


Устанавливаем и настраиваем систему управления конфигурациями сетевого оборудования Oxidized

 interface31.ru/tech_it/2023/06/ustanavlivaem-i-nastraivaem-sistemu-upravleniya-konfiguraciyami-setevogo-oborudovaniya-oxidized.html

Записки IT специалиста

Технический блог специалистов ООО "Интерфейс"

- [Главная](#)
- Устанавливаем и настраиваем систему управления конфигурациями сетевого оборудования Oxidized

О важности регулярного копирования конфигурации сетевого оборудования мы говорить не будем, это очевидно. При этом резервное копирование должно быть системным и централизованным, с единой точкой контроля и управления.

Также немаловажно не только делать резервные копии конфигурации сетевых устройств, но и иметь возможность контролировать изменения в них. Это способно сильно помочь при поиске неисправностей или при расследовании инцидентов. Мы предлагаем установить и использовать для этой цели Oxidized - простую систему управления конфигурациями с открытым исходным кодом.



Онлайн-курс по устройству компьютерных сетей

На углубленном курсе "[Архитектура современных компьютерных сетей](#)" вы с нуля научитесь работать с Wireshark и «под микроскопом» изучите работу сетевых протоколов. На протяжении курса надо будет выполнить более пятидесяти лабораторных работ в Wireshark.

Oxidized - это универсальное решение, поддерживающее более 130 типов устройств, поэтому вам достаточно установить и настроить его один раз и использовать затем без оглядки на применяемое оборудование. Это значительно удобнее, чем решения, предназначенные для оборудования какого-либо отдельного производителя.

В нашем примере мы будем рассматривать работу Oxidized совместно с сетевым оборудованием Mikrotik, для установки мы использовали Ubuntu 22.04 LTS, однако данная инструкция подойдет для Debian 10 и новее, а также Ubuntu 18.04 LTS и новее.

Установка и настройка Oxidized

Oxidized написан на Ruby и поэтому его установка будет несколько отличаться от привычной пользователям DEB-систем. Начнем с установки необходимых зависимостей:

```
apt install ruby ruby-dev libsqlite3-dev libssl-dev pkg-config cmake libssh2-1-dev  
libc-udeb zlib1g-dev g++ libyaml-dev
```

Также установим Git для хранения конфигураций и контроля изменений в них:

```
apt install git
```

А теперь приступим к установке, собственно, Oxidized и необходимых зависимостей, для этого будет использоваться RubyGems - система управления пакетами для языка Ruby:

```
gem install oxidized  
gem install oxidized-script oxidized-web  
gem install psych
```

Установка пакетов через RubyGems может занять некоторое время, поэтому наберитесь терпения.

Затем создадим пользователя oxidized:

```
useradd -s /bin/bash -m oxidized
```

Затем перейдем в эту учетную запись и разово запустим oxidized, чтобы он сформировал необходимую структуру директорий и шаблон конфигурационного файла:

```
su oxidized  
oxidized  
exit
```

Разработчики предлагают хранить конфигурационные файлы и данные в домашней директории пользователя oxidized и мы будем придерживаться этих рекомендаций. Откроем шаблон файла конфигурации **~oxidized/.config/oxidized/config** и внесем в него некоторые изменения, параметры перечислены нами в порядке их следования в файле.

Также обращаем ваше внимание, что oxidized использует для конфигурационного файла формат YAML, а следовательно все отступы в нем следует делать **только пробелами**, допускается два или четыре пробела на отступ.

Прежде всего зададим имя и пароль по умолчанию с которыми oxidized будет соединяться с сетевыми устройствами. Рекомендуем использовать для этого отдельного пользователя с нестандартным именем, которого потом следует завести на всех сетевых устройствах.

```
username: Oxidized  
password: Pa$$word
```

Затем укажем тип устройства по умолчанию, у нас это Mikrotik, поэтому:

```
model: routeros
```

Остальные типы поддерживаемых устройств можно посмотреть в официальной документации:

Supported OS types

Ниже зададим интервал создания копий в секундах, здесь следует исходить из того, как часто вы вносите изменения, в большинстве случаев достаточно будет делать копии раз в сутки:

```
interval: 86400
```

Таймаут по умолчанию имеет смысл увеличить, особенно если у вас много слабых устройств и медленные каналы, мы устанавливаем таймаут в 60 секунд и три повторных попытки:

```
timeout: 60
retries: 3
```

