2018

RunLabs Documentation

DOCUMENTATION FOR RUNLABS INSTALLATION, USAGE, AND LAB CREATION.

Contents

Introduction

This document provides information for operating RunLabs. This document will discuss installing RunLabs. RunLabs uses various components such as KVM, OpenVSwitch, Minimega, Python, and Flask. Installation will go through installing the required components and configurations. This document will also discuss usage/administration of RunLabs and some of the features. Usage will include use cases as well. Finally, this document will discuss lab creations and some of the issues that could occurred when creating labs.

RunLabs Installation

This section is for installing and configuring RunLabs.

System Requirements

CPU: CPU must support AMD-V feature or Intel VT-x.

Speed of the CPU depends on what RunLabs is being used for.

KVM requires that the CPU supports virtualization.

Memory: 4GB or more.

Memory requirements are dependent on what RunLabs is being used for.

Operating System: Ubuntu 16.04 or above.

Dependencies and RunLabs Installation

Python 2.7 is required.

There are two ways to download RunLabs source code. It can be downloaded as a zip file or cloned using git.

If it's cloned using git then you're required to have git installed. Installing git can be done by running 'sudo apt-get install git'

If zip file is downloaded then it can be extracted by running 'unzip runlabs.zip'

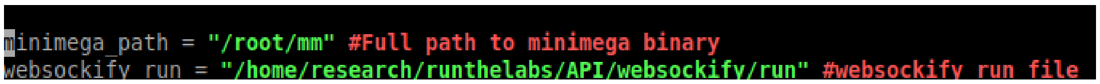
After RunLabs source code is obtained, move into RunLabs folder then the API folder. Installation script is called install.sh and it can be run by running 'sudo bash install.sh'

The installation script will do the following:

1. Update repositories and install dependencies
2. Install Python Flask, required for Web Interface and the API
3. Clone ITLivLab minimega repository and compile minimega
4. Compiled file will be named mm and a Python API file named minimega.py is generated
5. Source code for minimega is removed since it's no longer useful

RunLabs Configuration

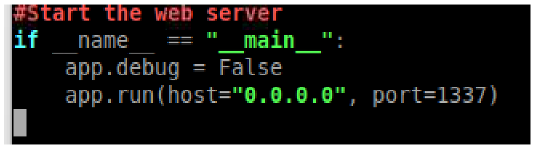
RunLabs needs to be configured. There are two important variables that need to be configured. First variable is the location of minimega binary or mm, which was compiled earlier. Second variable is the websockify run file. Example configuration can be seen below in figure 1.



```
minimega_path = "/root/mm" #Full path to minimega binary
websockify_run = "/home/research/runthelabs/API/websockify/run" #websockify run file
```

Figure 1

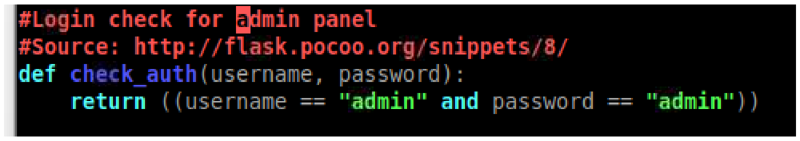
Additionally, api.py file needs to be edited and app.debug variable needs to be set to false, as shown in figure 2 below.



```
#Start the web server
if __name__ == "__main__":
    app.debug = False
    app.run(host="0.0.0.0", port=1337)
```

Figure 2

Finally, in api.py, username and password for RunLabs API or WebUI can be configured. In figure 3 below, the configuration is set to admin/admin.



```
#Login check for admin panel
#Source: http://flask.pocoo.org/snippets/8/
def check_auth(username, password):
    return ((username == "admin" and password == "admin"))
```

Figure 3

RunLabs Features

The feature RunLabs provides are the following:

- An API – for controlling RunLabs via external interface
- A WebUI – For controlling RunLabs via a browser
- NoVNC – Allows for interaction with the VM's without a VNC client
- VNC Password Change – Password to access VM's can be changed
- VM Reboot – In case of an issue, a VM can be easily rebooted
- GRE – For attaching two virtual networks located on two separate physical hosts
- Lab Configuration stays in memory after a lab is ran and turned off
- Support for DHCP – DHCP server can be ran on host side and connected to the virtual environment via network taps
- No persistent disk – No data is saved to the virtual disk
- No internet connection, unless setup with a network tap

RunLabs Usage

This section will discuss the actual usage of RunLabs.

Included Scripts and Their Functions

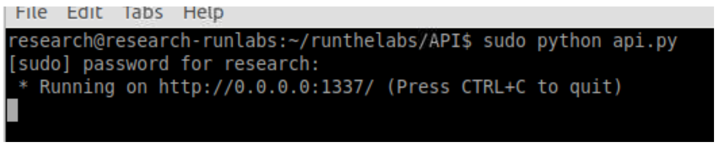
SCRIPT	USAGE
API.PY	The main RunLabs script. Needs to be ran to start RunLabs **EDIT IF NEEDED**
GENERATE_CONFIG.PY	To generate lab configuration JSON file **DO NOT EDIT**
DBCONTROL.PY	Interacts with SQLite database **DO NOT EDIT**
INSTALL.SH	To install RunLabs dependencies **EDIT IF NEEDED**
MMSTART.PY	Interacts with Minimega **DO NOT EDIT**
MMCONTROL.PY	Interacts with Minimega **DO NOT EDIT**
CONFIG.PY	RunLabs configuration file **EDIT**
TEMPLATES/WEBUI.HTML	WebUI for RunLabs **EDIT IF NEEDED**

API_README contains API documentation.

vminfo_db.sqlite3, tokenfile, iptables_old are based on the currently running session. They need to be left alone. Static folder, templates folder, websockify folder, and .pyc files need to be left alone. Templates/webui.html can be edited if needed.

