EDURange: A Cloud Based Framework for Teaching Cybersecurity Analysis Skills

Anonymous Submission #20 (USENIX CSET 2014)

Abstract

This paper reports on the design and implementation of the EDURange framework, a cloud–based resource for hosting on-demand interactive cybersecurity scenarios. EDURange is designed especially for the needs of teaching faculty. The scenarios we have implemented each are designed specifically to nurture the development of *analysis skills* in students as a complement to both theoretical security concepts and specific software tools.

EDURange has two features that make it unique compared to existing cybersecurity education infrastructure. First, EDURange is scalable because it is hosted on a commercial, large-scale cloud environment. Second, EDURange supplies instructors with the ability to dynamically change the parameters and characteristics of exercises so they can be replayed and adapted to multiple classes. EDURange has been used successfully in classes and workshops for students and faculty. We present our experiences building and testing the system.

1 Introduction

According to published reports by the SANS Institute and other groups [8], the US faces a major shortage of cybersecurity workers to defend our information infrastructure from attack. In recognition of this need, security has been included as a core topic in the new ACM/IEEE Computer Science 2013 Curricula [13]. Cybersecurity is also mentioned in more than half of the other knowledge areas in this report. At educational conferences such as SIGCSE and regional CCSC conferences, we are also seeing a growing interest in cybersecurity among faculty who do not have expertise in this area. Given the tight constraints of the Computer Science curriculum, most schools do not have the luxury of offering a separate class in cybersecurity. Thus, the first step is to integrate it into other classes both at the upper and lower division levels. We are seeing a growing concensus articulated in the CS 2013 Curricula and in the ACM report "Toward Curricular Guidelines for Cybersecurity" [11] to fully integrate cybersecurity into the Computer Science curriculum at multiple levels in multiple courses.

One of the major obstacles to integrating cybersecurity into the curriculum is the amount of work required to create and set up new hands-on exercises that can be easily adapted to any specific course. Few two-and four-year colleges have the facilities to set up their own hardware cluster dedicated to a security lab. In addition, we wanted hands-on exercises that teach analysis skills. For us, there was a gap between what we wanted and what we could access, so we decided to build our own tool. The criteria we used were:

1. *Flexibility* to specify exercises at a high level and create variations. DETER, The RAVE, and SEED provide sets of exercises. Many of them were good but not exactly what we wanted, and they were not easily modified. A big threat to those approaches to creating *long-term teaching tools* is that exercises become stale and answers become easy to Google.
2. *Ease-of-use for faculty*, which includes providing easy access to exercises, making them easy to create (not requiring configuration of VMs manually), and parameterizing exercises so that faculty can select the level of complexity that matches the level of their class.
3. *Educational goals*: we wanted to implement scenarios that would teach analysis skills, the *security mindset,* and address the CS2013 guidelines. The security mindset is the ability to think about how systems can fail, and be made to fail in different ways. This also extends to questioning assumptions and think analytically about their implications.

While there are large scale computing testbeds and facilities (DETER, The RAVE) that have been funded to meet the need of widely available hosted virtual machines for security teaching and research, they do not meet the need for flexibly creating new exercises, and they have significant limitations with respect to elasticity and scalability, particularly during busy times of the semester. While EDURange has significant advantages over existing infrastructure, it is not intended to supplant or replace such environments.

Rather, EDURange has been designed as a flexible complement to these facilities. EDURange provides a framework to support exercises in an elastic cloud environment. With EDURange, it is easy to modify an existing exercise so that students can repeat it multiple times, and the instructor does not need to worry about solutions being posted. It is easy to vary the difficulty of an existing exercise and update software versions by making small changes, and variations can be created for different courses.

The design and implementation of resources, frameworks, and exercises that support cybersecurity education is an active and lively area of work. A significant community of researchers and instructors from a variety of institutions have contributed everything from curriculum material to instructor/faculty training to VM-based labs to this landscape. Yet, we found as a practical matter that most deployed exercises and hosted environments have several shortcomings that make them difficult to leverage in our classrooms [15]. Towson's "Security Injections"[1] mainly focus on several important secure programming patterns. While the SEED [6] material presents a mature, well-documented set of exercises, they are not typically interactive or dynamic and they require faculty to set up and run them.

