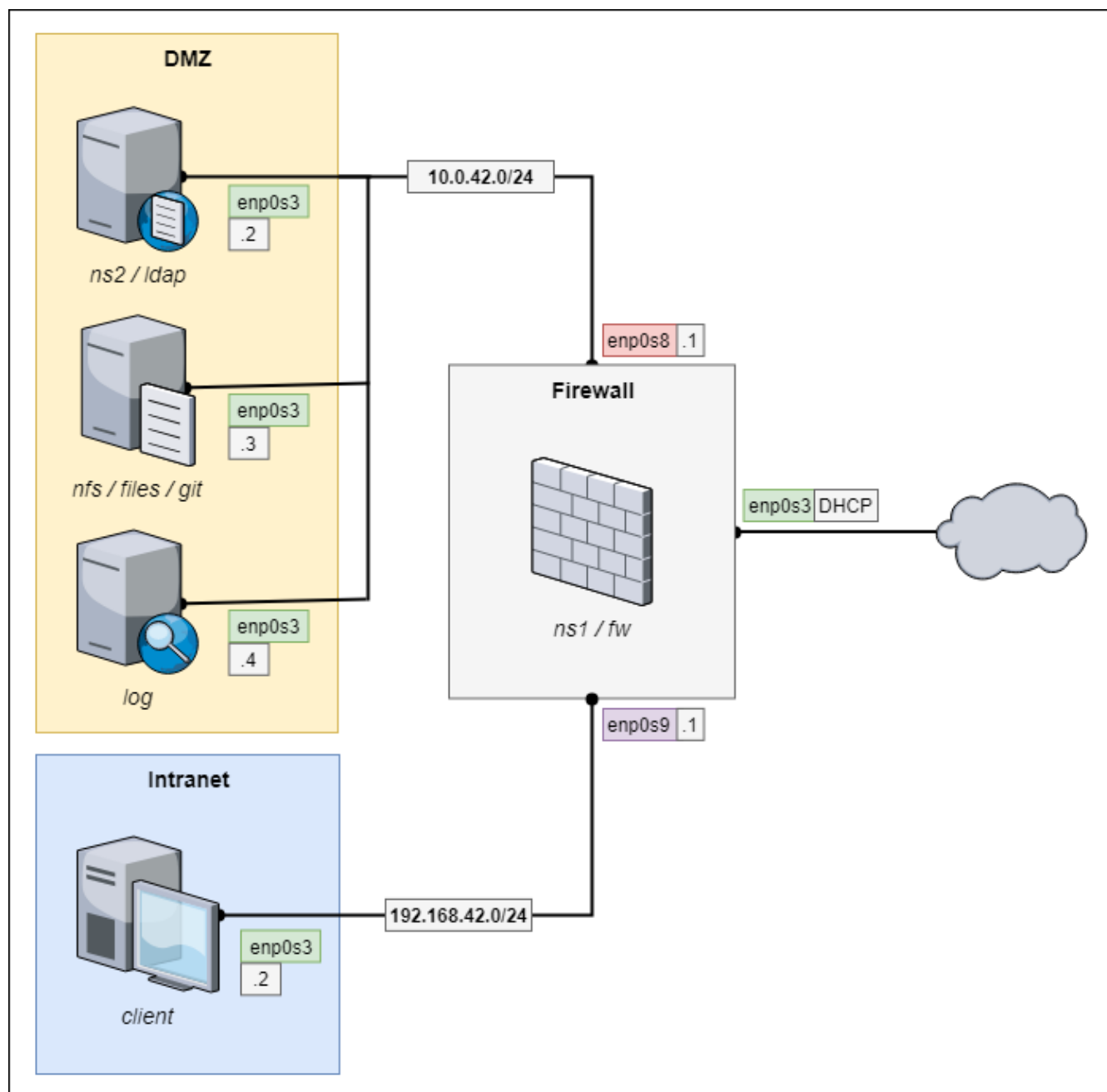


Figura 1. Topologia de rede desta sessão

Temos uma nova máquina, a saber:

- **log**, atuando como concentrador de logs dos servidores do *datacenter* e servidor NTP. Endereço IP `10.0.42.4/24`.
1. Os únicos registros DNS novos a serem criado são mapeamentos direto e reverso para a

máquina **log** — para fazer os ajustes, acesse a máquina **ns1** como o usuário **root**:

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

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

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

```
# grep log /etc/nsd/zones/intnet.zone
log      IN      A              10.0.42.4
```

Faça o mesmo para o arquivo de zona reversa:

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

```
# grep log /etc/nsd/zones/10.0.42.zone
4        IN      PTR          log.intnet.
```

Assine os arquivos de zona usando o *script* criado anteriormente:

```
# bash /root/scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 6 rrsets, 4 messages and 0 key entries
```

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

```
# dig log.intnet +short
10.0.42.4
```

```
# dig -x 10.0.42.4 +short
log.intnet.
```

## 2) Criação da VM de gestão de logs

1. Como visualizado na topologia que abre esta sessão, teremos uma máquina dedicada para gestão de logs e serviços auxiliares, como o NTP. Devemos começar clonando a máquina **debian-template** e criar uma nova, que chamaremos de **log**. Essa máquina estará conectada a uma única rede *host-only*, com o mesmo nome que foi alocado para a interface de rede da máquina virtual **ns1**, configurada durante a sessão 2, que está conectada à DMZ. O IP da máquina será 10.0.42.4/24.

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

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

```
# bash ~/scripts/changehost.sh -h log -i 10.0.42.4 -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
10.0.42.4/24 enp0s3
log
root
```

2. Agora, vamos configurar funcionamento do **sudo** na máquina **log**. Acesse a máquina **client** como o usuário **ansible**:

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

Insira a máquina **log** no inventário gerenciado pelo Ansible:

```
$ echo log >> ~/ansible/hosts
```

Execute a **role sudoers**, lembrando-se de limitar o escopo à máquina **log** e alterando o método de escalação de privilégio para o **su** (já que o **sudo** não foi configurado ainda):

```

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

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

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

ok: [log]

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

changed: [log]

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

changed: [log]

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

log                                : ok=3    changed=2    unreachable=0    failed=0

```

Fácil, não?

- Note que fizemos uma alteração, ainda que ligeira, ao repositório de configuração do Ansible: adicionamos a máquina **log** ao grupo **srv** no inventário:

```

$ cd ~/ansible/ ; git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   hosts

no changes added to commit (use "git add" and/or "git commit -a")

```

```
$ git diff hosts
diff --git a/hosts b/hosts
index f81ed68..be6cf78 100644
--- a/hosts
+++ b/hosts
@@ -3,3 +3,4 @@ ns1
    ns2
    nfs
    192.168.42.2
+log
```

Vamos adicionar essa mudança ao repositório, efetuar o *commit* e publicar as mudanças para o repositório remoto:

```
$ git add hosts
```

```
$ git commit -m 'Maquina log adicionada ao inventario do Ansible'
[master 24bcab5] Maquina log adicionada ao inventario do Ansible
1 file changed, 1 insertion(+)
```

```
$ git push
Counting objects: 3, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 350 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
0c673e4..24bcab5 master -> master
```

### 3) Ajuste das regras de firewall para o NTP

Iremos configurar a seguir um servidor NTP na máquina **log** para atuar como servidor de hora para os demais servidores do *datacenter*. O protocolo NTP opera por padrão na porta 123/UDP, para a qual não fizemos quaisquer liberações até aqui — caso não façamos a inserção de novas regras no firewall da rede, o serviço não funcionará.

Quais são os acessos a configurar? Vejamos:

- A máquina **log** deve conseguir consultar servidores NTP na Internet, exigindo regras de acesso na *chain* FORWARD da tabela **filter** e na *chain* POSTROUTING da tabela **nat**. Poderíamos limitar o conjunto de IPs de destino, mas por simplicidade iremos liberar a conexão com qualquer *host* remoto.
- Máquinas na DMZ devem conseguir acessar a máquina **log** na porta 123/UDP. Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.

c. Máquinas na Intranet devem conseguir acessar a máquina **log** na porta 123/UDP. Como esse tráfego passa **através do** firewall, devemos inserir a regra na *chain* FORWARD.

1. Vamos lá: acesse a máquina **ns1** como o usuário **root**:

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

Para atender o requisito (a), insira as regras:

```
# iptables -A FORWARD -s 10.0.42.4/32 -o enp0s3 -p udp -m udp --dport 123 -j ACCEPT
```

```
# iptables -t nat -A POSTROUTING -s 10.0.42.4/32 -o enp0s3 -p udp -m udp --dport 123 -j MASQUERADE
```

Para o requisito (b), nenhuma ação é necessária. Finalmente, para o requisito (c) insira a regra a seguir:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.4/32 -p udp -m udp --dport 123 -j ACCEPT
```

Não se esqueça de gravar as novas regras na configuração permanente do firewall:

```
# /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.
```

## 4) Configuração do NTP

Vamos ao NTP. Como estabelecido, a máquina **log** atuará como o servidor NTP da rede, mas quais serão seus clientes? Além dos servidores do *datacenter*, também iremos configurar a máquina **client** para consultar o servidor de hora na máquina **log**. Ao invés de instalar e configurar o *daemon* do OpenNTPD em cinco (!! ) máquinas diferentes, temos aqui uma excelente oportunidade para colocar nossos recém-adquiridos conhecimentos com o Ansible à prova.

1. Acesse a máquina **client** como o usuário **ansible**:

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

2. Para não termos que criar duas *roles* específicas — uma para gestão de servidores NTP, e outra para clientes — iremos usar a funcionalidade de grupos no inventário do Ansible para categorizar as máquinas-alvo. Edite o arquivo `/home/ansible/ansible/hosts`, deixando-o com o conteúdo a seguir:

```
$ nano ~/ansible/hosts
(...)
```

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

[ntp_server]
log
```

Note que incluímos a máquina `log` no novo grupo `ntp_server`, que conterà os servidores NTP da rede, deixando implícito que as demais máquinas atuarão como clientes NTP, como veremos a seguir.