Starting RunLabs

RunLabs can be started by executing api.py, as a root. Figure 4 shows api.py being ran.



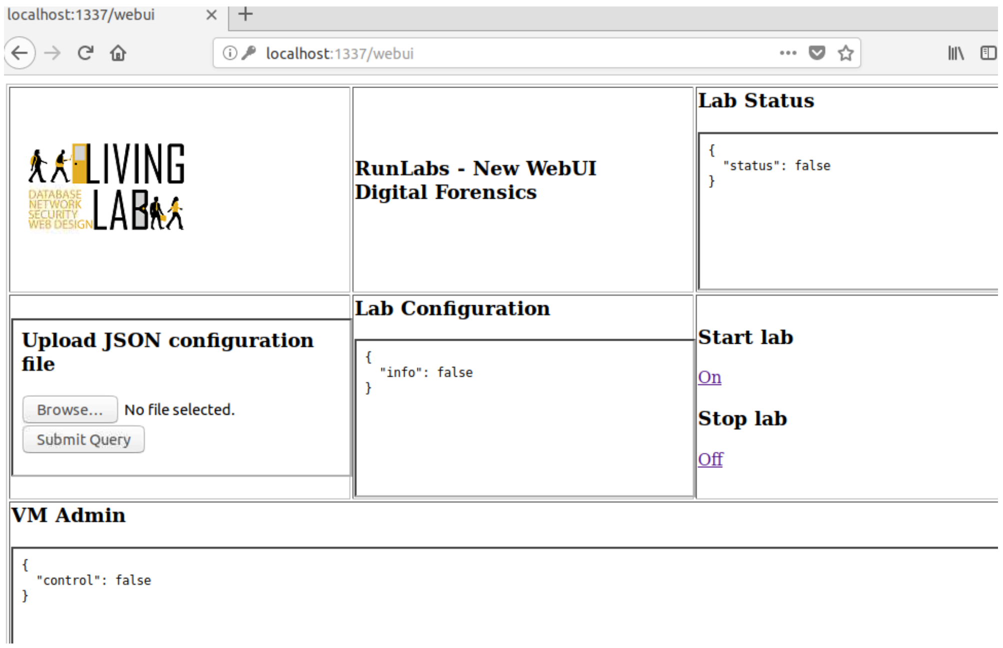
```
research@research-runlabs:~/runthelabs/API$ sudo python api.py
[sudo] password for research:
 * Running on http://0.0.0.0:1337/ (Press CTRL+C to quit)
```

Figure 4

RunLabs Web Interface

Web Interface can be accessed by visiting http://IP_OF_THE_RUNLABS_MACHINE:1337/webui

Visiting this page will prompt you to input your username and password. Screenshot in figure 5 shows what your interface should look like. Expect some of the parts of this page to refresh from time to time.


localhost:1337/webui
LIVING LAB
DATABASE NETWORK SECURITY WEB DESIGN
Browse... No file selected.
Submit Query
On
Off

RunLabs - New WebUI
Digital Forensics

Lab Status

```
{
  "status": false
}
```

Upload JSON configuration file

Lab Configuration

```
{
  "info": false
}
```

Start lab

Stop lab

VM Admin

```
{
  "control": false
}
```

Figure 5

Lab Status displays if any labs are currently running or not.

Lab Configuration displays JSON configuration file, if it has been uploaded.

Start and Stop lab have links that allow you to either start a lab or stop a lab.

VM Admin interface will give you access information for the VM's and allow you to reboot and change password for the VM's. It will also have NoVNC links.

RunLabs JSON File Format

Below is an example JSON file:

```
{
	"comment": "This is an example JSON file",
	"author": "Nobody",
	"title": "My title",
	"vm": {
		"Linux1": {
			"disk": "/storage/linux.iso",
			"ram": "1024",
			"network": "1"
		}
	},
	"gre": {
		"gre0": "10.5.7.1"
	},
	"dhcp": {
		"1": {
			"startip": "192.168.1.10",
			"tapip": "192.168.1.1",
			"cidr": "192.168.1.1/24",
			"endip": "192.168.1.50"
		}
	},
	"internet": {
		"cidr": "192.168.1.1/24",
		"outinterface": "eth0"
	}
}
```

Required fields for JSON files are:

- Comment – anything from lab info to passwords for VM's
- Author – people that made the lab
- Title – title of the lab
- VM
  - VM Name – name of the VM
  - Disk location – ISO or QCOW2 file for the VM, Full path
  - RAM – RAM for the VM
  - Network – Network/VLAN the VM should be on.

