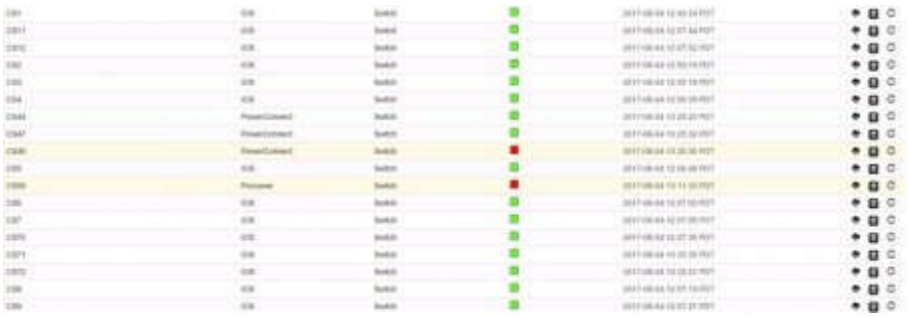
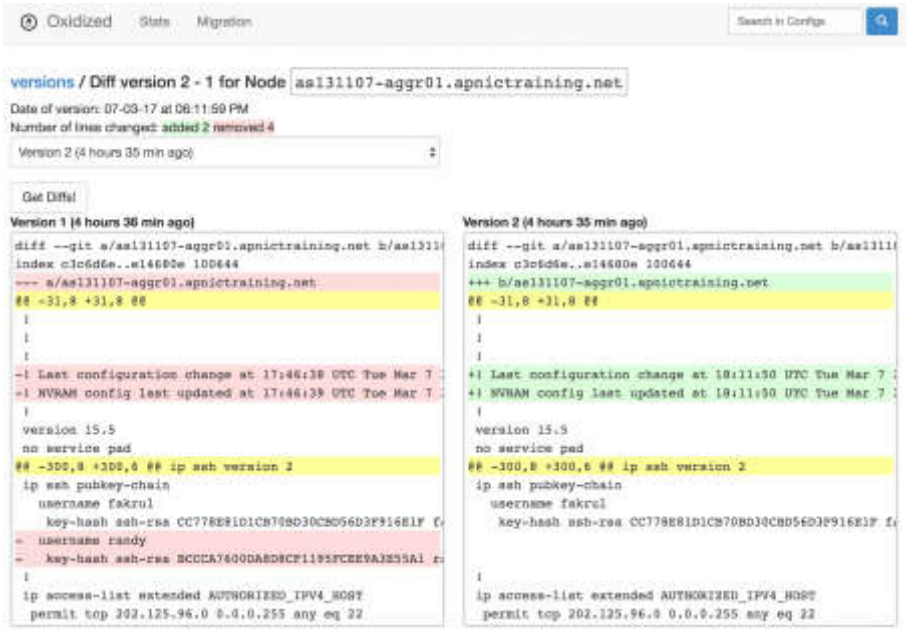
Система управления конфигурациями Oxidized Статьи

Oxidized создавался как альтернатива RANCID двумя разработчиками: Saku Ytti и Samer Abdel-Hafez.





Установка

apt install ruby ruby-dev libsqlite3-dev libssl-dev pkg-config cmake libssh2-1-dev
gem install oxidized oxidized-script oxidized-web

Конфигурация

Запускать oxidized с правами суперпользователя - не самая лучшая идея. Поэтому создадим пользователя oxidized, с правами которого будет запускаться система бэкапа:

useradd oxidized

Создадим для нового пользователя домашний каталог, в котором будут лежать конфигурационные файлы, список устройств и база git:

mkdir /home/oxidized

И дадим права на этот каталог юзеру oxidized:

chown oxidized:oxidized /home/oxidized

Первый запуск создаст все необходимые файлы и каталоги. Не забыли, что запускать надо под специальным пользователем?

su - oxidized
oxidized

Создался каталог /home/oxidized/.config/oxidized, в котором находится всё, что нужно для дальнейшей настройки. Основные настройки системы хранятся в файле config в формате YAML. Он состоит из нескольких частей:

- Input
- Output
- Source
- Основные настройки

Привожу свой конфиг:

```
debian# cat /home/oxidized/.config/oxidized/config
---
//пользователь и пароль, от которых система будет коннектиться к девайсам
username: oxidized
password: p@ssw0rd
//вендор. Я указывал вендора для каждого устройства в файле router.db. Но можно выставить и здесь
model: junos
//Периодичность снятия бэкапа в секундах
interval: 3600
use_syslog: false
debug: false
threads: 30
//таймаут сессии. Многие устройства не успевают выгрузить конфигурацию в дефолтные 20 секунд. Приходится увеличивать
timeout: 140
//количество попыток снять бэкап с каждого устройства, после чего считается, что бэкап сделать не удалось
retries: 3
prompt: !ruby/regexp /^([\w.@-]+[#>]\s?)$/
//IP адрес и порт, на котором будет работать REST API (веб интерфейс по простому)
rest: 127.0.0.1:8888
next_adds_job: false
Vars:
//позволяет исключить из бэкапа критичную информацию, такую как snmp-community, ключи шифрования и т.п.
 remove_secret: true
groups: {}
models: {}
```

```
pid: "/home/oxidized/.config/oxidized/pid"
log: "/home/oxidized/.config/oxidized/log"
//тип подключения к управляемым устройствам
input:
 default: ssh
 debug: false
 ssh:
   secure: false
//где хранится конфигурации (git, text, ...) и настройки хранилища
output:
 default: git
 git:
   user: oxidized
   email: admin@desktopbsd.ru
   repo: "/home/oxidized/.config/oxidized/devices.git"
//откуда берется информация о бэкапящихся девайсах
source:
 default: csv
 csv:
   file: "/home/oxidized/.config/oxidized/router.db"
   delimiter: !ruby/regexp /:/
   map:
    name: 0
    model: 1
    ip: 2
model_map:
 cisco: ios
 juniper: junos
```

Из важного здесь:

- Логин и пароль пользователя на устройствах, под именем которого будет делаться бэкап. Я рекомендую создать на всех девайсах специального пользователя с ограниченными правами, позволяющими бэкапить. К примеру, на RouterOS это группа read
- Разделы input, output, source. В них указывается как подключаться к устройствам, где взять их список и куда выгружать бэкап
- Адрес и порт REST API. Чуть позже я объясню, почему он должен быть 127.0.0.1:8888

В разделе source указано, что информацию о девайсах нужно брать из CSV файла, путь к нему и порядок полей в нем. В моем случае формат такой: имя_устройства:модель:ip

Сам файл router.db:

```
RB128:routeros:192.168.3.128
RB121:routeros:192.168.3.1:oxidized:drugoi_password
sw11:dlink:192.168.3.11:::2222
```

В первой строке указан роутер Mikrotik, о чем говорит поле routeros, с адресом 192.168.3.128 и называться в oxidized он будет RB128. Логин и пароль для подключения будет браться из файла config. Второе устройство RB121 имеет другую учетную запись, параметры которой

указаны после IP адреса - логин:пароль. А третий девайс производства D-Link, с логином/паролем из файла config, но SSH на порту 2222.

Стоит сказать, что устройства можно группировать по вендору или учетным записям. Об этом можно почитать в официальной документации.

Авторизация

На этом этапе уже можно запускать систему и набрав в браузере http://localhost:8888 увидеть её интерфейс. Но у oxidized есть большой недостаток: нет аутентификации в системе. То есть любой может открыть в браузере веб-интерфейс и увидеть все конфиги ваших сетевых устройств. Разработчик занимается только системой бэкапа и смежные фичи внедрять пока не планирует.

Обойдем этот недостаток с помощью Reverse-proxy.
Ставим nginx:
apt install nginx

И настраиваем его на работу как реверс прокси:
nano etc/nginx/sites-available/default

```
auth_basic “Username and Password Required”;
auth_basic_user_file /etc/nginx/.htpasswd;

location / {
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_pass http://127.0.0.1:8888;
}
```

И создаем юзера и пароль:
sudo htpasswd /etc/nginx/.htpasswd username

Теперь nginx будет работать на стандартном порту (или на том, который вы укажете в его настройках), при обращении к нему будет происходить авторизация и пользователь будет перенаправлен на адрес proxy_pass (127.0.0.1:8888 в нашем случае).

Хочу обратить ваше внимание на опцию remove_secrets в конфиге. Она удаляет критичную информацию из бэкапа, такую как SNMP-community, ключи шифрования, ключи Wi-Fi. Например, Mikrotik RouterOS в дефолте умеет скрывать эти данные, если экспорт выполнять с опцией hide-sensitive. Oxidized же может исключать эти данные из конфигов любого вендора. Узнать что именно удаляется из конфига, можно посмотрев в описание вашего вендора в списке. Например, для Cicso:

```
cmd :secret do |cfg|
  cfg.gsub! /^(snmp-server community).*/, '\\1 <configuration removed>'
  cfg.gsub! /username (\S+) privilege (\d+) (\S+).*/, '<secret hidden>'
  cfg.gsub! /^(encrypted radius-server key).*/, '\\1 <configuration removed>'
  cfg
end
```

Для того, чтобы oxidized стартовал как служба, сделайте следующее:
sudo cp /var/lib/gems/2.3.0/gems/oxidized-0.21.0/extra/oxidized.service /lib/systemd/system/
sudo systemctl enable oxidized.service
sudo systemctl start oxidized

Я же просто прописал в crontab запуск после ребута:
sudo -u oxidized crontab -l
@reboot /usr/local/bin/oxidized

Автор: **hellevil** | 15-01-2018, 12:59 | Просмотров: 4 301

Новости по теме

- Cacti - снимаем статистику устройств по SNMP
- Простейший центр сертификации
- Portmaster
- Копирование данных с помощью rsync + ssh
- Все об эмуляции Linux во FreeBSD