Kostis Netzwerkberatung
Talstr. 25, D-63322 Rödermark
Tel. +49 6074 881056
http://www.kostis.net/
kosta@kostis.net

HowTo
do-oob-device.py

created by

Kostis Netzwerkberatung
Konstantinos Kostis

do-oob-device.py HowTo

Document Details

Title:	do-oob-device.py..HowTo
Filename:	do-oob-device.py.HowTo.doc
Author:	Konstantinos Kostis, kosta@kostis.net
Manager:	
Version:	V1.01
Date:	18.08.2020

Audience:

`do-oob-device` script framework users

History (CTRL + click on page number to access text mark)

Page	Date	Editor	Reason
All	2019-03-06	kosta@kostis.net	initial documentation
All	2019-07-29	kosta@kostis.net	cleanup and some documentation about the inner workings of scanning

do-oob-device.py HowTo

Table of Contents (CTRL + click to access chapter)

1 Preface

Let's assume you have Cisco OOB router devices connected to console ports of network and/or server devices. Let's also assume you want to check whether the console connected to lines are active and what prompt is shown when connected to said console. You may also want to check if a login is required from the console once you connect.

This is exactly what `do-oob-device.py` was written for and you can use it free of charge at your own risk.

This document describes how to use `do-oob-device.py` allowing easy automated console connection verification for Cisco OOB router network devices. Log files created by `do-oob-device.py` can easily be analysed.

`do-oob-device.py` takes care of logging on to a device using `ssh` (via `paramiko`). It then checks the lines present on the device (using the "`show line`" command) and connects to one line after the other. It sends `[Enter]` in order to find out if anything is connected to a given line and if so log the prompt from the device. If a device is detected the console session is disconnected and the next line is tested. `do-oob-device.py` detects unused lines and will not crash because of consoles not providing utf-8 compatible output. However, if `do-oob-device.py` detect a "Unicode error" it most likely is due to a serial port speed mismatch.

User credentials must be stored encrypted in a personal "`credentials.txt`" file. The environment variable "`DO_DEVICE`" must point to the directory containing "`credentials.txt`" and "`devicelist.txt`".

For every device a log file is created.

`do-oob-device.py` originally was written in Python 3 (3.7.2) and uses the Python module `netmiko/paramiko` (containing `cryptography.fernet`).

If you haven't already, install `netmiko` using `pip`:

```
pip install netmiko
```

`do-oob-device.py` was created using Microsoft Windows. It should however work fine with Linux and the like as well.

https://www.python.org/downloads/

1.1 Personal Configuration Files

Personal configuration files should be located in a directory and the environment variable `DO_DEVICE` should contain that directory path. If you don't already have such a directory consider creating it like this:

```
REM Windows
c:
cd \
mkdir "%LOCALAPPDATA%\Kostis Netzwerkberatung"
mkdir "%LOCALAPPDATA%\Kostis Netzwerkberatung\do-device"
```

Set environment variable `DO_DEVICE` to

```
SET DO_DEVICE=%LOCALAPPDATA%\Kostis Netzwerkberatung\do-device
```

Expand `%LOCALAPPDATA%` before including it to `DO_DEVICE`. Create the files below there.

1.2 credentials.txt

Create a file `credentials.txt`.

```
################################################################################
# %DO_DEVICE%\credentials.txt example
################################################################################
#
# realm;username;password;secret
#
################################################################################
# realm=* means default realm, all realms must start with '*'
# secret=* means same as password
# secret=- means no secret (no enable)
# password or secret starting with " " is encrypted
################################################################################

*;username;encrypted-password;*
```

Replace "`username`" by the account name used to log on to devices.

Replace "`encrypted-password`" by your account password used to login to devices encrypted using `pwencrypt.py`.

If `enable` is not the same as the `encrypted-password`, replace `*` by the enable secret encrypted using `pwencrypt.py`.

Make sure to update this file every time after changing your device account passwords!