Optional Fields

- GRE – if used, tunnel name and IP are required
- DHCP – if used, network number, startip, tapip, cidr, and endip are required
  - Network number above is set to 1, which is the same network the VM is on since that VM needs to be provided an IP address
  - Start IP is the IP address from which DHCP will start assigning IP's. End IP is the last IP that DHCP server will assign

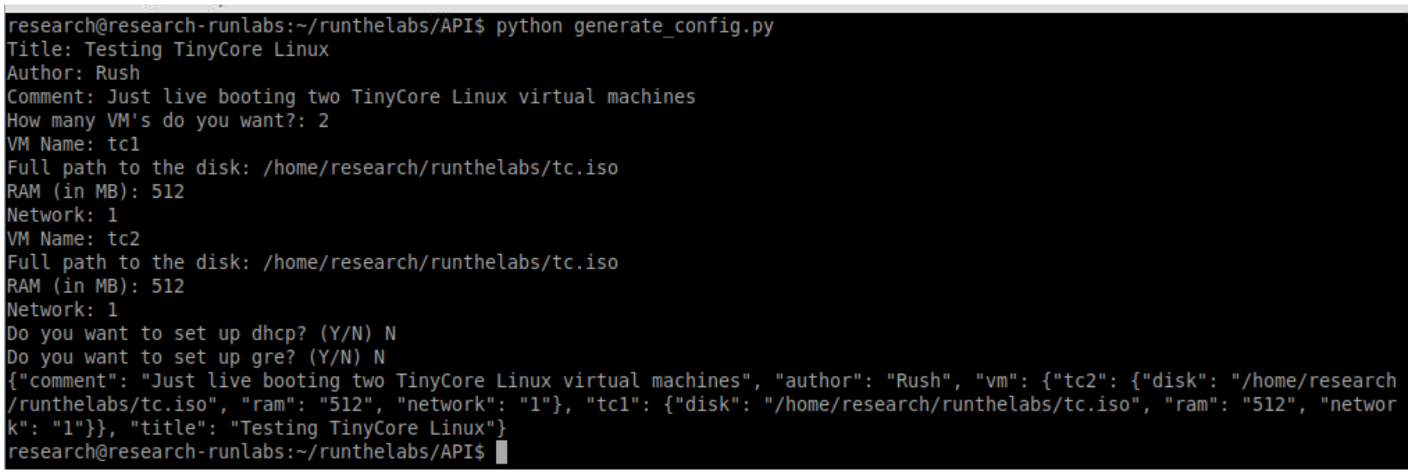
    - CIDR is network range
    - Tap IP is the IP address of the DHCP server
- Internet – If used, cidr and internet connected/outgoing traffic interface is required
    - DHCP MUST BE USED
    - CIDR should be same as DHCP CIDR for the specific network requiring internet/outgoing connection
    - Internet connected/outgoing traffic interface – the network interface you want to route your traffic

Building Your First Lab

Config.py file used in this exercise has been configured correctly. Api.py file also has been configured correctly.

In this lab, two TinyCore Linux virtual machines will be created and they will be put on the same network. These will be live booted based on an ISO file. This network will not be using DHCP or GRE.

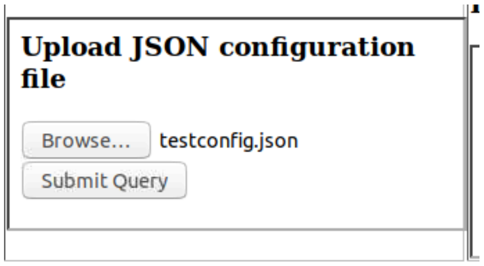
JSON file for this lab can be created by utilizing generate_config.py file. The JSON creation can be seen below in figure 6.



```
research@research-runlabs:~/runthelabs/API$ python generate_config.py
Title: Testing TinyCore Linux
Author: Rush
Comment: Just live booting two TinyCore Linux virtual machines
How many VM's do you want?: 2
VM Name: tc1
Full path to the disk: /home/research/runthelabs/tc.iso
RAM (in MB): 512
Network: 1
VM Name: tc2
Full path to the disk: /home/research/runthelabs/tc.iso
RAM (in MB): 512
Network: 1
Do you want to set up dhcp? (Y/N) N
Do you want to set up gre? (Y/N) N
{"comment": "Just live booting two TinyCore Linux virtual machines", "author": "Rush", "vm": {"tc2": {"disk": "/home/research
/runthelabs/tc.iso", "ram": "512", "network": "1"}, "tc1": {"disk": "/home/research/runthelabs/tc.iso", "ram": "512", "networ
k": "1"}}, "title": "Testing TinyCore Linux"}
research@research-runlabs:~/runthelabs/API$ 
```

Figure 6

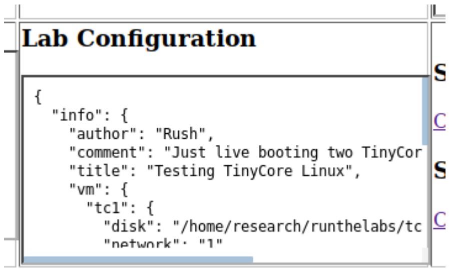
The JSON output can be saved to a text file and uploaded. In figure 7, testconfig.json file is uploaded.



Upload JSON configuration file

Figure 7

If the upload was successful, new tab should have opened with “true” result and lab configuration will be seen just like it’s shown in figure 8.

