2019 IEEE European Symposium on Security and Privacy Workshops

Realistic Cybersecurity Training via Scenario Progression Management

Razvan Beuran, Takuya Inoue, Yasuo Tan, Yoichi Shinoda
Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, Japan

***Abstract*—Cybersecurity training activities are being conducted worldwide in order to address the increase in security threats that is plaguing our network-centric society. However, most of these activities take place in restricted environments, such as military organizations, or require paying high training fees. In this paper we present an open-source architecture that makes it possible to easily conduct cybersecurity training, thus lowering the barrier to entry for a large number of students and IT professionals to benefit from such activities.**

The core element of our architecture is a set of mechanisms for automating the scenario progression management, thus enabling activities that combine attack, forensic and defense training in realistic scenarios. We present the design and implementation of the progression management module, and its interactions with the overall training framework into which it was integrated. Our evaluation of the architecture from both functionality and performance perspectives demonstrates that it meets the requirements for realistic training activities, and that progression management introduces only a low execution overhead.

***Index Terms*—cybersecurity training, hands-on training, scenario progression management, cyber range**

I. Introduction

Given that cybersecurity attacks are becoming an increasingly severe threat in the modern always-connected society, and that there is a widely-recognized lack of professionals with adequate skills for fighting this menace, cybersecurity training programs have proliferated significantly in recent years.

Cybersecurity training activities were initiated and are still mainly conducted by military and government organizations, taking place in environments called *cyber ranges*—a phrase modeled on the term "shooting range"—which are network environments created specifically for cyberspace security training purposes. Over the years, the span of available programs has, however, increased significantly. Thus, training activities now range from free-to-attend competitions—often in the form of Capture The Flag (CTF) events, such as the DEFCON convention [1]—to thorough education and training paid programs that cover a wide range of topics and include various certifications, e.g., the SANS NetWars training courses organized by the SANS Institute in the US [2].

Such public programs have to use different approaches to training, as they address target groups with different age, experience and motivation characteristics. On the one hand, CTF events are mainly intended for young participants, and try to attract and motivate them through gamification techniques. Paid training courses like SANS NetWars are, on the other hand, targeting professionals that need to acquire new skills and improve existing ones in order to be able to deal with the evolving security challenges at their workplace.

What existing training programs have in common is that the barriers to entry are relatively high, either in terms of prerequisite skills for CTF competitions—for which candidates need to go through a series of qualifying events—or in monetary terms, since training costs for paid programs are in the order of thousands of US dollars per participant/course.

Various online training programs and web sites have emerged as an attempt to lower the entry barrier to cybersecurity training, but due to logistic issues the realism of such programs is relatively low, as cyber range management is a difficult and resource-consuming task. Realism is also an issue for many of the easier-to-access training programs mentioned already. CTFs, for instance, can only address skills limited to particular security issues (such as binary analysis, cryptography, web exploits, etc.); thus, they cannot cover the *end-to-end* procedures that need to be mastered in order to successfully handle real-world security incidents.

Our research focuses on an architecture that makes possible security training activities that combine attack, forensic and defense training following realistic scenarios. The architecture is built on top of an open-source cybersecurity training framework, thus ensuring that the provided functionality is readily available to interested parties at no cost. The novel element of our architecture is represented by a set of mechanisms for automating the progression management of the training session, which enables the system to carry out various actions in the cyber range depending on each trainee's activity.

The main contributions of this paper are:

- Discuss general requirements for realistic cybersecurity hands-on training activities.
- Introduce an architecture designed to meet these requirements via scenario progression management mechanisms.
- Present the implementation of the automated progression management component, and its integration into the overall architecture.

The remainder of the paper is structured as follows. In Section II we provide an overview of cybersecurity training, and define requirements for realistic training activities. Then, in Section III, we discuss the framework on top of which our architecture is built, and the proposed architecture itself. This is followed by a detailed presentation of the scenario

This work was supported by JSPS KAKENHI Grant Number 17K0047.

DOI 10.1109/EuroSPW.2019.00014

progression management module that is the novel element of this architecture (Section IV), and of the manner in which the module is integrated with the framework (Section V). Our system is evaluated from several perspectives in Section VI, and a discussion of related work is given in Section VII. The paper ends with conclusions and a list of references.

II. Cybersecurity Training

In this section we discuss the overall characteristics of cybersecurity training, and map these characteristics into requirements for realistic training activities.

A. Training Activity Characteristics

As mentioned in [3], cybersecurity training activities have three main components:

- *Attack training:* Let trainees experience vulnerability exploitation techniques, including activities that make use of the same tools and methodologies employed by real attackers, such as penetration testing.
- *Forensic training:* Provide a deeper understanding of the mechanisms and effects of cybersecurity attacks through various analysis techniques for system logs, network traffic, and so on.
- *Defense training:* Focus on the design and implementation of vulnerability protection mechanisms aimed at strengthening computer system security.

