NPS-CS-19-003

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

LABTAINER FRAMEWORK EXTENSIONS: FINAL REPORT

by

Cynthia E. Irvine and Michael F. Thompson

August 2019

Approved for public release. Distribution is unlimited

Prepared for: National Science Foundation

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *08-28-2019*

2. REPORT TYPE Research

3. DATES COVERED ***(From-To)***
08-13-2015 to 05-31-2018

4. TITLE AND SUBTITLE

Labtainer Framework Extensions: Final Report

5a. CONTRACT NUMBER

5b. GRANT NUMBER
DUE-1438893

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Cynthia E. Irvine
Michael F. Thompson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Naval Postgraduate School
1 University Circle
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CS-19-003

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Science Foundation

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The material presented here is based upon work supported by the National Science Foundation under Grant No. DUE-1438893. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of the Department of the Navy.

14. ABSTRACT

Educational institutions are experiencing increasing demand for courses in computer and network security. Yet faculty members struggle to prepare new classes and compelling laboratory exercises, and must attempt to provide experiential learning to growing numbers of students. Meaningful laboratory activities are difficult to create and assess. However, laboratory exercises in which students learn fundamental concepts through exploration can help attract students, and ultimately into the cybersecurity workforce.

The Labtainer Framework for cybersecurity lab exercises was developed under NSF grant DUE-1438893. These fully packaged Linux-based cybersecurity labs run entirely within a modestly provisioned laptop computer. Labtainers currently offer:

- Automatically provisioned, consistent execution environments on diverse Linux distributions,
- Multi-component network topologies without running multiple virtual machines,
- Individualized labs to discourage the sharing of solutions and support reusability by instructors,
- Automated assessment of student progress and
- Over 45 labs covering topics ranging from software vulnerabilities, networking, cryptography, web security, system security and operations, and industrial control system security.

Labtainers are currently used by instructors worldwide to enhance cybersecurity education for their students. Each individual download enables the student to perform all existing Labtainer exercises as well as new Labtainer exercises introduced in the future. Labtainers has been well received by instructors from a range of educational institutions.

15. SUBJECT TERMS
Cybersecurity education, laboratory exercises, containers

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Cynthia E. Irvine
Unclassified	Unclassified	Unclassified	Unclassified	23	**19b. TELEPHONE NUMBER *(include area code)*** (831) 656 2461

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

The report entitled *Labtainer Framework Extensions: Final Report* was prepared for and funded by the National Science Foundation.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Cynthia E. Irvine
Distinguished Professor
Computer Science Department

Michael E. Thompson
Research Associate
Computer Science Department

Reviewed by:

Peter J. Denning, Chairman
Computer Science Department

Released by:

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

DEPARTMENT OF COMPUTER SCIENCE

NAVAL POSTGRADUATE SCHOOL

NPS-CS-19-003

Labtainer Framework Extensions: Final Report

Cynthia E. Irvine
Michael F. Thompson

August 2019

ACKNOWLEDGEMENT

A portion of the material presented here is based upon work supported by the National Science Foundation under Grant No. DUE-1438893. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of the Department of the Navy.

A Introduction

Workforce demands for individuals able to configure and defend computers and networks far outstrips production from the nation's educational institutions. [3, 16, 4]

Educational institutions are experiencing increasing demand for courses in computer and network security. Yet faculty members unfamiliar with security struggle to prepare new classes and compelling laboratory exercises, while experienced faculty members must attempt to provide experiential learning to growing numbers of students. In addition, instructors whose classes do not directly focus on security may wish to incorporate laboratory materials for a unit on security, but lack sufficient expertise to build them from scratch. Unfortunately, meaningful laboratory activities are difficult to create and assess. However, excellent laboratory exercises in which students learn fundamental concepts through exploration and discovery can help draw students to cyber security studies, and ultimately into the cybersecurity workforce.

To help meet this need, the Labtainer framework for Linux-based cybersecurity lab exercises was developed under the support of NSF grant DUE-1438893.

Labtainers are fully packaged Linux-based cybersecurity labs designed to run entirely within a modestly provisioned laptop computer [13, 12]. Labtainers currently offer:

- Automatically provisioned, consistent execution environments on diverse Linux distributions,
- Multi-component network topologies without running multiple virtual machines,
- Individualized labs to discourage the sharing of solutions and support reusability by instructors,
- Automated assessment of student progress [17], and
- Over forty-five cybersecurity labs, many derived from SEED labs [9, 7, 8, 23]. These labs cover the following categories of topics: software vulnerabilities, networking, cryptography, web security, system security and operations, and industrial control system security.