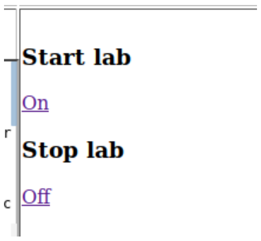


Lab Configuration

```
{
  "info": {
    "author": "Rush",
    "comment": "Just live booting two TinyCor
    "title": "Testing TinyCore Linux",
    "vm": {
      "tc1": {
        "disk": "/home/research/runthelabs/tc
        "network": "1"
```

Figure 8

“On” hyperlink can be clicked to run the configuration which was uploaded. This link with “Off” link are shown in figure 9. When “On” is clicked, it will take a few seconds for the machines to start. If everything worked correctly, you will be returned “true”



Start lab

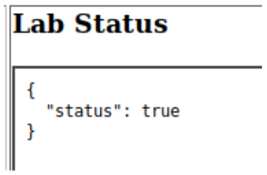
On

Stop lab

Off

Figure 9

When the Lab is running, Lab status will change to “true” as shown in figure 10.

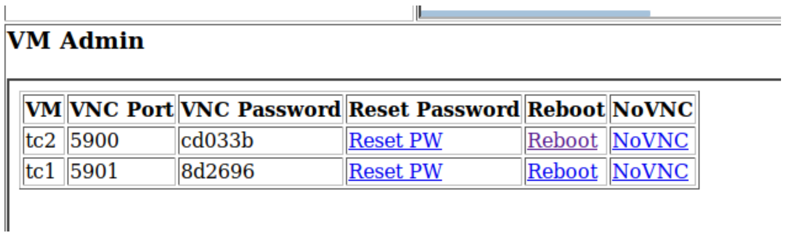


Lab Status

```
{
  "status": true
}
```

Figure 10

VM Admin area will have links for controlling VM’s as shown in figure 11.



VM Admin

VM	VNC Port	VNC Password	Reset Password	Reboot	NoVNC
tc2	5900	cd033b	Reset PW	Reboot	NoVNC
tc1	5901	8d2696	Reset PW	Reboot	NoVNC

Figure 11

NoVNC link can be clicked or shared to allow access to the VM. Additionally, a VNC client can be used with Port and Password information to access the VM. NoVNC screenshot can be seen in figure 12.

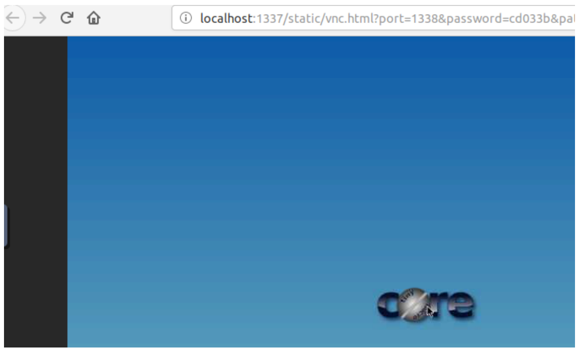


Figure 12

After Lab is complete or done being utilized, Off button displayed in figure 9 can be clicked to turn off all of the VM's.

RunLabs Use Cases

RunLabs can be used for almost any lab that uses virtual machines.

Since RunLabs does not enable persistence for virtual disks and does not connect to the internet, it's great for malware analysis or just playing with malware.

RunLabs can also be used for testing and researching offensive security related things such as hacking or network scanning.

RunLabs is also a great for CTF's, including CTF's that have offensive and defensive sides.

API

RunLabs includes an API, which allows the user to write their own application or UI to interact with RunLabs.

API returns messages in JSON format. Basic authentication is required to interact with the API.

```
Request_Type URL Returned_Data
POST /upload upload:true/false
GET /info json_file //uploaded json file
GET /on on:true/false
GET /off off:true/false
GET /status status:on/off
GET /passwords {passwords: {vmname:name,port:port,password:pw}...}
GET /changepassword/vmname change:true/false
GET /reboot/vmname reboot:true/false
```

VM Creation

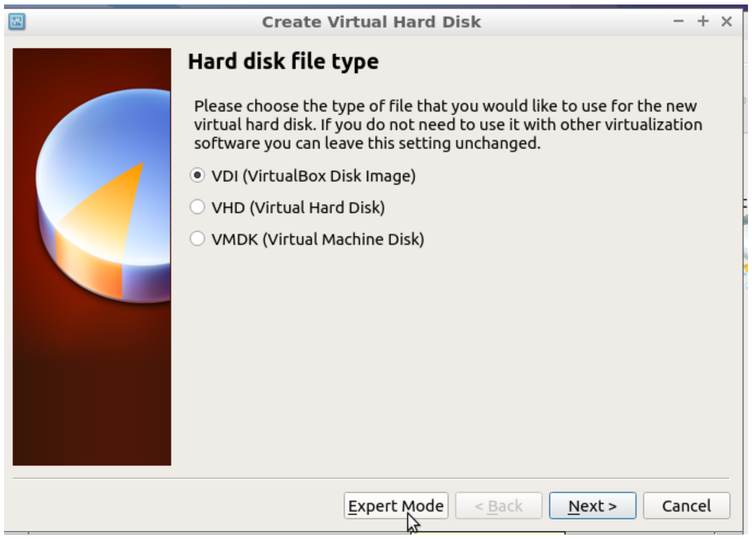
This section discusses VM creation for RunLabs. RunLabs can use the following disk formats: qcow2, raw, or iso. There are mainly two ways to create VM's for RunLabs. VirtualBox can be used to create a VM, if disk type is set to qcow2. Minimega can also be used to create VM's. Minimega utilizes KVM as for virtualization.

Lab VM Creation

Using VirtualBox

This is an example of data carving lab.

Start VM creation process as you normally would. When you get to Hard Disk creation section, click on Expert mode as shown in figure 13.