One of our primary motivations is to create exercises that would nurture analysis skills. When speaking of analysis skills, we largely mean the ability to reason about large, complex, and opaque data and systems. Strong analytical skills enable people to impose structure and meaning on such artifacts, reason about these relationships, and draw meaningful conclusions or inferences. These are precisely the kinds of skills that we believe are useful in many cybersecurity scenarios from security policy design to reverse engineering to vulnerability analysis. In designing EDURange exercises, we focus on the following list of analysis skills, and we welcome other suggestions.

- **Verify assumptions.** Checking network messages, protocols, file formats and other input data constraints to see if layers of abstraction are coherent and correct. Enumerating and checking if failure modes, exceptions, and errors are controlled, caught or anticipated.
- **Gaining understanding** of program, network, or system behavior and semantics, network topology or organization, or a defense posture. Observing and enumerating how software components or network elements are actually composed.
- **Extracting Information** from opaque artifacts. For example, analyzing a raw dump of network traffic or intrusion alerts or firewall logs.
- **Creating Emergent Resilience** Understanding a system well enough to design and propose ehancements to reliability, fault tolerance, or availability.
- **Create Deception** (or confusion) for an adversary by creating artificial diversity and applying randomization to selectively increase complexity.

[1]cis1.towson.edu/ cssecinj/

In the next section, we describe related work and how EDURange extends previous work on hands-on exercises. Note that this paper is about our design and decision process for creating scenarios, rather than the specific exercises that we have chosen. Section 3 describes the most interesting features of the EDURange infrastructure. We discuss the design of our current set of exercises in Section 3.1. Section 4 describes our experience creating the framework and delivering the exercises to a variety of audiences. The main philosophical takeaway (if a reader wanted to do this in another context) is the pattern for creating and considering the pros and cons of the types of scenarios. Section 5 contains a discussion of lessons learned and considers the mapping from the CS2013 curricula Security Knowledge Area to exercises and how they support pedagogical outcomes.

2 Related Work

Our philosphy on information security education stems from our understanding and teaching of the hacker curriculum as described by Bratus [1]. This approach is predicated on the utility of understanding failure modes. Rather than teaching students the "success" cases, we attempt to deliver a culture shock that makes them disrespect API boundaries and adopt a cross-layer view of the CS discipline as described by Bratus et al. [3]. We routinely encourage our students to adopt a dual frame of mind when solving problems to prevent artificial abstraction layers from becoming boundaries of competence [16]. The importance of analysis skills as explained by S. Bratus et al. [2] is based on linking expected behavior to actual behavior as seen in network traces, log files, program binaries, rules/policies, system call traces, network topologies, network interactions, unknown protocols, injected backdoor code, etc. NetCheck [17] is a tool that facilitates this type of analysis to debug network applications. Using a simplified model of normal network behavior, NetCheck collects information about network applications using strace.

Our work follows the tradition of creating cybersecurity games or exercises because they have applications to learning and assessment, and they are fun. This includes competitions such as CCDC[2], Plaid[3], notsosecure[4], iCTF[5] [5], CSAW[6] [9], TRACER FIRE[7], Packetwars[8], and many others. From our perspective, the problem with these competitions is that they require a significant amount of preparation for the organizers. For example, it took several grad students six months to create the exercises for iCTF [5]. Some competitions such as CCDC and Packetwars require the installation of physical hardware, and they require that students and their faculty travel to participate. There are also a number of non-technical games with the goal of interesting students with no technical background in cybersecurity. These include Control-Alt-Hack, [d0x3d!] [10] and Werewolves [7]. The last of these introduces players to the concept of covert channels in a non-technical context. Our exercises are intended to create scenarios that are closer to real-world situations.

EDURange is not the only framework for creating cybersecurity exercises. Two others are DETERlab [12] and The RAVE. As mentioned before, their limitations are lack of flexibility and scalability. The National Cyber Range is also of note, but its primary use is as a secure testbed for research. The Seattle Testbed[9] is a research environment with several security exercises including one on reference monitors [4].