Most of the training programs available for the general public focus only on one or at most two of these components. We illustrate this issue below with some relevant examples from the Japanese cybersecurity ecosystem:

a) SECCON (SECurity CONtest) [4]: CTF competition held in Japan that serves as qualifying stage for the DEFCON CTF. Given the use of the CTF format, SECCON exercises can be categorized into either attack or forensic training, and in most cases deal with a single issue, such as a certain type of web vulnerability (e.g., SQL injection, path traversal, and so on), a binary analysis technique (e.g., disassembly), etc. Rankings and other gamification techniques are used to motivate participants when competing with each other.

b) CYDER (CYber Defense Exercise with Recurrence) [5]: Training program initiated in 2013 by the Ministry of Internal Affairs and Communications in Japan, and currently managed by the National Institute of Information and Communication Technologies. The program aims to improve the competence in dealing with cyberattacks of IT and security-related personnel of central and local government offices, independent administrative agencies, as well as large companies. As such, CYDER focuses mainly on forensic techniques, by tasking participants to analyze log files, disk images, processes, etc. CYDER also addresses necessary abilities related to incident reporting and investigation outsourcing.

c) Hardening Project [6]: Two-day training contest organized by the Web Application Security (WAS) Forum in Japan starting from 2012. Participants are divided by organizers into teams before the competition, based on their self-declared skills. The teams compete in terms of the security hardening they can provide to a virtual e-commerce web site created for the purpose of the event. The winning team is decided based on the amount of virtual sales their web site generated during the duration of the competition, used as an objective measure of the overall effectiveness of the hardening. The focus of the event is on maximizing the strength of the defensive cybersecurity techniques of the participants in realistic settings, with attacks being conducted live by security experts who monitor each team's progress, including via cameras at the venue.

The fact that publicly-available training programs only focus on limited roles for the participants, and have a reduced realism—a situation that is not at all specific to Japan—is caused by the following factors:

- Attack training requires actual environments to be prepared in advance, which is a challenging task. Moreover, even if such environments are made available, most often there is no active defense of the target system, which reduces the training to static situations (e.g., CTFs). We note that active defense is sometimes "organized" by pitting trainees against each other, as for blue-team vs. red-team or king-of-the-hill type of contests.
- Forensic activities are the easiest to organize because, at a minimum, one only has to supply participants with the relevant files (e.g., for cryptography or binary analysis exercises). For more complex tasks, such as log analysis, actual environments could be created, but this kind of training too is mainly limited to static situations.
- Defense training is the most difficult to organize, since it requires active attack mechanisms the trainees must defend their systems from. Typically, this is achieved by employing king-of-the-hill scenarios, although realism in this case is debatable, since the attackers are themselves trainees, hence have potentially limited skills. Only occasionally, as in the case of the Hardening Project mentioned above, security experts are entrusted with this task—an approach that doesn't scale well.

B. Realistic Training Requirements

Given the above considerations, we propose the following requirements that should be met in order to conduct training activities that prepare participants for real-world situations:

1) The training activity should combine all three aspects of cybersecurity training: attack, forensics, and defense.
2) The system should actively "respond" to trainees' actions, e.g., via appropriate attacks for defense training, and via suitable defense tactics for attack training.
3) Training instructors should be able to fully control the hands-on activities, both in terms of training content and scenario reproducibility.
4) There should be a low entry barrier for participating to training, so as to improve its effectiveness and reach.

Requirement #1 is derived from the fact that the three aspects of security training are interdependent, as shown in Figure 1. Understanding attack techniques helps trainees learn how to conduct forensics, and both of these help them

prepare better defense mechanisms, which in its turn leads participants to develop improved attack techniques, and so on. Consequently, only by mastering all the three security aspects will participants be able to gain the necessary readiness for handling in a pro-active, efficient and timely manner the security incidents they will be confronted with in real life.

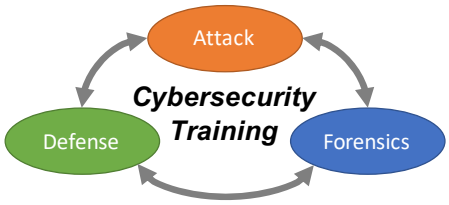

Fig. 1. Interdependence of the three cybersecurity training aspects.

Requirement #2 is justified by the fact that real systems are never static, and one can never assume that a system under attack is not defended, nor make assumptions about how a defended system will be attacked. Forensic investigations too may need to deal with systems in which logs change and network traffic is flowing in real time, as would happen in a regular production environment.

Requirement #3 is mainly related to the educational aspect of the training, as instructors need to be able to control the training content in order to oversee the learning process and evaluate the effectiveness of the training. In addition, scenario reproducibility is important for making sure that the training is both repeatable and fair for all trainees—which also contributes to its effectiveness.

Requirement #4 addresses the urgent need to increase the number of security specialists. Although it can be considered optional in a certain sense, our belief is that without giving all those interested a chance to undertake training, it will be difficult to compensate the lack of security professionals in a reasonable time span, thus risking more and more serious cyber attack consequences.

III. Architecture Overview

In this section we provide an overview of the open-source framework on top of which our architecture is built, and introduce the proposed architecture for realistic cybersecurity training that meets the requirements discussed above.

A. CyTrONE Training Framework

CyTrONE is an integrated cybersecurity training framework developed by Japan Advanced Institute of Science and Technology (JAIST), and was publicly released as open source in 2017 [7]. We consider that this framework provides all the basic functionality needed for security training, therefore we decided to build our architecture on top of it. The most important features of CyTrONE are (cf. Figure 2):

- Use text-based descriptions for the training content, so that it can be easily updated and improved by instructors; the content descriptions are stored in a training database together with other necessary resources for creating the training environment.
- Manage the entire training activity, including in terms of displaying training content to trainees and creating the corresponding cyber range; for this purpose, the framework employs two support tools, CyLMS and CyRIS.
- Make training content available to participants through the integration with a Learning Management System (LMS), mediated via the module called CyLMS (Cybersecurity Training Support for LMS).
- Create the training environment associated with the above content using the resources stored in the database by employing the module called CyRIS (Cyber Range Instantiation System).
- Trainees can then log in to the LMS for retrieving details about the training activity, and access the corresponding cyber range to address the questions provided.

