An Authoring Process to Construct Docker Containers to Help Instructors Develop Cybersecurity Exercises*

Jack Cook[1], Richard Weiss[2], Jens Mache[3], Carlos García Morán[3], Justin Wang[4]

[1]New York University
Brooklyn, NY 11201
jcc9838@nyu.edu
[2] The Evergreen State College
Olympia, WA 98505
weissr@evergreen.edu
[3]Lewis & Clark College
Portland, OR 97219
{jmache, carlos}@lclark.edu
[4]Marquette University
Milwaukee, WI 53233
hsiaoan.wang@marquette.edu

Abstract

As instructors, we are more likely to use exercises that we can modify or that we helped to develop. The problem we address is how to help instructors create their own hands-on exercises. This paper describes the authoring process for the creation of cybersecurity exercises and the experience of creating two very different exercises. One of these exercises is about vulnerable Web services and leverages the LAMP stack and the other is about cryptography and ransomware and uses VNC. They both use Terraform to create Docker containers. We address the issues of creating exercises that involve multiple containers and can be run in a cloud environment, as well as on a single server.

This paper describes the process of creating these cybersecurity exercises in our framework. The streamlined process enabled the development of totally new exercises much faster than previously experienced. In the case of the ransomware exercise, it was two weeks from start to finish, compared to months for previous exercises.

1 Introduction

The development of new security exercises is a cornerstone to cybersecurity education. A number of platforms for teaching cybersecurity through hands-on exercises have been developed in last 15 years. Many of them have more than a dozen exercises. Yet, they are not truly scalable from the perspective of developing a community unless they facilitate the ability of knowledgeable users to modify existing exercises or contribute new ones. There are only a couple of frameworks that are designed to make this easy. In this paper, we examine the creation of two very different exercises in order to reflect on how to help instructors to create their own exercises in the domain of cybersecurity. Cybersecurity exercises have some special requirements: 1) exercises may rely on installing specific versions of software, including ones with vulnerabilities, 2) software environments need to be complete, i.e. more than just the vulnerable applications, and 3) exercises should run on a variety of platforms, e.g. AWS, Azure, Google Cloud and desktop.

Even when a platform provides help for creating exercises, there still is going to be a learning curve. We have tried to make that learning curve as gentle as possible in our platform. EDURange uses two powerful tools Docker containers and Terraform. These are commonly used in IT for creating and configuring flexible computing environments. The use of Docker containers is becoming more popular than the use of virtual machines (VMs), especially when multiple virtual computing environments would be needed. Similarly, Terraform is becoming popular because it works with multiple cloud frameworks to configure hardware and software and interacts well with Docker. We have constructed a layer on top of both of those, so as to minimize the prerequisite knowledge that instructors would need in order to create or modify exercises. A prerequisite that instructors would need is some familiarity with the Linux command line interface. However, we believe that this is less of an issue for most cybersecurity instructors. Thus, we have not developed a graphical interface for creating exercises, although that would certainly be possible.

The two new exercises that we developed were Ransomware and WebFu. The learning goal of Ransomware is to teach some of the basics of cryptography in a context that would be very clear and motivating to students. It also promotes the security mindset because it illustrates a failure mode. One generally thinks of cryptography as protecting secret information, but in this

context it is about abusing it to prevent the owner from accessing data. WebFu teaches the basics of SQL injection. This exercise was developed as a gentle introduction to the topic and as our first attempt at a user interface that is different from the command line. While there are many CTF challenges that are based on SQL injection, we wanted an exercise for an introductory Web security course that would use a Web interface rather than the command line. In our experience, students sometimes struggle because of their limited understanding of SQL databases and while there are many tutorials on SQL, they generally focus on how to use the language rather than how to abuse it. We also wanted students to be aware of code injection and to recognize that code is also data. Both of these exercises demonstrate that EDURange can accommodate a wider range of exercises, not just ones limited to using the command line interface in a bash shell.

2 Related Work

The other academic frameworks that we consider are Labtainers, DETERLab, SecKnitKit, Security Injections, NICE-challenge, and KYPO.

