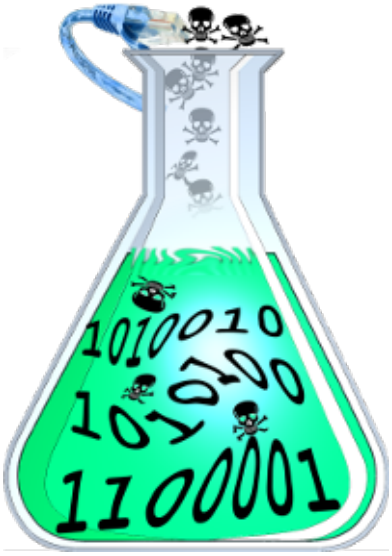
Labtainers Student Guide

Fully provisioned cybersecurity labs

January 21, 2022



1 Introduction

This manual is intended for use by students performing lab exercises with Labtainers. Labtainers provide a fully provisioned execution environment for performing cybersecurity laboratory exercises, including network topologies that include several different interconnected computers.

Labtainers assume you have a Linux system, e.g., a virtual machine appliance described below. If you are accessing a pre-defined Labtainers VM via a web browser, you can skip to section 2.

1.1 Obtaining and installing Labtainers

Labtainers requires an x86-based computer. It will not run on ARM-based processors such as Mac M1-based Powerbooks. If you have an M1 Mac, or otherwise cannot install a virtual machine, you can still access and run Labtainers exercises as described in section 5.

The easiest way to obtain Labtainers is to download one of the pre-configured virtual machines from `https://nps.edu/web/c3o/virtual-machine-images`, and import it into either VirtualBox or VMWare. Follow the brief instructions on that download page. When you first boot the resulting VM, Labtainers will take a moment to update itself. You are then provided a terminal that includes some hints, and can be used to run Labtainers. A video tutorial on installing Labtainers is at `https://nps.edu/web/c3o/labtainers-tutorials`.

Note that the VM's Ubuntu Linux distribution is configured to NOT automatically perform system updates. It may prompt you to download and install updates. That is typically not necessary and may tie up your network bandwidth. Yes, we are suggesting you not update your Linux VM unless and until you have the time and the bandwidth.

You may now skip to section 2.

1.2 Alternatives to the Labtainers VM Appliance

Skip this section and go to section 2 if you are using a Labtainers VM appliance or accessing Labtainers remotely via a browser.

Please note that Docker runs as a privileged service on your computer, and Labtainers containers run as privileged containers. If you have sensitive data on your computer, you should understand the isolation provided by Dockers on your system. An alternative is to use one of our virtual machine appliances rather than running Docker directly on your computer.

1.2.1 Installing Labtainers on an existing Linux system

The Labtainer framework is distributed as a tarball from: `https://nps.edu/web/c3o/labtainers` Click the link named: "Download the Labtainer framework", and untar the resulting file into a permanent directory on your Linux system, e.g., into `$HOME`. For example, if you downloaded the file from a browser on your Linux system:

```
cd
tar -xf ~/Downloads/labtainer.tar
```

From the directory into which you untarred the tarball start the installer script:

```
cd labtainer
./install-labtainer.sh
```

This script will install the latest version of Docker and packages required by the Labtainer framework. It will cause your Linux host to reboot when it completes.

After the Linux host reboots, open a terminal to your Linux host and change directory to wherever you untarred the tarball, e.g., your HOME directory.

1.2.2 Browser-based access to Labtainers

Labtainers can be run on servers, e.g., VMs on the a cloud service, and accessed via your browser. See section 5 for information on using cloud services. Alternately your school might have virtual desktop solution such as VMWare Horizons that can host the Labtainers VM appliance.

2 Selecting a Lab

All labs are run from the same Labtainer workspace directory, which is typically at:

```
cd $LABTAINER_DIR/scripts/labtainer-student
```

The prepackaged virtual machines automatically start a terminal in this directory.

To see a list of available labs, run the `labtainer` command with no arguments:

```
labtainer
```

Use the `-k` option to see a list of searchable keywords, and the `-f <keyword>` option to view a summary of labs having that keyword.