Если вы не хотите, чтобы резервные копии включали чувствительные данные, такие как ключи, пароли и т.п., то добавьте ниже опцию:

```
remove_secret: true
```

Теперь опускаемся к секции **input**, которая отвечает за связь с устройствами. По умолчанию используются протоколы ssh и telnet, но последний устарел и небезопасен, поэтому приводим секцию к виду:

```
input:
  default: ssh
```

Секция **output** отвечает за хранение собранных с устройств информации, для этих целей мы будем использовать Git, поэтому она будет выглядеть следующим образом:

```
output:
  default: git
  git:
    user: oxidized
    email: 1c@zaliv31.ru
    repo: "/home/oxidized/.config/oxidized/devices.git"
```

Пользователя и адрес почты можете указать произвольно, последняя опция указывает на расположение Git-репозитория с данными, как было сказано выше, он будет также располагаться в домашней директории пользователя oxidized.

И, наконец, секция source определяет базу данных устройств, мы будем использовать CSV-файл с двоеточием в качестве разделителя:

```
source:
  default: csv
  csv:
    file: "/home/oxidized/.config/oxidized/router.db"
    delimiter: !ruby/regexp /:/
    map:
      name: 0
      ip: 1
      username: 2
      password: 3
      model: 4
      port: 5
```

Здесь к важным параметрам относится опция **file** - путь к файлу с базой устройств и подсекция **map**, которая определяет в каком порядке в строке CSV файла перечисляются параметры подключения к устройству. Если какой-то параметр мы не задали, то он берется из глобальной конфигурации. В связи с этим мы используем собственную карту, но это не догма, можете делать как удобно вам.

Мы исходили из того, что имя и адрес устройства мы указываем всегда, поэтому они идут первыми. Следующая по частоте использованию опция - это отличные от дефолтных имя пользователя и пароль, затем уже тип устройства и нестандартный порт.

Сохраним файл конфигурации и займемся созданием базы устройств, создадим указанный в конфигурации файл и сразу откроем его на редактирование в **nano**, если вам больше нравится редактор MC, то замените **nano** на **mcedit**:

```
nano ~oxidized/.config/oxidized/router.db
```

Внутри этого файла задаем указанные в подсекции **map** параметры, разделяя двоеточием. В самом простом варианте запись будет выглядеть так:

```
Router_Office:192.168.150.1
```

Мы указали только имя и адрес, остальные параметры будут использованы из глобальной конфигурации.

Если нам нужно указать иные учетные данные, тогда добавим еще два параметра:

```
Router_Office:192.168.150.1:username:password
```

А если нужно указать другой тип устройства, но с теми же учетными данными. то просто ставим нужное количество двоеточий:

```
Router_Sklad:192.168.150.1:::dlink
```

Каждое двоеточие отделяет параметры, если между ними ничего нет, то будет подставлен параметр по умолчанию.

Сохраняем файл базы данных устройств.

Следующим шагом создадим юнит Systemd для работы в качестве службы, для этого скопируем готовый шаблон:

```
cp /var/lib/gems/3.0.0/gems/oxidized-0.29.1/extra/oxidized.service
/etc/systemd/system
```

Обратите внимание, что путь содержит номер версии продукта, поэтому, возможно, его придется уточнить.