Labtainers are currently used by instructors worldwide to enhance cybersecurity education for their students. For example, there were over fourteen hundred downloads of the Labtainers product during a recent four-month period. Each individual download enables the student to perform all existing Labtainer exercises as well as new Labtainer exercises introduced in the future. As can be seen in Section sec:testimonials, Labtainers has been well received by instructors from a range of educational institutions.

This report summarizes our current work on the Labtainer Framework. This section has introduced the work. In the next section we motivate the project. In Section C we provide an overview of the Labtainer framework.

B Why Labtainers?

Laboratory exercises provide students with experiential learning opportunities that reinforce concepts presented in readings and lectures. In addition, they allow students to learn how a variety of techniques can be used to solve problems. In the context of cybersecurity, laboratory exercises must cover a wide range of topics and should be organized so that their learning outcomes progress from fundamental concepts to in-depth understanding.

Despite the acknowledged value of experiential learning [10], cybersecurity educators often face a number of challenges when attempting to provide worthwhile laboratory exercises for classes.

First, they may lack the institutional infrastructure required to host complex lab exercises that often involve networked systems.

Second, they may not have the time or expertise required to develop high quality exercises. Just as educators often rely upon good textbooks that contain well-vetted homework problems, and that may be accompanied by slides and other materials, so can over-worked instructors benefit from materials that will help them provide useful laboratory activities.

Third, operating system and application configuration is a significant source of frustration and distraction for students and instructors when working with computer science labs. When students are required to use their personal systems, the consequent platform diversity can force instructors to manage problems associated with both the exercise setup and the potentially diverse results that can emerge due to platform differences. Platform diversity can also negatively impact students, who must devote a disproportionate amount of time to lab set up, relative to that devoted to doing the lab exercise and achieving the intended learning objectives. Furthermore, if the setup for even simple labs is arduous, students may perceive cybersecurity as laborious and uninteresting.

Fourth, often the best learning takes place when students are engaged in exploratory exercises. An innate characteristic of exploratory exercises is that they can involve many blind alleys and detours. When instructors encourage exploration, understanding what each student has done, where that student may have had difficulty and whether the lab was successfully completed can become very time-consuming.

Last, students may share or reuse exercise results. The undesirable consequences of this behavior include: implicit punishment of diligent students who complete the exercises themselves; student-cheaters who do not learn the material yet receive undeserved high marks; and instructors who are unable to identify and assist students having difficulty with the material. Individualized exercises could solve the problem; however, if each student is given an different exercise, the instructor is faced with a huge grading task. Also, it is possible that the individualized exercises will vary in difficulty, making the exercise unfair. Without individualized labs that are fair and consistent and despite the proven effectiveness of existing lab exercises, instructors are often forced to create new exercises each time a course is taught. A monumental task that may deter instructors from offering labs at all.

To address these problems, we developed *Labtainers* under a grant from the NSF (DUE-1438893).

C Overview of Labtainers

The Labtainer Framework builds on Docker containers [6] to automatically provision lab execution environments that ensure consistency in software packages and system configurations for all students and instructors. An instance of *OS-level virtualization*, Docker containers, which are sometimes referred to as *lightweight virtual machines*, require substantially fewer computing resources than do full virtual machines [2]. This allows a Labtainer exercise that includes multiple networked components to run on a student computer that could not support multiple virtual machines due to memory or processor limitations. When beginning a Labtainer exercise, the student is presented with one or more virtual terminals connected to what appear to the student as independent computers linked by one or more networks. Each of these computers is, in fact, a Docker container that runs multiple programs, e.g., network daemons, and has its own set of users, files, and device interfaces. Students interact with these containers via virtual terminals, e.g., *bash* shells, GUI-based applications, such as a browser running within the container, or both. When the student stops a lab exercise, the Labtainer Framework collects a set of artifact files from the student's session. The resulting file archive is forwarded to the instructor, who uses similar containers to review student work and perform automated assessment of student progress against the lab exercise's pre-defined goals.

The Labtainer Framework's automated provisioning features relieve students and instructors from the distractions and frustrations often associated with configuring environments to support labs. The Docker containers described in the previous paragraph are primarily defined by Docker images, which are sets of files and configuration settings that will exist in the running container. The Labtainer Framework pre-provisions software packages and networking configurations within Docker images stored on a publicly accessible Docker Hub or other Docker registries. When a student first starts a given lab, the corresponding Docker images are automatically pulled from the registry and incorporated into running Docker containers on the student's computer.