Vamos agora criar uma nova *role* para gerenciar a instalação e configuração do OpenNTPD, com o nome `ntp`. Crie-a:

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles ntp
```

3. O próximo passo consiste em editar os arquivos de configuração da *role*. Vamos começar com os *templates* de arquivos de configuração: crie o arquivo novo `/home/ansible/ansible/roles/ntp/templates/ntp_server.conf.j2` com o seguinte conteúdo:

```
1 listen on 127.0.0.1
2 listen on {{ ansible_default_ipv4.address }}
3 servers pool.ntp.br
```

O arquivo acima será o arquivo de configuração do OpenNTPD nos servidores NTP (no caso, a máquina `log`). As diretivas `listen` definem em quais interfaces de rede o *daemon* irá escutar: além da interface `loopback`, note que estamos utilizando a variável `ansible_default_ipv4.address` — ela contém o endereço IPv4 da interface de saída padrão da



máquina sendo configurada (10.0.42.4, no caso da VM `log`). A diretiva `servers` simplesmente define quais servidores remotos na Internet serão consultados para descobrir a hora correta.

Vamos agora para o *template* do arquivo de configuração dos clientes: crie o arquivo novo `/home/ansible/ansible/roles/ntp/templates/ntp_client.conf.j2` com o seguinte conteúdo:

```
1 server {{ ntpsrv }}
```

Temos apenas uma diretiva: `server`, que define o servidor remoto a ser consultado. Note que estamos usando uma variável para configurar esse campo, `ntpsrv`, que será definida a seguir.

4. Que variáveis são essas? Edite o arquivo `/home/ansible/ansible/roles/ntp/vars/main.yml` com o seguinte conteúdo:

```
1 ---
2 ntpsrv: "{{ groups['ntp_server'][0] }}.intnet"
3 fname: "{{ 'ntp_server' if 'ntp_server' in group_names else 'ntp_client' }}"
```

Primeiro, `ntpsrv`: essa variável contém o servidor NTP sendo configurado no momento — para obter seu valor, consultamos no arquivo de inventário qual grupo possui o nome `ntp_server`, e atribuímos a ela o nome do primeiro *host* desse grupo (a máquina `log`).

Depois `fname`: esse variável define qual nome de arquivo será usado para configurar um *host* — se a máquina pertencer ao grupo `ntp_server`, então o valor da variável será também `ntp_server`. Do contrário, ela valerá `ntp_client`.

5. Vamos às tarefas da *role*. Edite o arquivo `/home/ansible/ansible/roles/ntp/tasks/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Install OpenNTPD
3   apt:
4     name: openntpd
5     state: present
6     update_cache: true
7
8 - name: Copy OpenNTPD configuration
9   template:
10     src: '{{ fname }}.conf.j2'
11     dest: /etc/openntpd/ntpd.conf
12     owner: root
13     group: root
14     mode: 0644
15   notify:
16     - Restart OpenNTPD
```

Temos duas tarefas:

- *Install OpenNTPD* se encarrega de instalar o pacote do OpenNTPD, usando o módulo `apt` do Ansible. Antes da instalação, atualizamos a *cache* dos repositórios de forma análoga ao `apt-get update`.
  - *Copy OpenNTPD configuration* copia o arquivo de configuração apropriado para `/etc/openntp/ntp.conf`. Note que o arquivo-origem é definido a partir da variável `fname`, que discutimos no passo anterior. O arquivo é ajustado para usuário-dono `root` e grupo-dono `root`, e permissões `rw-r--r--`. Ao final da cópia, invocamos um *handler* para reiniciar o OpenNTPD.
6. Que *handler* é esse? Edite o arquivo `/home/ansible/ansible/roles/ntp/handlers/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Restart OpenNTPD
3   service:
4     name: openntp
5     state: restarted
```

O *handler* acima simplesmente utiliza o módulo `service` do Ansible para garantir que o *daemon* `openntp` esteja com o estado "reiniciado" após sua execução.

7. A *role* está pronta! Para executá-la, vamos editar o arquivo `/home/ansible/ansible/srv.yml`:

```
$ echo '    - ntp' >> ~/ansible/srv.yml
```

```
$ cat ~/ansible/srv.yml
---
- hosts: srv
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - sudoers
    - ntp
```

Note que agora ao invocarmos o arquivo `/home/ansible/ansible/srv.yml` iremos não apenas configurar o `sudo` de forma automática, mas também instalar e configurar o OpenNTPD nas máquinas-alvo. Basta, é claro, que editemos o arquivo de inventário de antemão.

8. Ufa! Agora sim, vamos testar:

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

```
PLAY [srv]
```

```
*****
*****
```

```
(...)
```

```
TASK [ntp : Install OpenNTPD]
```

```
*****
****
```

```
changed: [192.168.42.2]
```

```
changed: [log]
```

```
changed: [nfs]
```

```
changed: [ns1]
```

```
changed: [ns2]
```

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

```
*****
```

```
changed: [ns2]
```

```
changed: [ns1]
```

```
changed: [log]
```

```
changed: [nfs]
```

```
changed: [192.168.42.2]
```

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

```
*****
```

```
changed: [log]
```

```
changed: [nfs]
```

```
changed: [ns1]
```

```
changed: [ns2]
```

```
changed: [192.168.42.2]
```

```
PLAY RECAP
```

```
*****
*****
```

192.168.42.2	: ok=6	changed=4	unreachable=0	failed=0
log	: ok=6	changed=4	unreachable=0	failed=0
nfs	: ok=6	changed=4	unreachable=0	failed=0
ns1	: ok=6	changed=4	unreachable=0	failed=0
ns2	: ok=6	changed=4	unreachable=0	failed=0

Terá nossa "magia" funcionando? A princípio, o OpenNTPD foi instalado e configurado corretamente nas máquinas-alvo. Vamos checar:

```
$ ansible -i ~/ansible/hosts srv -b -m shell -a 'which openntpd ; ps auxmw | grep
'[/]usr/sbin/ntpd' ; cat /etc/openntpd/ntpd.conf'
log | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      9286  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
listen on 127.0.0.1
listen on 10.0.42.4
servers pool.ntp.br

ns1 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      7163  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

ns2 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      7117  0.0  0.0   7564   124 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

nfs | CHANGED | rc=0 >>
/usr/sbin/openntpd
root      6967  0.0  0.0   7564   120 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet

192.168.42.2 | CHANGED | rc=0 >>
/usr/sbin/openntpd
root     14410  0.0  0.0   7564   120 ?        -    17:32   0:00 /usr/sbin/ntpd -f
/etc/openntpd/ntpd.conf
server log.intnet
```

Uhm... o OpenNTPD está instalado em todas as máquinas, como verificado pelo **which**. O processo do **ntpd** também está ativo, como verificado via **ps**. E, finalmente, os arquivos de configuração de todas as máquinas cliente possui apenas a linha **server log.intnet**, como esperado, e o arquivo de configuração da máquina **log** corresponde ao que objetivávamos.

Muito legal, não é mesmo?

9. Muitas alterações em nosso repositório local — vamos fazer um *commit* e enviá-las:

```
$ cd ~/ansible/
```

```
$ git add .
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   hosts
        new file:   roles/ntp/README.md
        new file:   roles/ntp/defaults/main.yml
        new file:   roles/ntp/handlers/main.yml
        new file:   roles/ntp/meta/main.yml
        new file:   roles/ntp/tasks/main.yml
        new file:   roles/ntp/templates/ntp_client.conf.j2
        new file:   roles/ntp/templates/ntp_server.conf.j2
        new file:   roles/ntp/tests/inventory
        new file:   roles/ntp/tests/test.yml
        new file:   roles/ntp/vars/main.yml
        modified:   srv.yml
```

```
$ git commit -m 'Adicionada role para instalacao e configuracao do servico NTP'
[master 17486a0] Adicionada role para instalacao e configuracao do servico NTP
12 files changed, 146 insertions(+)
create mode 100644 roles/ntp/README.md
create mode 100644 roles/ntp/defaults/main.yml
create mode 100644 roles/ntp/handlers/main.yml
create mode 100644 roles/ntp/meta/main.yml
create mode 100644 roles/ntp/tasks/main.yml
create mode 100644 roles/ntp/templates/ntp_client.conf.j2
create mode 100644 roles/ntp/templates/ntp_server.conf.j2
create mode 100644 roles/ntp/tests/inventory
create mode 100644 roles/ntp/tests/test.yml
create mode 100644 roles/ntp/vars/main.yml
```

```
$ git push
Counting objects: 19, done.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (19/19), 1.66 KiB | 0 bytes/s, done.
Total 19 (delta 1), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
24bcab5..17486a0  master -> master
```

## 5) Registro de comandos digitados com SnoopyLog

1. Vamos agora instalar o Snoopy (<https://github.com/a2o/snoopy>), um programa bastante simples que serve para registrar todos os comandos digitados no terminal, por qualquer usuário, nos logs do sistema. Assim, é possível gerar uma trilha de auditoria de comandos para realização de

análise forense e garantia de não-repúdio em caso de incidentes.

Assim como no caso no NTP, vamos automatizar a instalação e configuração desse pacote como uma *baseline* de segurança em todos os servidores (atuais e futuros) do *datacenter* usando o Ansible. Crie a *role*:

```
$ ansible-galaxy init --init-path=/home/ansible/ansible/roles snoopy
```

2. Felizmente, a configuração do Snoopy é bem mais simples que a do OpenNTPD, já que não temos servidores/clientes distintos no inventário. Edite o arquivo `/home/ansible/ansible/roles/snoopy/tasks/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Install Snoopy
3   apt:
4     name: snoopy
5     state: present
6     update_cache: true
7
8 - name: Configure ld.so.preload
9   lineinfile:
10    path: /etc/ld.so.preload
11    line: '/lib/snoopy.so'
12    insertafter: EOF
13    create: yes
```

Novamente, usamos o módulo `apt` para instalar o pacote `snoopy`. Em seguida, adicionamos ao final do arquivo `/etc/ld.so.preload` (criando-o se ele não existir) a linha `/lib/snoopy.so`, garantindo que a biblioteca compartilhada do Snoopy será carregada antes da execução de qualquer binário no sistema — garantindo assim que os comandos serão de fato logados.