Перечитаем содержимое юнитов Systemd:

```
systemctl daemon-reload
```

Создадим директорию в /run:

```
mkdir /run/oxidized
```

И сделаем oxidized ее владельцем:

```
chown oxidized:oxidized /run/oxidized
```

Теперь добавим службу в автозагрузку:

```
systemctl enable oxidized
```

И запустим ее:

```
systemctl start oxidized
```

Проверить работу службы можно командой:

```
systemctl status oxidized
```

А также убедимся, что она работает на порту 8888 локального узла.

```
ss -tln
```

```
root@OXI:~# systemctl status oxidized
* oxidized.service - Oxidized - Network Device Configuration Backup Tool
   Loaded: loaded (/etc/systemd/system/oxidized.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-06-23 17:16:55 UTC; 7s ago
     Main PID: 274 (oxidized)
       Tasks: 8 (limit: 18696)
      Memory: 69.3M
         CPU: 1.728s
    CGroup: /system.slice/oxidized.service
            └─274 "puma 3.11.4 (tcp://127.0.0.1:8888) [/]" * * *

Jun 23 17:16:55 OXI systemd[1]: Started Oxidized - Network Device Configuration Backup Tool.
Jun 23 17:16:55 OXI oxidized[274]: I, [2023-06-23T17:16:55.854540 #274] INFO -- : Oxidized starting, running as pid 274
Jun 23 17:16:55 OXI oxidized[274]: I, [2023-06-23T17:16:55.856228 #274] INFO -- : lib/oxidized/nodes.rb: Loading nodes
Jun 23 17:16:56 OXI oxidized[274]: I, [2023-06-23T17:16:56.137043 #274] INFO -- : lib/oxidized/nodes.rb: Loaded 1 nodes
Jun 23 17:16:56 OXI oxidized[274]: Puma starting in single mode...
Jun 23 17:16:56 OXI oxidized[274]: * Version 3.11.4 (ruby 3.0.2-p107), codename: Love Song
Jun 23 17:16:56 OXI oxidized[274]: * Min threads: 0, max threads: 16
Jun 23 17:16:56 OXI oxidized[274]: * Environment: development
Jun 23 17:16:56 OXI oxidized[274]: * Listening on tcp://127.0.0.1:8888
Jun 23 17:16:56 OXI oxidized[274]: Use Ctrl-C to stop
root@OXI:~# ss -tln
State  Recv-Q  Send-Q  Local Address:Port  Peer Address:Port  Process
LISTEN  0        4096    127.0.0.53:53       0.0.0.0:*          users:((("systemd-resolve",pid=107,fd=14)))
LISTEN  0        1024    127.0.0.1:8888      0.0.0.0:*          users:((("oxidized",pid=274,fd=5)))
LISTEN  0        100     127.0.0.1:25       0.0.0.0:*          users:((("master",pid=270,fd=13)))
LISTEN  0        4096    *:22               *:.*               users:((("systemd",pid=1,fd=36)))
LISTEN  0        100     [::]:25            [::]:.*            users:((("master",pid=270,fd=14)))
```

Если вы получили аналогичный результат, то oxidized был успешно установлен и работает.

Настройка NGINX как обратного прокси

Ну вот, oxidized запущен и работает, а как получить к нему доступ? Дело в том, что разработчики сосредоточились только на системе управления конфигурациями и не реализовали никаких механизмов контроля доступа, поэтому будет лучше, если oxidized будет работать только на локальном узле, а во внешний мир будет смотреть через NGINX, прежде всего установим его:

```
apt install nginx
```

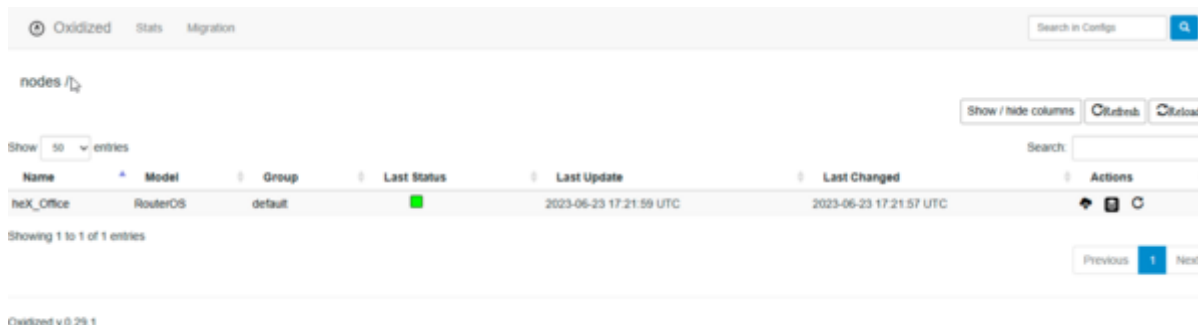
Так как других веб-служб здесь не предполагается, то просто перезапишем конфигурацию NGINX по умолчанию из шаблона oxidized:

```
cat /var/lib/gems/3.0.0/gems/oxidized-0.29.1/extra/oxidized.nginx > /etc/nginx/sites-available/default
```