Lab exercises are also organized into *Labpacks* that are a collection multiple related labs that you may wish to perform in sequence (e.g., based on direction from your instructor.) Use the

```
labpack
```

to view a list of Lab Packs, and provide the name of a Labpack as an argument to see a list of the labs within a Labpack. That command output also includes an indication (`[Y] or [N]`) of whether you've generated any results from each lab. Your instructor may provide you with custom Labpacks in the form of a URL. You may add those to your system by using the

```
labpack -a <url>
```

Your instructor may direct you to add new or custom lab exercises to your installation by providing you with a URL of an *IModule*. To get access to those labs, use:

```
imodule <url>
```

Additional lab exercises created by other instructors are available as IModules, whose URLs are listed at `https://nps.edu/web/c3o/imodules`.

3 Performing a Lab

To run a specific lab, include the name of the lab in the `labtainer` command:

```
labtainer <labname>
```

where *labname* is the name of the lab to run. The first time any given lab is run, a set of files are downloaded, and that progress is reported on the screen. The size of the downloads varies between labs.

Most labs direct you to a PDF version of a lab manual, can be viewed by right clicking on the displayed path, or you can open the file in a browser. Please note that some of the initial lab instructions repeat the steps you've already taken, and you need not perform those again.

A list of Labtainer commands can be found in Appendix A of this document. A video tutorial on performing Labtainer labs is at `https://nps.edu/web/c3o/labtainers-tutorials`.

Once you start the lab, you will typically see one or more virtual terminals connected to computers within the lab. While running the lab, if you require more virtual terminals, use:

```
moreterm.py <labname> <container>
```

where *container* is the host name of the component on which to attach a terminal. It can be omitted for labs having a single component. See Appendix B for information on customizing terminal window colors and text.

The virtual terminals for most labs present bash shells via which you can interact with the attached computer, (which is actually a Docker container designed to appear like a separate computer). A single computer may have multiple virtual terminals attached to it. Each computer is independent, and may use networks to interact with other Labtainer computers within the lab.

Many labs automatically gather results of your work, which you will provide to your instructor. Note that, unless otherwise directed, exploration and experimentation you perform either before or after performing the expected activity will not diminish or dilute your results. And you typically do not have to take actions to collect or record your results. This occurs automatically as noted in the next section.

3.1 Interrupting and Completing Labs

When you want to stop working for a while or are finished and ready to turn it in to your instructor, type:

```
stoplab
```

from the Linux system from which you issued the `labtainer` command. All changes to the files, etc. will be preserved and you will be able to resume the lab just the way you started it. You can resume your work, as needed.

The `stoplab` command always displays the directory containing a file with a `.lab` extension that should be provided to your instructor. It shows the current results of your work.

The easiest way to forward the complete `.lab` file to the instructor is to start a browser, e.g., Firefox, on the VM from which you are running Labtainers. Then use the browser to either email the file, or upload it into an LMS system, e.g., Sakai. Alternately, you can configure the VM to use a shared folder, and use that to copy the `.lab` file to the host computer.

3.2 Redoing a Lab

Sometimes you might want to redo the lab from the beginning. In this case, type:

```
labtainer -r  <labname>
```

This will delete any previous containers associated with this lab and start it fresh. **Warning**: this will cause all previous data from the named lab to be lost.

3.3 Checking your work

Some labs include criteria by which to automatically assess your progress. Where enabled and supported, this feature can be utilized by issuing the `checkwork` command from Linux system. That command can be run while the lab is still running. If the lab has been stopped, you must provide the lab name to the checkwork command, e.g.,

```
checkwork telnetlab
```

The meaning and value of the `checkwork` output varies by lab. The command output includes a description of what is being measured, which in some cases may be quite mundane such as the quantity of times you tried a particular command. Please note that the `checkwork` output is not a "score" or a grade.

3.4 Submitting your work

When you've completed a lab and run the `stoplab` command, your results are stored in a file with a `.lab` extension in the directory at:

```
$HOME/labtainer_xfer/<lab name>
```

That file should be provided to your instructor. There are several ways to transfer the file.

1. Use the browser on the VM to email the file to your instructor.
2. Use the browser on the VM to access your school's LMS system such as Saki or Blackboard, and upload the file.
3. Configure the VM to enable *drag and drop*, then move the file to your host computer to email or upload to an LMS.
4. Configure the VM and host to share folders and copy the `.lab` file to the shared folder to email or upload to an LMS.

