YubiX Virtual Machine Howto
With Local Validation, RADIUS, and OpenVPN

jerichod505

Table of Contents

1.0 Overview

This howto documents the steps involved in setting up a virtual machine that performs the following multiple functions:

- YubiKey authorization infrastructure (yubiauth)
- YubiKey local key store (yubiksm)
- YubiKey One Time Password (OTP) validation server (yubval)
- freeRADIUS infrastructure
- OpenVPN server

Although in a real-world deployment these functions should reside on separate servers for security, for the purposes of this howto they will be located on a single Ubuntu Server VM.

By the end of this howto we will have a Ubuntu Server that will serve as the VPN-accessed entry point into a network, secured with the YubiKey two factor authentication / one time password device.

Note – YubiKey had a previous virtual machine project called YubiRADIUS. As of 11/27/2013 YubiKey is no longer supporting the YubiRADIUS vm, and instead is moving to the YubiX reference authentication software stack. See more about this announcement here. This howto is focused on the YubiX infrastructure and not YubiRADIUS.

Yubico does provide a yubix-vm, and a script to build a virtual appliance. I preferred to start with a standard Ubuntu server, and roll the YubiKey software stack and the OpenVPN server into it, documenting and understanding all the component parts along the way.

2.0 Preparation

The following items are needed to complete this setup:

- A YubiKey – For this example I used a NEO,
  - http://www.yubico.com/products/yubikey-hardware/yubikey-neo/
- A Virtual Machine hosting system – For this example I used VMWare Fusion on Mac OS-X 10.9, and later moved the VM to an ESXi host.
- A Guest Virtual Machine to host the software stack – For this example I used Ubuntu 12.04 LTS x86_64 server.
- A separate VM or physical machine to use as a OpenVPN client. It will need the OpenVPN client software installed. For this example I used a Kali Linux VM.

2.1 YubiKey Configuration

At points in this howto you will need to configure your YubiKey. When the YubiKey arrives at your doorstep, it should be pre-configured at the factory with one of the ID's, called 'slots' in Yubico terminology, preprogrammed to work with Yubico's web-based OTP validation tool. I strongly suggest that you not mess with the pre-programmed slot 1, at least not at first. This will give you a good reference point to return to if you get things confused. Instead, for the purposes of customizing your YubiKey use the second slot, slot 2, as the one you modify for testing.

To perform the customizations, you will need a tool, referred to in YubiKey's documentation as a 'Personalization Tool'. See their web page at http://www.yubico.com/products/services-software/personalization-tools/ for more information, and the download link to the configuration tools. In my case I downloaded the YubiKey Personalization Tool from the Mac App Store, and installed it.

The user guide for the cross platform personalization tool is a PDF file, found at http://static.yubico.com/var/uploads/pdfs/Cross_Platform_YubiKey_Personalization_Tool_3.0.1_User_guide_v5.pdf. It is pretty detailed and comprehensive.

Once you install the personalization tool, start it up, and take a quick tour of the features, without your YubiKey plugged in. Then plug your YubiKey in, and review the information for your key by clicking on the "*About*" tab. If you are already on the "*About*" tab, move over to the "*Tools*" tab, and then back to "*About*" to refresh the information.

You should also do a quick test to verify that everything is working ok, and that you know how to use the YubiKey. Point your browser to the YubiKey demonstration page at http://demo.yubico.com/. Follow the instructions on this page, and verify that the identity programmed into slot 1 works.

> *Note: If you have not done so already, it is a good idea to stop and read through the YubiKey Technical Manual. In it you will learn the difference between generating a OTP out of slot 1 (short press on the button) and slot 2 (long press on the button), along with a much better understanding of how the OTP is generated and validated.*

You can experiment with performing different personalizations on the YubiKey at this point if you want to, just understand that it would be best to stay with personalizing slot 2. Later in the howto we will set up slot 2 to perform our local authentication, so we will overwrite anything you configure in slot 2 now.

3.0 References

Here is a list of some web pages that are pertinent to this howto:

Yubico's home page
http://www.yubico.com/

The YubiKey Technical Manual is actually the best place to start to learn about the YubiKey. Don't let the term 'Technical Manual' or its 41 page length put you off of it. Read the Technical Manual before you even open up and hold your YubiKey. It will save you lots of time down the road.
http://www.yubico.com/wp-content/uploads/2013/07/YubiKey-Manual-v3_1.pdf

Yubico's personalization tools, for customizing and programming (setting the public and private identities, as well as the secret key) the YubiKey.
http://www.yubico.com/products/services-software/personalization-tools/

The reference guide for the Cross-platform YubiKey Personalization Tool is not only very useful for explaining what the various screens, options, and settings in the tool do, but it also gives a great deal of additional useful information on the YubiKey itself.
http://static.yubico.com/var/uploads/pdfs/Cross_Platform_YubiKey_Personalization_Tool_3.0.1_User_guide_v5.pdf

Yubico has a web page for you to test the validation of the OTP shipped in the YubiKey from the factory. This page also allows you to experiment with the concept of using two factor with and without usernames. Note that these credentials are only used in this demo page, so dummy values are best. Certainly don't use anything meaningful here.
http://demo.yubico.com/

The Yubico reference authentication software stack page on GitHub. This is page to start with when setting out to build your own YubiX software reference stack, along with this howto.
https://github.com/Yubico/yubix

The YubiX getting started Wiki page on GitHub. This is a pretty short document, but it spells out the essential steps necessary. For this howto, we are NOT using the Virtual Appliance, but instead are Installing YubiX on our own Ubuntu VM.
https://github.com/Yubico/yubix/wiki/GettingStarted

The OpenVPN home page.
http://openvpn.net/

OpenVPN Access Server software packages, for adding the OpenVPN server to the Ubuntu VM.
http://openvpn.net/index.php/access-server/download-openvpn-as-sw.html

OpenVPN quickstart. The best and quickest read to get you up and running with OpenVPN initially.
http://openvpn.net/index.php/access-server/docs/quick-start-guide.html

OpenVPN client instructions for connecting with linux (why connect with anything less? :})
http://openvpn.net/index.php/access-server/docs/admin-guides/182-how-to-connect-to-access-server-with-linux-clients.html

Other pages that may be of interest

Although written for the older YubiRADIUS project, and not specific to the YubiX VM, more information on using the YubiKey two factor authentication can be found the document
http://static.yubico.com/var/uploads/pdfs/How%20to%20deploy%20YubiRADIUS%20TFA%20solution.pdf

Web page with scripts to build a YubiX virtual appliance (not used in this howto)
http://opensource.yubico.com/yubix-vm/

4.0 YubiX VM build

These steps document my installation of the basic YubiX authentication framework and the OpenVPN Access Server using the items described above in Section 2.

4.1 Ubuntu Server Build

From within Fusion, build a new VM from the Ubuntu server distribution disk. Use the Linux Easy Install option. Set the user/account name of yxadmin[1], and set a password. Click continue, but then customize the settings. Save the VM as YubixVMC. In the settings dialog, select the Network Adapter, and configure for Bridged Networking, using the ethernet interface. Start the VM and let the installation process complete.