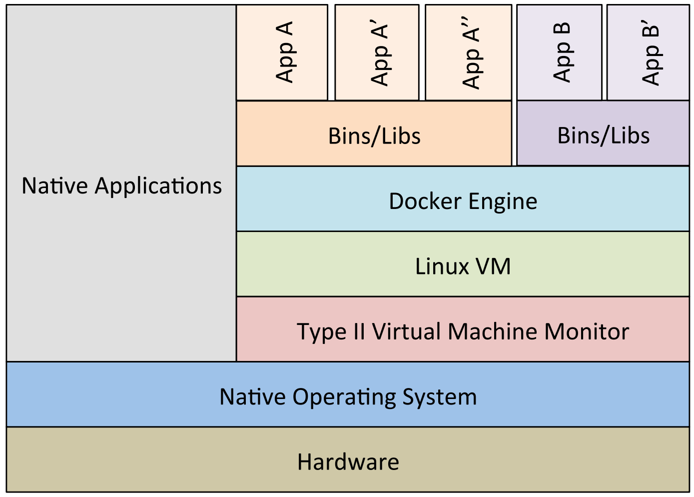

Figure 1: Typical Labtainer software stack

Students typically obtain Labtainers by downloading and importing a VM appliance into either Virtual Box or VMWare. Alternately, the Labtainer Framework installs on any Linux system that supports Docker. The software stack of a typical student system is illustrated in Figure 1

C.1 Labtainer User Types

.

The Labtainer Framework supports three types of users.

Lab Designers are responsible for for creating laboratory exercises so that they meet intended learning objectives. Each lab designer determines if and how the lab is parameterized and whether automated assessment will be supported.

Instructors assign labs to students and assess their work. Instructors may or may not work with lab designers to create exercises.

Students perform the laboratory exercises. They are oblivious to the underlying framework that configures and individualizes their labs, and which will later gather any artifacts required for assessment.

C.2 Labtainer Primary Goals

.

The overarching goal of the Labtainers project has been to develop a framework for cybersecurity laboratory exercises that is:

Consistent and Fair Laboratory exercises are delivered to students in a preconfigured environment. Different labs offer different environments. Both students and instructors generate consistent, reproducible results. Each lab environment is homogeneous across the entire instructor and student population, despite heterogeneity across the underlying platforms. (Of course, per-platform performance can vary, so temporal heterogeneity will still be evident.) Students do not have to struggle to create lab environments, and, instead, can concentrate on and learn from the lab assignment. Instructors can assess student work without having to contend with differences resulting from the way the labs were set up.

Parameterizable Each student has a laboratory exercise that cannot be accomplished by simply copying the results of another student. Parameterization choices ensure that all students have labs of the same difficulty. Lab designers can parameterize the generation of expected results on a per student basis, thus streamlining the assessment process. Labtainers are not an absolute guarantee that students will not engage a surrogate to do the lab or that a student might outwit the parameterization mechanism for a particular exercise, but the work factor required to cheat is intended to disincentivize such misbehavior.

Support Automatic Assessment The Labtainer Framework is designed so that evidence can be collected to demonstrate that students have achieved the intended intermediate and final goals of exercises. Information is automatically collected so that the instructor can determine where students become confused or go off track. The Framework includes automated assessment of artifacts gathered from student work, based on criteria provided by the lab designer.

C.3 Efficient Use of Computing Resources

The primary efficiency that containers offer over VMs is that the containers all share the host platform's OS kernel, as shown in Figure 1. For example, a lab consisting of three computers, each based on a different Linux distribution, can have all three share the underlying Linux platform's kernel and thus does not incur the overhead of running multiple kernels.

Leveraging Docker containers allows the Labtainer Framework to efficiently manage the space needed to store multiple labs on the student computer, and to reduce the size of downloads used to import a lab into the student computer. Docker manages each container image as a set of layers each of which represents an incremental addition, e.g., the addition of a set of software packages. The lowest layer is a stripped-down Linux distribution, e.g., Ubuntu. The final layer is a command that is to execute within the container, e.g., to launch the Linux initial process. Docker manages these images such that layers are not duplicated on disk or in memory when multiple containers are running. As a result, even though the image for a single container may require several hundred megabytes of storage, most of that storage is not duplicated as additional containers are added. Once present on the student computer, the Docker system will not pull that layer again from the Docker registry. For example, if the image for a new lab is similar to that used for a previous lab, only the new layers will be pulled from the registry when the new lab is started.