3.5 Getting Help and Things to Avoid

To get help, type:

```
labtainer -h
```

from the Linux system from which you issued the `labtainer` command. A list of useful labtainer commands will be displayed. Also see our support page at `nps.edu/web/c3o/support1`

Do not run multiple labs simultaneously. Consistent results cannot be guaranteed when more than one lab runs at the same time.

4 Other Considerations

4.1 Networking

In addition to network properties defined for the lab, each component `/etc/host` file includes a "my_host entry" that names the Linux host, e.g., the VM. Most containers will include a default gateway that leads to the Linux host. This allows students to scp files to/from the container and Linux host. It also allows the student to reach external networks, e.g., to fetch additional packages in support of student exploration.

In some instances, the lab requires one or more components to a have different default route. Typically, these components will include a *togglegw.sh* script that the student can use to toggle the default gateway between one that leads to the host, and one defined for the lab. This allows students to add packages on components having lab-specific default gateways. Use of the *togglegw.sh* script is not necessary to reach the Linux host, (e.g., to scp files).

4.2 Installing and Using Labtainers Behind a Web Proxy

If you are not behind a web proxy, ignore this section (most school environments are not behind proxies). If you are behind a web proxy, Labtainer installation requires that you have configured your Linux package management configuration to reflect the proxy, e.g., the /etc/atp/apt.conf or /etc/dnf.conf files.

Additionally, you will need to configure your Docker service as described at: `https://docs.docker.com/engine/admin/systemd/#httphttps-proxy` And set the HTTP_PROXY environment variable to your proxy, e.g.,

```
HTTP_PROXY=http://myproxy:3128
```

If you wish to use apt-get from within a container to add new software to a container, you must first modify the container's /etc/apt/apt.conf file to reflect your proxy.

4.3 Limitations

The Labtainer "computers" are individual Docker containers that are interconnected via virtual networks. These containers each share the Linux kernel of your host. Thus, a change to the kernel configuration on one computer, (e.g., enabling ASLR), will be visible on other containers, as well as your host.

It is suggested that the student's Linux host be a virtual machine that is not used for purposes requiring trust. Software programs contained in cybersecurity lab exercises are not, in general, trusted. And while Docker containers provide namespace isolation between the containers and the Linux host, the containers run as privileged. Labtainers run as Docker containers and use the Docker group which is root-equivalent. In other words, even though you start a Docker container as a non-privileged user, software in the resulting container can modify the Linux host, e.g., the VM.

The computers each include a `.local/` directory beneath the HOME directory. This is used by the Labtainer framework and includes results that get packaged up for forwarding to the instructor. Do not modify any files beneath the .local directory. Otherwise, you can treat those containers as Linux systems, and explore them.

Pasting multiple commands into a labtainer terminal may result in the not all of the commands being executed.

4.3.1 Network Limitations

Labtainer containers do not include typical OS network configuration files such as /etc/network/interfaces or /etc/netplan. Nor do the containers include networking daemons such as `networkd`. The initial post-boot network interface configurations are managed by Docker as prescribed by the labs design. Users may alter network configurations, e.g., via the `ip` command, and may control DNS naming by directly modifying the `/etc/resolv.conf` file. Persistent changes to the `resolv.conf` DNS naming can be achieved using `/etc/rc.local`.

5 Cloud Labtainers

Labtainers can be run on cloud services and accessed via a browser. Cloud service providers may offer free accounts for students or others looking to learn about their cloud services. Currently, Labtainers works with the Azure and Google cloud platforms as described below.

5.1 Azure Cloud

These instructions assume the user has an Azure account, e.g., `https://azure.microsoft.com/en-us/free/students/`

This approach requires that the Azure CLI be installed on the Mac, Windows or Linux: `https://docs.microsoft.com/en-us/cli/azure/install-azure-cli`

In the following command examples, use the "ps1" file extension instead of "sh" when using PowerShell.

- Open a terminal on Mac/Linux, or a PowerShell window on Windows.

