NAVAL POSTGRADUATE SCHOOL
CENTER FOR INFORMATION SYSTEMS SECURITY STUDIES AND RESEARCH

Labtainers:
A Docker-based Framework for Cybersecurity Labs

USENIX Workshop on Advances in Security Education
15 August 2017

Cynthia Irvine, Michael Thompson,
Michael McCarrin, and Jean Khosalim
Department of Computer Science
Naval Postgraduate School

Experiential learning is desirable, but …

- Institutional infrastructure may be absent
- Labs are difficult to build and difficult to maintain
  - Overworked instructors need well-vetted labs
- Student platform diversity introduces problems
  - Different operating systems, libraries, software tools, etc.
  - Platform setup for lab distracts from learning objectives
  - Lab results vary widely due to configuration differences
- Experiential labs require exploration
  - How is this observed?
- Students may share or reuse other work
  - Need individualized labs, but grading effort becomes large

Labtainers Objectives

Consistent and Fair

- Students execute labs in identical environments
- Instructors see consistent results and assess students on their work rather than environmental effects

Parameterizable

- Labs configured so each student's work can be unique
- Labs are same level of difficulty for all students
- Expected results are parameterized to streamline grading

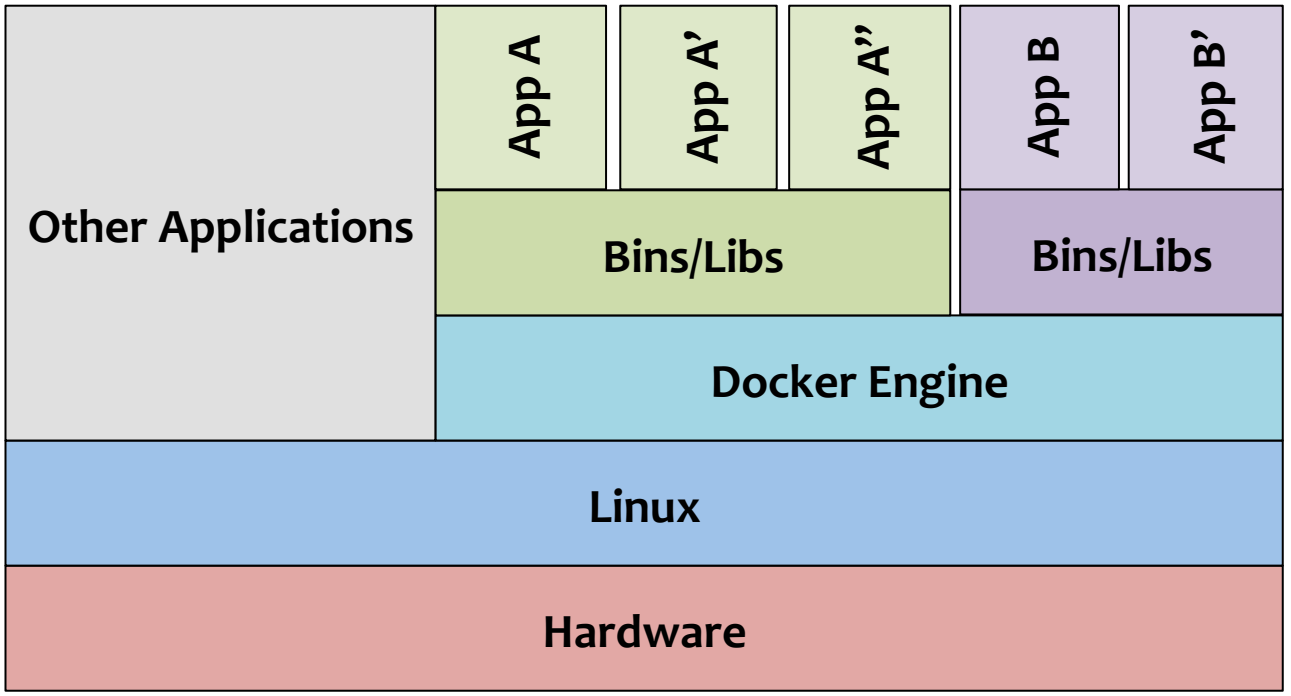
Support for Automatic Assessment

- Collected student work is parsed for specific outputs
- Tools may be developed to support assessment of particular aspects of exercise

Stand-alone Linux Cybersecurity Labs

- Multi-component network topologies
  - Packaged using Docker containers
  - Pre-configured execution environments
- Local to student's computer
  - One Linux host, (e.g., VM) runs many containers
  - No per-lab provisioning required by the student
- Public repository of labs & open framework

Architecture



- Linux can run in a VM on a non-Linux platform
- Preconfigured containers ensure consistency across heterogeneous platforms

Why Containers?