The Labtainer Framework manages the building of containers from images on the student computer when a lab is first started. It uses lab-specific configuration files to define virtual networks and their connections to individual containers. These configuration settings are image-independent, and thus two labs that share the same images can have different network topologies.

C.4 Lab Exercise Definition

The most challenging and critical part of creating a new cybersecurity lab is the design of the lab itself, i.e., identifying learning objectives and organizing exercises to achieve those objectives. The Labtainer Framework does not specifically address any of that. Rather, the Framework allows the lab designer to focus more time on lab design and less time on mitigating and explaining system administration burdens typically placed on students and instructors in other lab environments. The Framework does not require lab designers to program or create scripts. The lab designer primarily interacts with the Framework by editing configuration files that affect the student's execution environment and the optional automated assessment of student activity.[19]

In the remainder of this section, we outline the major steps in lab definition.

C.4.1 Define the lab execution environment

A given lab typically requires some set of software packages, and some system configuration, e.g., network settings. Identifying an expected environment is a typical element of any lab design. The Framework captures most configuration details within a standard Dockerfile.[5] Additionally, the lab designer identities the set of lab-specific files, e.g., vulnerable programs, that are to reside in the student's home directory within the container. Simple labs can use the default Dockerfiles created by the Labtainers script.

The Framework includes a set of baseline images available for lab designers to draw from. A single lab may contain multiple containers, built from different baselines. The baseline containers include:

Table 1: Baseline container elements

base	A basic Ubuntu distribution with software development tools, e.g., *gcc*, and software analysis tools such as *gdb* and *hexedit*.
network	Extends the base image to include networking tools such as *openssl*, an *ssh* server, *netcat* and *tcpdump*.
firefox	Extends the base to include the Firefox browser.
java	Extends the firefox image include a Java development kit.
centos	A CentOS distribution including networking utilities and analysis tools
lamp	Extends Centos to include the Apache web server, MySQL and PHP, i.e., the 'LAMP' stack.

C.4.2 Lab Parameterization

When a student starts a lab, the Labtainer Framework incorporates a student-supplied email address into a seed for generation of pseudo random values. A watermark file is automatically created for each student lab, and this file becomes one of the artifacts within the student's archive. The watermark value is validated as part of the assessment process initiated by the instructor. This simple strategy ensures (albeit weakly), that the archive provided by the student originated with the student commencing the lab.

Additional assurances of the originality of student work rely on choices made by the lab designer, who identifies specific properties of the lab that are to be individualized for each student. [17] Parameterization is achieved by defining symbols within source code or data files. During container initialization, the Framework will replace these symbols with randomized, student-specific values. A configuration file identifies the files, and the symbols within those files, that are to be replaced by the computed values. An example might be a constant controlling a buffer size in the source code of a program that the student is to interact with. A given student's instance of the lab will have a cryptographically generated seed that is unique to that student for that lab. The configuration file entry for the example buffer size value would be expressed as a random value within a given range. When the student first starts the lab, functions within the Framework will replace the indicated constant in the source code file prior to code compilation.

C.4.3 Automated assessment of student labs

Labs can be configured for automated assessment of student work. This is optional; the Framework does not require that labs include automated assessment, e.g., the "results" of a lab may consist entirely of a written report submitted by the student.

The goal of automated assessment is to provide instructors with some confidence that students performed the lab, and to give instructors insight into which parts of a lab students may be having difficulty with. It is possible to create exercises in which the student must execute a specific sequence of operations, and the lab designer can easily create assessments for such labs. However, it is more interesting to design labs that encourage experimentation. Yet such experimentation does not lend itself to highly prescriptive assessment techniques. What is needed is a means to ensure that students achieve a desired set of goals, albeit through individual explorative steps. Thus, the automated assessment functions implemented in the Framework are not intended to standardize each student's approach to a lab, rather the objective is to permit *ad hoc* exploration by students. To do so, the Framework gives the lab designer a means to identify evidence that steps of a lab were performed, rather than trying to identify everything a student may have done in the course of the lab.

The Framework's automated assessment functions generally assume the student will interact with one or more programs or system utilities. Each time the student invokes a selected program or utility, the Framework captures copies of standard input and standard output (*stdin* and *stdout*) into timestamped file sets. This is transparent to the student. For example, one artifact might be the value that follows the string `The result is:` within the *stdout* file. The lab designer then references these tagged values within a configuration file that defines the expected results of the lab. The expression of expected results can include indirect references to values that are specific to the student's instance of the lab, i.e., are a function of the lab-specific seed described earlier.