The above functionality is sufficient for basic training activities, such as forensics or simple attack training. However, since no security-related component of CyTrONE is active during the actual hands-on session, more complex attack/defense training is only possible through role playing by the various participants. As we have discussed already, this approach is not ideal in terms of the quality of the training, nor reproducible in terms of action sequence and timing.

To address these limitations, we propose to extend the CyTrONE architecture by using an additional module that actively manages the progression of the training activity by interacting with the existing framework components.

B. Proposed Architecture

The architecture that we propose as an extension of the CyTrONE framework is shown in Figure 2. The novel element is the scenario progression management module, nicknamed CyPROM, which integrates with CyTrONE and the other framework elements in order to manage the manner in which the training scenario advances depending on the actions and progress of the participants.

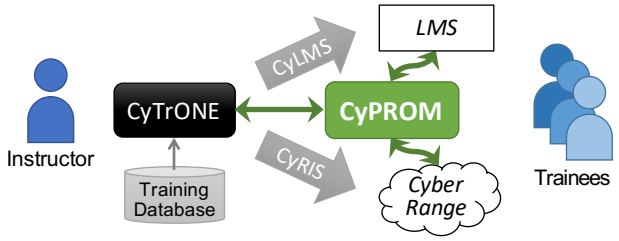

Fig. 2. Proposed architecture for realistic cybersecurity training via scenario progress management mechanisms.

The interactions between CyPROM and the other elements of the training environment can be summarized as follows:

a) CyTrONE: As overall manager of the training activity, the CyTrONE framework is in charge of starting CyPROM once the training session preparation is finished, more specifi-

cally after the training content is registered into the LMS, and the corresponding cyber range is created.

b) Cyber Range: Trainees access the cyber range during the hands-on activity, and the main function of CyPROM is to put into practice the training scenario by conducting various actions within the cyber range, while adjusting the scenario progression according to the effects of trainees' actions.

c) LMS: CyPROM is aware of the scenario progression for each participant, hence, it can provide additional information to them by employing the LMS as a user interface, e.g., to dynamically display messages regarding the scenario state, result of actions, etc. CyPROM can also get feedback from participants via the LMS, for instance by means of buttons, so as to allow for interactive training sessions.

IV. CyPROM Design & Implementation

In this section we discuss the overall design of CyPROM, and provide implementation details for each of its modules. Please refer to Figure 3 for an architecture overview. Note that a defense training prototype of CyPROM was presented in [8] (in Japanese) under the name DeTMan; the present paper introduces the generalized version of the tool and its internal modules, as well as the integration with CyTrONE.

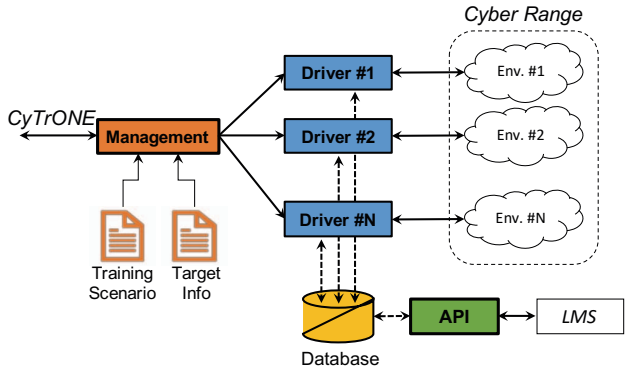

Fig. 3. Design of the CyPROM scenario progression management module.

A. Management Module

The management module is in charge of the basic functionality of CyPROM, such as reading and validating the input files, initializing the database, etc. Its most important function is to start, in parallel, one instance of the scenario driver module for each participant in the training. The parallel execution of the driver processes ensures that scenario progression takes place independently, according to the actions of each trainee and the state of their environment.

B. Training Scenario

The main input of the CyPROM management module is the actual training scenario. We conceived the training scenario as a set of steps; each step contains information about the action to be executed at that step, as well as various options regarding that action: what is the target machine, action-specific parameters, potential triggers for the action (e.g., timers), and so on. Each scenario step also includes information regarding execution branching, so that the scenario driver can decide what step should be executed next.

The training scenario description is provided as a YAML file, similar to how CyTrONE handles the training content description for the content to be displayed to trainees via the LMS, and the cyber range description needed to create the cyber range. YAML is a text-based representation that is both human and machine readable [9], therefore we consider it very suitable for our purposes, as it allows instructors to easily specify training scenarios via a regular text editor.

The structure of a training scenario file starts with the keyword `scenario`, followed by a list of step blocks. Each step uses specific keywords to denote its properties, such as `name` for the step label, `target` for the id of the machine that is to be the targeted by the action included in the current step, an optional `trigger` block for defining details about the action trigger, an `action` block to define the action details, and two branching elements, `success` and `failure`, for defining the labels of the steps that are to be taken in case action execution is successful or fails, respectively.

Due to space limitations we cannot provide here the full details on the scenario file syntax, but we will illustrate its use with an example. In Figure 4 we show the flowchart for a basic scenario that includes two attacks on a web site: (i) a command injection based exploit followed by an SSH dictionary attack and remote command execution; (ii) an exploit of a PHP authentication bypass vulnerability in WordPress v4.7.0 that is used for remote command execution. If the first attack steps are successful, that attack continues, otherwise the second attack is employed. The attacks are repeated until both of them fail—meaning that the trainee's defense tactic was valid, which is the condition for the training to finish.