3. Adicione a *role* do Snoopy ao arquivo `/home/ansible/ansible/srv.yml`:

```
echo '    - snoopy' >> ~/ansible/srv.yml
```

4. Finalmente, execute a *role*:

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

```
PLAY [srv]
```

```
*****
*****
```

```
(...)
```

```
TASK [snoopy : Install Snoopy]
```

```
*****
***
```

```
changed: [ns2]
```

```
changed: [ns1]
```

```
changed: [nfs]
```

```
changed: [log]
```

```
changed: [192.168.42.2]
```

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

```
*****
```

```
changed: [ns1]
```

```
changed: [ns2]
```

```
changed: [nfs]
```

```
changed: [log]
```

```
changed: [192.168.42.2]
```

```
PLAY RECAP
```

```
*****
*****
```

192.168.42.2	: ok=7	changed=3	unreachable=0	failed=0
log	: ok=7	changed=3	unreachable=0	failed=0
nfs	: ok=7	changed=3	unreachable=0	failed=0
ns1	: ok=7	changed=3	unreachable=0	failed=0
ns2	: ok=7	changed=3	unreachable=0	failed=0

5. Terá funcionado? Logue via SSH na máquina **ns1**, ainda como o usuário **ansible**:

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

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

Last login: Fri Nov 16 00:16:07 2018 from 192.168.42.2
ansible@ns1:~$
```

Use o comando `sudo -i` para escalar privilégio:

```
$ sudo -i
```

Agora, verifique o conteúdo do arquivo `/var/log/auth.log` — de fato, procure por linhas que possuam a *string* `snoopy`:

```
# grep 'ns1 snoopy' /var/log/auth.log | grep -v '(none)'
Nov 16 00:17:07 ns1 snoopy[10150]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/bash]: -bash Nov 16 00:17:07 ns1 snoopy[10152]:
[uid:10005 sid:10150 tty:/dev/pts/0 cwd:/home/ansible filename:/usr/bin/id]: id -u
Nov 16 00:17:07 ns1 snoopy[10154]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:07 ns1 snoopy[10156]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/usr/bin/dircolors]: dircolors -b
Nov 16 00:17:07 ns1 snoopy[10158]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:11 ns1 snoopy[10161]: [uid:10005 sid:10150 tty:/dev/pts/0
cwd:/home/ansible filename:/usr/bin/sudo]: sudo -i
Nov 16 00:17:11 ns1 snoopy[10162]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/bash]: -bash
Nov 16 00:17:11 ns1 snoopy[10164]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/usr/bin/id]: id -u
Nov 16 00:17:11 ns1 snoopy[10167]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/ls]: ls /etc/bash_completion.d
Nov 16 00:17:11 ns1 snoopy[10168]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/usr/bin/mesg]: mesg n
Nov 16 00:20:27 ns1 snoopy[10185]: [uid:0 sid:10150 tty:/dev/pts/0 cwd:/root
filename:/bin/grep]: grep ns1 snoopy /var/log/auth.log
```

Note como o Snoopy registrou todos os nossos comandos desde o login no sistema — desde o carregamento de opções do perfil do *shell* Bash, passando pela escalação de privilégio usando o `sudo`, e chegando até o próprio `grep` que acabamos de executar!

Agora, se seus usuários fizerem qualquer ação indevida nos servidores, ficará bem mais difícil negar o ocorrido! Especialmente com o uso do concentrador de logs e SIEM que instalaremos a seguir.

6. Como fizemos mais alterações nos repositórios locais, vamos enviá-las:



```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para instalacao e
configuracao do Snoopy logger' ; git push
[master ada4705] Adicionada role para instalacao e configuracao do Snoopy logger
10 files changed, 130 insertions(+)
create mode 100644 roles/snoopy/README.md
create mode 100644 roles/snoopy/defaults/main.yml
create mode 100644 roles/snoopy/handlers/main.yml
create mode 100644 roles/snoopy/meta/main.yml
create mode 100644 roles/snoopy/tasks/main.yml
create mode 100644 roles/snoopy/tests/inventory
create mode 100644 roles/snoopy/tests/test.yml
create mode 100644 roles/snoopy/vars/main.yml
create mode 100644 srv.retry
Counting objects: 16, done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (16/16), 1.34 KiB | 0 bytes/s, done.
Total 16 (delta 1), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
17486a0..ada4705 master -> master
```

Já estamos construindo um histórico interessante no repositório, não é mesmo? Confira com o **git log**:

```
$ git log
commit ada4705b62e53e0802c06fc75e67a89d83424467
Author: Ansible <ansible@intnet>
Date:   Fri Nov 16 00:29:32 2018 -0200

    Adicionada role para instalacao e configuracao do Snoopy logger

commit 17486a03a71405cb22737c026f068ac0a6a17384
Author: Ansible <ansible@intnet>
Date:   Fri Nov 16 00:03:36 2018 -0200

    Adicionada role para instalacao e configuracao do servico NTP

commit 24bcab569ee2f9504c54a65081f75f5cd5c400e5
Author: Ansible <ansible@intnet>
Date:   Thu Nov 15 16:12:51 2018 -0200

    Maquina log adicionada ao inventario do Ansible

commit 0c673e48dc7aaf2bd6738c8033c33815f10cc6
Author: Ansible <ansible@intnet>
Date:   Thu Nov 15 02:04:14 2018 -0200

    Importacao inicial, role sudoers adicionada
```

## 6) Instalação e configuração inicial do Graylog

Vamos proceder à instalação do SIEM *open-source* Graylog. Como a instalação será feita em uma única máquina (**log**), e realizaremos um grande número de passos, faremos o processo "à moda antiga" — via comandos diretos no terminal. Seguiremos os passos de instalação indicados na documentação oficial do Graylog, em <http://docs.graylog.org/en/latest/pages/installation/os/debian.html>.

1. Antes de mais nada desligue a VM **log**. O Graylog exige uma quantidade bastante significativa de recursos — na console do Virtualbox, acesse *Settings > System > Base Memory* e aumente-a para 4096 MB (4 GB). Em seguida, religue a VM **log** e acesse-a como o usuário **root**:

```
# hostname ; whoami
log
root
```

2. Agora, instale as dependências do Graylog com o comando a seguir:

```
# apt-get update ; apt-get install -y apt-transport-https openjdk-8-jre-headless
uuid-runtime pwgen
```

3. O próximo passo é a instalação do MongoDB, uma base de dados NoSQL *open-source* orientada ao armazenamento e gestão de documentos:

```
# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
Executing: /tmp/apt-key-gpghome.yMHAQrCigl/gpg.1.sh --keyserver
hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
gpg: key 58712A2291FA4AD5: public key "MongoDB 3.6 Release Signing Key
<packaging@mongodb.com>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

```
# echo "deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/3.6 main" >
/etc/apt/sources.list.d/mongodb-org-3.6.list
```

```
# apt-get update ; apt-get install -y mongodb-org
```

Em seguida, inicie o serviço do MongoDB:

```
# systemctl daemon-reload
```

```
# systemctl enable mongod.service
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service →
/lib/systemd/system/mongod.service.
```

```
# systemctl restart mongod.service
```

4. Agora, vamos instalar o Elasticsearch, um *engine* de busca distribuída e *open-source* baseada na biblioteca Apache Lucene:

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
OK
```

```
# echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" >
/etc/apt/sources.list.d/elastic-5.x.list
```

```
# apt-get update ; apt-get install -y elasticsearch
```

Temos que configurar o Elasticsearch para operar com o Graylog—para isso, basta informarmos um nome de *cluster* no arquivo de configuração [/etc/elasticsearch/elasticsearch.yml](#):

```
# sed -i 's/^#\(\cluster\.name:\).*\/\1 graylog/'
/etc/elasticsearch/elasticsearch.yml
```

Feito isso, basta iniciar o serviço do Elasticsearch:

```
# systemctl daemon-reload
```

```
# systemctl enable elasticsearch.service
Synchronizing state of elasticsearch.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service
→ /usr/lib/systemd/system/elasticsearch.service.
```

```
# systemctl restart elasticsearch.service
```

5. Finalmente, vamos instalar o Graylog:

```
# wget -q https://packages.graylog2.org/repo/packages/graylog-2.4-  
repository_latest.deb
```

```
# dpkg -i graylog-2.4-repository_latest.deb ; rm graylog-2.4-repository_latest.deb
```

```
# apt-get update ; usermod -e -1 root ; apt-get install -y graylog-server ; usermod  
-e 0 root
```

Agora, temos que configurar uma senha randômica para garantir a segurança do armazenamento de senhas dos usuários do Graylog, bem como uma senha de acesso administrativo para o Graylog (como de costume, iremos usar **rnpesr**). Os comandos a seguir irão realizar essas ações:

```
# SECRET=$(pwgen -s 96 1) ; sed -i -e 's/password_secret =.*/password_secret =  
'$SECRET'/' /etc/graylog/server/server.conf ; unset SECRET
```

```
# PASSWORD=$(echo -n 'rnpesr' | shasum -a 256 | awk '{print $1}') ; sed -i -e  
's/root_password_sha2 =.*/root_password_sha2 = '$PASSWORD'/'  
/etc/graylog/server/server.conf ; unset PASSWORD
```

O Graylog utiliza a *timezone* UTC (*Universal Time Coordinated*) como padrão, que não é a mesma que utilizamos no Brasil. Assumindo que o curso está sendo realizado no fuso de Brasília, o comando a seguir irá ajustar a *timezone* corretamente:

```
# sed -i 's/^#\(\root_timezone =\).*/\1 America\Sao_Paulo/'  
/etc/graylog/server/server.conf
```

Finalmente, vamos iniciar o Graylog:

```
# systemctl daemon-reload
```

```
# systemctl enable graylog-server.service  
Synchronizing state of graylog-server.service with SysV service script with  
/lib/systemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install enable graylog-server  
Created symlink /etc/systemd/system/multi-user.target.wants/graylog-server.service  
→ /usr/lib/systemd/system/graylog-server.service.
```

```
# systemctl start graylog-server.service
```

6. Para não termos que configurar o Graylog escutando diretamente por conexões do mundo externo, iremos instalar o servidor HTTP Nginx para atuar como um *proxy* reverso, filtrando e repassando as conexões para o Graylog. Primeiro, instale o Nginx:

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

Remova o arquivo de configuração do website padrão do Nginx — iremos substituí-lo a seguir:

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

Crie o arquivo novo `/etc/nginx/sites-available/graylog` com o conteúdo a seguir. **IMPORTANTE:** ao editar o arquivo abaixo, substitua as duas ocorrências da *string* `NS1_ENP0S3_IPADDR` pelo endereço da interface `enp0s3` da sua máquina `ns1`.