These timestamped file sets, and everything else germane to the exercise in the student's container-based home directory, are automatically packaged when the student completes the lab. These packages of artifacts are then transferred to the instructor, (e.g., via email or a learning management system), and ingested into the instructor's system where lab assessment occurs.

Thus, for assessments, the Framework requires the lab designer to create configuration files identifying what has been parameterized for each lab, and what artifact values constitute success. This extra work by the lab designer greatly simplifies the instructor's task of assessing each student's performance of the labs. While many labs may still include an essay component in which the student is asked to describe the lab activity, the automated evaluation of artifacts from individualized labs gives the instructor confidence that the student essays reflect work performed by the student.

Labtainer Framework currently includes over 45 lab exercises. These are listed in the tables that follow.

Table 2: Crypto Labs

Lab Name	**Description**	**Difficulty**
macs-hash	Exploration of cryptographic hashes and the potential for hash collisions.	2
onewayhash	Introduction to generating cryptographic hashes using the openssl utility.	1
pubkey	Explore public key certificates from a variety of web sites	1
sshlab	Use of a public/private key pair to access a server via ssh.	1
ssh-agent	Use an SSH agent to manage your private key and avoid retyping your passphase	1
ssl	Use of SSL to authenticate both sides of a connection, includes creating and signing certificates using a CA.	
symkeylab	Exploration of symmetric key encryption modes.	1
vpnlab	Example use of OpenVPN to protect network traffic.	2
vpnlab2	Similar to vpnlab, but with the use of a vpn gateway.	2

Table 3: Web Security Labs

Lab Name	**Description**	**Difficulty**
webtrack	Illustrates web tracking techniques and the role of ad servers	1
xforge	Cross Site Request Forgery with a vulnerable web site	2
xsite	Cross site scripting attacks on a vulnerable web server.	2
sql-inject	SQL injection attacks and countermeasures.	2

Table 4: Networking Labs

Lab Name	Description	Difficulty
telnetlab	The student uses telnet to access a remote computer, and employs the tcpdump tool to view plaintext passwords, and to observe how use of ssh mitigates that vulnerability.	1
nmap-discovery	The *nmap* utility is used to locate an ssh server on a network and to discover the port number being used by the service.	2
pcapanalysis	The tshark network traffic analysis tool is used to identify and display a specific network packet containing a plaintext password.	2
wireshark-intro	Introduction to the use of Wireshark analyze network traffic.	2
nmap-ssh	The nmap utility is utilized in combination with the tshark network traffic analysis utility to demonstrate a security problem with an ssh server.	2
routing-basics	A simple routing example with two LANs and an internet connection via NAT	2
iptables	The iptables utility is used to configure a firewall component to only forward selected application service traffic between a client and a server.	2
tcpip	TCP/IP protocol vulnerabilities, including SYN flooding, RST attacks and session hijacking. Derived from the SEED lab.	2
arp-spoof	Use of ARP spoofing for Man-in-the-middle attacks.	2
local-dns	DNS spoofing and cache poisoning on a local area network. Derived from the SEED lab.	3
snort	Use of snort for network intrusion detection	2
dmz-lab	Set up a DMZ for an enterprise.	2
radius	Use a Radius authentication service to authenticate network devices.	2
ldap	Authenticate users of Linux servers using an LDAP service.	2

Table 5: Software Vulnerability Labs

Lab Name	Description	Difficulty
bufoverflow	An example program vulnerable to a stack buffer overflow, derived from a SEED lab	3
formatstring	Explor C library printf function vunerabilities, derived from a SEED lab	2
retlibc	Exploit a program using a buffer overflow and return-to-libc, derived from a SEED lab.	3
gdblesson	An introduction to using gdb to debug a simple C program.	1
metasploit	Use metasploit on a Kali Linux system to attack a ”metasploitable” host.	1
setuid-env	Risks of the setuid feature, including environment variables, derived from a SEED lab.	2