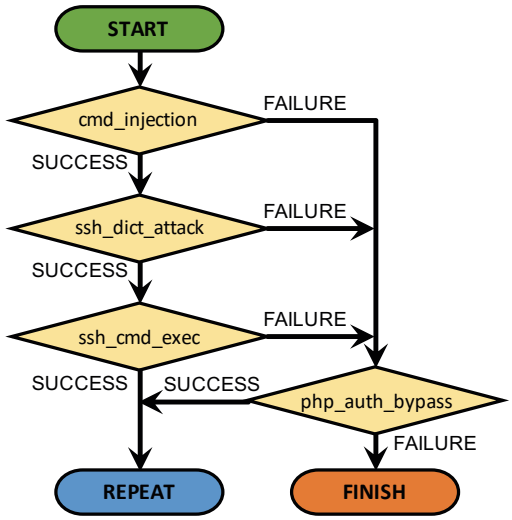

Fig. 4. Execution flowchart of a training scenario example.

The corresponding scenario description that needs to be created in order to represent this scenario for use in CyPROM is shown in Figure 5. Note how the target and trigger are set for the first step, labeled "OS command injection", followed by the configuration of the `cmd_injection` action. On failure, the step "WordPress v4.7.0 attack" will be executed, otherwise the scenario continues automatically to the next step, "SSH dictionary attack", and so on.

```
scenario:
 - step: OS command injection
   target: web-server1
   trigger:
     module: timer
     delay: 5
   action:
     module: cmd_injection
     path: /injection.php?cmd=<command>
     command: cat /etc/passwd
   failure: WordPress v4.7.0 attack
 - step: SSH dictionary attack
   target: web-server1
   action:
     module: ssh_dict_attack
     passwd: True
   failure: WordPress v4.7.0 attack
 - step: Command execution
   target: web-server1
   action:
     module: ssh_cmd_exec
     shell:
       - touch /var/www/html/wordpress/
index.html
       - echo "Hacked by CyPROM" > /var/www/
html/wordpress/index.html
     success: REPEAT
 - step: WordPress v4.7.0 attack
   target: web-server1
   action:
     module: php_auth_bypass
     content: Hacked by CyPROM
   success: REPEAT
   failure: FINISH
```

Fig. 5. CyPROM scenario description corresponding to the example flowchart.

C. Target Information

In order for it to be a generic representation, training scenario description doesn't contain network details about the action targets, which are represented by labels. Thus, network information must be provided each time CyPROM is executed, depending on the properties of the actual environment.

For this purpose, we use a separate training session dependent file that contains a section for each trainee or team by using the typical square bracket syntax of configuration files. Each participant section includes a sequence of target machine labels and the actual IP address those machines have in the training environment associated to that trainee. In Figure 6 we show an example of such a target information file. The sample file contains actual network details for an activity with three participants; two targets are defined per participant, in this case two web servers.

```
[Trainee #1]
web-server1 = 10.1.1.1
web-server2 = 10.1.1.2

[Trainee #2]
web-server1 = 10.2.1.1
web-server2 = 10.2.1.2

[Trainee #3]
web-server1 = 10.3.1.1
web-server2 = 10.3.1.2
```

Fig. 6. Sample target information file.

After the management module verifies both the training scenario and target information files, it creates a number of scenario driver instances equal to the number of participants specified in the latter. Each scenario driver then receives the name of the training scenario file, and network details about targets that apply to that particular instance.

We note that CyPROM also supports the CyRIS environment details output format to define the target information, as it will be explained in Section V-B. However, in order to ensure that CyPROM is not overly dependent on CyRIS, we have also introduced the specific format presented here.

D. Scenario Driver

The core functionality of CyPROM is provided via the scenario driver module, which uses the trigger-action-branching mechanism illustrated in Figure 7. This mechanism is applied repeatedly to decide when an action is to be executed, and what scenario step should be executed next. The scenario driver can also check the state of some of the services running on the target machines, and only run triggers and actions when those services are available. This is done to avoid undesired situations in which, for instance, an attack is performed while the service is not yet running because the machine has just been rebooted. In case the service is found not to be running, execution is paused and the check is repeated periodically.

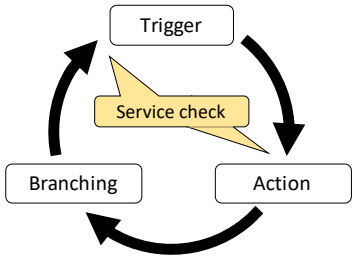

Fig. 7. The scenario progression mechanism at the core of CyPROM.

The workflow of the scenario driver is as follows. Initially, the step that is located top-most in the scenario description file is executed. If the step includes a trigger, such as a timer,

then the module waits for it to complete before executing the action specified in that step. Then, depending on the outcome of the action, the next step to be performed is selected from the other steps in the scenario via the branching mechanism, and the processing flow continues. Details about the three main elements of the workflow follow.

1) Triggers: CyPROM makes use of triggers to allow scenario progression to be delayed or interrupted, based on the needs and specificities for each type of training. Consequently, the two trigger modules made available so far are:

- `timer`: Delay action execution by a predefined amount of time—for example, to allow trainees to get their defense mechanisms ready—before carrying out an attack.
- `signal`: Prevent action execution until a notification is received; for this purpose, an HTTP socket is opened by the scenario driver, which listens on it until an appropriate message is received.

Timers allow actions to be executed at certain moments, in what we call *time-driven training*. On the other hand, signals make possible a more generic type of training that we name *event-driven training*. These two types of triggers make possible realistic activities in a wide range of conditions.