1.3 deviceinfo.txt

`deviceinfo.txt` contains information about the device OS and credentials and may also be used to map hostnames to IP addresses if there is no DNS resolution of said hostnames and also no permission to modify your `hosts` file.

```
################################################################################
# %DO_DEVICE%\deviceinfo.txt example
################################################################################
#
# device;ipaddr;type;username;password;secret
#
################################################################################
#
# ipaddr is optional (when there's no hosts/DNS available)
# type is a netmiko type (e.g. cisco_ios, cisco_nxos)
# username/password/secret=* means default (realm=*) value from credentials.txt
# *realm means realm (not default) value from credentials.txt
# otherwise specify different username/password/secret for device
# password/secret must be encrypted
# secret=- means no secret (no enable)
#
################################################################################

hostname1;;cisco_nxos;*;*;*
hostname2;;cisco_nxos;*;*;*
hostname3;;cisco_nxos;*;*;*
hostname4;;cisco_nxos;*;*;*
hostname5;;cisco_nxos;*;*;*
hostname6;;cisco_nxos;*;*;*
hostname7;;cisco_nxos;*;*;*
hostname8;;cisco_nxos;*;*;*
hostname9;;cisco_nxos;*;*;*
```

If no `ipaddr` is given, DNS/hosts will have to be available for the given `hostname`.
`type` (such as "`cisco nxos`") are defined by the Python `netmiko` module.
'`*`' for `username`, `password`, `secret` means use the values from "`credentials.txt`".
You can use a different `username`, `password`, `secret` for each device…

2 Script File Location

Put the following files in a directory included in your `PATH` environment variable:

- `do-oob-device.py`
- `pwdecrypt.py`
- `pwdecrypt.py`

If you don't have such a directory, place the files in a directory of your choice and add that directory path to your `PATH` environment variable.

2.1 Syntax

2.1.1 do-oob-device.py

Test lines/consoles on a given Cisco OOB device.

Syntax:

```
do-oob-device.py device
```

`do-oob-device.py` logs on to device scanning console ports. A log file is created.

2.1.2 pwencrypt.py

Encrypt a password read from the console for use in "`credentials.txt`" and "`devicelist.txt`".

Syntax:

```
pwencrypt.py
```

2.1.3 pwdecrypt.py

`pwdecrypt.py` is a module imported by `do-oob-device.py`. It is not meant to be run directly.

3 On Password Encryption

You may ask yourself: why encrypt a password? Or if you're beyond this, how do I make sure the decrypted password cannot be seen by anyone else?

3.1 Why Encrypt A Password?

The main reason why passwords should never be stored in clear text in configuration files, permanent scripts or anywhere else is that people might find those files and use the credentials in there to do harm. Even if they don't get access to the file directly they might look over your shoulder and see passwords while you display or edit them in your scripts.

If you think that's paranoid, think again. It happens all the time and unless you can be really sure nobody will be ever be able to look at your screen, your devices may not be safe.

3.2 What to Avoid Using Decodable Passwords

One mistake people tend to make is to make files containing password information readable by more than the owner. Make sure your file permissions are such that nobody else can read your files if at all possible.

Another problem is that if you extract a password and then use it as a parameter for a script or program the clear text password is shown in the process list. Never use clear text passwords in command line parameters.

3.3 Make Decodable Passwords More Safe

Use a mechanism that looks up the credentials in your scripts. Decrypt them inside your script (use the Force of Python, Luke).

4 Script Functions Step by Step

4.1 Gather Information About OOB Router Device

`do-oob-device.py` connects to a given ip address/hostname and logs on using the credentials given in `credentials.txt` or `credentials.txt` using netmiko.

While connected runs the command "`show line`" on the OOB router device to figure out the number of lines and interface names. The result is kept im memory and the connection using netmiko is severed.

Now `do-oob-device.py` goes through the output line by line and looks only at lines that contain "TTY". If they start with '`*`' that character is removed.

4.2 Connect to OOB Router Device For Each Line

In the next step the line is split using white space and the number of columns is determined.

For each line the column for "Tty, "Line" and "Roty" is extracted.
If the content of "Roty" is equal to "-" the content of "Line" is used instead.

After that function `connect()` is called handling logon on again using paramiko, which allows for interactive communication and then doing interactive work for each line.

```
def connect(hostname, username, password, line):
```

`connect()` connects to a give host using credentials given using paramiko.
A command with the following syntax is created and executed:

```
ssh –p line hostname
```

`connect()` checks if the login banner of the Cisco OOB router device is returned. If so it will run the function `receive()` on that line.

4.3 Handle Console Port (Sending '\n')

Each port status detection is handled by function `receive()`:

Before anything else happens `receive()` sends `'\n'` (CR/LF), which should return a login prompt of the console device attached. With some luck that includes the device hostname.

Since some console devices take a little time before they actually react to `'\n'` on the console the function sends `'\n'` up to 5 times. After 5 attempts and no reaction an idle console port is assumed.

4.4 Console Port Cleanup

If `receive()` returns data and successfully connected, connect() will send `[Ctrl]^ x` and execute "`disco 1`" on the command line. If "assword:" is read `[Ctrl]C` is sent.