Table 6: System Security and Operations

Lab Name	Description	Difficulty
acl	Access Control Lists (ACLs) on Linux	2
backups2	Using tar and dump/restore for file backups, including remote backups.	1
capabilities	Use of Linux capabilites to limit program privileges.	2
sys-log	System log basic usage and configuration on an Ubuntu system.	2
centos-log2	System log basic usage and configuration on a CentOS system.	2
file-deletion	Data recovery from deleted files within EXT2 and NTFS file systems.	2
file-integrity	File integrity checking and intrustion detetion with AIDE	2
pass-crack	Introduction to passwords and elementary cracking schemes.	2
denyhost	Use of the denyhost utility to block brute force attacks on SSH	2
nix-commands	Introduction to Linux and shell commands.	1

Table 7: Industrial Control System Security

Lab Name	Description	Difficulty
softplc	Program a software-based programmable logic controller (PLC)	3
plc-forensics	Forensic analysis of a PLC session from a rouge client.	2
plc-forensics-adv	Forensic analysis of a PLC session from a rouge client, including CIP & EtherNet/IP protocols.	4
plc	Simulated example of a vulnerable Programmable Logic Controller system.	2
plc-app	Application firewall and whitelisting to protect a PLC.	2
iptables-ics	Use iptables to limit traffic destined for a PLC through a firewall.	2
grassmarlin	Introduction to the GrassMarlin SCADA/ICS network discovery tool.	2
grfics	A graphical realism framework for industrial control simulations https://github.com/djformby/GRFICS	2

D Impact

This project advances cybersecurity education. Labtainer exercises cover foundational cybersecurity topics, and individual labs can also address specific topics resulting from new research and developments in the field. For example, new vulnerability assessment tools designed for use in Linux environments can be easily deployed within a Docker container and incorporated into a Labtainer exercise. The tool's environment and configuration can be entirely defined by the lab designer to focus on a specific set of learning objectives.

The Labtainer Framework simplifies creation, deployment and assessment of new cybersecurity exercises, allowing instructors and other content developers to focus more on the pedagogical content of the labs and less on administration and mechanics of lab deployment. The assessment functions provide insight into student results. This can be particularly valuable when creating labs that address new topics that do not yet have an established base of educational material.

The overarching goal of this project has been to provide quality cybersecurity lab exercises to instructors and students at a low cost (measured in work required to administer and assess labs as well as license and service fees.) To date, counts of virtual machine downloads (over 2000 over six recent months), suggest that Labtainers is being used in courses and that its use is growing.

Our graduate students have used many of the laboratory exercises and we have had successful results. Three high school and one university student have participated in the project as interns. While learning about Linux, scripting, and programming, the interns brought fresh eyes to the project and have contributed to all three Labtainer roles as students, instructors, and lab designers. They have been invaluable in debugging documentation and procedures.

D.1 Dissemination and Outreach

Table 8 lists the various publications and demonstrations associated with this project.

Table 8: Labtainer Publications, Presentations, and Demonstrations

Type	Title
Article	Labtainers: A framework for parameterized cybersecurity labs using containers. [12]
Article	Labtainers: A Docker-based cyber lab framework [13]
Article	Individualizing Cybersecurity Lab Exercises with Labtainers [17]
Article (review periodical)	Individualizing Cybersecurity Lab Exercises with Labtainers [18]
Extended Abs.	CyberCIEGE and Labtainers: Promoting Exploration in Cyber Security Labs [11]
Manual	Labtainer lab designer user guide [19]
Manual	Labtainer Lab Manuals [1]
Manual	Labtainer Student Guide [15]
Manual	Labtainer Instructor Guide [14]
GitHub website	Labtainers: A Docker-based cyber lab framework [21]
NPS website	Labtainers [22]
Docker Hub	Docker Hub - Labtainers [20]
Presentation	Labtainers: A framework for parameterized cybersecurity labs using containers. [12]
Presentation	CyberCIEGE and Labtainers: Promoting Exploration in Cyber Security Labs [11]
Presentation	Labtainers: A Docker-based cyber lab framework [13]

Labtainers is distributed from our website as virtual machine appliances, and as an alternative archive installable on any Linux system. All source code and documentation is maintained on GitHub, and the Labtainer Docker images are all available on the public Docker Hub web site. The Labtainers project maintains an email list server that we use to announce new releases of Labtainers.

The Labtainers VM appliances, and the alternate installation package, are available at the Labtainers website [22] The Docker images used by each lab are automatically pulled from the Docker Hub. [20]

Labtainer framework and all lab source code is available on GitHub. [21]

D.2 Educators' Testimonials of the Value of Labtainers

The following statements of support are from external educators who are using Labtainers.