1 You can pick whatever account name and password you want – just be sure and record the ones you use.

After the installation completes, reboot the VM, log in, update the network settings as is appropriate for your environment, and apply any updates, including any kernel updates. Reboot the VM after applying the updates.

Log back into the VM, and using apt-get, install gcc and make.

```
sudo apt-get install gcc make
```

Install (or re-install) the VMWare tools in the VM. Note that since the VM is Ubuntu Server, the VMWare Tools CD may not automount when connected. In this case, after selecting to install vmware-tools from Fusion's Virtual Machine menu, mount the CD manually, copy over the files, and start the vmware tools installation manually

```
sudo mount /dev/sr0 /media
tar -xvzf /media/VmwareTools-<version>.tar.gz
cd vmware-tools-distrib
sudo ./vmware-install.pl
```

Answer the prompts as needed. I accepted the defaults, except for the drag-and-drop feature. I did not see a need for it on server. Reboot.

Log back into the VM, and using apt-get, install some additional packages that will be useful down the road: openssh-server, firefox (and its dependancies), gedit, lynx, and curl. Note that the installation of firefox is so that we can connect easily to services that are bound to and accessible only via the localhost interface. Note the IP address of the YubiX VM server (assuming that the IP address was set using DHCP), and reboot.

```
sudo apt-get install openssh-server firefox gedit lynx curl gnome-terminal
wireshark
```

In my case the IP address picked up by the VM was 10.0.0.21, which is what I will use for the rest of this howto. Yours may be different.

At this point you should be able to log into the VM from a terminal session on the Mac, allowing for more flexibility in screen size, and facilitating cut and paste more easily. Might be a good idea here to check the /etc/ssh/sshd_config file on the YubiX VM, and the /etc/ssh_config file on the Mac and make sure that you can forward X windows to your Mac.

Also, change the hostname of the VM to something that is specific to this function. In my case I chose to change it to *yubixvmc*. The name change will take effect on the next reboot. Note that I had to actually manually edit the /etc/hostname file and the /etc/hosts file to make the name change take proper effect. If you make a mistake at this point, the freeRADIUS server will act strangely, or the radiusclient will not be able to connect. See section 7 for more details on this if needed.

If you simply do the following using the hostname command, things will work until the next reboot.

```
yxadmin@ubuntu:~$ sudo hostname yubixvmc
```

```
[sudo] password for yxadmin:
yxadmin@ubuntu:~$ hostname
yubixvmc
```

The correct thing to do is to edit the /etc/hostname file and make sure it contains the hostname, e.g.:

```
yxadmin@yubixvmc:~$ more /etc/hostname
yubixvmc
yxadmin@yubixvmc:~$
```

Also be sure and change the /etc/hosts file so that the name matches. The IPv6 lines are not used in this example.

```
yxadmin@yubixvmc:~$ more /etc/hosts
127.0.0.1    localhost
# comment out line below for name change.
# 127.0.1.1  ubuntu
127.0.1.1    yubixvmc

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
yxadmin@yubixvmc:~$
```

It is a good idea to test this to be sure that both the machine name, and localhost, resolve and work. Since so many pieces of this puzzle make local connections, if this is broken the system will be broken.

```
yxadmin@yubixvmc:~$ ping -c 1 localhost
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_req=1 ttl=64 time=0.225 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.225/0.225/0.225/0.000 ms
yxadmin@yubixvmc:~$ ping -c 1 yubixvmc
PING yubixvmc (127.0.1.1) 56(84) bytes of data.
64 bytes from yubixvmc (127.0.1.1): icmp_req=1 ttl=64 time=0.238 ms

--- yubixvmc ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.238/0.238/0.238/0.000 ms
yxadmin@yubixvmc:~$
```

Of course, if you are going to use this VM in an actual installation, with outward facing IP addresses, you will need to adjust accordingly.

Note – this VM is not set up for running in production. It is only for testing. You will need to harden and configure the VM securely as is appropriate for your specific environment.

4.2 YubiX Software Installation

At this point we pick up with the section Installing YubiX on Ubuntu, found on the getting started guide, https://github.com/Yubico/yubix/wiki/GettingStarted#wiki-installing-yubix-on-ubuntu.

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:yubico/stable
sudo add-apt-repository ppa:yubico/yubix
sudo apt-get update
sudo apt-get install yubix
```

For me 101 new packages were downloaded and installed.

Once the download and installation of the packages is complete, the various post installation scripts will automatically run, prompting for the initial configuration settings. These settings are important to take note of, particularly the passwords for the database and other components. They will be needed in later steps, so be sure that you record them as you go.

The steps you go through are summarized in bullet form below:

- Configuring mysql-server
  - This is the root user for the MySQL database, which is used in several places by the framework. Select a good password here.

- Configuring python-yubiauth.
  - You can make changes to the database used by the python yubiauth package, if you have one that you need to use. For this howto, I am choosing to let the dbconfig-common tool configure the database for yubiauth.
  - Select the “yes” option.
  - Select the database of choice here. Note that some of the getting started wiki appears to be written as if pgsql were used. However, I chose to use mysql here. Select the “mysql” option, then select “yes”.
  - Next you will need to provide the script with the root password of the mysql installation that you set up in the previous step. For the entry “Password of the database's administrative user:” enter the mysql root password, and press “Ok”, and confirm when prompted.
  - I elected to set a password for the python-yubiauth application. Enter the password, and press “Ok”, confirm when prompted.

- Configuring yubikey-ksm.
  - Similar to the yubiauth subsystem, the key storage manager, yubikey-ksm, needs to have a database configured. Again, I chose to let dbconfig-common do the heavy lifting, using mysql, and prompting me where needed.
  - Select the “yes” option.
  - Select the “mysql” option, then select “yes”.
  - Again, provide the script with the root password of the mysql installation that you set up in the previous step. For the entry “Password of the database's administrative user:” enter the mysql root password, and press “Ok”, and confirm when prompted.

- Set a password for yubikey-ksm. Enter the password, and press “Ok”, confirm when prompted.

- Configuring yubikey-val.
  - Similar to the two prior subsystems, yubikey-val, needs to have a database configured. Again, I chose to let dbconfig-common do work, using mysql, and prompting me where needed.
  - Select the “yes” option.
  - Select the “mysql” option, then select “yes”.
  - Again, provide the script with the root password of the mysql installation that you set up in the previous step. For the entry “Password of the database's administrative user:” enter the mysql root password, and press “Ok”, and confirm when prompted.
  - Set a password for yubikey-val. Enter the password, and press “Ok”, confirm when prompted.

Note that an initial API client has been set up. Should you need the information from it, you can retrieve the secret for that client by running:

```
sudo ykval-export-clients
```

At this point, the YubiKey YubiX reference architecture is installed, and the initial configuration done. Probably not a bad idea to take a reboot here.

Next, move on to the OpenVPN installation.

4.3 OpenVPN Installation

Log into the YubiX VM, and start up a Firefox browser session from the VM (here is the first place where the remote X display comes in handy). Browse to the OpenVPN page for downloading the Access Server (AS) package, http://openvpn.net/index.php/access-server/download-openvpn-as-sw/113.html?osfamily=Ubuntu
and select the proper one for the version of Ubuntu you are using – in this case Ubuntu 12 amd/x86 64bit. Download the package and save the file.

Refer to the OpenVPN Quick Start Guide http://openvpn.net/index.php/access-server/docs/quick-start-guide.html for more detail on installing the Access Server. Although it is pretty detailed, some areas were a bit different from the instructions, so I will note those here.

After the OpenVPN AS package has downloaded successfully, install it using dpkg. Note that your version of the software package may be later than 2.0.3

```
cd ~/Downloads
sudo dpkg -i openvpn-as-2.0.3-Ubuntu12.amd_64.deb
```

Several items worth noting are displayed, assuming the installation was successful.