```
1 server
2 {
3     listen      80 default_server;
4     listen      [::]:80 default_server ipv6only=on;
5     server_name NS1_ENP0S3_IPADDR;
6
7     location /
8     {
9         proxy_set_header    Host $http_host;
10        proxy_set_header    X-Forwarded-Host $host;
11        proxy_set_header    X-Forwarded-Server $host;
12        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
13        proxy_set_header    X-Graylog-Server-URL
14        http://NS1_ENP0S3_IPADDR:9080/api/;
15        proxy_pass           http://127.0.0.1:9000;
16    }
17 }
```

Suponha, por exemplo, que o endereço IP da interface `enp0s3` da sua máquina `ns1` seja 200.130.46.254; você poderia usar o comando `sed` para fazer as substituições de forma automática da seguinte forma:

```
# sed -i 's/NS1_ENP0S3_IPADDR/200.130.46.254/' /etc/nginx/sites-available/graylog
```

Agora, crie um link simbólico do arquivo recém-criado para o arquivo `/etc/nginx/sites-enabled/graylog`, habilitando-o no Nginx, como se segue:

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

Finalmente, reinicie o Nginx:

```
# systemctl restart nginx
```

## 7) Ajuste das regras de firewall para o Graylog

O uso do Graylog exigirá alguns ajustes no firewall, a saber:

- Iremos acessar a interface web do Graylog a partir da máquina física, usando o endereço IP público da máquina **ns1**. Para que consigamos atingir o Graylog, será necessário criar uma regra de DNAT na tabela **nat**, **chain** PREROUTING, fazendo o redirecionamento de endereço/porta, além de uma regra na tabela **filter**, **chain** FORWARD, correspondente. Especificamente, faremos o mapeamento da porta externa 9080/TCP para a porta interna 80/TCP.
- Máquinas na DMZ devem conseguir acessar a máquina **log** na porta 5140/UDP, para envio dos logs locais usando o Rsyslog. Como essas máquinas estão todas na mesma sub-rede, os acessos não passam pelo firewall e nenhuma configuração adicional se faz necessária.
- Máquinas na Intranet devem conseguir acessar a máquina **log** na porta 5140/UDP, para envio dos logs locais usando o Rsyslog. Como esse tráfego passa **através do** firewall, devemos inserir a regra na **chain** FORWARD.

1. Vamos lá: acesse a máquina **ns1** como o usuário **root**:

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

Para atender o requisito (a), insira as regras:

```
# iptables -t nat -A PREROUTING -i enp0s3 -p tcp -m tcp --dport 9080 -j DNAT --to  
-destination 10.0.42.4:80
```

```
# iptables -A FORWARD -i enp0s3 -d 10.0.42.4/32 -p tcp -m tcp --dport 80 -j ACCEPT
```

Para o requisito (b), nenhuma ação é necessária. Finalmente, para o requisito (c) insira a regra a seguir:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.4/32 -p udp -m udp --dport 5140  
-j ACCEPT
```

Grave as novas regras na configuração permanente do firewall:

```
# /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.
```

## 8) Visualizando logs de máquinas no Graylog

1. Tudo pronto? Vamos acessar a interface do Graylog: em sua máquina física, abra o navegador e aponte-o para [http://NS1\\_ENP0S3\\_IPADDR:9080](http://NS1_ENP0S3_IPADDR:9080) (substitua a *string* `NS1_ENP0S3_IPADDR` pelo endereço IP da interface `enp0s3` da sua máquina `ns1`). Você deverá ver a tela a seguir:

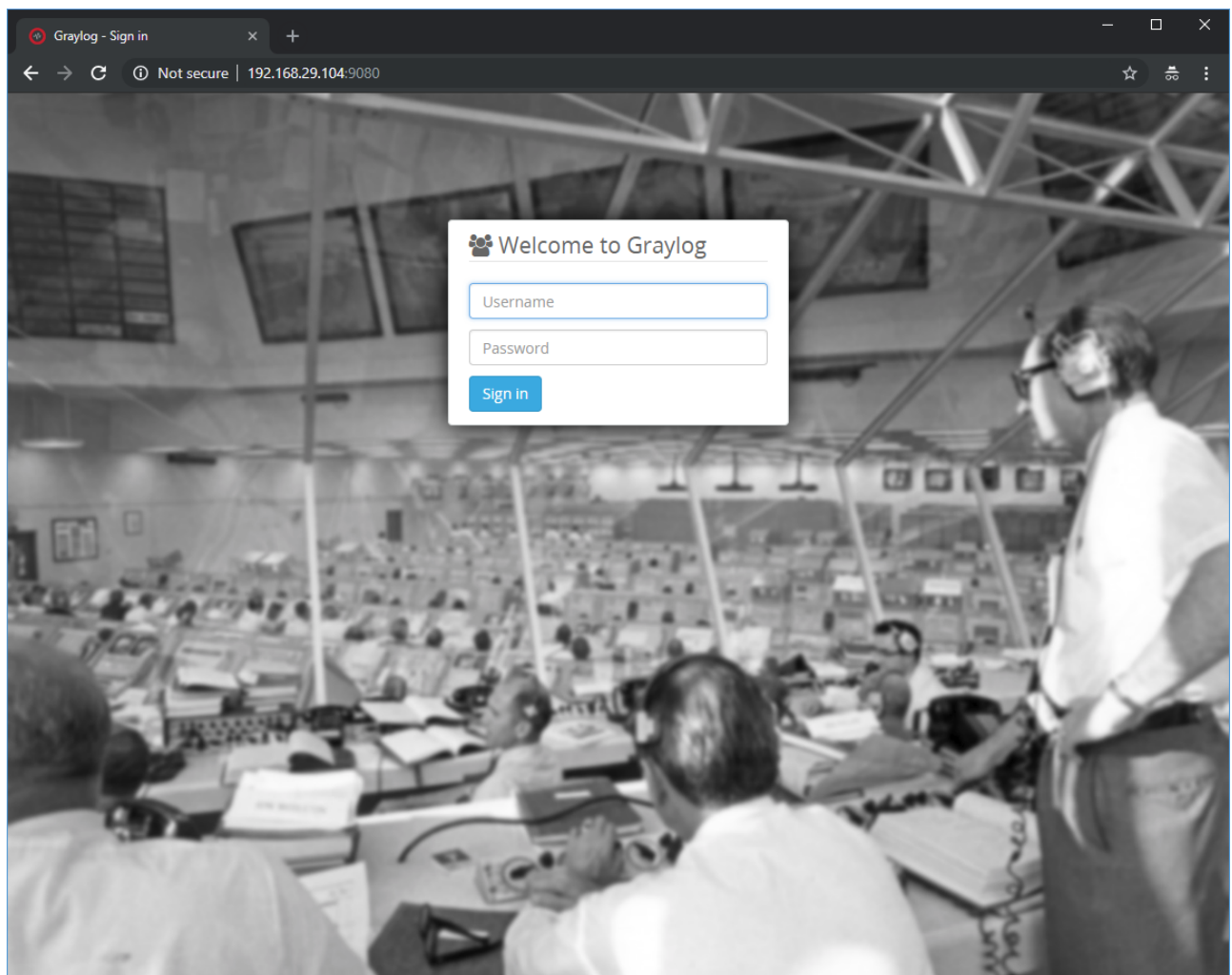


Figura 2. Tela de login do Graylog

Faça login com o usuário `admin` e senha `rnpesr`, como definimos anteriormente.

2. Vamos fazer alguns ajustes iniciais. Acesse o menu *System > Indices* e clique em *Edit*. Na nova tela, configure o valor do campo *Index Shards* como 1, e em seguida clique em *Save* na base da página. Como estamos em um ambiente simulado, essa configuração irá auxiliar na economia de recursos da máquina.
3. Agora, acesse o menu *System > Inputs*. Na caixa *Select input* desça a barra de rolagem, selecione a opção *Syslog UDP* e clique em *Launch new input*.