> Labtainers is a great way to introduce students to the technical aspects of cybersecurity. ...
> As a cybersecurity researcher and instructor, I highly recommend the adoption of Labtainers in cybersecurity classes as a viable alternative to virtual labs that are designed by instructors from scratch, or virtual labs of questionable quality that are offered to students at a steep price by academic publishers.

> I used about 12 of the Labtainer exercises in my graduate class in Computer and Network Security ... I thought they were excellent hands-on exercises with enough depth to provide some very nice learning opportunities. ...
> I talked to the students frequently about their thoughts on these assignments as a learning tool - and every one of them felt very positively. I hope to use them again the next time I teach this class and perhaps add more of them to the assignment list. I truly appreciate this tool - I feel it helped me offer a deeper experience in learning security to the students.

> ... I was asked to build and deliver a "300-level" technical elective Cyber Security for Software Developers, for Spring 2019. ... I can safely say that the Labtainer Project is and will be essential to the success of my Cyber Security for Software Developers course.
> From my brief experience, I can see that the Labtainer Project can offer any school and every student the ability to build hands-on skills in the configuration and operation of a range of cyber systems in a virtualized lab environment, with no capital investment, and minimal lab administration overhead. We have installed Labtainers on our lab machines, and many students have opted to install on their personal systems, so they have access to the capability all the time.

> Labtainers brings a very interesting virtual labs environment for teaching cybersecurity. We're adopting it for a MOOC on cybersecurity which is part of a series of MOOCs in taught in French on networking, and will be hosted on the french platform FUN-MOOC, reaching thousands of learners in all francophone countries. Not only does Labtainer include many already useable labs on cybersecurity, but it brings great tools for personalization and auto-evaluation of learner progress, which is precious in a context of distant learning like a MOOC. The publicly available source code and use Python and Linux containers make it quite flexible and help us integrate it in the PaaS platform used for giving learners seamless access to virtual labs.

> As a professor of cybersecurity and an instructional designer, I can attest to the fact that it can be quite the challenge to find excellent hands-on learning resources that can support a cybersecurity curriculum. There are many products out there, but their quality is often not great, and for the better ones, the cost is quite high. I was extremely delighted to find out about Labtainers. The exercises are well thought out, easy to follow and provide good feedback to the professors. The Labtainers cover a wide range of topics and provide the possibility of creating your own. I use Labtainers in several of the classes I teach. My students have ALL enjoyed the labs and have state so to me and in the evaluations of the courses.

My students and I work in security and privacy and the research that we do is very high-impact (multiple top-tier papers published every year with our work regularly finding its way to the popular press).

I regularly teach introductory and advanced security courses at an undergraduate level as well as at a graduate level. These courses are very well attended (typically more than 100 students each time) ... the large number of students puts a strain on myself and the TAs in terms of providing realistic, hands-on assignments and somehow grading hundreds of these assignments each time we offer a security course.

I was impressed not only with the large number of realistic exercises across a wide range of topics (including access control, authentication, buffer overflows, and cross-site scripting) but also with the auto-grading functionality available for most of the offered exercises.

In summary, I am very thankful for the Labtainers platform and I am unreservedly supportive of any efforts to keep this platform maintained and expand it with more labs and additional functionality.

E Summary

The **intellectual merit** of the Labtainers project is reflected in its creation of a framework for cybersecurity labs. That framework has been proven through creation of a variety of cybersecurity labs, and by importing several of the highly popular laboratory exercises from the SEED project. The resulting labs are entirely provisioned using Docker containers so that students are no longer required to make administrative or configuration changes to perform the labs. As a result, students can run labs containing multiple networked components all within the confines of a laptop computer, and do so without having to make configuration and networking changes particular to the lab exercises. Automated assessment of student progress supports instructors by relieving the burden for significant portions of the grading effort. Individualization of exercises discourages sharing of results between students and allows instructors to reuse successful labs in subsequent classes.

The **broader impacts** of the work include:

- Availability of a large number of prepackage laboratory exercises for use by instructors,
- Modest resource requirements for running labs, even those involving many distributed components. This supports students at institutions with limited resources.
- Tools for the creation of new labs,
- Availability of all results of the Labtainers project, and
- Parameterization to prevent students from sharing results.