2.1 Hands-on Cybersecurity Exercises

Judging from the Birds of a Feather sessions on Security at the last two years of SIGCSE, there has been a modest increase in the number of hands-on cybersecurity exercises, not all of which address the security mindset. It is also clear from the workshops we have attended that faculty want to use them but are frustrated because of the difficulty in disseminating and setting them up. The other frameworks that we have tried have some useful features that we can emulate and some limitations that we try to avoid.

With respect to our design goals, Table 1 shows the strengths and weaknesses of other existing cybersecurity labs, exercises, and curricula. EDURange addresses the primary weaknesses that are listed. The PacketWars[10] project is probably the closest existing piece of work to what EDURange proposes, and there has been some work on providing students with access to "live" exercises on a small scale [14].

[2]http://nationalccdc.org
[3]http://www.pwning.net
[4]http://ctf.notsosecure.com/
[5]http://ictf.cs.ucsb.edu/
[6]https://csaw.isis.poly.edu/
[7]http://csr.lanl.gov/tf/
[8]`http://packetwars.com/`
[9]seattle.poly.edu
[10]`http://packetwars.com`

3 Design of EDURange Framework

The main requirements for the framework are that it be easy for faculty and students to use, the exercises should teach analysis skills, it must be secure, and the exercises should address the learning goals of CS2013. The last requirement is addressed by scoring event elements in the YAML scripts. The use of YAML as an intermediate representation provides a concise way for an instructor to understand and modify exercises.

EDURange's security requirement is that a student should not be able to unintentionally or intentionally negatively impact resources outside of the EDURange framework. For example, in the Recon exercise this would include mapping a network outside of the battlespace. In addition, teams should not be able to access other teams' VMs, but their members should be able to collaborate.

Ease of use involves both accessibility and simplcity of the user interface. We address this accessibility requirement by deploying the EDURange framework on Amazon's AWS EC2 cloud. Students and faculty do not need to sign up in advance for resources. When they are ready to do an experiment or if they decide at the last minute to do a demo in class, the resources are always available. Students can work from anywhere — all they need is an SSH client. We decided against the alternative of running VMs on a local cluster because that would have been more expensive and would require work to maintain — skills and resources that our target audience may not have access to in any significant amount.

An important way in which we achieved flexiblity is through the use of tools such as Chef that install software in a programmed fashion. We didn't want to write separate scripts for each operating system on each VM. A Chef script can install packages for a wide range of operating systems. The base VMs don't change frequently, and the Chef scripts that install software can handle upgrades transparently.

Each EDURange exercise is specified by a YAML file. A small number of types of entities recur in all of our scenarios, so we have made them primitives: networks, instances (hardware, computers), software that is directly involved in the exercise, participants (users), groups (teams of users), artifacts (flags), and goals (scoring events). Some exercises clearly involve networks and subnets. For example, Recon 1 and Recon 2 have a subnet for the battlespace and a subnet for each team. The ELF Infection exercise also has a network topology with a subnet for each team and a subnet for the infected VM.

Table 1: *A Comparison of EDURange and Existing Projects.* EDURange focuses on developing cybersecurity analysis skills. This table is not a criticism of existing efforts, but rather meant to highlight the ways in which EDURange differs from the main characteristics of existing projects — note that these projects may have been built with different criteria in mind.

Project	Primary Weakness	Primary Strength
Cybersiege	shallow analysis	interactive training scenarios
SEED	lacks competitive interaction	comprehensive documentation
Security Injections	focus on defensive coding patterns	introduction to basic security
CCDC	requires travel; limited remote access	interactive and competitive
PacketWars	requires travel; limited availability	contains well-structured scenarios
ITSEED	minimal instructor support, distrib by flash drive	good documentation for students
Google Gruyere	narrow focus (web apps)	cloud-based; well-documented
Security Knitting Kit	not distributed	difficult to judge
The RAVE	not very flexible	cloud-based; existing lab manual
Seattle Testbed	limited in scope	easy-to-use; P2P; includes mobile devices
DETERlab	limited scalability	range of exercises

An "instance" is usually a VM with an operating system. The special software for Recon 1 includes nmap and tcpdump. The goals in this exercise would include the IP addesses of the instances in the battlespace.