Na nova janela, clique no campo *Select Node* e selecione a única opção disponível; no campo *Title*, escreva `syslog`; em *Bind address*, digite o endereço local da máquina, `10.0.42.4`; finalmente, no campo *Port* informe o valor `5140`. Sua tela deve ficar assim:

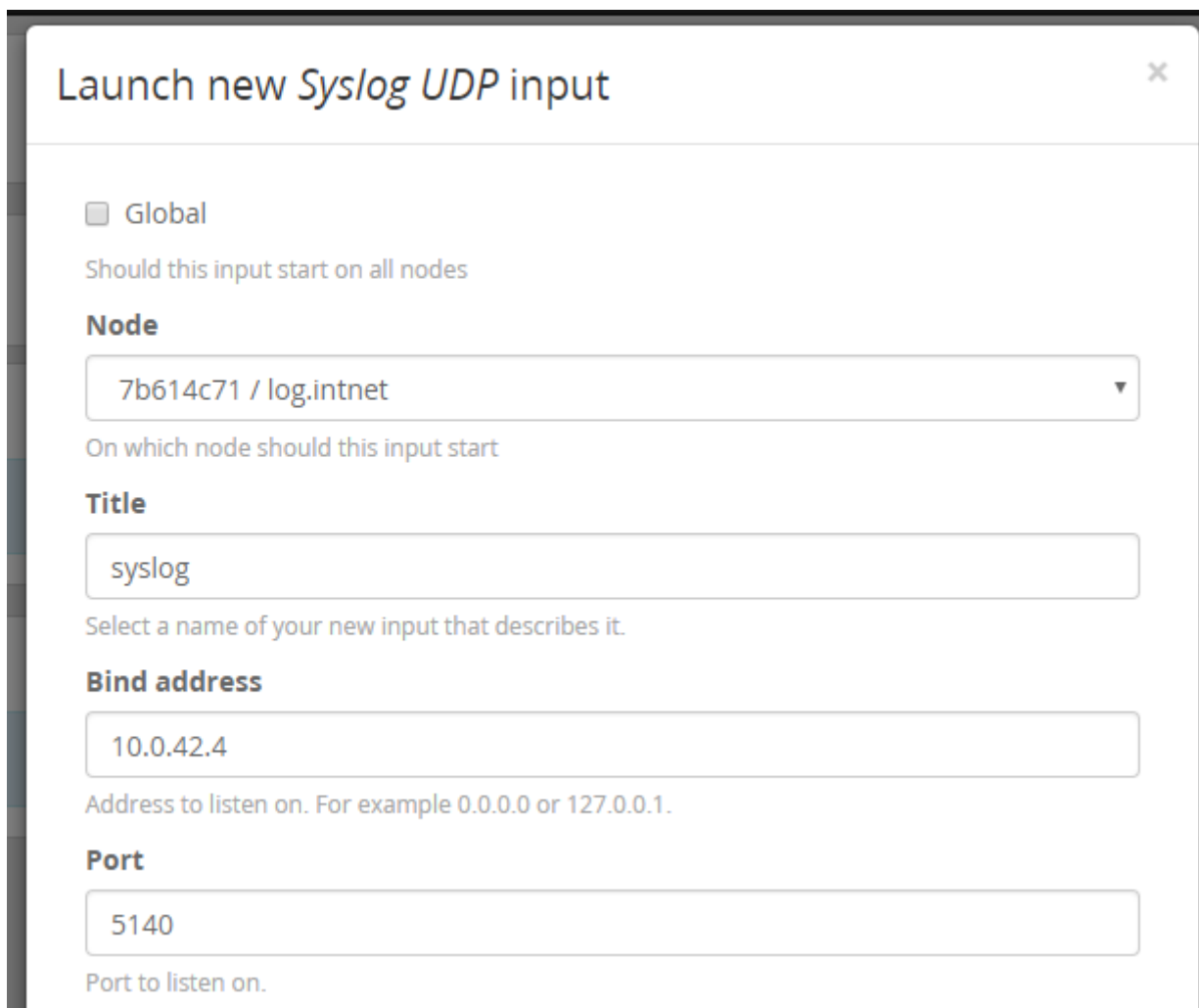


Figura 3. Criação de um input Syslog/UDP no Graylog

Na base da janela, clique em *Save*.

4. O *input* via Syslog está configurado; temos, agora, que configurar o envio de logs para esse *input*. Novamente, o Ansible é a ferramenta para a tarefa.

Acesse a máquina `client` como o usuário `ansible`:

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

Vamos criar uma *role* que envie todos os logs das máquinas do *datacenter* para o *input* que configuramos no Graylog:

```
$ ansible-galaxy init --init-path /home/ansible/ansible/roles/ syslog
- syslog was created successfully
```



Vamos usar uma técnica similar à que fizemos no caso do NTP: insira um novo grupo, `log_server`, no inventário do Ansible:

```
$ echo -e '\n[log_server]\nlog' >> ~/ansible/hosts
```

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

[ntp_server]
log

[log_server]
log
```

Agora, edite o arquivo `/home/ansible/ansible/roles/syslog/vars/main.yml` com o seguinte conteúdo:

```
1 ---
2 logsrv: "{{ groups['log_server'][0] }}.intnet"
```

Onde vamos chegar com isso? Edite o arquivo `/home/ansible/ansible/roles/syslog/tasks/main.yml` com o seguinte conteúdo e confira:

```
1 ---
2 - name: Configure centralized syslog
3   lineinfile:
4     path: /etc/rsyslog.d/99-graylog.conf
5     line: '*. * @{{ logsrv }}:5140;RSYSLOG_SyslogProtocol23Format'
6     insertafter: EOF
7     create: yes
```

A tarefa acima irá criar o arquivo `/etc/rsyslog.d/99-graylog.conf`, se inexistente, em todos os *hosts* aplicáveis e instruir o Rsyslog a enviar todos os logs para a máquina `logsrv` (que definimos no arquivo `vars`, acima) na porta 5140/UDP. Feito isso, chama-se um *handler* que reinicia o Rsyslog.

É claro, temos que criar o *handler*. Edite o arquivo `/home/ansible/ansible/roles/syslog/handlers/main.yml` com o seguinte conteúdo:

```
1 ---
2 - name: Restart Rsyslog
3   service:
4     name: rsyslog
5     state: restarted
```

O de sempre: usamos o módulo `service` para reiniciar o Rsyslog local. Adicione a nova *role* ao arquivo de amarração de inventário `/home/ansible/ansible/srv.yml`:

```
$ echo '    - syslog' >> ~/ansible/srv.yml
```

Vamos testar? Dispare a *role*:

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

```
PLAY [srv]
```

```
*****
*****
```

```
(...)
```

```
TASK [syslog : Configure centralized syslog]
```

```
*****
```

```
changed: [ns1]
```

```
changed: [ns2]
```

```
changed: [nfs]
```

```
changed: [192.168.42.2]
```

```
changed: [log]
```

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

```
*****
```

```
changed: [nfs]
```

```
changed: [ns1]
```

```
changed: [ns2]
```

```
changed: [192.168.42.2]
```

```
changed: [log]
```

```
PLAY RECAP
```

```
*****
*****
```

192.168.42.2	: ok=9	changed=3	unreachable=0	failed=0
log	: ok=9	changed=3	unreachable=0	failed=0
nfs	: ok=9	changed=3	unreachable=0	failed=0
ns1	: ok=9	changed=3	unreachable=0	failed=0
ns2	: ok=9	changed=3	unreachable=0	failed=0

5. Como de costume, não se esqueça de enviar as mudanças para o repositório remoto do Git:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para envio de logs para o servidor Graylog' ; git push
[master b1e7a24] Adicionada role para envio de logs para o servidor Graylog
10 files changed, 127 insertions(+)
create mode 100644 roles/syslog/README.md
create mode 100644 roles/syslog/defaults/main.yml
create mode 100644 roles/syslog/handlers/main.yml
create mode 100644 roles/syslog/meta/main.yml
create mode 100644 roles/syslog/tasks/main.yml
create mode 100644 roles/syslog/tests/inventory
create mode 100644 roles/syslog/tests/test.yml
create mode 100644 roles/syslog/vars/main.yml
Counting objects: 16, done.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (16/16), 1.39 KiB | 0 bytes/s, done.
Total 16 (delta 3), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
ada4705..b1e7a24 master -> master
```

6. Volte para o navegador em sua máquina física, e acesse a aba *Search*. Você deverá ver algo parecido com a imagem abaixo:

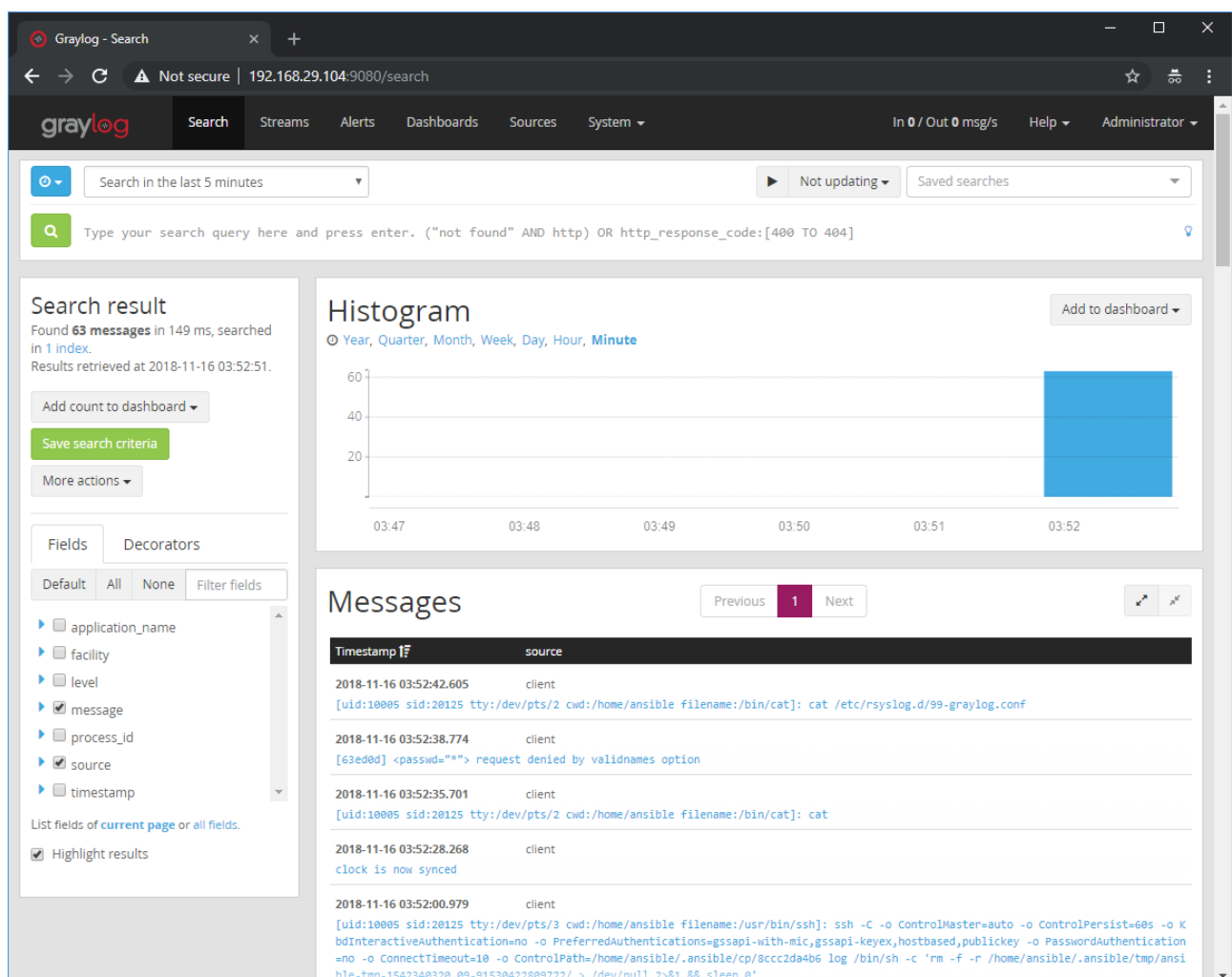


Figura 4. Visualização de logs remotos no Graylog

Legal, não é mesmo? O Graylog está recebendo todos os logs enviados pelos servidores do *datacenter* (e também da máquina *client*), tornando-os acessíveis de forma fácil e pesquisável através de uma interface bastante poderosa.

Vamos testar, por exemplo, as funções de busca do Graylog. Faça um login via SSH na máquina *ns1*, e *sudo* para *root*:

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

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

Last login: Fri Nov 16 02:07:12 2018 from 192.168.42.2
ansible@ns1:~$
```

```
$ sudo -i
```

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

Agora, volte à interface web do Graylog e busque (no canto superior esquerdo da tela, num campo identificado por uma lupa de cor verde) pelo termo *source:ns1 AND application\_name:snoopy*. Esse termo de busca irá mostrar todos os logs oriundos da máquina *ns1* e que tenham sido gerados pela aplicação Snoopy, como vemos abaixo:

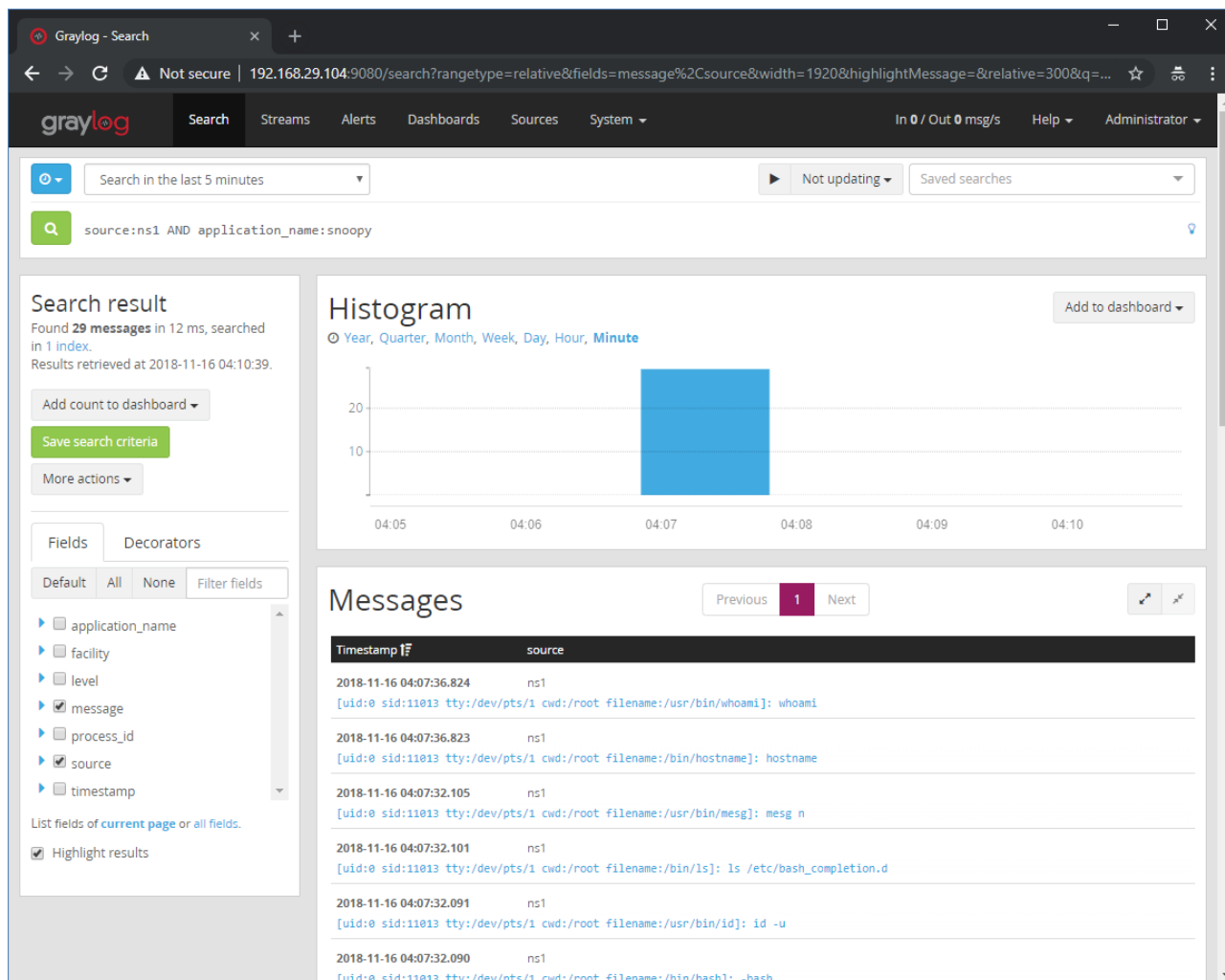


Figura 5. Filtros de busca no Graylog

Vamos um passo além: imagine que você quer encontrar todos os eventos de `sudo` executados na máquina `ns1`. Expanda o termo de busca para `source:ns1 AND application_name:snoopy AND message:sudo`. Você deverá encontrar o evento de escalção de privilégio que fizemos logo acima — clique sobre o evento para expandir os campos do log:

The screenshot shows the Graylog Messages interface. At the top, there's a 'Messages' header with navigation buttons 'Previous', '1' (selected), and 'Next'. Below this is a table with columns 'Timestamp' and 'source'. The first row shows a message from 'ns1' at '2018-11-16 04:07:32.087'. The message content is '[uid:10005 sid:11013 tty:/dev/pts/1 cwd:/home/ansible filename:/usr/bin/sudo]: sudo -i'. Below the message, there's a detailed view with fields like 'Received by', 'Stored in index', 'Routed into streams', 'application\_name', 'facility', 'level', 'message', 'process\_id', 'source', and 'timestamp'. Each field has a search icon and a dropdown menu.

Figura 6. Visualizando eventos específicos

Note que a mensagem mostra, inclusive, o UID do usuário que executou o comando: 10005. Conseguimos descobrir nosso "culpado" facilmente, consultando a base de usuários do LDAP:

```
$ getent passwd | grep 10005
ansible:*:10005:10001:ansible:/home/ansible:/bin/bash
```

## 9) Autenticação centralizada via LDAP no Graylog

Estamos usando o usuário **admin** para realizar todas as ações no Graylog, o que obviamente não é sustentável — e se quisermos que vários colaboradores tenham acesso à ferramenta e possam pesquisar eventos e investigar logs dos sistemas? Felizmente, a integração do Graylog com sistemas externos de autenticação, como o LDAP, é bastante fácil.

1. Acesse o menu *System > Authentication*. Na aba à esquerda, selecione *LDAP/Active Directory* e clique na caixa *Enable LDAP*.
2. Em *Server configuration*, mantenha *Server Type* como *LDAP*; em *Server Address*, digite **ldap://ldap.intnet:389**; informe *System Username* como **cn=admin,dc=intnet**; finalmente, em *System Password* digite **rnpesr**.

Clique em *Test Server Connection* para verificar a correta conexão com o servidor LDAP.

3. Continuando, em *User Mapping* defina *Search Base DN* como **ou=People,dc=intnet**; em *User Search Pattern*, digite **(&(objectClass=posixAccount)(uid={0}))**; depois, em *Display Name Attribute* informe **cn**.

Até o momento, sua tela de configuração deverá estar da seguinte forma:

## LDAP Settings

This page is the only resource you need to set up the Graylog LDAP integration. You can test the connection to your LDAP server and even try to log in with an LDAP account of your choice right away.

☒ Enable LDAP

User accounts will be taken from LDAP/Active Directory, the administrator account will still be available.

### 1. Server configuration

Server Type ☒ LDAP ☐ Active Directory

Server Address   :

☐ SSL ☐ StartTLS ☐ Allow self-signed certificates

System Username

The username for the initial connection to the LDAP server, e.g. `uid=admin,ou=system`, this might be optional depending on your LDAP server.

System Password

The password for the initial connection to the LDAP server.

### 2. Connection Test

[Test Server Connection](#)

Performs a background connection check with the address and credentials above.

### 3. User mapping

Search Base DN

The base tree to limit the LDAP search query to, e.g. `cn=users,dc=example,dc=com`.

User Search Pattern

For example `(&(objectClass=inetOrgPerson)(uid={0}))`. The string `{0}` will be replaced by the entered username.

Display Name attribute

Which LDAP attribute to use for the full name of the user in Graylog, e.g. `cn`.

Try to load a test user using the form below, if you are unsure which attribute to use.

Figura 7. Configuração do Graylog com o LDAP, parte 1

4. Em *Group Mapping*, defina *Group Search Base DN* como `ou=Groups,dc=intnet`; em *Group Search Pattern*, digite `(objectClass=posixGroup)`; para *Group Name Attribute*, informe `cn`; finalmente, mantenha *Default User Role* como *Reader - basic access*.

A segunda parte da tela de configuração ficará assim:



## 4. Group Mapping (optional)

Group Search Base DN

ou=Groups,dc=intnet

The base tree to limit the LDAP group search query to, e.g. `cn=users,dc=example,dc=com`.

Group Search Pattern

(&amp;(objectClass=posixGroup)(cn=sysadm))

The search pattern used to find groups in LDAP for mapping to Graylog roles, e.g. `(objectClass=groupOfNames)` or `(&(objectClass=groupOfNames)(cn=graylog*))`.

Group Name Attribute

cn

Which LDAP attribute to use for the full name of the group, usually `cn`.

Default User Role

Reader - basic access

The default Graylog role determines whether a user created via LDAP can access the entire system, or has limited access. You can assign additional permissions by [mapping LDAP groups to Graylog roles](#), or you can assign additional Graylog roles to LDAP users below.

Changing the static role assignment will only affect to new users created via LDAP/Active Directory! Existing user accounts will be updated on their next login, or if you edit their roles manually.

Additional Default Roles

Choose additional roles...

Choose the additional roles each LDAP user will have by default, leave it empty if you want to map LDAP groups to Graylog roles.

Changing the static role assignment will only affect to new users created via LDAP/Active Directory! Existing user accounts will be updated on their next login, or if you edit their roles manually.

Figura 8. Configuração do Graylog com o LDAP, parte 2

Na base da tela, clique em *Save LDAP settings*.

5. Após salvar as configurações, note que em *Group Mapping > Default User Role* há um link destacado em azul que diz *mapping LDAP groups to Graylog roles* — clique neste link. Você verá a tela abaixo:

## LDAP Settings

This page is the only resource you need to set up the Graylog LDAP integration. You can test the connection to your LDAP server and even try to log in with an LDAP account of your choice right away.

fwadm

None

ldapadm

None

setup

None

sysadm

Admin

Save

Cancel

Figura 9. Mapeamento de grupos do LDAP e roles no Graylog

Mapeie o grupo `sysadm` com a role `Admin`, e clique em *Save*. Depois, no canto superior da tela, clique em *Administrator > Logout*.

6. Vamos testar? Logue com o usuário `luke`, membro do grupo `sysadm`. Use a mesma senha do usuário no LDAP. Observe o nível de acesso do usuário — ele consegue ver todas as abas e configurações às quais o usuário `admin` possui acesso.

Agora, faça logout e acesse como o usuário `han`. Note: apenas um pequeno número de abas e opções estão disponíveis, e todas como somente leitura. O usuário possui acesso apenas a

*streams* de logs e *dashboards* pré-configurados pelo administrador, e não consegue alterar quaisquer configurações do sistema.

O sistema de *roles* do Graylog é bastante poderoso, permitindo a customização do nível de acesso de usuários com boa granularidade.

## 10) Configurando inputs customizados no Graylog

Acesse o Graylog novamente com o usuário **admin** (ou **luke**, se preferir). Busque por mensagens com o padrão **source:ns2 AND application\_name:slapd AND message:dc\=intnet**, e expanda uma qualquer, como mostrado abaixo:



Figura 10. Mensagens não interpretadas pelo Graylog

Observe: a mensagem acima é proveniente do log do OpenLDAP, e possui várias informações relevantes no campo **message** — temos a base de busca, escopo e filtro utilizado. Porém, como o Graylog não está configurado para processar e interpretar a mensagem acima, todos os campos ficam agrupados em uma "massa" única, dificultando operações de busca e criação de filtros e alertas. Não podemos, por exemplo, usar um termo como **filter:uid\=ansible** no campo de busca do Graylog.

Vamos instalar um *content pack* para o Graylog que irá instruí-lo sobre como processar logs de aplicações específicas; usaremos, para o nosso exemplo, o servidor web Nginx que está instalado na máquina **log**.

1. No navegador em sua máquina física, faça o download do arquivo [https://raw.githubusercontent.com/graylog-labs/graylog-contentpack-nginx/master/content\\_pack.json](https://raw.githubusercontent.com/graylog-labs/graylog-contentpack-nginx/master/content_pack.json) para sua Área de Trabalho.
2. Na interface web do Graylog, acesse *System > Content Packs*. Clique na caixa *Import Content Pack* e em seguida em *Choose File*, apontando o arquivo que baixamos no passo anterior. Depois, clique em *Upload*.

Uma nova caixa, *Web Servers*, irá surgir. Clique nessa caixa, marque o botão *nginx* e, na aba à direita, clique em *Apply Content*.

3. Vá para *System > Inputs*, e note que temos dois novos *inputs*, **nginx\_access\_log** e **nginx\_error\_log**. Em ambos, clique no botão *Start Input* — frequentemente, essa operação irá reportar erro. Se for esse o caso, acesse a máquina **log** como o usuário **root** e reinicie o *daemon* do Graylog:

```
# hostname ; whoami  
log  
root
```

```
# systemctl restart graylog-server
```

Após o reinício, volte a *System > Inputs* e verifique que ambos os novos *inputs* estão em estado **RUNNING**, como mostrado abaixo:

**Local inputs** 3 configured

**nginx access\_log** Syslog UDP **RUNNING**

On node ★ 7b614c71 / log.intnet

```
allow_override_date: true  
bind_address: 0.0.0.0  
override_source: <empty>  
port: 12301  
recv_buffer_size: 1048576
```

**Static fields**

from\_nginx: true ✖

nginx\_access: true ✖

---

**nginx error\_log** Syslog UDP **RUNNING**

On node ★ 7b614c71 / log.intnet

```
allow_override_date: true  
bind_address: 0.0.0.0  
override_source: <empty>  
port: 12302  
recv_buffer_size: 1048576
```

**Static fields**

from\_nginx: true ✖

nginx\_error: true ✖

Figura 11. Inputs do nginx ativos

4. Agora, temos que instruir o Nginx instalado na máquina **log** a enviar seus logs para os *inputs* que acabamos de configurar. Vamos ao Ansible!

Acesse a máquina **client** como o usuário **ansible**:

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

Vamos criar uma *role* que envie os logs de acesso e erro do Nginx de servidores web do *datacenter* para o *input* que configuramos no Graylog:

```
$ ansible-galaxy init --init-path /home/ansible/ansible/roles/ nginxlog
- nginxlog was created successfully
```

Desta vez, apenas um subconjunto de máquinas do *datacenter* deve ser o alvo da nossa *role*. Crie o novo grupo *web\_server* no inventário do Ansible:

```
$ echo -e '\n[web_server]\nlog' >> ~/ansible/hosts
```

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

[ntp_server]
log

[log_server]
log

[web_server]
log
```

Agora, edite o arquivo */home/ansible/ansible/roles/nginxlog/vars/main.yml* com o seguinte conteúdo:

```
1 ---
2 logsrv: "{{ groups['log_server'][0] }}.intnet"
```

Sem surpresas até aí — queremos configurar o envio de logs para o concentrador de eventos da rede, que está no grupo *log\_server*. Vamos editar o *template* de configuração do Nginx que irá usar a variável acima: crie o arquivo novo */home/ansible/ansible/roles/nginxlog/templates/nginx\_graylog.conf.j2* com o seguinte conteúdo:

```

1 log_format graylog2_format '$remote_addr - $remote_user [$time_local]
"$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
"$http_x_forwarded_for"
<msec=$msec|connection=$connection|connection_requests=$connection_requests|cache_s
tatus=$upstream_cache_status|cache_control=$upstream_http_cache_control|expires=$up
stream_http_expires|millis=$request_time>';
2
3 access_log syslog:server={{ logsrv }}:12301 graylog2_format;
4 error_log syslog:server={{ logsrv }}:12302;
```

O arquivo acima será incluído na configuração padrão do Nginx instalado na máquina **log** (e, de fato, de qualquer outro servidor web futuro que adicionemos ao *datacenter*), informando que os logs de acesso e erros devem ser enviados para a máquina remota **logsrv** (variável que definimos no arquivo **vars**, acima) num formato compatível com o esperado pelo processador do Graylog.

Vamos às tarefas: edite o arquivo `/home/ansible/ansible/roles/nginxlog/tasks/main.yml` com o seguinte conteúdo e confira:

```

1 ---
2 - name: Setup Nginx <> Graylog logging
3   template:
4     src: nginx_graylog.conf.j2
5     dest: /etc/nginx/conf.d/99-graylog.conf
6     owner: root
7     group: root
8     mode: 0644
9   notify:
10    - Restart Nginx
```

Usando o módulo **template**, a *task* acima copia o *template* que configuramos anteriormente para o diretório `/etc/nginx/conf.d` (o qual é incluído pelo arquivo-padrão `/etc/nginx/nginx.conf`), ajusta suas permissões e reinicia o *daemon* do Nginx.

É claro, temos que criar o *handler*. Edite o arquivo `/home/ansible/ansible/roles/nginxlog/handlers/main.yml` com o seguinte conteúdo:

```

1 ---
2 - name: Restart Nginx
3   service:
4     name: nginx
5     state: restarted
```

Nada de novo: usamos o módulo **service** para reiniciar o Nginx após a execução da *task*.