Of these, the only one that has addressed the issue of instructor-generated scenarios is Labtainers [7]. Labtainers has a collection of base Docker images that can be combined in a variety of ways to produce new exercises using Docker-Compose. This has some pros and cons. One advantage is that they have implemented a GUI that is aware of the base containers and allows the user to select them and compose them. The disadvantage is that if the user wants to go beyond the existing types of exercise, then they need to be familiar with Docker-Compose syntax, craft a unique Dockerfile, and also define networking rules from scratch. Labtainers can be used anywhere that has Docker installed, which could be a laptop or a Cloud environment.

In comparison, EDURange [9] takes a hybrid approach to this problem, by providing templates for instructors to modify while also allowing the use of custom container images. As a result, instructors can either provide their own pre-configured images, or extend the base SSH server with a list of their own bash scripts. One disadvantage is that there is no GUI, so exercise designers need to be familiar with basic Docker commands. Nevertheless, they don't need to know Docker Compose and instead can use JSON to combine Docker commands. EDURange can be used anywhere that has Docker installed, which could be a laptop or a Cloud environment.

DETERLab [4] also allows instructors to design their own exercises. It uses a combination of bash scripts and NS scripts. The NS scripts are not a commonly used format. There is not much documentation on the procedure for creating new exercises. The platform is tied to specific hardware. While it

is free, there are resource limitations.

NICE-challenge has on the order of 100 exercises and has a staff of developers. The advantage for the instructor is that there is no expense and little effort to use the exercises. The disadvantage is that it is not possible for instructors to modify or contribute exercises or to host exercises on their own hardware resources. As with DETERLab this limits scaling because the hardware resources are not easily expanded.

Security Injections [6], SecKnitKit [5], and SEED [1] are also valuable. While, an instructor cannot contribute or modify exercises, they are scalable in terms of the number of instances of a course. Security Injections does not require provisioning of VMs or containers, so it is easier to use than the other systems. SecKnitKit does use VMs that can be run locally on the instructor's hardware. SEED has one large VM that the students run and an associated textbook.

KYPO [8] is a very interesting system in terms of the exercises provided, and it is open source. However, it is not easy for instructors to add exercises or to run it on their own hardware.

There are several free non-academic frameworks such as Portswigger and overthewire.org. Instructors cannot modify or extend them, and they are generally harder to integrate into a course, in terms of assessment and prerequisite material.

The tools that make EDURange extensible, portable and scalable for instructors are Docker and Terraform. Section 3 steps through the process one would go through in EDURange to create a new exercise. Then, in sections 4 and 5, we discuss the particular requirements for those exercises.

3 Recipe for Creating New Cybersecurity exercises

Developing good hands-on exercises and homework assignments can be a difficult and time-intensive task. One standard method is backward design [10]. The author of an exercise would start with specifying the learning goals and develop a high-level description. They would translate the goals into concrete objectives and create a plan for assessing them. In the case of hands-on exercises, the objectives and assessment are often realized as tasks and criteria for determining that those tasks have been completed satisfactorily. Once the tasks have been described, they need to be implemented by creating the virtual hardware and software environment. In EDURange, we use a collection of containers running on Ubuntu. The author of a scenario would describe each container in detail. They would specify the software and services they should provide. They specify accounts for students and services, and create files/artifacts that the students need to retrieve. The author also needs to configure the

network, e.g. assigning IP addresses and redirecting ports. All of this should be done using scripts (mostly bash), so that it is easy to modify the exercise and create new containers.

Another approach that we have seen is to start with an existing virtual environment, where the instructor wants students to learn to work in that environment. In this approach, the instructor must then create the goals and learning objectives, usually based on introspection to understand why that environment is important and what the essential goals and objectives are. Then, the author could generate tasks that would demonstrate those objectives in that environment. An example of such an environment is Metasploit on Kali Linux. Metasploit is a penetration testing framework. It is easy to find VMs for both Kali (the attacker) and Metasploitable (the target). Both of these can be ported to containers and networked together. Student accounts can be created in the Kali container.

Both of these approaches are reasonable. In practice, we often see a hybrid or middle-first approach where there are some ideas for exercises that are attractive, and they suggest goals and objectives. The part that EDURange can help with in both of these approaches is to make it easy to configure the virtual environment.