Having a scenario description language has the following benefits: flexiblity: exercises can be modified to keep them exciting. In our pedagogical model, students repeat exercises. For example, with Recon 1, a student may try a set of options for nmap and discover that it takes too long. We want the student to have time to think about trying different options after experiencing the problem. Analysis takes time, which is a limitation of most competitions. They tend to reward speed and don't allow time for in-depth analysis. Repeating exercises is not viable if the network configuration is static.

3.1 Scenarios

The overriding requirement for each scenario was that it must support the development of analysis skills; in other words, the student must come away from the experience with not only an appreciation for the knowledge involved in the subject matter or a basic understanding of some of the tools used, but also with *insight* and a *logical approach* for understanding the conceptual issues at play. EDURange is a work in progress, and we have a number of exercises under development:

- **Recon 1** is about mapping a network and understanding network protocols, such as TCP, UDP, ICMP.
- **Recon 2** includes intrusion detection and prevention. The student trades off speed with stealth, the attacker must be able to map a network without triggering the defenses.
- **ELF Infection** is about forensics and reverse engineering. The student is given a VM, which has an infected utility. They must discover which utility is infected and what the malicious behavior is.
- **ScapyHunt** is a puzzle set in a software defined network. The puzzle involves finding data on a target host that is behind a gateway. It involves passively examining network traffic and crafting packets to reveal specific information.
- **Firewall** is about creating a set of rules to control traffic in and out of a network. More generally, it requires understanding how a complex set of rules implements an access conrol policy.
- **Fuzzing** In the simplest version, the defender is given the grammar for a calculator and must implement an interpreter for that grammar. The attacker tries to fuzz the interpreter to produce incorrect results or get it to reject a valid expression. Students interact with their peers by assuming different roles.
- **Process Records (strace)** involves deciphering what programs were running on a machine given a large trace of system calls (exeve(2) records are purposefully mangled). The simplest form of this exercise is to provide a complete short trace for a simple program (e.g., running date(1)). Variations involve larger traces, more programs, incomplete traces (sampling). The exercise demands that students begin to impose some modeling and structure on this large, overwhelming source of data, and it encourages them to read the documentation on a

number of system calls. Similar exercises could be based on log files.

4 Experience Generating and Disseminating Exercises

In this section, we discuss how we implemented the EDURange framework. EDURange has several components working at different levels. At the intermediate level, the YAML file containing primitives described previously is translated into API calls to AWS/EC2, which create the instances. For software installation, every instance created runs a startup script via cloud-init which installs all dependencies required for us to configure it, including Chef Solo. Additionally, each instance is sent a private AWS S3 URL which contains instance-specific Chef instructions, such as creating accounts for players or installing and configuring software for a scenario.

Sample YAML that can be used to modify the instance-specific chef code is included below. Roles contain both packages, which use chef to install a package from the package manager, as well as recipes which reference either EDURange or third party chef recipes. In a manner similar to CSS classes, after being defined roles are referenced by name within each instance declaration.

Chef has a wide variety of third party cookbooks which EDURange supports using, but EDURange also has its own library of recipes, created as needed to support our scenarios. The ELF Infection exercise mentioned earlier in this paper is simply implemented as a shell script within chef, as the code compiles using the standard make, make install process.

The final component of EDURange is the scoring engine. While we have not implemented this parsing side yet, we have designed a specification for defining goals in the YAML file. Each goal (associated with teams) may contain any combinations of triggers, actions, checks, and point changes in order to assess progress throughout the scenario, as well as over multiple playthroughs. Triggers define what event must occur in order for the goal to be tested. It could be once, 20 minutes into a game, or every minute. The action defines what is tested, using any scoring modules we make available such as testing a web sites status or checking that a port is open. The check field in a goal takes the result from the action and determines whether or not the point modification made, if specified, should be made. A sample goal is included below, although the syntax is likely to change.