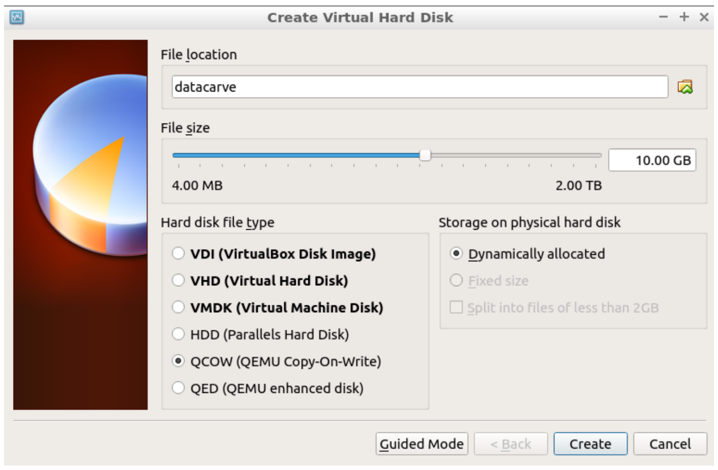
Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- [✓ VDI (VirtualBox Disk Image)]
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)

Figure 13

Select QCOW option as shown in figure 14.



File location: datacarve

File size: 10.00 GB (4.00 MB – 2.00 TB)

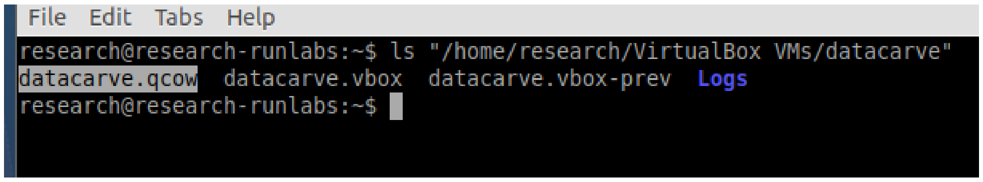
Hard disk file type
- VDI (VirtualBox Disk Image)
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- [✓ QCOW (QEMU Copy-On-Write)]
- QED (QEMU enhanced disk)

Storage on physical hard disk
- [✓ Dynamically allocated]
- Fixed size
- Split into files of less than 2GB

Figure 14

After you’re done creating the VM, start the VM and install whatever OS and software that is required for the RunLabs exercise.

Shutdown the VirtualBox VM then go into the VM disk directory. You will be using datacarve.qcow as your RunLabs lab disk.



```
research@research-runlabs:~$ ls "/home/research/VirtualBox VMs/datacarve"
datacarve.qcow  datacarve.vbox  datacarve.vbox-prev  Logs
research@research-runlabs:~$
```

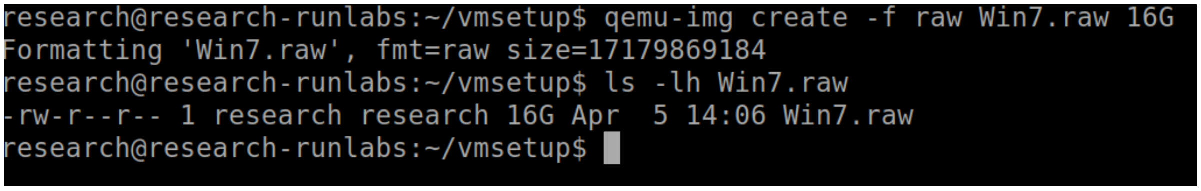
Using Minimega

To create a Virtual Machine utilizing Minimega, there are prerequisites. Qcow or Raw hard drive image is required. This can be created using qemu-utils. Another requirement is the installation source, such as an ISO or a setup image. Minimega, by default, uses snapshot mode (snapshot setting is enabled) which means that qcow or raw harddrive will not be modified and saved. When creating a VM, snapshot mode needs to be disabled. The process involves the following steps:

1. Create a virtual harddrive
2. Start minimega
3. Disable snapshot mode in Minimega
4. Create a VM with the harddrive and the ISO attached
5. Enable networking/internet connection if needed
6. Install the OS and add additional tools as needed
7. Shutdown the VM properly

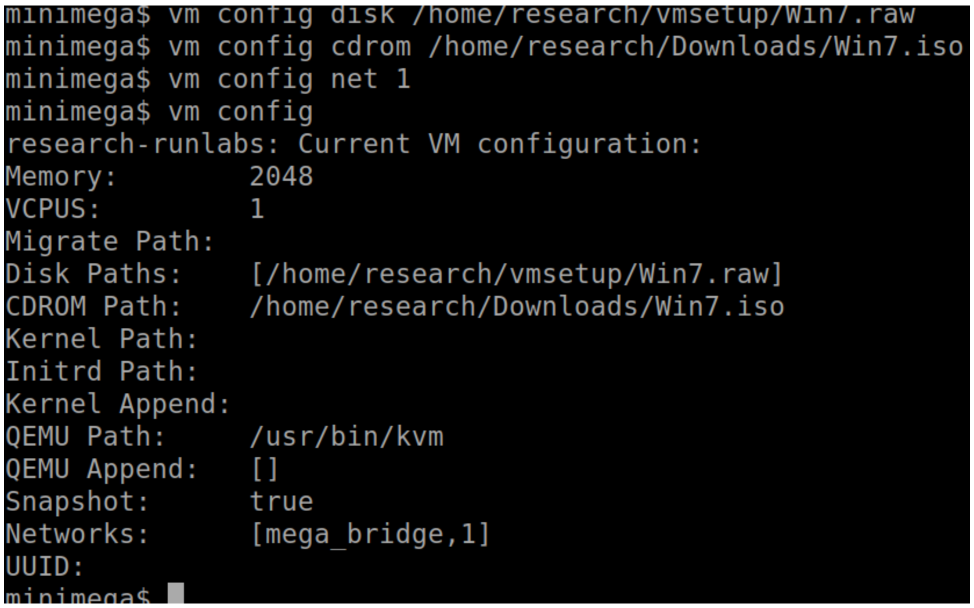
Harddrive can be created by running the following command:

qemu-img create -f HARD_DRIVE_FORMAT FILENAME SIZE



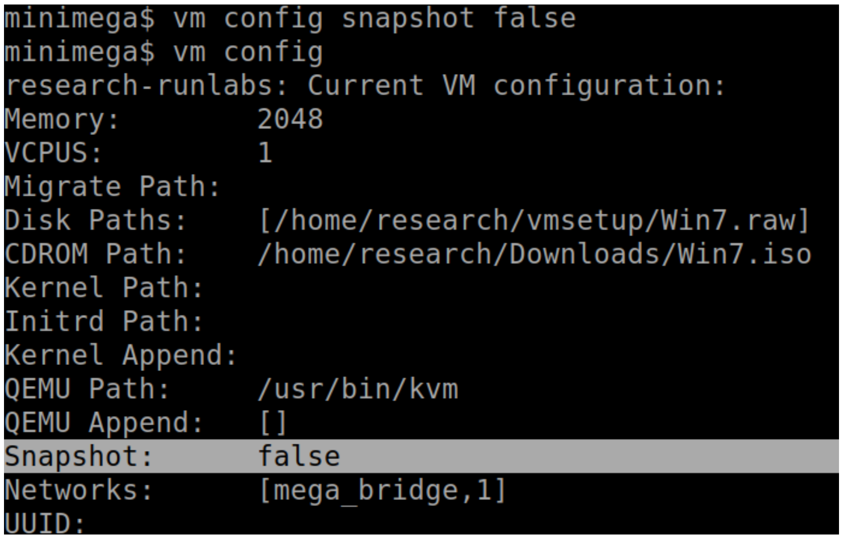
```
research@research-runlabs:~/vmsetup$ qemu-img create -f raw Win7.raw 16G
Formatting 'Win7.raw', fmt=raw size=17179869184
research@research-runlabs:~/vmsetup$ ls -lh Win7.raw
-rw-r--r-- 1 research research 16G Apr  5 14:06 Win7.raw
research@research-runlabs:~/vmsetup$
```

Start minimega and configure your virtual machine like shown below. Disk will be the harddrive, cdrom will be the ISO, and network needs to be set to 1. More virtual network cards can be added as well. Check the minimega documentation.



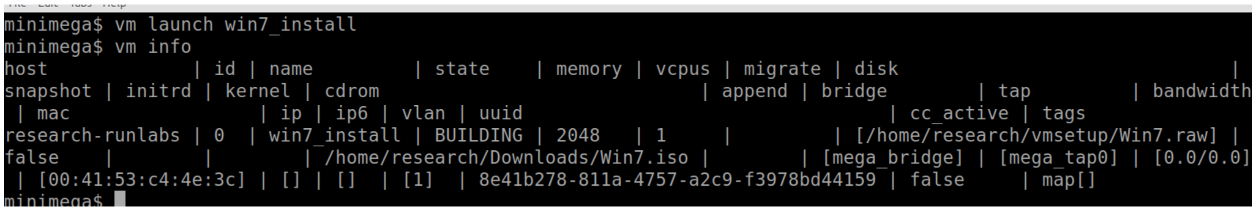
```
minimega$ vm config disk /home/research/vmsetup/Win7.raw
minimega$ vm config cdrom /home/research/Downloads/Win7.iso
minimega$ vm config net 1
minimega$ vm config
research-runlabs: Current VM configuration:
Memory:         2048
VCPUS:          1
Migrate Path:
Disk Paths:     [/home/research/vmsetup/Win7.raw]
CDROM Path:     /home/research/Downloads/Win7.iso
Kernel Path:
Initrd Path:
Kernel Append:
QEMU Path:      /usr/bin/kvm
QEMU Append:    []
Snapshot:       true
Networks:       [mega_bridge,1]
UUID:
minimega$
```

Disabling snapshot mode can be done by the command shown below:



```
minimega$ vm config snapshot false
minimega$ vm config
research-runlabs: Current VM configuration:
Memory:         2048
VCPUS:          1
Migrate Path:
Disk Paths:     [/home/research/vmsetup/Win7.raw]
CDROM Path:     /home/research/Downloads/Win7.iso
Kernel Path:
Initrd Path:
Kernel Append:
QEMU Path:      /usr/bin/kvm
QEMU Append:    []
Snapshot:       false
Networks:       [mega_bridge,1]
UUID:
```

Vm launch command in minimega can be used to start the VM, however, it does not boot up the VM yet. Below is the example of that command. Win7_install is the name of the VM.