We note that the signal trigger is currently used to allow trainees to control scenario execution via buttons in a UI, but it is generic enough to make it possible to even employ external tools to unblock scenario progression.

2) Actions: Each scenario step contains an action, and creators need to specify the module to be executed for that action. We consider that these modules form collectively an "action library" that can be used as base for various scenarios. We have already implemented several such modules, and more actions are planned for the near future. Below are some examples of already-available modules:

- `message`, `hint`, `question`: UI-related modules that can be used to send messages or hints to trainees, and even ask for their input.
- `metasploit`: Interface for employing the Metasploit [10] penetration testing framework to conduct attacks on the target of the current step.
- `cmd_injection`: Perform command injection tasks on a given target web server.
- `ssh_dict_attack`: Conduct a simple dictionary attack on the SSH service running on a given target.
- `ssh_cmd_exec`: Remotely execute a command on a target machine via the SSH service.
- `php_auth_bypass`: Perform an attack that exploits a PHP authentication bypass vulnerability specific to WordPress v4.7.0 to alter the home page of a web site.

For each action, the user can specify some arguments specific to that action, such as the text to be sent by the module `message`. Moreover, in addition to returning the exit status of the underlying command, modules can also return specific data that becomes available to the next command. For instance, `ssh_dict_attack` will return a list of user names and passwords for which the login to a given target was successful, which could be used by the `ssh_cmd_exec` module to log in and execute a command on that target machine.

Actions executed by the scenario drivers target remotely the machines in the cyber range environment associated to each trainee, according to the network details provided in the target information file. No agent-like module is running on those machines, thus preventing trainees to interfere with scenario progression, either willingly or by mistake.

3) Branching: The scenario driver decides what step to execute next depending on whether the result of the current action was success or failure. This makes it easy for creators to design scenarios using flowcharts with two possible outcomes for each decision block. The reasoning behind such a binary decision becomes clear if we consider defense training. When an attack performed by CyPROM succeeds, it means that the trainees' defense tactic was not correct, and the system can continue with the next steps in that attack chain. However, if the attack fails, it means that some defense mechanisms are in place, and a completely different attack sequence must be used (or scenario execution can end).

A syntax extension to allow for more than two branches per decision block—for instance, by using program exit codes to make the decision—is straightforward, but the increased complexity may make scenarios difficult to design.

Each scenario step contains information about what step should be executed in case the included action was successful or failed by using the keywords `success` and `failure`, respectively. If an action return status occurs but the corresponding keyword is missing, then the subsequent step in the scenario file will be executed. Two special step labels are used to control scenario execution: (i) `REPEAT`, to indicate that scenario execution should be repeated from the beginning (e.g., to conduct again the same attacks in case a trainee's defense tactics failed), and (ii) `FINISH`, to specify that scenario execution should end, no matter what other steps appear in the scenario file.

E. Database

The database used by CyPROM serves to store data needed for several purposes, as follows: (i) UI interaction information; (ii) configuration settings; (iii) log of the actions performed by each scenario driver instance; (iv) information about the current step in a scenario; (v) service check information.

The database is initialized by the management module, and its tables are read and updated by the scenario driver instances and the API module. Our implementation uses the SQLite relational database management system for this purposes.

F. API

The API server is an optional module of CyPROM that makes it easy for the system to interact with external components. Currently, this module is mainly used to communicate with the user interface, in particular the LMS, in order to provide feedback to trainees via messages and hints, and to get input from them via buttons or input forms.

V. Framework Integration

In this section we discuss the integration of CyPROM with the architecture presented in Section III-B. Please refer to Figure 3 for an illustration of the relations between internal CyPROM modules and framework components.

A. Interaction with CyTrONE

Some extensions of the CyTrONE source code were necessary in order to integrate CyPROM with the framework. The modifications are as follows:

1) Once CyTrONE confirms that the training content was registered into the LMS and the cyber range was created, it also starts CyPROM.
2) Upon starting CyPROM, CyTrONE provides the training scenario description for that particular training session as parameter; this supplementary description file must be registered in advance in the CyTrONE training database, and minor modifications were needed to add support for such descriptions into the database.
3) Another parameter provided to CyPROM by CyTrONE is a file containing the network details for the cyber range created by CyRIS, such as IP addresses for the VMs. This file was already generated by CyRIS, and only needs to be made available to CyPROM at start.
4) Once CyPROM is started, its execution is independent from that of CyTrONE, and continues until the training scenario ends; optionally, CyPROM can inform CyTrONE about the completion of the scenario.

B. Cyber Range Access

During scenario progression, CyPROM needs to access the target machines in the cyber range environment for each participating trainee. In addition to the target information format presented in Section IV-C, CyPROM is also able to take as input the cyber range description file generated automatically by CyRIS after the range is created, as mentioned above. The location of this file is provided to CyPROM by CyTrONE, which also manages CyRIS execution, and CyPROM will pick the target information it needs from it.

The condition for this integration to work as presented here is to make sure that the guest names in CyRIS match the target labels in the CyPROM training scenario. For CyRIS, the guest names are included in the cyber range description file used to create the cyber range. The match can be realized simply by using the same name as target labels in the CyPROM training scenario description. The only difference is that CyRIS does not differentiate between multiple identical guests, whereas CyPROM has to make the difference; hence, a guest index needs to be added at the end of the target label. Thus, for a guest type named `desktop` in CyRIS, the target labels `desktop1` and `desktop2` must be used in CyPROM, assuming that there are two such guests in the cyber range.