The cost of running a two-day hackathon was only $28.

4.1 A Detailed View: The Recon Exercise

The recon exercise was inspired by a largely similar execise from Packetwars. We chose it as the first exercise to implement because it is relatively simple, we understood it well enough, everyone in the project had actually completed it a few times, and it would enable us to concentrate on building the EDURange infrastructure without having to worry too much about simultaneously specifying a “scenario-in-progress”. Supporting the Recon exercise would be an existence proof of EDURange’s viability as a teaching tool, and it would allow us to check that we had met our design requirements for the infrastructure itself. The Recon exercise amply exhibits the characteristics of the types of scenarios we are interesting in developing, deploying, and supporting in EDURange. In particular, this exercise demonstrates how each EDURange scenario we have designed and implemented can act as a jumping off point for further lessons and study. The setting can serve as a starting point for exploration of a range of topics. The recon exercise can be accomplished at a fairly basic level, but then provide the instructor with the opportunity to discuss or teach about a variety of networking and security concepts and ideas.

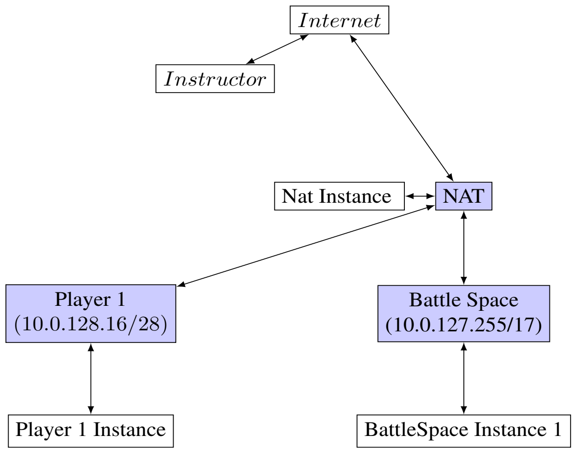

Figure 1: Conceptual diagram of the Recon I game. Note subnets are shaded (blue)

4.2 Feedback on Recon Exercise

In this section we provide a detailed review of the feedback and experiences of user in the Recon 1 exercise. We have now run the Recon exercise with 110 students in 6 different settings spanning four different institutions as shown in the table below. The first roll out of the Recon exercise was at the SISMAT 2013 program held at Dartmouth. Recon was one of 6 lab experiences that the students participated in during the 10 day workshop. Student feedback indicated that there were implementation issues (e.g. insufficient background information and instructions). However, student responses on a post-class survey revealed that students felt that the exercise was worthwhile and that it increased their interest in cybersecurity education (one-tailed t-test found ratings well above neutral $p < .01$).

Eleven Evergreen and Lewis & Clark College students along with four professors spent two days in August 2013 in an intensive hackathon that included 2 run throughs of the Recon 1 exercise, once with advanced students with previous computer security coursework and once with students with no prior computer security training. This workshop uncovered several issues with remote configuration and set up of the infrastructure, that were subsequently fixed. We also learned that some important prerequisite knowledge is needed in order for the exercise to be meaningful, including the OSI model for network layers We also uncovered unreasonably long latencies in configuring the exercise and in running some nmap commands that led to improvements to the instructions an changes in the scope of the network searches in the exercise.

The four remaining student deployments were all done in the context of undergraduate and graduate level courses on security at three different educational institutions. In each case the full exercise was successfully implemented based on improvements made from SISMAT and the Hackathon. In one of these courses (CS 495 at Lewis & Clark College) student surveys conducted at the end of the semester indicate that students found the Recon exercise worthwhile in their learning (M=5.25 on a 7 point likert scale, $p < .005$).

Along with testing the Recon 1 exercise with over 100 students, we also held workshops for 29 faculty at three different conferences. As you can see in the table below, we targeted a variety of institutions, including instructors at two year colleges, small liberal arts colleges and research universities. Our workshop attendees also differed widely in experience teaching Computer Security courses, some have no experience yet while others have taught several different courses at both undergraduate and graduate levels. At each conference, we made adjustments based on feedback and challenges experienced in the how to present the Recon 1 Scenario. We also gave post workshop surveys at each conference. At each conference participants felt that taking the workshop increased their interest in the topic of Cyber Security (all $ps < .02$).