4 Developing an SQL-injection exercise (WebFu) using the LAMP stack

The goal of this exercise was to teach SQL injection in a hands-on fashion. This led to defining objectives, such as dumping tables from a database and bypassing a basic Web Application Firewall (WAF). These needed to be translated into an implementation in a concrete environment. For WebFu, the author chose the MySQL database system, and created the database schema and then the queries. The next section describes the experience of applying the tools in EDURange.

4.1 Applying the Tools

First, the author copied Terraform templates from another scenario and changed the container names to match the new scenario name. The author also copied the YAML file containing the assessment questions for the students and the Markdown file containing the scenario's student guide. Of course, the text in these files needed to be changed for the new scenario, but that was not difficult. Next, the author made a copy of the existing EDURange base image from DockerHub, which is based on a minimal Ubuntu installation. The author then set up a LAMP stack by extending the image with a) a MySQL database server;

b) an Apache web server; and c) a PHP installation. These elements formed the infrastructure of the web application. After populating the database tables with data (made up of public datasets and the hidden artifacts or flags), the author pushed this modified image to EDURange DockerHub repository and edited a single line in the Terraform template to invoke it. Lastly, the author wrote a bash script for starting the MySQL and Apache services at scenario launch time and added it to the JSON file. The development of this infrastructure took about a month (the author was a full-time student and this was a side-project). However, this set up can now be reused to create a wide range of scenarios for practising web security auditing skills. Potential labs include a website vulnerable to Cross-site Request Forgery (CSRF), Server-side Request Forgery (SSRF), Cross-site scripting (XSS), Local File Inclusion (LFI), and many other techniques. With the current infrastructure-as-code, the only task left to the author is writing or copying the vulnerable application(s).

On the scenario's development, it must be said that gaining familiarity with the Docker workflow was challenging at first. This was where most of the time was spent, since the author had a background in Linux system administration, but not in container-related technologies such as Docker. Every time the Docker image was changed, a commit needed to be made for the new container which resulted in a new image. Then, this image had to be tagged and pushed to the DockerHub repository. Finally, the instance of EDURange had to pull from the remote repository to update its changes. This is similar to Git's workflow, and the method of container deployment results in an agile and effective process. Once the author became familiar with this, the process went more quickly.

When trying to deploy this exercise in the classroom, the author ran into an unexpected problem. The exercise was being run on a cloud environment, but students needed to connect through the school's network through HTTP. Students were experiencing problems connecting, and it turned out that an internal firewall was blocking malicious traffic (i.e., the SQL injection strings) over plain-text HTTP. The solution was to use HTTPS, but we wanted to avoid requiring an instructor to create and deploy an SSL/TLS certificate. Using Terraform's "bind" property for SSL support and redirecting ports (from the container to the host) were the most significant changes. Using "bind" allowed us to easily set up HTTPS for the web application. The Let's Encrypt directory with the certificate and the private key on the host VM was made available to the guest container through a bind mount. This removed the need for creating and maintaining additional SSL certificates. What's more important about this, is that the whole process occurs at the scenario's creation time, thus not having the private key stored in the repository's image, ensuring confidentiality. Lastly, the port redirection implemented with the Terraform API spared us

from having to write and maintain iptables rules. This was particularly helpful due to how easy it was to add redirection rules in the Terraform file.

4.2 Scripts and Files

This exercise required a working database. The tables were created using scripts. How and where to run the scripts was specified in a JSON file. The author found it is easy to use an existing file from another exercise as a template, but ideally this would be produced by a user interface that would prompt the user for the information and produce the JSON file. The JSON file defines a list of the containers to be provisioned for this scenario, as well as three categories of other files that are used by Terraform to create the container: user files, system files, and global files. For each type of file, Terraform will take different actions to copy or execute them in order to prepare the scenario environment. Terraform copies a list of "User Files" into each students' home directory, "System Files" are executed once at scenario launch time for system configuration, and "Global Files" are added to the "/bin" folder so they can be run in a bash shell.

4.3 Using Docker and Terraform in WebFu

Terraform is a scripting platform most commonly used by system administrators and cloud/DevOps engineers to create and configure (provision) virtual machines (VMs) In the case where networks of VMs are needed, Terraform can be used with a VM orchestration configuration file which is similar to the Docker YAML-based language for defining networks of containers. One of the common uses of Terraform is to modify the state of a VM running on a Cloud by applying rules while the VM is running. This takes the place of the administrator logging in to all of the VMs in the network and running update commands. However, this is not how we are using it. Instead, we focus on rapidly setting up containers and configuring them.