We note that CyPROM only includes mechanisms for accessing the targets externally, without actually logging in to them. For defense training this is enough, since CyPROM only has to attempt attacks on those targets. For attack training, however, scenario actions may include tasks that are to be executed directly on the target hosts. The solution that we currently use for this purpose is to have special accounts dedicated to training purposes on such target machines; these accounts are not supposed to be modified by trainees, and allow CyPROM to use remote execution techniques (e.g., via SSH) to carry out actions on those machines.

C. LMS / Web UI

We use an HTTP-based API server to handle user interface operations in CyPROM, as discussed in Section IV-F, which is a flexible solution allowing us to integrate the scenario progression module with various other tools.

The LMS integration is currently implemented for Moodle [11]—the main LMS supported by CyTrONE—in a lightweight manner, as follows:

- The trigger mechanism in CyPROM, in particular the `signal` trigger mentioned in Section IV-D, is used to allow trainees to control the training session interactively via buttons, for instance, to start and resume it.
- Another integrated functionality is to display messages via the UI-related actions mentioned in the same section. This makes it possible to inform trainees of the scenario progression, give them hints, get answers to questions.

Figure 8 shows a screenshot of a training session as viewed via Moodle. The training session page has three tabs; the ones named "STATE" and "LOG" can be used to display public information, such as the scenario state and action logs for all trainees. The tab named "BOARD", which is pictured, is a message board page customized for each participant, showing information which depends on the scenario progression for that participant. As examples, the screenshot includes a status report about the attack outcome for a given trainee, and a question regarding the targeted vulnerability.

In the future we plan a more tight integration with Moodle via the PHP plugin functionality that this LMS supports. CyPROM could interact with such a custom plugin to provide specific security training functionality, and allow for interactive training sessions in a more flexible manner.

In addition to the LMS integration, we have also implemented a simple web UI that can be used instead of Moodle for a lighter installation, in case the full LMS features are not required. The functionality in this case includes displaying the state, log and message board pages, and can be used to conduct CTF-style training, for example.

VI. Evaluation

In this section we evaluate the proposed architecture from the point of view of the provided functionality, and assess the execution performance of the scenario progression module.

A. Functionality Assessment

We summarize in Table I the features of several free cybersecurity training systems and programs that are representative examples of current training approaches. For comparison purposes, our proposed architecture that integrates the CyPROM

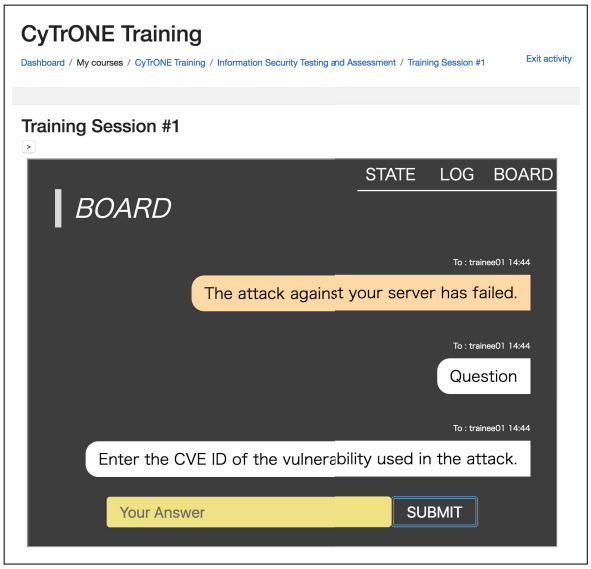

Fig. 8. Screenshot of the scenario progression user interface displayed within the Moodle LMS.

module with CyTrONE is denoted simply by "CyPROM", as we also compare it to the original CyTrONE framework.

The table is divided into two main sections, with the upper half discussing features related to attack, forensic and defense training. Thus, our proposed architecture can be used for all training aspects, and includes dynamic environment features, such as real-time changes and automatic attack/defense, in this way providing a clear advantage over the other systems.

The second half of the table addresses some more general features, such as training realism, flexibility, scalability, and so on. Due to the training characteristics discussed above, our architecture allows for realistic training, although programs such as CYDER and Hardening—in which staff help via role playing in conducting the training—are perhaps more realistic. Our system also allows for flexibility and interactivity when conducting the training, and due to its open-source nature the architecture is both scalable and extensible regarding the training environment. Furthermore, the automation mechanisms allow for control and reproducibility of the actions that take place during the training, making our system superior overall in comparison with existing approaches.

We also mention participation opportunities and educational aspects as built-in features of our proposed architecture that make possible unattended training activities at large scale, another important advantage compared to other systems.

B. Requirement Evaluation

The detailed feature assessment presented in the previous section will be used to determine how each of the evaluated systems addresses the requirements we formulated in Section II-B. This comparison is summarized in Table II, where again we denote our architecture by CyPROM.

We note that our proposed architecture is the only one that supports all three types of security training. As for reacting to trainees' activities, we emphasize that CYDER and Hardening can only achieve this by having staff and white-hat hackers play the active roles, whereas our architecture makes it possible to automate part of this process.

Regarding control and reproducibility, SECCON contests are done without any instructor involvement, hence they are controlled and reproducible, similar to CyTrONE; none of the other systems though extend these features to scenario actions, like attack and defense tactics, as it is done in our architecture. Finally, all systems have a low-to-medium barrier to entry, except for CYDER, which is dedicated to government employees and such. However, the only open-source systems that truly provide anytime/anywhere training opportunities are CyTrONE and our derived architecture.