4.3 A Detailed View: ScapyHunt

We presented a small class of graduate students with both the Recon and ScapyHunt exercises. This group of students felt that Recon was easier than ScapyHunt – in fact, they were unable to make much progress at all in the scenario (this was partly due to class dynamics: they did not function effectively as a team). Similar to students using other scenarios, they expressed a wish for a simple canned demo or hint to start off. The exercise itself is somewhat intimidating because it begins with a login prompt and little else beyond the directive "find the hidden resource in this hidden network topology." This style of play is similar to text adventure games.

The students required prompting during most of the interaction. For example, the students needed prompting to open a terminal and use standard network command line tools and utilities (e.g., Wireshark) to discover network information. They quickly fixated on nmap, although nmap is ultimately of little utility for this exercise. To us, this seemed like a side-effect of security training that is too heavily tool-oriented. We prompted students to both "write" to the network (via ping, nmap, netcat, and some packet-crafting tools) and simultaneously "read" from the network to observe both their own actions and the actions of the entities in the software defined network (SDN). They also did not use any notes, drawings, or aids — they tried to hold most of the information in their heads. This kind of mistake is unlikely to be repeated when they revisit a similar task because they saw how "big" the task was in terms of the amount of information generated by the tools they eventually used.

5 Discussion, Lessons Learned and Future Work

There is a demand for hands-on cybersecurity exercises and a framework for creating them. One of the most important lessons we learned from feedback from faculty is that they really want more exercises that they can use right now in their classrooms and they want it to be easy. They have expressed interest in EDURange, they want it now, and they want enough exercises to fill out a course. There is a wide range of faculty who want this. At our SIGCSE workshop, some felt that Recon 1 was too challenging, while others said it was too easy.

EDURange provides starting points for discussing important topics. For example, Recon 1 can be an opportunity to discuss the OSI model, subnet masking, broadcast addresses, even using the command line.

EDURange can be used by students for formative self-assessment. We learned that unlike with exams, where students do not want to admit what they don't know, with hands-on exercises such as Recon 1, students were able

Table 2: Student classes and workshops

Date	Site	Class	Number of Students
June 2013	Dartmouth	SISMAT	12
Aug 2013	Lewis & Clark	Hackthon	11
Nov 2013	Evergreen	Network Security	40
Feb 2014	Lewis & Clark	CS 495 Cbyersecurity	18
Feb 2014	Univ. of Calgary	CPSC 601 Seminar: Security Analysis	4
March 2014	Univ. of Calgary	CPSC 525 Network Security	25

Table 3: Faculty workshops

Date	Audience	Workshop	Number of Faculty
Oct 2013	Liberal Arts Colleges	CCSC-NW[11]	8
Jan 2014	2-year Colleges	MPICT[12]	7
March 2014	Broad Scope	SIGCSE[13]	14

to reflect on what they didn't know in the context of what they wished they had known when trying the exercise. Since Recon 1 is a very focused exercise, students were able to identify the need for more tutorials and canned demos on specific topics, such as TCP, ICMP and subnetworks. With exercises such as Recon 1, there is no way to fake the knowledge needed. On the other hand, we learned that the game must be appropriate for the audience. If students haven't studied networking and don't have experience with the command line, they will have difficulty and may get frustrated. We learned that some students may need videos and screen captures to help them get a foothold. We have started to create some tutorials and "level zero" versions.

6 Conclusion

EDURange provides a scalable, easy-to-use infrastructure to an audience of instructors that have few local resources or capacity to set up complex systems. By using a public industry "best of breed" cloud, EDURange is unique and cost effective, and avoids some of the limitations associated with dedicated testbeds. Many of our student and instructor audiences were positive about its potential.