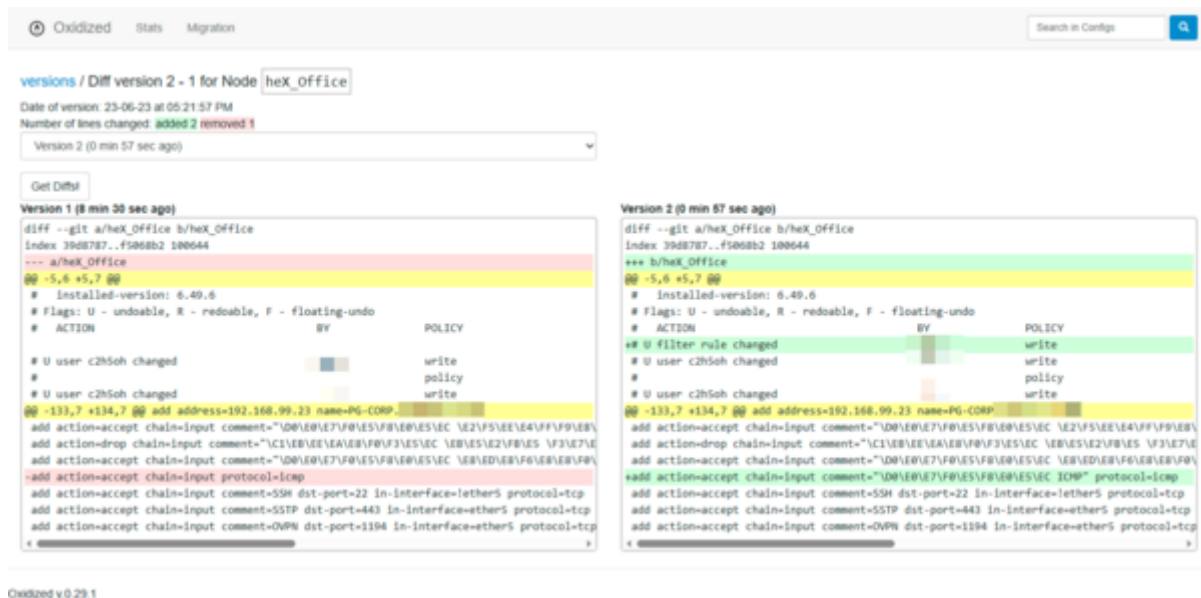
Перезапустим NGINX:

```
nginx -s reload
```

Теперь можем набрать в браузере адрес узла с oxidized и попадем в его веб-интерфейс.



Он довольно прост, но понятен и все необходимое в нем есть. Отсюда же можно скачать файл конфигурации, просмотреть изменения или принудительно выполнить резервное копирование - за это отвечают три кнопки в колонке **Actions**. Давайте посмотрим, как это работает, внесем какие-нибудь изменения в конфиг и заново прочитаем его. Теперь нам доступен просмотр изменений.



При этом сравнить выбранную версию конфигурации можно не только с предыдущей, но и с любой другой. Для этого выбираем нужную версию в выпадающем списке и нажимаем **Get Diffs!** Это очень удобно если вам нужно выяснить кем и когда были внесены некоторые изменения.

Настройка TLS-защиты

В целом, на этом можно и закончить, но имейте ввиду, что абсолютно любой желающий может набрать в браузере адрес oxidized и получить полный доступ ко всем конфигурациям и данным в них. А это неправильно, поэтому примем меры по ограничению доступа. Первое, что нужно сделать в нынешних реалиях - это включить шифрование.

Так как наш сервис внутренний, то можно обойтись самоподписанным сертификатом:

```
openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -keyout  
/etc/ssl/private/oxidized-selfsigned.key -out /etc/ssl/certs/oxidized-  
selfsigned.crt
```

Указанная выше команда выпустит самоподписанный сертификат на 10 лет. При выпуске сертификата желательно указать в поле CN реальное FQDN имя сервера, даже локальное, скажем **oxi.it-31.lab**.

Также сразу создадим файл параметров Диффи-Хеллмана:

```
openssl dhparam -out /etc/ssl/private/dhparam.pem 2048
```

Теперь откроем файл **/etc/nginx/sites-available/default** и изменим существующую секцию следующим образом:

```
server {
    listen 80;
    server_name oxl.it-31.lab;
    return 301 https://$host$request_uri;
}
```