C. Performance Assessment

The overall performance of CyTrONE was already analyzed in [12], therefore we focus here on the novel element in our proposed architecture, CyPROM. This component is relatively simple in terms of implementation, being composed mainly of text processing, command execution, and database operation tasks. We used a Supermicro SuperServer 5018D-FN4T server having an 8-core Intel Xeon CPU @ 2.10 GHz and 128 GB memory to assess CyPROM performance, with all the necessary tools and the cyber range being hosted on it.

In Table III we summarize the results of our measurements, calculated as averages over three experiment runs. The most time-consuming tasks are related to initialization, as shown in the upper part of the table. For the management module, this includes the time to initialize the database, process the input files (training scenario and target information), start the scenario driver processes, etc. For the driver modules, this overhead represents mainly the time to prepare the database for running the scenario. Note that these two initialization tasks only have to be carried out once in the beginning of the execution of CyPROM, and we consider that the total initialization overhead of about 83 ms is acceptably small.

The lower part of the table indicates the overhead for the tasks that are executed repeatedly for each step in a scenario, such as initialization, service check, trigger/action/branching execution, and database accesses. The total service check duration of under 5 ms shown in the table represents the case in which the checked service is running, since a timeout delay occurs if the service is not running (this delay is not very important, as in that case we pause execution anyway). The other larger overhead, again below 5 ms, was measured for the database access operations needed to update the tables related to scenario progression. The total overhead per scenario step is of about 10 ms, generally smaller than the actual execution time of actions. Hence, we conclude that the overhead introduced by the actual operation of CyPROM is negligible, and has no significant influence on scenario execution.

D. Discussion

Our architecture is mainly intended for stand-alone use as a cybersecurity training system, but other applications can be

TABLE I
FUNCTIONALITY ASSESSMENT

Program/System Features	SECCON	CYDER	Hardening	CyTrONE	*CyPROM*
Attack training capability	Yes	No	No	No	Yes
Automatic defense support	No	No	No	No	Yes
Forensic training capability	Yes	Yes	No	Yes	Yes
Real-time changes support	No	No	Yes	No	Yes
Defense training capability	No	No	Yes	No	Yes
Automatic attack support	No	No	No	No	Yes
Training realism	Low	High	High	Low	Medium
Flexibility & interactivity	High	Low	Medium	Low	High
Scalability & extensibility	High	Medium	Low	High	High
Control & reproducibility	High	High	Medium	Low	High
Participation opportunities	Medium	Low	Medium	High	High
Educational aspects	Medium	High	Low	Medium	High

TABLE II
REQUIREMENT EVALUATION

No.	Realistic Training Requirements	SECCON	CYDER	Hardening	CyTrONE	*CyPROM*
#1	Support attack/forensics/defense					○
#2	Respond to trainees' actions		○	○		○
#3	Provide control & reproducibility	○			○	○
#4	Ensure a low barrier to entry	○		○	○	○

TABLE III
PERFORMANCE ASSESSMENT

System Function	Overhead [ms]
Management module initialization	64.99
Scenario driver initialization	17.97
Step initialization	0.05
Service check (total, on success)	4.55
Trigger execution	0.03
Action execution	0.06
Branching decision	0.42
Database access	4.63

imagined, especially when considering the scenario progression module. For instance, one could implement scenarios that follow best-practice penetration test techniques, both to automate the procedure, but also for educational purposes.

Furthermore, we consider extending the functionality of the API, so that the training scenario can be modified dynamically, for instance, with steps being added and/or branching decisions being made externally, in real time. In this way, advanced technologies could be integrated, such as artificial intelligence, so that the enacted scenarios become more complex than what can be expressed statically in input files.

Although we have not done it yet, CyPROM could also be used for other purposes of realistic training environment emulation, such as for traffic generation. Thus, it would become possible to create background traffic in the cyber range as needed, so that participants perform the training in conditions similar to an actual live network environment.

VII. RELATED WORK

Related work for our paper can be split into three main categories, as follows.

1) Training programs: In addition to SECCON, CYDER and Hardening Project that were mentioned in Section II-A, there are several other training programs in Japan that are worth mentioning. Thus, Secure Eggs, offered by Nomura Research Institute (NRI) SecureTechnologies [13], is a suite of paid basic and specialized courses that take place over one or two days. Despite its technical content, the short duration makes it that Secure Eggs does not represent a thorough education and training program.

Another program worth mentioning in Japan is Micro Hardening [14], a stripped-down version of the Hardening competition. In Micro Hardening, the trainees' environments are repeatedly attacked automatically in exactly the same manner, in particular 3 or 4 times at 45 minute intervals. The idea is that trainees are thus able to learn from their mistakes, but this is a simple time-driven approach in which the same attack steps are repeated blindly, without considering trainees' activity. In contrast, our system takes this factor into account for action selection and execution timing.

Globally, SANS Institute that we have mentioned already, in addition to its on-site course suite, also provides an online course named "NetWars Continuous Online Skill-Sharpening Range" [15]. While this course is relatively thorough, barrier to entry is high because of its high cost; moreover, the only dynamic training provided is in king-of-the-hill form.

2) Training systems: Facebook CTF is an open-source platform that supports quiz, flag and king-of-the-hill types of CTF [16]. This platform does not provide any assistance with cyber range setup in the manner of CyTrONE, neither uses any scenario progression as we introduced via CyPROM.