- Install the local scripts by getting this script (make it executable on Mac or Linux): `https://raw.githubusercontent.com/mfthomps/Labtainers/master/azure/install_labtainers.sh` Or on Windows: `https://raw.githubusercontent.com/mfthomps/Labtainers/master/azure/install_labtainers.ps1`

  On Mac or Linux:

  - `curl -L https://raw.githubusercontent.com/mfthomps/Labtainers/master/azure/install_labtainers.sh --output install_labtainers.sh`
  - `chmod a+x install_labtainers.sh`

  On Windows:

  - `wget https://raw.githubusercontent.com/mfthomps/Labtainers/master/azure/install_labtainers.sh -OutFile install_labtainers.ps1`

- Then run it (Mac/Linux).

  ```
  ./install_labtainers.sh
  ```

  Windows:

  ```
  ./install_labtainers.ps1
  ```

  That will create a `$HOME/labtainers_azure` directory.

- Change to the `$HOME/labtainers_azure directory`

  ```
  cd $HOME/labtainers_azure
  ```

- Log into your Azure account:

  ```
  az login
  ```

  NOTE: If your account has access to more than one Azure Subscription, you need to change these parameters to specify the student subscription before running the install_labtainers script:

  1. Change the /.azure/clouds.config to show your student subscription number
  2. Change the entries in /.azureProfile.json so that only your student subscription shows "isDefault"= true, the rest being set to "false".

- Once logged into Azure, run the create_vm.sh (or create_vm.ps1 for windows) script, passing in a user ID. The ID can be any name, e.g.,

  ```
  ./create_vm.sh myname
  ```

  The create_vm script may take a while to run. The process is complete when you see *Labtainers is up.* Point a local browser to `localhost:6901` and perform the labs. When prompted for a password in the browser, just click submit or OK, i.e., leave the password blank. The password for the *labtainer user* in the VM is "labtainer".

  Select and perform the lab as described in section 2. Then refer to the items below.

- When done with labs, run the get_results.sh (or get_results.ps1) script:

  ```
  ./get_results.sh <user ID>
  ```

  This will store your Labtainer results in `$HOME/labtainer_xfer`. Provide those results to your instructor.

- If you become unable to reach the Labtainers via your browser, e.g., after shutting down your computer, use the restart.sh script:

  ```
  ./restart.sh <user ID>
  ```

- The create_vm.sh script will create an SSH key pair named id_labtainers within your `$HOME/.ssh` directory. The private key in id_labtainers is not passphrase protected, so you must protect it. You may move the keys to a different computer and access your Labtainers from that computer's browser. You must first run the install_labtainers.sh script on that computer, and then run the restart.sh script.

- When done with a lab, use

  ```
  ./deallocate_vm <user ID>
  ```

  to stop incurring most charges. Note however that any work you've performed on the Labtainers might be lost (unless you've retrieved your results with get_results.sh), depending on how long the VM is dormant.

- To restore a VM after you deallocated it, use:

  ```
  ./restore_vm.sh <user ID>
  ```

- When completely done with the VM, use the delete_vm.sh script to stop incurring all charges:

  ```
  ./delete_vm.sh <user ID>
  ```

- Shutting down the VM without deallocating or deleting it will not stop charges.

5.2 Google Cloud Platform

These instructions assume you have a google cloud account. `https://cloud.google.com/`

This requires that the Google Cloud SDK be installed on the Mac, Windows or Linux: `https://cloud.google.com/sdk/docs/quickstart`

On Linux/Mac, add the `google-cloud-sdk/bin` directory to your PATH environment variable. For example, if you put the SDK in your home directory, then add this to your `$HOME/.bash_profile`

```
PATH=$PATH:$HOME/google-cloud-sdk/bin
```

and then run

```
source $HOME/.bash_profile
```

On Windows, just reopen a new PowerShell window after installing the SDK.

In the following command examples, use the "ps1" file extension instead of "sh" when using PowerShell.

- Open a terminal on Mac/Linux, or a PowerShell window on Windows.
- Install the local scripts by getting this script (make it executable on Mac or Linux): `https://raw.githubusercontent.com/mfthomps/Labtainers/master/google/install_labtainers.sh` Or on Windows: `https://raw.githubusercontent.com/mfthomps/Labtainers/master/google/install_labtainers.ps1`

  On Mac or Linux:

  - `curl -L https://raw.githubusercontent.com/mfthomps/Labtainers/master/google/install_labtainers.sh --output install_labtainers.sh`
  - `chmod a+x install_labtainers.sh`

  On Windows:

  - `wget https://raw.githubusercontent.com/mfthomps/Labtainers/master/google/install_labtainers.sh -OutFile install_labtainers.ps1`
- Then run it (Mac/Linux).

  ```
  ./install_labtainers.sh
  ```