```
To reconfigure manually, use the /usr/local/openvpn_as/bin/ovpn-init tool.
```

```
Access Server web UIs are available here:
Admin  UI: https://10.0.0.21:943/admin
Client UI: https://10.0.0.21:943/
```

Before doing anything else, the password for the openvpn user needs to be set. This is the user, installed on the Ubuntu VM during the OpenVPN package installation, that is the administrative user for configuring the OpenVPN AS. Again, especially if you are going to use this in a production environment, be sure and set a strong password here.

```
yxadmin@ubuntu:~/Downloads$ sudo passwd openvpn
Enter new UNIX password: <enter strong password>
Retype new UNIX password: <re-enter strong password>
passwd: password updated successfully
yxadmin@ubuntu:~/Downloads$
```

At this point it is probably wise to invoke a reboot of the VM, to make sure that all the daemons start up normally.

4.4 Checking The Installation

After reboot, log back into the VM, and check the status of listening services with `netstat`. You should see something similar to the results below. The IP addresses and PID numbers may be different, but the daemons of particular interest are highlighted with a grey background.

```
yxadmin@ubuntu:~$ sudo netstat -plunt4
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:909           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 10.0.0.21:943           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1379/apache2
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      992/python
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      827/sshd
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      1672/0
tcp        0      0 10.0.0.21:443           0.0.0.0:*               LISTEN      1564/openvpn
tcp        0      0 127.0.0.1:5348          0.0.0.0:*               LISTEN      951/python
tcp        0      0 127.0.0.1:904           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 127.0.0.1:905           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 127.0.0.1:906           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      988/mysqld
tcp        0      0 127.0.0.1:907           0.0.0.0:*               LISTEN      1534/python
tcp        0      0 127.0.0.1:908           0.0.0.0:*               LISTEN      1534/python
udp        0      0 0.0.0.0:68              0.0.0.0:*                           805/dhclient3
udp        0      0 10.0.0.21:1194          0.0.0.0:*                           1572/openvpn
udp        0      0 127.0.0.1:18120         0.0.0.0:*                           1309/freeradius
udp        0      0 0.0.0.0:1812            0.0.0.0:*                           1309/freeradius
udp        0      0 0.0.0.0:1813            0.0.0.0:*                           1309/freeradius
udp        0      0 0.0.0.0:1814            0.0.0.0:*                           1309/freeradius
yxadmin@ubuntu:~$
```

The service listening on tcp/8080 is the YubiKey web administration infrastructure. Note that it is only listening on localhost. You can change this and other parameters of this service by editing `/etc/yubico/admin/yubiadmin.conf`, but for security purposes it is recommended that you leave it accessible only via the localhost interface. You will access it using a Firefox browser session started from the VM using the remote X display on the Mac.

Note the mysqld service listening on localhost tcp:3306. If you set up to use mysql, and this service is not running, then probably the rest of the YubiKey framework will not work either.

The freeradius services are listening on all interfaces on udp ports 1812, 1813, and 1814, as well as on the localhost interface on udp:18120. These services will be used by the OpenVPN (and any other service you want to use the RAIDUS server for) in conjunction with the YubiKey.

The python service listening on tcp:943 is the OpenVPN administration interface.

The openvpn service listening on tcp:443 is the web interface that OpenVPN clients can use to check their logins, and download the configuration file for a command-line or automated establishment of the VPN tunnel.

The openvpn service listening on udp:1194 is where the openVPN clients will look to connect to the AS.

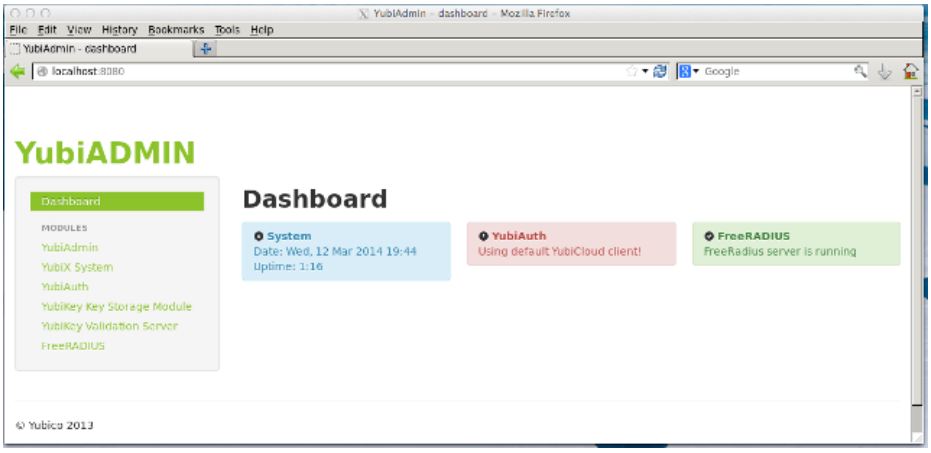
At this point the installation and the first phase of the setup is done. Time now to work on the detailed configuration and testing of the infrastructure.

5.0 YubiX Configuration and Testing

In this section we will complete the configuration in several steps. The first will be to use your VM to authenticate users using the YubiKey's cloud based service for the OTP validation. Later on we will convert the VM to perform all the validation locally.

5.1 YubiX Configuration

As we noted earlier the web-based YubiX configuration tool can only be accessed from localhost. Login to the YubiX VM, and run Firefox, displaying the window on your Mac via X forwarding. In the location bar, enter http://localhost:8080. You should see a pop-up requesting authentication. Unless you changed it during installation, the default username is yubiadmin, and the default password is yubiadmin. You should see the YubiAdmin dashboard, similar to the one below.



The first item is to change the yubiadmin user access password. Navigate to the "*YubiAdmin"* link on the left, and under the "*General"* tab, scroll down to the password field. Enter a new password, making sure to note it. Since you don't get a chance to confirm it, it is best to copy and paste it into this

form. Then press the "*Restart Server*" button. You will need to log back into the admin page. If you bungle something here, you can manually change the values in /etc/yubico/admin/yubiadmin.conf to get back going again.

Navigate to the "*YubiAuth"* page, and select the "*General*" tab. Un-click the option to Autoprovision the keys, and ensure that the Security Level is set for "*When Provisioned*". Then click "*Save*", and then "*Reload web server*". You can change these options later, once you have a working system setup and have more familiarity with the ramifications of doing so.

Naviagate to the "*OTP Validation"* tab. Review the settings here, but don't change anything at this point. These settings are valid for using the YubiCloud services for validation of OTPs.

> *Note: The YubiKey you received should have a valid identity and key, known also to the YubiCloud, programmed into it. The way the validation settings are right now, the YubiX VM will go out to the cloud to validate the OTP. The username and password will stay local to the YubiX VM. Only the validation of the OTP will be done in the cloud.*

Now it is time to create some test users. For now, set these users up as just dummy accounts, making it meaningful and easy to remember. For now, just set the usernames and passwords. We will associate the YubiKey slot 1 identity in a moment. For the purposes of this howto, I set the following users:

username	**password**	**YubiKey identity associated.**
yubislot1	yubipass1	<varies based on your key>
yubislot2	yubipass2	<varies based on your key>
yubislotnone	yubipassnone	None.

Click on the *"Manage Users"* tab, and then on "*Create new user*". Enter the three users described above, pressing "*Save*" each time, and checking for the green 'User created!' message. Once this is done, click back on the "*Manage Users*" tab and verify that all three users are created.

Next we are going to associate the YubiKey you have, with the yubislot1 user. Assuming that you followed the preparation steps above, including testing your YubiKey on the yubico demo page, you should be familiar with its operation by now. Insert the key into the Mac, select the yubilsot1 username in the "*YubiAuth*" page, "*Manage users*" tab, and click inside the Assign Yubikey field. Press the YubiKey button with a short press, for slot 1. The automatic return key included in the OTP string submitted the form for you. Click on the "*Manage Users*" tab again, and check that your key identity for slot 1 of your YubiKey is listed now. The modhex converted public identity for your key, slot 1, should be listed there.

> *Note that one of the cool things about the YubiAuth system is that multiple YubiKeys can be associated with a given user. Simply select that user again, and enter a new key, keeping the old one.*

5.2 YubiX OTP Testing

Now that you have a valid username (yubislot1) and password (yubipass1) associated with a YubiKey, you are ready to test the use of your VM and Yubico's Cloud service to validate your key. Your local system will check the username and password, while passing the OTP off to Yubico to verify that it is a valid OTP. To test this, click on the "*FreeRADIUS*" page and select the "*General"* tab. The client secret has been pre-programmed in for this service – leave it be. Click inside the username field and type the username **yubiuser1**, then click inside the password field, and type the yubiuser1 password of *yubipass1*. Before taking any additional actions, while the insertion point is sitting at the end of the password you typed, press the button on the YubiKey to generate the OTP appended to the end of the password. The form will auto-submit at the end of the OTP string. You should see a green bordered box at the top of the page, and text similar to the following:

```
Sending Access-Request of id 223 to 127.0.0.1 port 1812
      User-Name = "yubislot1"
      User-Password = "yubipass1<your unique OTP>"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=223, length=20
```

The highlighted text (highlighted in this document, not on the screen) that says Access-Accept indicates that everything worked! If you did not get the green-bordered box, and the Access-Accept message, then something went wrong.

If you see a message similar to the following:

```
rad_recv: Access-Reject packet from host 127.0.0.1 port 1812, id=49, length=27
      Reply-Message = "false"
```

It means that the authentication failed for one or more reasons. The authentication could fail because:

- The username or password is wrong,
- The OTP is not the right OTP for that user,
- You did not include an OTP
- You tried to use the same OTP more than once.
- The YubiX VM can't get to the internet. Since the slot 1 OTP is validated against Yubico's Cloud service if the VM can't get to the internet, even if the username and password is right, the authentication will fail.

You can get more information as to why things failed by looking at the log files. The two log files that are of most interest here are:

- /var/log/yubiauth.log
- /var/log/freeradius/radius.log

However, don't forget to consult the apache logs as well if things seem broken.

Assuming that everything works, then take a few moments to try some different combinations of things that should and should not work, so that you confirm that you understand the YubiX VM operation. You might even fire up a packet capture and look at some of the network traffic.

5.3 YubiX Local Validation

At this point we are ready to convert the YubiX VM over to local OTP validation. If this is the end state you want to achieve, then continue on with this section. If, however, you want to use the YubiCloud validation with your OpenVPN configuration, then you can skip this section and set up the OpenVPN using the information in section 6.0.

Converting to local OTP validation will require that the Key Store Module (yubiksm) on the YubiX VM know the public and private identities as well as the secret that you are going to program into the YubiKey slot2. The YubiX Getting Started wiki has a brief section (OTP validation, Using your own validation server) covering this process at https://github.com/Yubico/yubix/wiki/GettingStarted#wiki-otp-validation. However, I found that some of the information in the three sections (Generate KSM Key, Generate Keys, and Import Keys) to be incorrect or lacking for my setup. As a result, we are going to walk through the steps necessary in this howto.

First, we need to set up the KSM key, that will be used to import the Yubikey private information. You can refer to the Yubico wiki page at https://github.com/Yubico/YubiKey-ksm/wiki/GenerateKSMKey. We will make a key named yubixvmc, with a unique ID. One wrinkle that we will encounter is the entropy generation. The process of generating the GPG key requires a source of entropy, and normally moving the mouse around performs that function. However, since we are using a server, that is not a viable option. In my case, when the gpg key generation process complained about not having enough entropy, I opened a second window and login to the YubiX VM, and generated network traffic via pings, nslookup, and other network and keyboard actions. Once you achieve enough entropy the key generation should proceed. During this process you will need to enter a passphrase to use to unlock the gpg key. Be sure and record that passphrase, and we will call that the **yk-ksm** password.

```
yxadmin@yubixvmc:~$ gpg --gen-key
gpg (GnuPG) 1.4.11; Copyright (C) 2010 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory `/home/yxadmin/.gnupg' created
gpg: new configuration file `/home/yxadmin/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/yxadmin/.gnupg/gpg.conf' are not yet active during this
run
gpg: keyring `/home/yxadmin/.gnupg/secring.gpg' created
gpg: keyring `/home/yxadmin/.gnupg/pubring.gpg' created
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 2
DSA keys may be between 1024 and 3072 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: yk-ksm yubixvmc import key
Email address:
```

```
Comment:
You selected this USER-ID:
    "yk-ksm yubixvmc import key"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key. <use a strong password here>

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: WARNING: some OpenPGP programs can't handle a DSA key with this digest size
++++++++++.+++++..++++++++++..+++++++++++++++...+++++++++++++++..+++++.++++++++++..++++++
+++++++++..+++++.....++++++++++.+++++.+++++...++++++++++++++++++..+++++>....+++++.>++++
++++++

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 282 more bytes)
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
++++++++++.++++++++++.+++++++++++++++++++++++++++++++......+++++.++++++++++++++++++++++
+..+++++.++++++++++++++++++++++++++++++.++++++++++.++++++++++.++++++++++>+++++..+++++>.++
+++>..+++++>.....+++++..>+++++....<+++++>+++++...................<+++++.....>.++++
+.........................<.+++++>+++++...........<+++++.+++++^^^
gpg: /home/yxadmin/.gnupg/trustdb.gpg: trustdb created
gpg: key <id> marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub
      Key fingerprint = <fingerprint>
uid                  yk-ksm yubixvmc import key


yxadmin@yubixvmc:~$
```

The next step is to generate the information needed to program the second slot in the YubiKey for use with the local keystore. Yubico has a tool that can generate the necessary fields easily, which can then be imported both into the local KSM, and programmed into the YubiKey. The wiki page https://github.com/Yubico/YubiKey-ksm/wiki/GenerateKeys decribes this in more detail for generating a number of keys. However, we only need one, and we want to see what is happening in detail, so we will do this slightly differently.

First, generate the key data. You can select a starting number for the key identity that suits you. For this one I just picked 2014 out of thin air.