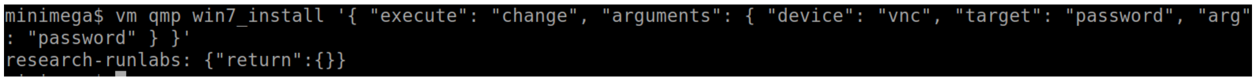
```
minimega$ vm launch win7_install
minimega$ vm info
host             | id | name         | state    | memory | vcpus | migrate | disk                              |
snapshot | initrd | kernel | cdrom                             | append | bridge        | tap           | bandwidth
 | mac                  | ip | ip6 | vlan | uuid                                 | cc_active | tags
research-runlabs | 0  | win7_install | BUILDING | 2048   | 1     |         | [/home/research/vmsetup/Win7.raw] |
false    |        |        | /home/research/Downloads/Win7.iso |        | [mega_bridge] | [mega_tap0] | [0.0/0.0]
 | [00:41:53:c4:4e:3c] | [] | []  | [1]  | 8e41b278-811a-4757-a2c9-f3978bd44159 | false     | map[]
minimega$
```

To access the VM, we will have to utilize VNC. VM’s created in minimega will require VNC password.

Below is the command which can be used for modifying the password for the VM. Arg variable is the password.

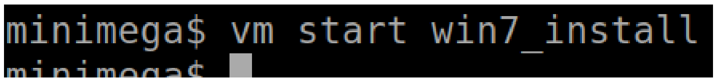
```
vm qmp win7_install '{ "execute": "change", "arguments": { "device": "vnc",
"target": "password", "arg": "password" } }'
```

It should return without an error.



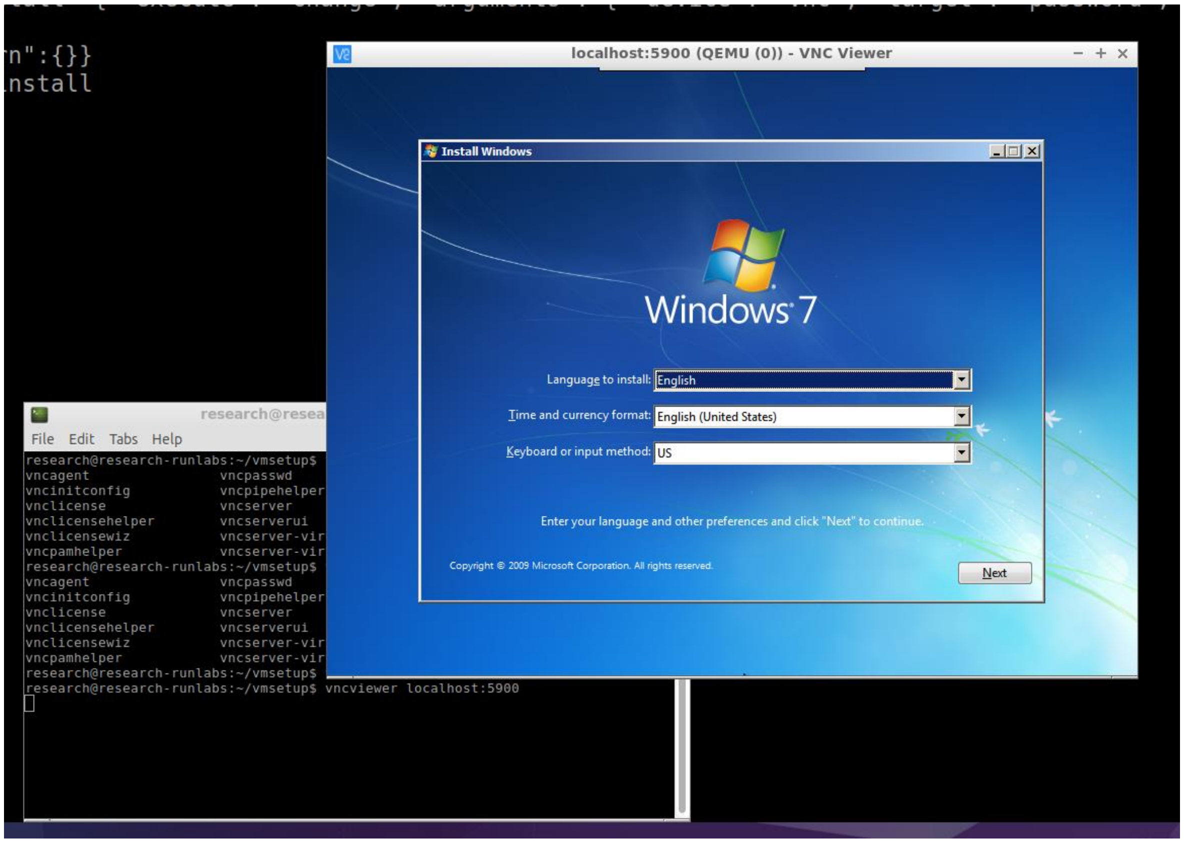
```
minimega$ vm qmp win7_install '{ "execute": "change", "arguments": { "device": "vnc", "target": "password", "arg"
: "password" } }'
research-runlabs: {"return":{}}
```

Vm start will boot up the VM.



```
minimega$ vm start win7_install
minimega$
```