Ниже добавим секцию:

```
server {
    listen 443 ssl http2;
    server_name oxl.it-31.lab;

    ssl_certificate /etc/ssl/certs/oxidized-selfsigned.crt;
    ssl_certificate_key /etc/ssl/private/oxidized-selfsigned.key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:50m;
    ssl_session_tickets off;
    ssl_dhparam /etc/ssl/private/dhparam.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
    ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
    POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
    SHA384;
    ssl_prefer_server_ciphers off;

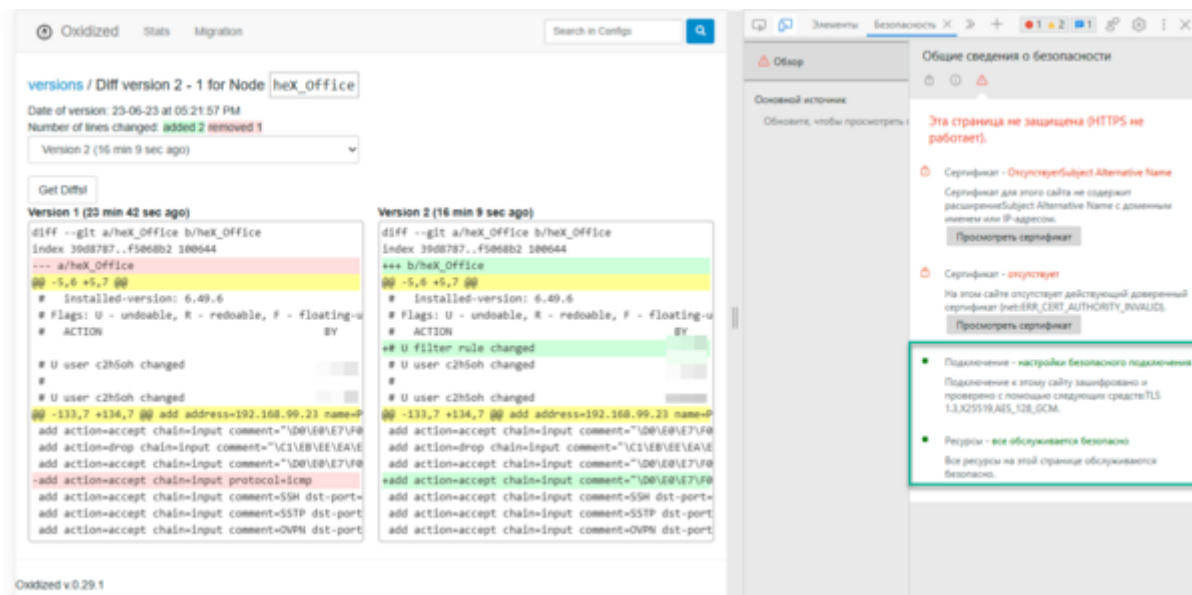
    location / {
        proxy_pass http://127.0.0.1:8888/;
    }

    access_log /var/log/nginx/access_oxidized.log;
    error_log /var/log/nginx/error_oxidized.log;
}
```

Проверяем конфигурацию NGINX и перезапускаем его:

```
nginx -t
nginx -s reload
```

Теперь снова открываем веб-интерфейс и убеждаемся, что соединение зашифровано:



Предупреждения красным можем проигнорировать, это издержки самоподписанного сертификата, главное - два пункта внизу, страница шифруется при помощи TLS 1.3 шифром AES-128 и включена совершенная прямая секретность на основе эллиптической кривой Curve25519.

Настраиваем аутентификацию доступа

И завершающая часть - настройка аутентификации доступа. Для этого установим пакет **apache2-utils**:

```
apt install apache2-utils
```

Теперь создадим базу пользователей, первого пользователя заводим с использованием ключа **-c**, в этом случае будет создан новый файл паролей, а существующий перезаписан:

```
htpasswd -c /etc/nginx/.htpasswd user1
```

Для следующих пользователей используйте команду:

```
htpasswd /etc/nginx/.htpasswd user2
```

Теперь снова откроем **/etc/nginx/sites-available/default** и приведем секцию **location /** к следующему виду:

```
location / {
    auth_basic "Administrator's Area";
    auth_basic_user_file /etc/nginx/.htpasswd;
    proxy_pass http://127.0.0.1:8888/;
}
```

Перезапустим NGINX:

```
nginx -s reload
```

И снова попробуем зайти в веб-интерфейс, теперь у нас это не получится без ввода логина и пароля.

Для удаления пользователя из базы используйте:

```
htpasswd -D /etc/nginx/.htpasswd user2
```

Перезапуск веб-сервера после манипуляций с базой пользователей не требуется.

Надеемся данный материал окажется вам полезен и поможет быстро установить, настроить и начать использовать Oxidized.

Онлайн-курс по устройству компьютерных сетей

На углубленном курсе "Архитектура современных компьютерных сетей" вы с нуля научитесь работать с Wireshark и «под микроскопом» изучите работу сетевых протоколов. На протяжении курса надо будет выполнить более пятидесяти лабораторных работ в Wireshark.