```
yxadmin@yubixvmc:~$ ykksm-gen-keys 2014 |tee key2014.txt
# ykksm 1
# serialnr,identity,internaluid,aeskey,lockpw,created,accessed[,progflags]
2014,ccccccccitu,1534a03b37fb,b6c6ac96af7d0faa8643ac554ff633fb,3f6c10471086,20
14-03-13T15:40:46,
# the end
yxadmin@yubixvmc:~$
```

Normally you would want to pipe this output straight to the gpg encryption that you set up in the previous step, so that the identity and the secrets are not floating around in clear text. For this howto, we will leave it this way for now, so that we can work with the output saved in the file 'key2014.txt' more later.

Now we are going to use this information in two places. First we are going to make an encrypted

version that we can import into the KSM. Then, we are going to program the information into the YubiKey slot 2.

To make the encrypted version, run the gpg utility with the options shown below. Note that '16405BDA' is the ID of the ksm key we made a few steps prior. You will need to enter the password you used to protect that key, the yk-ksm password. The result is an ascii-armored encrypted version of the key2014.txt file, called key 2014.txt.asc.

```
yxadmin@yubixvmc:~$ gpg -a --encrypt -r 16405BDA -s key2014.txt

You need a passphrase to unlock the secret key for
user: "yk-ksm yubixvmc import key"
2048-bit DSA key, ID 16405BDA, created 2014-03-13

gpg: gpg-agent is not available in this session
Enter passphrase:
yxadmin@yubixvmc:~$ ls -l key2014*
-rw-rw-r-- 1 yxadmin yxadmin  193 Mar 13 15:40 key2014.txt
-rw-rw-r-- 1 yxadmin yxadmin 1299 Mar 13 15:59 key2014.txt.asc
yxadmin@yubixvmc:~$
```

Now the key2014.txt.asc file is in the correct format to import into the KSM database. This process is described in the Yubico wiki at https://github.com/Yubico/YubiKey-ksm/wiki/ImportKeysToKSM. Be aware, however, that the command listed in the wiki is for a pgsql database, not for a mysql database. So assuming you use mysql, follow the steps below.

You will need two passwords in this step. In the ykksm-import command invocation you use the yubikey-ksm application password that you set during the database setup. When prompted for the passphrase you use the yk-ksm import key password you set when you made the GPG key earlier.

```
yxadmin@yubixvmc:~$ ykksm-import --verbose --database
'DBI:mysql:dbname=ykksm;host=127.0.0.1' --db-user ykksmreader --db-passwd
<yubikey-ksm password> < ./key2014.txt.asc

You need a passphrase to unlock the secret key for
user: "yk-ksm yubixvmc import key"
2048-bit ELG-E key, ID C083EF24, created 2014-03-13 (main key ID 16405BDA)

Enter passphrase: <enter the yk-ksm import key password>

Verification output:

<snipped for brevity>

You need a passphrase to unlock the secret key for
user: "yk-ksm yubixvmc import key"
2048-bit ELG-E key, ID C083EF24, created 2014-03-13 (main key ID 16405BDA)

Enter passphrase: <enter the yk-ksm import key password>

line:
2014,ccccccccccitu,1534a03b37fb,b6c6ac96af7d0faa8643ac554ff633fb,3f6c10471086,20
14-03-13T15:40:46,
      serialnr 2014 publicname ccccccccccitu internalname 1534a03b37fb aeskey
b6c6ac96af7d0faa8643ac554ff633fb lockcode 3f6c10471086 created 2014-03-
13T15:40:46 accessed  eol

yxadmin@yubixvmc:~$
```

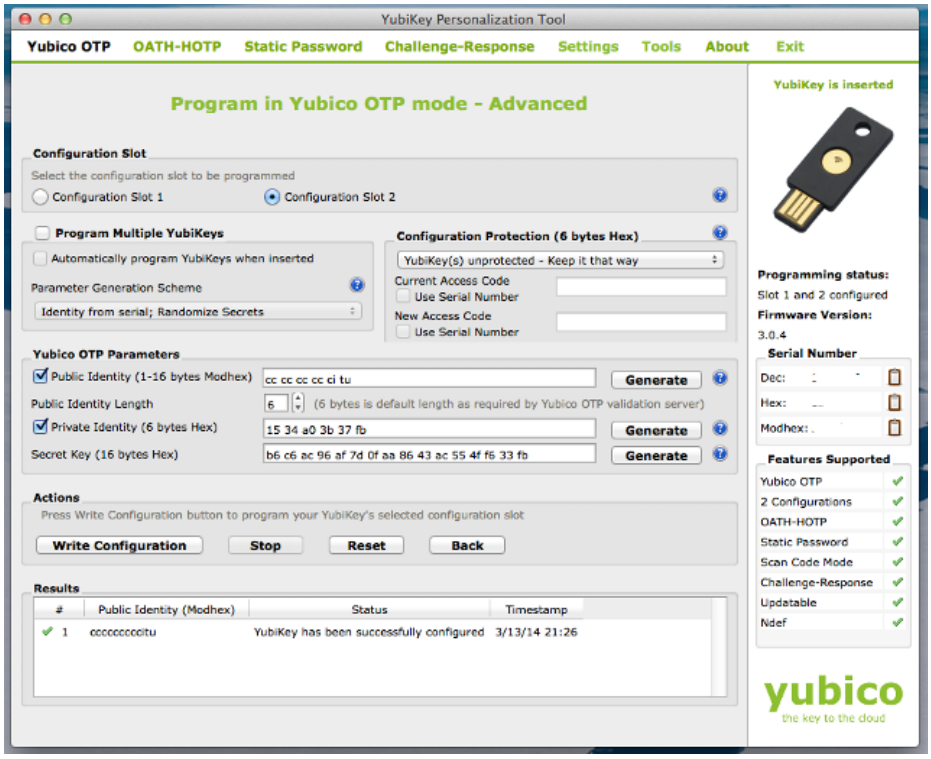
So now we have an identity programmed into the local KSM. We need to put that same information in the YubiKey slot 2. To do this, we will return to the YubiKey Personalization Tool. Insert your YubiKey into the Mac, and start the Personalization Tool. Select the Yubico OTP option on the top line, and then select the Advanced button.

Next select the Configuration Slot 2 option. The Public Identity, Private Identity, and Secret Key need to be set to the values that were generated when we made the key data a few steps ago. To refresh, when we made the key data the commands looked like this:

```
yxadmin@yubixvmc:~$ ykksm-gen-keys 2014 |tee key2014.txt
# ykksm 1
# serialnr,identity,internaluid,aeskey,lockpw,created,accessed[,progflags]
2014,ccccccccitu,1534a03b37fb,b6c6ac96af7d0faa8643ac554ff633fb,3f6c10471086,20
14-03-13T15:40:46,
# the end
yxadmin@yubixvmc:~$
```

- The value we will use for the Public Identity is the value highlighted in blue, ccccccccitu
- The value we will use for the Private Identity is the value highlighted in green, 1534a03b37fb
- The value we will use for the Secret Key is the value highlighted in yellow, b6c6ac96af7d0faa8643ac554ff633fb

Copy and paste these values into the proper places in the personalization tool. You don't have to worry about the spacing between the bytes, as the tool takes care of that automatically. Once they are pasted in, press the "*Write Configuration*" button. Name a log file if you wish, for instance key2014.csv, and press the "*Save*" button. You should see that the key has been successfully configured in the results box. A screen shot of the personalization tool window is shown below:



You can test that the key's slot 2 has been programmed by opening a text editor, and doing a long-press (for slot 2) on the key. In my case this resulted in the string below:

ccccccccitucbbrtcfhlvfcugekbtgiikktcgrrecej

The first 12 characters are recognizable as the public identity as programmed in above. The rest are generated according to the OTP algorithm based in part on the private identity and secret key.

The next step is to configure the YubiX VM to use the local validation service, including the local KSM. A new client, able to be used to access the local validation server, was generated during installation time. You can retrieve it's information from the YubiAdmin web interface by clicking on the "*YubiKey Validation Server*" link, then the "*API Clients*" tab. The new Client ID (most likely 1) and the new API key are listed. Copy those two to a temporary text editor, or open a second browser tab from "*YubiAuth*" link. Within the "*YubiAuth*" page, select the "*OTP Validation*" tab. Since we will be overwriting these values, it is a good idea to print the page to a PDF file so that you have the values in case you need them again.

Then, substitute the new Client ID and API key that were found in the "*Yubikey Validation Server*" page / "*API Clients*" tab for the current values in the "*YubiAuth*" page. Then replace the list of Validation Server URLs with the single line:

```
http://127.0.0.1/wsapi/2.0/verify
```

Then press the "*Save*" button. Click on the "*General*" tab and press the "*Reload web server*" button.

The final step in this process is to associate the slot 2 values with a specific user. In this case, we will will leave the slot 1 user alone, and will assign the slot 2 key to the yubislot2 user, much the same was as we did for the slot 1 user in section 5.1.

Click the "*YubiAuth*" link, and then the "*Manage Users*" tab. Click on the yubislot2 user, and put the insertion point in the "assign YubiKey" text box, and long-press the YubiKey button. Return to the Manage Users tab and verify that the slot2 public ID is assigned to the slot 2 user.

We are ready to test the slot 2 and local validation.

5.4 YubiX Local Validation Testing

Once again, click on the "*FreeRADIUS*" page link and select the "*General*" tab. The client secret is still the same. Click inside the username field and type the username **yubiuser2**, then click inside the password field, and type the yubiuser2 password of *yubipass2*. Before taking any additional actions, while the insertion point is sitting at the end of the password you typed, long-press the button on the YubiKey to generate the OTP from the slot 2 identity and append it to the end of the password. The form will auto submit at the end of the OTP string. You should see a green bordered box at the top of the page, and text similar to the following:

```
Sending Access-Request of id 37 to 127.0.0.1 port 1812
      User-Name = "yubislot2"
      User-Password = "yubipass2ccccccccccituvchilugvelvubdeuglikidbbnfhglgbj"
      NAS-IP-Address = 127.0.0.1
      NAS-Port = 0
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=37, length=20
```

Similar to before, the highlighted text that says Access-Accept indicates that everything worked! If you did not get the green-bordered box, and the Access-Accept message, review the steps above to be sure you did not miss anything, and ensure that you are using the credentials and OTP for the YubiKey slot2.

If you want to convince yourself that all the action is being conducted locally, you can fire up a packet capture and run a few tests. There should be no access to the YubiCloud for OTP validation at this point.

Although written for the older YubiRADIUS project, and not specific to the YubiX VM, more information on using the YubiKey two factor authentication can be found the document http://static.yubico.com/var/uploads/pdfs/How%20to%20deploy%20YubiRADIUS%20TFA%20solution.pdf

We are ready to use our infrastructure for authenticating remote users via VPN.

6.0 OpenVPN Configuration

The next task is to setup the OpenVPN access server. The YubiKey and freeRADIUS setup we have

built only supports PAP authentication, so we need to configure that during this process.

The OpenVPN project's quickstart is probably one of the best and quickest ways to get you up and running with OpenVPN, at http://openvpn.net/index.php/access-server/docs/quick-start-guide.html.

You have already done the installation, so we are now in the Configuring phase. The web-based administration tool is reachable external to the VM, so you should be able to access it at the IP of the YubiX VM. In my case I access the admin interface at

https://10.0.0.21/admin.
Or at
https://10.0.0.21:943/admin

If you have a different external facing IP, then use that.

> *Note that the OpenVPN does not listen on the localhost interface by default, so you will need to go outside the yubixvmc to access the configuration page.*

Login with the OpenVPN administrative password set during the installation. Review and accept the EULA.

> *Note that you will have two user accounts that you can use for testing purposes. If you need more, and before you shift into production, you will need to purchase the appropriate number of OpenVPN licenses.*

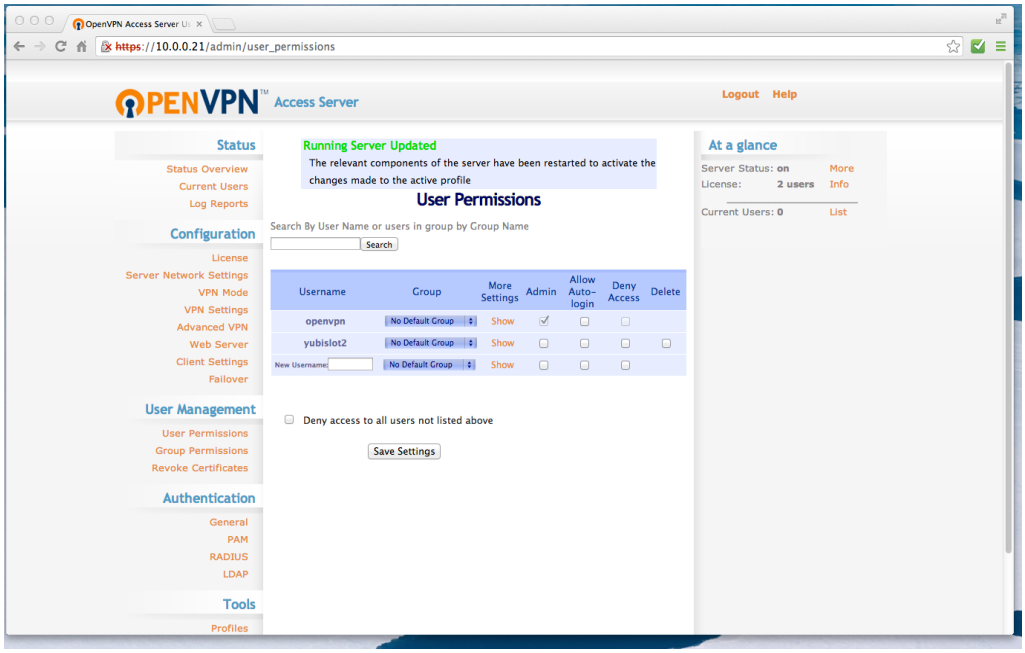
If you want to make the adjustment so that OpenVPN Admin Web UI listens on all interfaces, or only on localhost, do so under "*Configuration*" then "*Server Network Settings*", scrolling down to the "*Admin Web UI*" section. If you do add localhost to the list of interfaces to listen on, then you can access the OpenVPN admin UI at:

https://localhost:943/admin

If you want to make the adjustment so that OpenVPN listens on all interfaces, do so under "*Configuration*" then "*Server Network Settings*".

I left most everything set at the default setting, with the following exceptions:

Click "*User Permissions*" (under "*User Management*"). Add a second user called yubislot2. Press Save, and then press "*Update Running Server*".



Click on "*General*" (under "*Authentication*"), and select RADIUS as the method of user authentication. "*Save*", and then "*Update the Running Server*".

You will need to ensure that the OpenVPN configurations are correct for the addresses used. Note that in particular you will need to ensure that the proper hostname or IP address is entered in the VPN server configuration, under "*Configuration*", then "*Server Network Settings*", "*VPN Server*" section. Note that this is how clients connect, so it cant be localhost only. This address is passed to the clients in the client configuration file.

Click on "*RADIUS*" (under "*Authentication*"), and ensure that the PAP RADIUS Authentication method is selected. Then add the YubiX VM information in the RADIUS settings fields. Since the OpenVPN and RADIUS servers are on the same VM, you have two choices on the address you specify here.

If you want to use the localhost internal connection to the RADIUS server, that works great. Simply specify 127.0.0.1 in the address field, enter the RADIUS shared secret, click to enable accounting, save the settings, and update the running server. Note that by default the RADIUS server listens on localhost only, so you dont need to make an adjustment to its configuration.

You can have two or more RADIUS server connection methods, say one to localhost, and one to a backup external host.

> *Note that for security you best enter only the RADIUS servers you actually will use, so that you are not reaching out to bogus or non-existing RADIUS servers.*

If, on the other hand you want to use the external facing IP you will need to make an additional change to the YubiX freeRADIUS portion.. Enter the IP address, the RADIUS Shared Secret (go back to the YubiAdmin interface and find it under the "*FreeRADIUS*" link if you forgot what it was), click to enable accounting, and save the settings and update the running server. Before you move on, though, you will need to ensure that the freeRADIUS server is listening on the external interface. By default in this setup it is not. The RADIUS server can listen on both internal and external interfaces.

To do this, you can either edit the file /etc/freeradius/clients.conf by hand, or you can update the file in the YubiAdmin interface by clicking on the "*freeRADIUS*" link, and the "*RADIUS clients*" tab.

Either way you need to add a stanza to the client configuration file to specify how you want the freeRadius server to listen. You can either specify a specific host (the most secure) or add a subnet of networks to allow. For the purposes here, we will just specify the Yubix VM, which is also our openVPN host. Add the following to the clients.conf file:

```
client yubix_openvpn {
    ipaddr = 10.0.0.21
    netmask = 32
    secret = testing123
    nastype = other
    }
```

Save the file, and be sure and restart the radius server from the "*General*" tab.

Thats it. The OpenVPN configuration should be done. It is time to move on and test the whole system out from end to end.

7.0 System Testing

If the Yubix VM was implemented in an ESXi environment or on a physical server, we would use a separate system to establish a VPN tunnel. My recent experience with VMWare Fusion on the Mac has been problematic in trying to connect to Fusion-hosted VMs from outside of the Fusion host. As a result, we will test this by using a separate VM hosted on the same machine as the YubiXVM that we will use to make a VPN connection.

In my case, I had a Kali Linux VM handy on my Mac, so we will use it as the VPN client.

Start up the Kali VM, and ensure that you establish a network connection via Bridging to the ethernet interface. Your YubiX VM and your Kali VM should be bound to the same ethernet interface on your Host Mac.

Since you have a Kali VM with all the tools, perhaps a quick nmap scan of the YubiX VM would be a good idea, to get a picture of the running services.

```
root@kali:~# nmap -n -sS -F 10.0.0.21

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-14 09:49 MDT
Nmap scan report for 10.0.0.21
Host is up (0.00010s latency).
Not shown: 97 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
MAC Address: 00:0C:29:75:A0:58 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
root@kali:~# nmap -n -sU -p 1194,1812-1814 10.0.0.21

Starting Nmap 6.40 ( http://nmap.org ) at 2014-03-14 09:49 MDT
Nmap scan report for 10.0.0.21
Host is up (0.00041s latency).
PORT     STATE         SERVICE
1194/udp open|filtered openvpn
1812/udp open|filtered radius
1813/udp open|filtered radacct
1814/udp open|filtered tdp-suite
MAC Address: 00:0C:29:75:A0:58 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.42 seconds
root@kali:~#
```

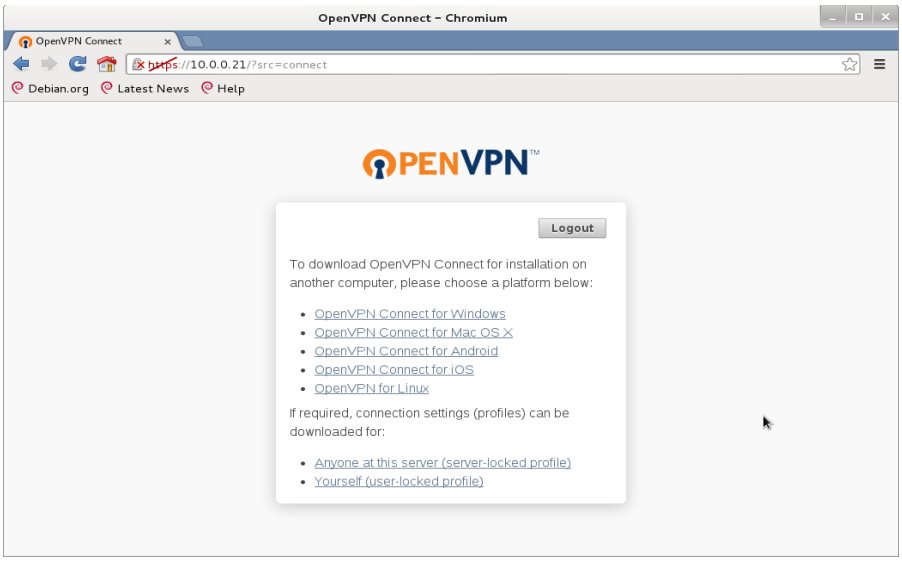
This all looks right, so lets proceed. We can start the client connection process from the Kali VM using

a web browser. Start up a web browser (I use Chrome on the Kali VM) and point it to the Yubix VM using https:

https://10.0.0.21

You will need to accept the self-signed certificate, and proceed anyway. You should see an OpenVPN login prompt. At this point you will use your YubiKey slot 2 credentials, including the slot 2 password with the OTP concatenated (remember, this is a long press) to login.

If everything worked, you should be able to login using these credentials, and see a page similar to the one below:



If the login fails, then perhaps a trip through the troubleshooting section below will help. If the login process seems to hang for a long time, then fail, check the OpenVPN logs for an indication that the RADIUS communication timed out. If so, check to see that the freeRAIDUS server is listening on the interface (eth0 or localhost) for the connection information you entered into the RADIUS configuration section of the OpenVPN server, and that the RADIUS secret is right.

Assuming you are able to login, then the next thing is download the profile for the openVPN client. Click on the link titled "*Yourself (user locked profile)*" and download and save the client.ovpn file.

Another nice touch to using the Kali VM is that the openVPN client is already there. As a final test, start up the openVPN client from the command line, and authenticate using your YubiKey slot 2 identity and OTP. Note that when you are prompted for the Auth Password, nothing will be echoed, even when the YubiKey is feeding the OTP in. Since you are doing a long-press, hold the button until you see the additional messages starting to scroll by.