Future work on Labtainers can include:

- A broader suite of labs with which to gain hands-on experience with cybersecurity concepts,
- Facilitation of collaborations with other educators,
- Further automation of testing, integration and distribution of the Labtainer framework,
- Support to help educators effectively integrate Labtainers into cybersecurity courses,
- Increased support for contributions by outside educators, which should, in turn, increase the pool of available labs,
- Tools to simplify the lab designer's task, which again will result in additional cybersecurity exercises available within the Labtainer framework,
- Tools to simplify lab parameterization of labs will result in more exercises that instructors can assign without fear of students sharing results, and
- Improved tools for instructors that might facilitate their understanding of student progress and elements of labs that students have trouble with.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] —–. *Labtainer Lab Manuals*. Naval Research Laboratory, Monterey, California, October 2018.

[2] Abel Arvam. Docker: Automated and consistent software deployments. https://www.infoq.com/news/2013/03/Docker, 27 March 2013.

[3] BurningGlass. Job market intelligence: Cybersecurity jobs, 2015. http://burning-glass.com/wp-content/uploads/Cybersecurity_Jobs_Report_2015.pdf, 2015.

[4] Cisco. Mitigating the cybersecurity skills shortage. http://www.cisco.com/c/dam/en/us/products/collateral/security/cybersecurity-talent.pdf, 2015.

[5] Docker docs. Dockerfile reference. https://docs.docker.com/engine/reference/builder/, November 2017.

[6] Docker.com. Docker. https://www.docker.com, 2017.

[7] W. Du, K. Jayaraman, and N. B. Gaubatz. Enhancing security education with hands-on laboratory exercises. In *Proceedings 5th Annual Sympoisum on Inforamtion Assurance (ASIA'10)*, June 2010.

[8] Wenliang Du. Seed: Hands-on lab exercises for computer security education. *IEEE Security and Privacy Magazine*, 9(5):70–73, September 2011.

[9] Wenliang Du and Ronghua Wang. Seed: A suite of instructional laboratories for computer security education. *J. Educ. Resour. Comput.*, 8(1):3:1–3:24, March 2008.

[10] Janet Eyler. The power of experiential education. *Liberal Education*, 95(4), 2009.

[11] Cynthia Irvine and Michael Thompson. Cyberciege and labtainers: Promoting exploration in cyber security labs. https://www.fbcinc.com/e/nice/presentations/2017/lightning/L-2_Irvine.pdf, November 2017.

[12] Cynthia E. Irvine, Michael F. Thompson, and Jean Khosalim. Labtainers: A framework for parameterized cybersecurity labs using containers. In *Proceedings of the National Security Summit*, June 2017.

[13] Cynthia E. Irvine, Michael F. Thompson, Michael McCarrian, and Jean Khosalim. Labtainers: A Docker-based framework for cybersecurity labs. In *Proceedings of the 2017 USENIX Workshop on Advances in Security Education*. USENIX, August 2017.

[14] NPS Labtainer Team. Labtainer instructor guide. https://my.nps.edu/documents/107523844/109121513/labtainer-instructor.pdf/595fe4b8-9858-48a7-a822-9e9374f1ddd9, 17 November 2017.

[15] NPS Labtainer Team. Labtainer student guide. https://my.nps.edu/documents/107523844/109121513/labtainer-student.pdf/a6c49134-4bd4-457f-bb88-f5cb79b34fd0, 17 November 2017.

[16] Ariha Setavad. Demand to fill cybersecurity jobs booming. http://peninsulapress.com/2015/03/31/cybersecurity-jobs-growth/, 31 March 2015.

[17] M. F. Thompson and C. E. Irvine. Individualizing cybersecurity lab exercises with labtainers. *IEEE Security Privacy*, 16(2):91–95, March 2018.

[18] M. F. Thompson and C. E. Irvine. Individualizing cybersecurity lab exercises with labtainers. *Computing Edge (reprint)*, pages 29–33, May 2018.

[19] Michael F. Thompson. Labtainer lab designer user guide. http://my.nps.edu/documents/107523844/109121513/labdesigner.pdf, 27 October 2017.

[20] Michael F. Thompson. Docker hub - labtainers. https://hub.docker.com/u/mfthomps, April 2019.

[21] Michael F. Thompson. Labtainers: A docker-based cyber lab framework. https://github.com/mfthomps/Labtainers, April 2019.

[22] Michael F. Thompson and Cynthia E. Irvine. Labtainers. https://my.nps.edu/web/c3o/labtainers, April 2019.

[23] Xinli Wang, Yan Bai, and Guy C. Hembroff. Hands-on exercises for it security education. In *Proceedings of the 16th Annual Conference on Information Technology Education*, SIGITE '15, pages 161–166, New York, NY, USA, 2015. ACM.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943