- Similar resource and naming isolation
  - Dockerfiles simplify provisioning of containers
- Student laptop can run several containers
  - But may be bogged down by 2 or more VMs
  - Enables labs with many networked components
- All containers share Linux kernel with host
  - But can have distinct packages & library versions
  - Containers limited to Linux

Parameterization

- Individualizes labs for each student (optional)
- Random number seed based on student email
- Example: size of buffer to overflow
  - Symbolic replacement of value in source code
  - Vulnerable program compiled during first run
  - Affects offset of return address to overwrite

Automated Assessment

- Student activity and files collected as artifacts
  - Mostly transparent to students, they see Linux
  - Bash hooks capture stdin & stdout
  - Artifacts forwarded to instructor
- Instructor tools assess student performance
  - Expected results as defined by lab designer
  - View of student's file and ability to run programs
- Lab exercises individualized for students
  - Discourages sharing answers & solution mining
  - Automated assessment makes this practical

Roles in the World of Labtainers

Designer

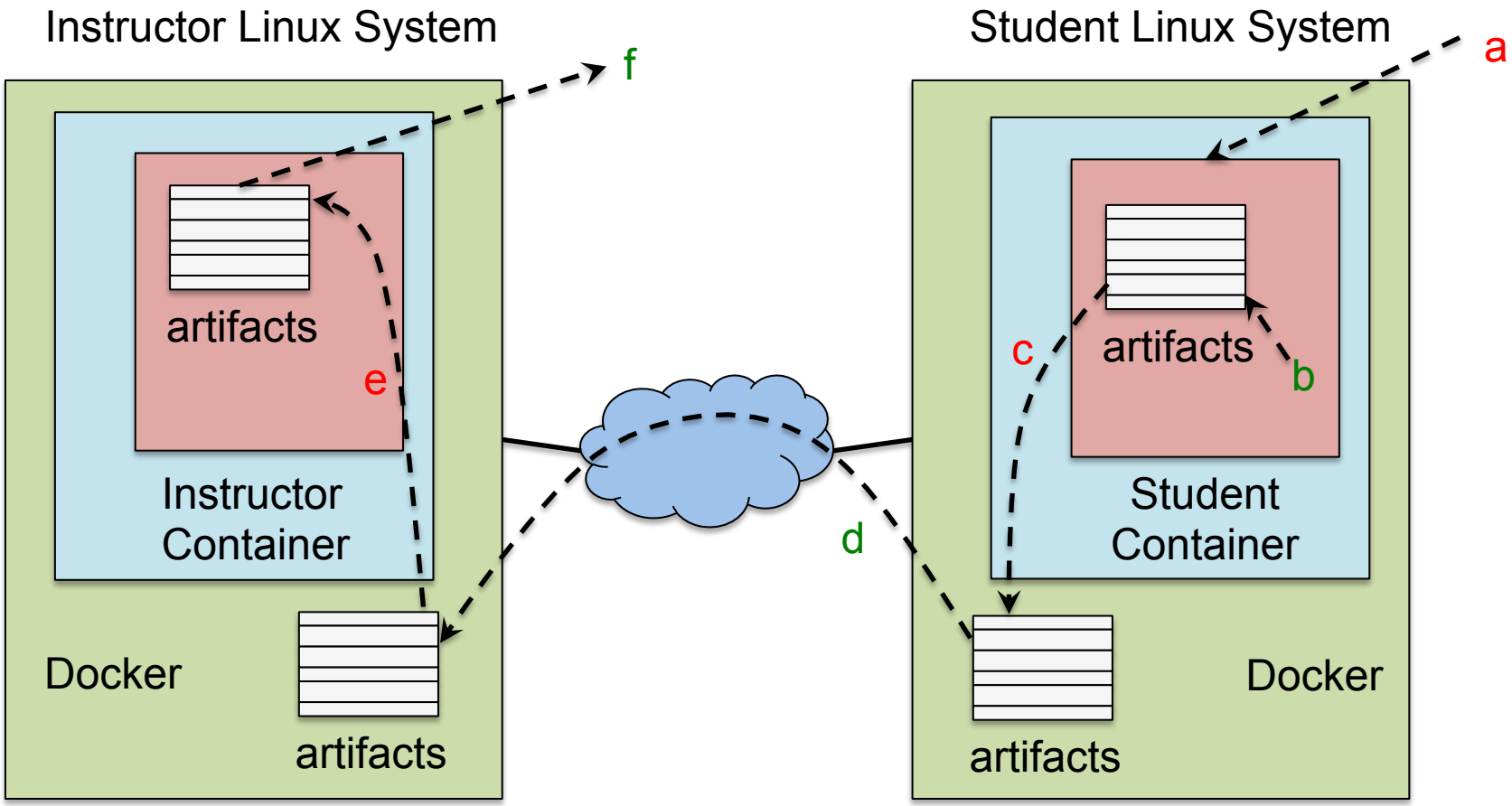
SME who works with instructor to create labs based on learning objectives. Fine tunes and updates labs. May support auxiliary assessment tools.