Como essa nova *role* se aplica apenas a um subconjunto de *hosts* do inventário (os membros do grupo **web\_server**), vamos adicionar uma nova seção ao *playbook* `/home/ansible/ansible/srv.yml`:

```
$ cat << EOF >> ~/ansible/srv.yml

- hosts: web_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - nginxlog
EOF
```

```
$ cat ~/ansible/srv.yml
---
- hosts: srv
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - sudoers
    - ntp
    - snoopy
    - syslog

- hosts: web_server
  become: yes
  become_user: root
  become_method: sudo
  roles:
    - nginxlog
```

Agora, vamos executar o *playbook*. Como apenas a máquina **log** será alvo das modificações que fizemos, o uso da opção **--limit** (ou, de forma mais simples, **-l**) é interessante para acelerar a execução do Ansible:

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

```
PLAY [srv]
```

```
*****
*****
```

```
(...)
```

```
PLAY [web_server]
```

```
*****
*****
```

```
TASK [Gathering Facts]
```

```
*****
*****
```

```
ok: [log]
```

```
TASK [nginxlog : Setup Nginx <> Graylog logging]
```

```
*****
```

```
changed: [log]
```

```
RUNNING HANDLER [nginxlog : Restart Nginx]
```

```
*****
```

```
changed: [log]
```

```
PLAY RECAP
```

```
*****
*****
```

```
log                                : ok=11   changed=3    unreachable=0    failed=0
```

5. E agora? Se temos alterações no repositório local, é hora de enviá-las para o Git remoto:

```
$ cd ~/ansible/ ; git add . ; git commit -m 'Adicionada role para envio de logs
formatados do Nginx de servidores web para o servidor Graylog' ; git push
[master ad7b12d] Adicionada role para envio de logs formatados do Nginx de
servidores web para o servidor Graylog
11 files changed, 138 insertions(+)
create mode 100644 roles/nginxlog/README.md
create mode 100644 roles/nginxlog/defaults/main.yml
create mode 100644 roles/nginxlog/handlers/main.yml
create mode 100644 roles/nginxlog/meta/main.yml
create mode 100644 roles/nginxlog/tasks/main.yml
create mode 100644 roles/nginxlog/templates/nginx_graylog.conf.j2
create mode 100644 roles/nginxlog/tests/inventory
create mode 100644 roles/nginxlog/tests/test.yml
create mode 100644 roles/nginxlog/vars/main.yml
Counting objects: 16, done.
Compressing objects: 100% (11/11), done.
Writing objects: 100% (16/16), 1.69 KiB | 0 bytes/s, done.
Total 16 (delta 2), reused 0 (delta 0)
To nfs:/home/ansible/ansible.git
b1e7a24..ad7b12d master -> master
```

6. Vamos voltar para o Graylog. Em seu navegador, vá para *System > Inputs* e em **nginx access\_log**, clique no botão *Show received messages*.



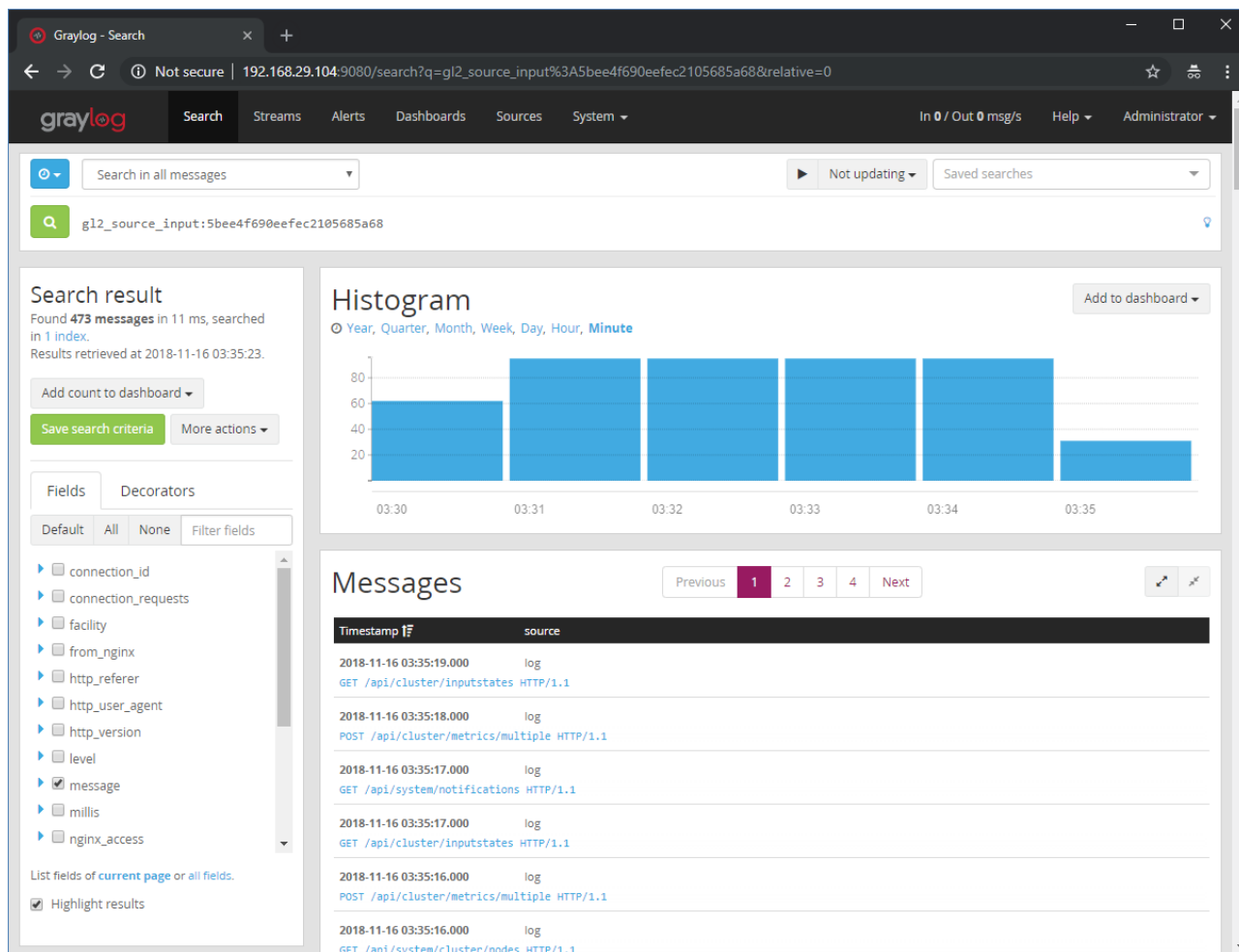


Figura 12. Recebimento de logs formatados do Nginx

Primeiro bom sinal: os logs do Nginx estão sendo recebidos e processados pelo *input* do Graylog, o que significa que nossa *role* no Ansible funcionou a contento. Mas, tirando esse fato, o que mudou? As mensagens mostradas pelo Graylog parecem, em grande parte, as mesmas de antes.

7. Expanda um dos eventos de log recebidos pelo Graylog nesse *input* `nginx_access_log`, como mostrado abaixo:

2018-11-16 03:38:52.000 log	
POST /api/cluster/metrics/multiple HTTP/1.1	
✉ e6b066e1-e961-11e8-8961-0800274a38ea	
<div>Received by</div> <div>nginx access_log on 7b614c71 / log.intnet</div>	
<div>Stored in index</div> <div>graylog_0</div>	
<div>Routed into streams</div> <ul style="list-style-type: none"> <li>All messages</li> <li>nginx</li> <li>nginx requests</li> </ul>	
<div>connection_id</div> <div>696</div>	
<div>connection_requests</div> <div>33</div>	
<div>facility</div> <div>local7</div>	
<div>from_nginx</div> <div>true</div>	
<div>http_referer</div> <div>http://192.168.29.104:9080/search?q=gl2_source_input%3A5bee4f690eefec2105685a68&amp;relative=0</div>	
<div>http_user_agent</div> <div>Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36</div>	
<div>http_version</div> <div>1.1</div>	
<div>level</div> <div>6</div>	
<div>message</div> <div>POST /api/cluster/metrics/multiple HTTP/1.1</div>	
<div>millis</div> <div>0.005</div>	
<div>nginx_access</div> <div>true</div>	
<div>remote_addr</div> <div>192.168.29.106</div>	
<div>remote_user</div> <div>db6a107c-e790-4d4f-af7f-b106f6208f21</div>	
<div>request_path</div> <div>/api/cluster/metrics/multiple</div>	
<div>request_verb</div> <div>POST</div>	
<div>response_bytes</div> <div>295</div>	
<div>response_status</div> <div>200</div>	
<div>source</div> <div>log</div>	
<div>timestamp</div> <div>2018-11-16T05:38:52.000Z</div>	

Figura 13. Processamento de campos individuais em mensagens

Observe o evento acima, e compare-o com o do OpenLDAP que analisamos no começo desta atividade — em lugar de um campo **message** pouco específico, temos agora um grande conjunto de campos pesquisáveis, como:

- **http\_referer**
- **http\_user\_agent**
- **remote\_addr**
- **request\_path**
- **request\_verb**
- **response\_bytes**
- **response\_status**

Todos esses campos são extremamente relevantes em um pacote HTTP, e seu processamento e pesquisa facilita enormemente o trabalho do analista de segurança. E se quisermos descobrir quais pacotes com método POST foram enviados para uma URL específica do servidor web, filtrando pelo IP de origem? Com os campos acima, pesquisas complexas como essa tornam-se totalmente viáveis.

- Encerradas as nossas atividades com o Graylog, recomenda-se que o aluno mantenha a máquina **log** desligada a partir desta sessão. Apesar de ser interessante que recuperemos os logs de todas as VMs do *datacenter* simulado para análise, o fato de essa máquina exigir 4 GB de

memória RAM para operar a contento torna-a um peso muito grande na execução das atividades das próximas sessões.



O *Content Pack* que estamos usando para processar essas mensagens do Graylog é um entre muitos que podem ser encontrados no *Graylog Marketplace*: <https://marketplace.graylog.org/> . Esse website contém centenas de *add-ons* e *plugins* para as versões gratuita e empresarial do Graylog, que permitem estender suas funcionalidades de forma conveniente.

A construção de expressões regulares e regras de processamento de logs para o Graylog é um trabalho árduo, porém necessário para tornar a ferramenta SIEM verdadeiramente efetiva e produtiva para os analistas de segurança. Para facilitar seu trabalho, consulte o nome das ferramentas mais utilizadas na sua organização no *Graylog Marketplace* — quem sabe algum outro usuário já fez um *plugin* que irá facilitar bastante seu trabalho de integração?