  Windows:

  ```
  ./install_labtainers.ps1
  ```

  That will create a `$HOME/labtainers_google` directory.
- Change to the `$HOME/labtainers_google directory`

  ```
  cd $HOME/labtainers_google
  ```
- Log into your Google cloud account from the command line:

  ```
  gcloud auth login
  ```
- Define your default region and zone by editing and running the set_defaults.sh script. And then initialize using:

  ```
  gcloud init
  ```
- Once logged into the Google Cloud with default region/zone defined, run the create_vm.sh (or create_vm.ps1 for windows) script, passing in a user ID. The ID can be any name, e.g.,

  ```
  ./create_vm.sh myname
  ```
- On Linux/Mac, you will be prompted for an ssh passphase, leave it blank. On Windows, ignore the warnings about ssh keys.

- The create_vm script may take a while to run. The process is complete when you see Labtainers is up. Point a local browser to http://localhost:6901 and perform the labs. When prompted for a password in the browser, just click submit or OK, i.e., leave the password blank. The password for the labtainer user in the VM is labtainer.

- When done with labs, run the get_results.sh (or get_results.ps1) script:

  ```
  ./get_results.sh <user ID>
  ```

  This will store your Labtainer results in /labtainer_xfer. Provide those results to your instructor.

- If you become unable to reach the Labtainers via your browser, e.g., after shutting down your computer, simple use the restart.sh script:

  ```
  ./restart.sh <user ID>
  ```

- The create_vm.sh script will create an SSH key pair named id_labtainers within your /.ssh directory. The private key in id_labtainers is not passphrase protected, so you must protect it. You may move the keys to a different computer and access your Labtainers from that computer's browser. You must first run the install_labtainers.sh script on that computer, and then run the restart.sh script.

- When done with a lab, use

  ```
  ./stop_vm.sh <user ID>
  ```

  to stop incurring processing charges. Note you may still incur storage charges until the VM is delete.

- To restore a VM after you stopped it, use:

  ```
  ./start_vm.sh <user ID>
  ```

- When completely done with the VM, use the delete_vm.sh script to stop incurring all charges:

  ```
  ./delete_vm.sh <user ID>
  ```

- Shutting down the VM without deleting it will not stop all charges, but will stop processing charges. See the Google Cloud dashboard and pricing for more information.

Appendices

Appendix A Labtainer Command Summary

The following labtainer commands are available from the `labtainer-student` directory. Most of these commands include a `-h` option for help:

- `labtainer <lab> --` Start the named lab. If no name is given, a list of available labs will be displayed. Command completion is supported, e.g., typing `labtainer tel` followed by the tab key will display all labs starting with `tel`.
- `stoplab --` Stop the currently running lab.
- `moreterm.py <lab> <container> --` create a new virtual terminal for the container.
- `labpack` – List the installed Labpacks, i.e., groups of related labs.
- `imodule` – Manage local IModule labs, e.g., labs distributed by your instructor.
- `labtainer <lab> -r --` Delete any previous containers associated with this lab and start it fresh. **Warning**: this will lose any previous data from the named lab.
- `checkwork` – Performs automated assessment for selected labs and provides you with information about your progress. Note this is not a grade and is not a score. It simply reflects a lab-dependent set of goals.
- `quiz` – Provides a quiz for selected labs to help prepare you to perform the lab.
- `update-labtainer.py` – Update the Labtainer installation to include bug fixes and new labs.
- `check_nets.py` – Runs diagnostics to potentially resolve Docker related problems.

Appendix B Customizing terminals

Terminal colors and text size can be customized by right clicking on a terminal and selecting `Preferences`. From there, select the `Unnamed` or `Default` profile and click its down-arrow and select "clone". Give the new profile a name, and then select your new profile. Adjust the colors and text appearance by selecting the tabs on the top of the window. Experiment by creating a new terminal window, right-click and select your profile from the `Profiles` submenus.

If you want all of your terminals to look like a new profile, click the down arrow on your new profile and make it the "default".

If you create a terminal profile named *labtainers*, that profile will be used with Labtainers lab terminals. This can be helpful to distinguish the Labtainers terminals from other terminals on your desktop. A video tutorial on customizing terminals is at `https://nps.edu/web/c3o/labtainers-tutorials`.