Instructor

Defines learning objectives. Works with (or is) designer. Ensures student readiness to perform labs and conducts assessments.

Student

Performs lab exercise. Learns! Delivers results to instructor for assessment.



- Designer-created configuration files provide automation (red) for
  - Container setup and artifact analysis
- Manual steps (green) : student runs lab, student sends artifacts to instructor, instructor reviews assessment table

```
# results.config
#
# Please see labdesigner.md

# The following are meant to identify artifacts from telnet lab

fileview = client:telnet.stdout : 4 : STARTSWITH : My string is:
sshfileview = client:ssh.stdout : 4 : STARTSWITH : My string is:
tcpdumprun = server:tcpdump.stdout : CONTAINS : mydoghas
failed_login = server:/var/log/auth.log : CONTAINS : FAILED LOGIN
```

```
# goals.config
#
# Please see labdesigner.md

# Recorded actions or results of student work to be summarized
# for instructor

telnetview = matchany : string_equal : fileview : parameter.FSTRING
sshview = matchany : string_equal : sshfileview : parameter.FSTRING
tcpdump_plain = is_true : tcpdumprun
failed_login = is_true : failed_login
```

```
# Filename : start.config
# Description:
#       A simple configuration file read by start.py
#       Defines the containers used by both students and instructors

GLOBAL_SETTINGS
        # GRADE_CONTAINER - container where instructor will perform grading
         GRADE_CONTAINER client
         # HOST_HOME_XFER - directory to transfer artifact to/from containers
         HOST_HOME_XFER seed_dir/
         # LAB_MASTER_SEED - this is the master seed string specific to this laboratory
         LAB_MASTER_SEED telnetlab_jean_seed

# SUBNETS
NETWORK SOME_NETWORK
         MASK 172.20.0.0/24
        GATEWAY 172.20.0.100

# Container name and settings
CONTAINER client
         USER ubuntu
         TERMINALS 2
         SOME_NETWORK 172.20.0.2
CONTAINER server
         USER ubuntu
         TERMINALS 1
         SOME_NETWORK 172.20.0.3
```

Note

For a simple single-container lab, there is a default configuration file.

Parameterization

- Labs parameterized using
  - Per student unique string, e.g. email address
  - Both student and instructor know string

```
# parameter.config
#
# Please see labdesigner.pdf

# This string provides per-student parameterization of telnet lab
FSTRING : HASH_REPLACE : telnetlab.server.student=filetoview.txt :
TELNET_STRING : mytelnetfilestirng
```

Assessment Support

- Instructor starts the lab and automatically
  - Containers created
  - Student artifacts pulled in
  - Results configuration sets up assessment environment
  - Loop through all students using
    - Parameter configuration
    - Goals configuration
  - Table of per-student goals produced to support assessment

	telnet	SSH	tcpdump	failed_login
Alan	Y	X	X	X
Alice	Y	Y	X	X
Barbara	Y	Y	Y	Y
Bill	X	Y	Y	X
Chuck	X	X	X	X
Corrine	Y	Y	Y	Y

Status and Near-Term Objectives

- A few proof of concept labs from SEED
- Current Labs
  - Format string (printf) vulnerabilities
  - Buffer overflow
  - Forensics
  - Cryptographic hashes
  - Telnet (plaintext password on the network)
  - VPN (configuring Openvpn to protect traffic)
  - nmap
  - gdb introduction
  - Several more

Status and Near-Term Objectives

- Summer 2017
  - Internal testing
  - Three highschool interns
  - Additional labs, e.g. ICS security
- Early Fall 2017, initial general release

Join the Labtainers Team

- The beta version of Labtainers can be found at http://my.nps.edu/web/cisr/labtainers

Contact

Cynthia Irvine irvine@nps.edu
Mike Thompson mftomps@nps.edu

Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943 U.S.A