Our general approach is to create a base Docker image and write configuration files to customize the image for each specific exercise. One could imagine using Docker scripts to do this, but there are potential problems with synchronization. For example, when configuring a network, some steps are dependent on others. Instead, EDURange uses Terraform scripts to create containers and network them together. In this case, the network must be configured before the containers can use it, otherwise there will be errors. Docker scripts do not provide a simple and reliable way to do that, while Terraform does. The Terraform scripts can be generated by EDURange as JSON files. This makes it possible to implement a user interface that would make it easier for instructors and authors to create their own scenarios. In EDURange currently, we have

defined our exercises using Terraform templates that can be copied and adjusted. At the lowest level, this allows exercise developers to write bash scripts which modify an existing docker image to create the desired environment.

In practice, once contributors have written their desired scripts, they can just list them in JSON format to apply them and extend the docker image. This can be contrasted with other testbeds, in which manual editing of a Dockerfile or a NS file is required in addition to preparing low level scripts. Alternatively, authors can create their own Docker images and incorporate them by modifying a single line to reference them.

Two Terraform templates are used to configure the virtual network. One of these templates defines how the host appears to the external network. It defines the IP address and external network, allowing Docker to expose ports publicly, as well as an internal network for hosting potentially vulnerable containers. Secondly, at least a single container Terraform file must be copied, which in the case of most EDURange exercises is the file "nat.tf.json". All of this can and will be automated. This file provisions a container with a basic SSH server running. The container is connected to both the external and internal networks, and Terraform will automatically add any of the students' user accounts to it, as well as any additional scripts listed in the JSON User Files. For all of the base Docker containers, the OpenSSH package must be installed because Terraform uses it to install files. For most of the exercises, SSH is also used by students to interact with the container. In one of the new exercises, VNC is used for student interaction.

At this point, with a new folder created and templates copied, contributors can make a choice of how to proceed based on the requirements of their scenario. If their scenario does not require the installation of new software or specialized containers beyond the capabilities of the base SSH server, then they can write bash scripts and list files in the JSON description file as their only means of customizing the scenario. On the other hand, if they need containers that are running databases, web servers, or other complex applications, then they can choose to build a Docker image that fulfills their requirements and list that image in the Terraform file instead of writing any configuration scripts.

With those steps done, the scenario would be ready to be tested. In the remainder of this paper, we describe the experience of a different author in creating a ransomware exercise.

5 Developing a Ransomware Exercise

In the midst of ever increasing incidents that are caused by ransomware attacks around the world, it is critical for students who are learning about cybersecurity to understand that a ransomware attack is based on asymmetric key

encryption. This exercise mimics the execution of a ransomware attack. The goals are for students to learn how an adversary can weaponize public key cryptography and how that can be deployed on a vulnerable system. The newly added ransomware exercise introduces students to the foundations of ransomware and asymmetric key encryption. Through this scenario, the students learn how asymmetric key pairs can be weaponized and how end users can potentially stop such an attack by interrupting the corresponding cyber kill-chain.

5.1 Converting an Existing Exercise with Novel Requirements

This is an example of converting an exercise that was developed independently and then ported to EDURange. The first version of the exercise was developed on Windows [3]. It consisted mainly of a collection of Python scripts that installed a key pair, encrypted files, popped up some windows, and then decrypted the files if the user complied with some file modifications. The author developed this into an exercise with learning objectives and tested it on a Docker container for Windows. The last step was to integrate the container into EDURange.

We made some adjustments and converted the script into a Linux compatible program to ensure that it can be deployed within EDURange. We also adapted an existing ubuntu-VNC desktop docker container [2] to make the experience more realistic while providing the users a visual effect as the program gets executed. This was a significant extension because the previous exercises had only used the command line interface with SSH, so this involved a major change to exercise structure. The authoring process took about two weeks (part-time) starting from the Python scripts for Windows.

6 Results

The development of the SQL injection exercise was spread over 1-2 months, including 1 month for developing the infrastructure. At the end of that time, it was used in the classroom. Developing the exercise only required about 150 lines of PHP and HTML code, and about 250 lines of Terraform templates, mostly copied. The exercise process was very flexible and iterative because Terraform keeps track of interconnected components.