```
root@kali:~# openvpn --config ~/Downloads/client.ovpn
Fri Mar 14 10:44:14 2014 OpenVPN 2.2.1 x86_64-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11]
[eurephia] [MH] [PF_INET6] [IPv6 payload 20110424-2 (2.2RC2)] built on Jun 18 2013
Enter Auth Username:yubislot2
Enter Auth Password:<enter the password, and long-press the YubiKey button>
Fri Mar 14 10:44:33 2014 NOTE: OpenVPN 2.1 requires '--script-security 2' or higher to
call user-defined scripts or executables
```

```
Fri Mar 14 10:44:33 2014 Control Channel Authentication: tls-auth using INLINE static key
file
Fri Mar 14 10:44:33 2014 Outgoing Control Channel Authentication: Using 160 bit message
hash 'SHA1' for HMAC authentication
Fri Mar 14 10:44:33 2014 Incoming Control Channel Authentication: Using 160 bit message
hash 'SHA1' for HMAC authentication
Fri Mar 14 10:44:33 2014 LZO compression initialized

<snipped for brevity>

Fri Mar 14 10:44:35 2014 /sbin/ifconfig tun0 172.27.232.2 netmask 255.255.248.0 mtu 1500
broadcast 172.27.239.255
Fri Mar 14 10:44:40 2014 /sbin/route add -net 10.0.0.21 netmask 255.255.255.255 gw
10.0.0.1
Fri Mar 14 10:44:40 2014 /sbin/route add -net 0.0.0.0 netmask 128.0.0.0 gw 172.27.232.1
Fri Mar 14 10:44:40 2014 /sbin/route add -net 128.0.0.0 netmask 128.0.0.0 gw 172.27.232.1
Fri Mar 14 10:44:40 2014 Initialization Sequence Completed
```

Once you see the route being added and the initialization sequence completed, your VPN tunnel is established. You can further verify this by looking at ifconfig:

And after:

```
root@kali:~# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-
00-00
          inet addr:172.27.232.2  P-t-P:172.27.232.2  Mask:255.255.248.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:504 (504.0 B)
```

At this point we have accomplished our objectives. Once this all is working, it may be useful to take some time and look through the log files, and perform various different tests to solidify your understanding of the operation of the infrastructure.

7.1 Troubleshooting.

This is a pretty complicated setup, and there are a number of places were you could pop off the rails. Fortunately there are logs in various places that will help you sort out what is going wrong.

You can get more information as to why things failed by looking at the log files. The two log files that are of most interest are:

- /var/log/yubiauth.log
- /var/log/freeradius/radius.log

However, don't forget to consult the apache logs as well.

The OpenVPN web administration page has a "*Log Reports*" link that takes you to its logs.

7.1.1 freeradius Testing

The testing of the username, password, and OTP are described above in section 5.2 If there are issues with freeradius, this will not work. When you are working on the "*FreeRADIUS*" page in the YubiADMIN console, the command that is used to access the RADIUS daemon is listed above the text box of the results.

This **`radtest`** command is useful from the command line as well. You can run the **`radtest`** command from within a login on the yubixvmc VM (you dont have to use a legitimate username and password, although you can use one if you wish) to test the radius server outside of the YubiKey infrastructure.

```
yxadmin@yubixvmc:~$ radtest y y localhost 0 testing123
Sending Access-Request of id 231 to 127.0.0.1 port 1812
      User-Name = "y"
      User-Password = "y"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
rad_recv: Access-Reject packet from host 127.0.0.1 port 1812, id=231, length=27
      Reply-Message = "false"
```

Here are several variants you can consider to help troubleshoot the radius communication and function:

- Substitute a different name (e.g. a FQDN, or the VM hostname {yubixvmc in this case}) for localhost. Look to see if you get a name resolution failure (fix DNS resolution), or no response from the server (ensure that the RADIUS daemon is listening to the interface you intend it to).

- Check to see that the secret is correct. If it is not, the response will be something similar to:

```
rad_verify: Received Access-Reject packet from home server 127.0.1.1 port 1812
with invalid signature!  (Shared secret is incorrect.)
```

- Test it with a username and password that you have set in section 5.1 above, that is NOT associated with a YubiKey. This should succeed:

```
yxadmin@yubixvmc:~$ radtest yubislotnone yubipassnone localhost 0 testing123
Sending Access-Request of id 222 to 127.0.0.1 port 1812
      User-Name = "yubislotnone"
      User-Password = "yubipassnone"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=222, length=20
```

- Of course you can test this with an actual legitimate YubiKey OTP (ignore the line wrap in the command):

```
yxadmin@yubixvmc:~$ radtest yubislot2
yubipass2ccccccccciturljdchijvrvebtktuvlfhcfrblnjgilt localhost 0 testing123
Sending Access-Request of id 78 to 127.0.0.1 port 1812
      User-Name = "yubislot2"
      User-Password = "yubipass2ccccccccciturljdchijvrvebtktuvlfhcfrblnjgilt"
      NAS-IP-Address = 127.0.1.1
      NAS-Port = 0
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=78, length=20
```

8.0 Revision History

Version 0.2c 7/9/14

- Initial version for posting on yubikey forum.

8.1 Issues remaining

The following is a short list of issues that may need to be addressed in a future revision

- Hostname. The shorthand hostname, FQDN, localhost, and internal vs external connections to the VM and servers within the VM is very confusing. It will probably need to be straightened up in a future revision.
- The FreeRADIUS server, or it's accessing middleware, is not very stable and has to be restarted occasionally.