In the academic world, Raj et al. propose the use of application containers as a solution for improving the scalability

of CTF competitions [17]. Their work focuses on deployment only, and doesn't address issues such as content creation and management, or training scenario progression.

There are also commercial systems for security training, such as the Boeing Cyber-Range-in-a-Box (CRIAB) [18]; however, such systems use proprietary technologies that create a vendor lock-in both in terms of software and hardware. Our open-source solution, on the other hand, allows to freely create training content, for instance by designing custom scenarios that best suit a certain target audience.

3) Management tools: Although not directly related to security training, there are tools used to configure network systems similarly to how CyTrONE is operating. We can mention here cloud controllers, such as OpenStack [19], and some more generic management tools, such as Ansible [20] and Chef [21]. Although our architecture shares some environment creation features with them, the overall training functionality, including the scenario progression mechanisms described in this paper are an important differentiator.

VIII. Conclusion

In this paper we have presented an architecture for cybersecurity training that makes it possible to conduct realistic activities that combine attack, forensic and defense training in live cyber range environments. The architecture is built on top of the CyTrONE cybersecurity training framework, and was conceived as a solution to meet the training requirements that we have defined in the beginning of the paper.

The key new element of the proposed architecture is the scenario progression management module named CyPROM. This module was designed to take a training scenario description as input, and execute the steps in the scenario independently for each participant. Each step contains an action to be performed, and additional information about this action, such as when it should be triggered, and what step should be executed next depending on the outcome of the action; step execution is done according to a trigger-action-branching mechanism.

CyPROM was integrated with CyTrONE and the other training environment components, the Moodle LMS and the cyber range used for hands-on practice. We have evaluated the extended architecture from a functionality point of view, and demonstrated that it meets the defined training requirements, while providing significant advantages with respect to existing solutions. Thus, the proposed architecture is the only system to support all types of training in a controllable and reproducible way; furthermore, scenario progression is carried out in a manner that takes into account trainees' activity. We have also conducted a performance assessment of CyPROM, showing that the new module introduces an initial 83 ms overhead, and a 10 ms overhead per scenario step, both having only a minimal impact on overall execution performance.

As future work, we consider extending the existing CyPROM API, so as to make it possible to modify the training scenario dynamically. This would allow CyPROM to interface with external modules, such as AI technologies, to decide the particular actions that should be taken at any given time, depending on the current state of the scenario and associated training environment. The required actions are to be selected from an action library that will also be made available as open source on the GitHub site of CROND [7], together with CyPROM, CyTrONE and so on.

References

[1] "DEF CON Hacker Convention," https://www.defcon.org/.

[2] SANS Institute, "SANS NetWars Training Courses," https://www.sans.org/netwars/.

[3] R. Beuran, K. Chinen, Y. Tan, and Y. Shinoda, "Towards Effective Cybersecurity Education and Training," Japan Advanced Institute of Science and Technology (JAIST), Tech. Rep. IS-RR-2016-003, October 2016.

[4] Japan Network Security Association (JNSA), "Security Contest (SECCON)," (in Japanese), http://seccon.jp/.

[5] Ministry of Internal Affairs and Communications, Japan, "Cyber Defense Exercise with Recurrence (CYDER) Training Program Press Release," http://www.soumu.go.jp/main_sosiki/joho_tsusin/eng/Releases/Telecommunications/130925_02.html.

[6] Web Application Security Forum, "Hardening Project," (in Japanese), https://wasforum.jp/hardening-project/.

[7] Cyber Range Organization and Design (CROND), "GitHub Repositories," https://github.com/crond-jaist.

[8] T. Inoue and R. Beuran, "Proposal of scenario progress automation system for defense exercises in cybersecurity training (in Japanese)," in *Internet Conference (IC 2018)*, 2018.

[9] C. Evans, "The Official YAML Website," 2017, http://www.yaml.org/.

[10] Rapid7 LLC, "Metasploit: Penetration Testing Software," 2017, https://www.metasploit.com/.

[11] The Moodle Project, "Moodle – Open-source Learning Platform," 2017, https://moodle.org/.

[12] R. Beuran, C. Pham, D. Tang, K. Chinen, Y. Tan, and Y. Shinoda, "CyTrONE: An Integrated Cybersecurity Training Framework," in *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP 2017)*, 2017, pp. 157–166.

[13] Nomura Research Institute (NRI) Secure Technologies, "Secure Eggs Training Program," (in Japanese), http://www.nri-secure.co.jp/service/learning/secureeggs.html.

[14] H. Kawaguchi (organizer), "Micro Hardening," (in Japanese), https://microhardening.connpass.com/.

[15] SANS Institute, "NetWars Continuous Online Skill-Sharpening Range," https://www.sans.org/netwars/continuous.

[16] Facebook, Inc., "Platform to host Capture the Flag competitions," 2017, https://github.com/facebook/fbctf/.

[17] A. S. Raj, B. Alangot, S. Prabhu, and K. Achuthan, "Scalable and Lightweight CTF Infrastructures Using Application Containers," in *Proceedings of the 2016 USENIX Workshop on Advances in Security Education (ASE '16)*, 2016.

[18] Boeing, Inc., "Cybersecurity & Information Management," 2017, http://www.boeing.com/defense/cybersecurity-information-management/.

[19] The OpenStack Foundation, "OpenStack – Open Source Cloud Computing Software," 2017, https://www.openstack.org/.

[20] Red Hat, Inc., "Ansible is Simple IT Automation," 2017, https://www.ansible.com/.

[21] Chef Software, Inc., "Chef – Automate Your Infrastructure," 2017, https://www.chef.io/chef/.