The development of the Ransomware exercise was even more rapid (two weeks), but it has not yet been tested in the classroom. Most importantly, the ease of importing a unique and pre-existing exercise to our platform illustrates the potential for adding many more exercises that are not just based on the current ones.

Both new exercises introduced completely novel interfaces for student interaction - a web application in the case of SQL Injection and a VNC desktop in the case of the Ransomware. In both cases, the authors had access to EDURange developers and were able to ask questions, but now this expands the range of topics that can be taught.

7 Conclusion and Future Work

Two new exercises were developed rapidly by people who were not familiar with the EDURange framework, which demonstrates that the framework has the flexibility for instructors to create new exercises. In one case, they were able to learn enough about the framework and develop an exercise in a matter of weeks. In the other case, it took a little over a month. In both cases, the authors had some specific learning goals in mind: in one case, teaching SQL injection, in the other, teaching how ransomware works and how it connects to modern cryptography. They differed in terms of the starting point. In one case, there was already a script that could be used on Windows, and that had to be translated from Windows to Linux and then integrated into EDURange. In the other case, everything had to be created from scratch. Potentially most cases will fall in between these two.

We expect that many instructors who teach cybersecurity have some tools that they often use and are familiar with. In that case, finding or constructing Docker containers with those tools and targets would be relatively fast, even more so if Linux is already being used. If the instructors are developing something new, provided the scope is reasonable, they should still be able to develop something in less than one term and have it ready for the next one.

This process has uncovered several new features that we want to add to EDURange. We plan to automate all of file copying and editing described in Section 3. Beyond that, we would create a GUI that could run those scripts. In addition, the Ransomware author has thought of a new exercise to be added. The exercise was inspired by the challenge-based learning pedagogy where an improperly configured Linux image and applications will be presented to the students. By mitigating the challenges or adjusting the configuration of the image, the student will receive flags to enter into the forms within EDURange for a score. This exercise will add system security and proper privilege configuration of users and the file system infrastructure to our list of topics. In addition to the image, a descriptive list of exercise objectives will be provided to the students as guiding reference so that they are properly informed of what the ideal configuration should be.

References

[1] Wenliang Du. Seed labs: Using hands-on lab exercises for computer security education (abstract only). In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, page 704, New York, NY, USA, 2015. Association for Computing Machinery.

[2] github.com/fcwu. Docker-ubuntu-vnc-desktop. `https://github.com/fcwu/docker-ubuntu-vnc-desktop`, 2021.

[3] github.com/ncorbuk. Python-ransomware. `https://github.com/ncorbuk/Python-Ransomware`, 2021.

[4] Jelena Mirkovic and Terry Benzel. Teaching cybersecurity with DeterLab. *IEEE Security & Privacy*, 10(1):73–76, 2012.

[5] Ambareen Siraj and Sheikh Ghafoor. Crest-security knitting kit: Readily available teaching resources to integrate security topics into traditional cs courses (abstract only). In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 1058, New York, NY, USA, 2018. Association for Computing Machinery.

[6] Blair Taylor and Siddharth Kaza. Security injections@towson: Integrating secure coding into introductory computer science courses. *ACM Trans. Comput. Educ.*, 16(4), June 2016.

[7] Michael F. Thompson and Cynthia E. Irvine. Individualizing cybersecurity lab exercises with labtainers. *IEEE Security & Privacy*, 16(2):91–95, 2018.

[8] Jan Vykopal, Radek Ošlejšek, Pavel Čeleda, Martin Vizváry, and Daniel Tovarňák. Kypo cyber range: Design and use cases. In Cardoso J., Cardoso J., Maciaszek L., Maciaszek L., van Sinderen M., and Cabello E., editors, *Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT*, pages 310–321, Madrid, Spain, 2017. SciTePress.

[9] R. Weiss, F. Turbak, J. Mache, and M. E. Locasto. Cybersecurity education and assessment in EDURange. *IEEE Security & Privacy*, 15(3):90–95, 2017.

[10] Grant Wiggins and Jay McTighe. What is backward design. *Understanding by design*, 1:7–19, 1998.