Via VNC, you should be able to access your virtual machine.



```
n":{}}
nstall
```

```
research@research-runlabs:~/vmsetup$
vncagent                vncpasswd
vncinitconfig           vncpipehelper
vnclicense              vncserver
vnclicensehelper        vncserverui
vnclicensewiz           vncserver-vir
vncpamhelper            vncserver-vir
research@research-runlabs:~/vmsetup$
vncagent                vncpasswd
vncinitconfig           vncpipehelper
vnclicense              vncserver
vnclicensehelper        vncserverui
vnclicensewiz           vncserver-vir
vncpamhelper            vncserver-vir
research@research-runlabs:~/vmsetup$
research@research-runlabs:~/vmsetup$ vncviewer localhost:5900
```

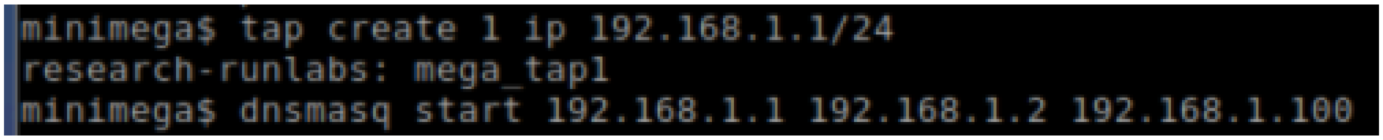
Language to install: English
Time and currency format: English (United States)
Keyboard or input method: US

Enter your language and other preferences and click "Next" to continue.

Copyright © 2009 Microsoft Corporation. All rights reserved.

Adding network capability

Add a network tap and start dnsmasq like shown below:



```
minimega$ tap create 1 ip 192.168.1.1/24
research-runlabs: mega_tap1
minimega$ dnsmasq start 192.168.1.1 192.168.1.2 192.168.1.100
```

Check to see if your virtual machine has received an IP address from the DHCP server.

Now it's time to connect the virtual machine to the internet. The following commands can make that possible: (note: enp3s0 is my outgoing interface)

```
echo 1 | tee -a /proc/sys/net/ipv4/ip_forward
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A POSTROUTING -o enp3s0 -s 192.168.1.1/24 -j MASQUERADE
iptables -P FORWARD DROP
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -s 192.168.1.1/24 -j ACCEPT
iptables -A FORWARD -s 192.168.1.1/24 -d 192.168.1.1/24 -j ACCEPT
```

And now your VM should have internet connectivity.

Mounting and modifying QCOW files

Mounting and modifying QCOW files can be required if a file needs to added or removed or just modified. This could be used for adding challenge files QCOW files or adding malicious files to QCOW files.

Install qemu-utils by running: 'sudo apt-get install qemu-utils'

Run the following commands as root as well.

Run 'modprobe nbd' to load nbd module.

To mount a qcow file, the following command is used:

'qemu-nbd --connect=/dev/nbd0 datacarve.qcow'

Command above mounts qcow image to nbd0, which can be treated similarly to sdb. To access partition 1 on sdb, sdb1 is mounted. With nbd0, to access partition 1, nbd0p1 would be mounted.

To mount a partition the following command can be used:

'mount /dev/nbd0p1 /mnt/mypartition'

The command above mounts partition 1 to /mnt/mypartition. Any modifications that need to be made to the file system can be made.

To unmount the file system the following command can be ran:

'umount /mnt/mypartition'

To disconnect qcow from nbd, the following command can be ran:

'qemu-nbd --disconnect /dev/nbd0'

Possible VM Issues

Network Adapter MAC address or type being change may cause issues. For example, the VM OS may add a new interface. VM MAC address can be set by setting network info in JSON to networknumber,MAC address for example like this: 1,AA:BB:CC:00:11:22

CPU cores and type can also cause an issue for certain operating systems. For example, Windows based operating systems may require reboot if CPU type has changed from AMD to Intel or if an additional CPU core is added. Since the disk is not persistent, rebooting the VM in RunLabs will not help. This problem has to be avoided and it can be done by making sure that CPU type and core count remains the same during VM creation and RunLabs usage.

Security Issues

RunLabs

RunLabs does not utilize HTTPS. HTTPS was disabled to avoid issues with NoVNC and Websockets. HTTPS can be enabled by editing api.py and NoVNC can be served on a different port or websockify can be used with SSL. This setting depends on what your threats are.

Command and SQL injections are also a concern. Due to the way RunLabs is designed to work, these issues cannot be fully avoided. Currently, this threat is accepted. If the person administrating RunLabs has access to execute commands and does not gain anything from doing SQL injection, efforts were not made to fix those specific issues.

JSON configuration files are not validated until it’s time to run the lab. If a bad JSON file is uploaded, there will be no way of knowing that until an error occurs when turning on the labs. In case of an error, RunLabs can be quickly reset to allow another upload with a fixed JSON file.

VM

Vulnerabilities in OpenVSwitch or KVM/QEMU may lead to VM escape, network escape, or information leakage. These risks have to be accepted but the impact can be minimized by placing RunLabs on the correct network and applying appropriate measures to secure assets that need securing.

Enabling DHCP and allowing traffic to the Internet

Enabling DHCP feature creates a host to internal network tap and host works as a DHCP server. This creates an issue by allowing VM to touch any service running on host that is bound to 0.0.0.0. Firewall rules and other configuration should be put into place to avoid VM’s from having access to certain services.

Allowing outgoing traffic to the internet will also allow the VM to access anything on the network that host can access. Proper firewall rules should be put into place to avoid VM’s from having access to certain hosts on the network. With internet configuration, multiple CIDRs and Outgoing interface could be used, however, it has not been tested.