While EDURange has significant advantages over existing infrastructure, it is not intended to supplant or replace such environments (RAVE, DETER). Our main focus is on providing dynamic, flexible cybersecurity scenarios that teach analysis skills (rather than toolsets or specific attacks). Our design criteria for exercises is stringent in that each exercise must nurture a student's ability to analyze large, opaque artifacts. We also identify how the exercise maps to the CS2013 curriculum knowledge areas as a service to the students and instructors. We suggest that EDURange's support for customizing scenarios represents a natural evoluation of cybersecurity education infrastructure.

References

[1] Sergey Bratus. What hackers learn that the rest of us don't: Notes on hacker curriculum. *IEEE Security and Privacy*, 5:72–75, 2007.

[2] Sergey Bratus, Nihal DCunha, Evan Sparks, and Sean W Smith. Toctou, traps, and trusted computing. In *Trusted Computing-Challenges and Applications*, pages 14–32. Springer, 2008.

[3] Sergey Bratus, Anna Shubina, and Michael E. Locasto. Teaching the principles of the hacker curriculum to undergraduates. In *Proceedings of the 41st ACM technical symposium on Computer science education*, SIGCSE '10, pages 122–126, New York, NY, USA, 2010. ACM.

[4] Justin Cappos and Richard Weiss. Teaching the security mindset with reference monitors. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 523–528, New York, NY, USA, 2014. ACM.

[5] Adam Doupé, Manuel Egele, Benjamin Caillat, Gianluca Stringhini, Gorkem Yakin, Ali Zand, Ludovico Cavedon, and Giovanni Vigna. Hit 'em

where it hurts: A live security exercise on cyber situational awareness. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, pages 51–61, New York, NY, USA, 2011. ACM.

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

[7] Roya Ensafi, Mike Jacobi, and Jedidiah R. Crandall. Students Who Don'T Understand Information Flow Should Be Eaten: An Experience Paper. In *Proceedings of the 5th USENIX conference on Cyber Security Experimentation and Test (CSET'12)*, pages 10–10, 2012.

[8] Z. FRYER-BIGGS. Dod faces cyber expert talent shortage. *Computer*, 33(12):52–59, 2000.

[9] Efstratios Gavas, Nasir Memon, and Douglas Britton. Winning cybersecurity one challenge at a time. *Security & Privacy, IEEE*, 10(4):75–79, 2012.

[10] Mark Gondree and Zachary N.J. Peterson. Valuing security by getting [d0x3d!]: Experiences with a network security board game. In *Presented as part of the 6th Workshop on Cyber Security Experimentation and Test*, Berkeley, CA, 2013. USENIX.

[11] Andrew McGettrick, Lillian N. Cassel, Melissa Dark, Elizabeth K. Hawthorne, and John Impagliazzo. Toward curricular guidelines for cybersecurity. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 81–82, New York, NY, USA, 2014. ACM.

[12] Peter AH Peterson and Peter L Reiher. Security exercises for the online classroom with deter. *Proc. of the 3rd USENIX CSET*, 2010.

[13] Mehran Sahami, Mark Guzdial, Andrew McGettrick, and Steve Roach. Setting the stage for computing curricula 2013: computer science–report from the acm/ieee-cs joint task force. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 161–162. ACM, 2011.

[14] G. Vigna. Teaching Network Security through Live Exercises. In *Proc. 3rd Ann. World Conf. Information Security Education (WISE 03)*, pages 3–18. Kluwer Academic, 2003.

[15] Richard Weiss, Jens Mache, and Erik Nilsen. Top 10 hands-on cybersecurity exercises. *Journal of Computing Sciences in Colleges*, 29(1):140–147, 2013.

[16] G. White and G. Nordstrom. Security across the Curriculum: Using Computer Security to Teach Computer Science Principles. In *Proceedings of the 19^{th} National Information Systems Security Conference*, pages 483–488. NIST, 1996.

[17] Yanyan Zhuang, Eleni Gessiou, Steven Portzer, Fraida Fund, Monzur Muhammad, Ivan Beschastnikh, and Justin Cappos. Netcheck: Network diagnoses from blackbox traces. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI14), USENIX.*