Evaluating Two Approaches to Assessing Student Progress in Cybersecurity Exercises

Valdemar Švábenský
Masaryk University
Czech Republic
svabensky@ics.muni.cz

Richard Weiss
The Evergreen State College
Washington, USA
weissr@evergreen.edu

Jack Cook
New York University
New York, USA
cookjackc@gmail.com

Jan Vykopal
Masaryk University
Czech Republic
vykopal@ics.muni.cz

Pavel Čeleda
Masaryk University
Czech Republic
celeda@ics.muni.cz

Jens Mache
Lewis & Clark College
Oregon, USA
jmache@lclark.edu

Radoslav Chudovský
Masaryk University
Czech Republic
chudovsky@mail.muni.cz

Ankur Chattopadhyay
Northern Kentucky University
Kentucky, USA
chattopada1@nku.edu

ABSTRACT

Cybersecurity students need to develop practical skills such as using command-line tools. Hands-on exercises are the most direct way to assess these skills, but assessing students' mastery is a challenging task for instructors. We aim to alleviate this issue by modeling and visualizing student progress automatically throughout the exercise. The progress is summarized by graph models based on the shell commands students typed to achieve discrete tasks within the exercise. We implemented two types of models and compared them using data from 46 students at two universities. To evaluate our models, we surveyed 22 experienced computing instructors and qualitatively analyzed their responses. The majority of instructors interpreted the graph models effectively and identified strengths, weaknesses, and assessment use cases for each model. Based on the evaluation, we provide recommendations to instructors and explain how our graph models innovate teaching and promote further research. The impact of this paper is threefold. First, it demonstrates how multiple institutions can collaborate to share approaches to modeling student progress in hands-on exercises. Second, our modeling techniques generalize to data from different environments to support student assessment, even outside the cybersecurity domain. Third, we share the acquired data and open-source software so that others can use the models in their classes or research.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Security and privacy**;

KEYWORDS

cybersecurity education, command-line history, educational data mining, learning analytics, assessment, modeling

ACM Reference Format:
Valdemar Švábenský, Richard Weiss, Jack Cook, Jan Vykopal, Pavel Čeleda, Jens Mache, Radoslav Chudovský, and Ankur Chattopadhyay. 2022. Evaluating Two Approaches to Assessing Student Progress in Cybersecurity Exercises. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499414

1 INTRODUCTION

Cybersecurity is an essential topic in the ACM/IEEE Computing Curricula 2020 [10]. However, it is challenging for students to learn since it encompasses skills from many areas of computing, such as programming, operating systems, and networking. To promote deep understanding, students must practice these skills hands-on.

Subsequent assessment of students' learning is vital [15]. However, the assessment of practical tasks is challenging for several reasons. If it is performed manually, it is time-consuming and can be inaccurate due to the quantity and complexity of student interaction data. If automated, it is often superficial, including only the information about whether the solution was correct or not [36].

To help instructors overcome this challenge, we propose and evaluate two methods for supporting semi-automated, timely, accurate, and in-depth assessment of students. The methods are based on visualizing command-line histories from solving cybersecurity tasks, resulting in graphical progress models. Instructors can use these models to better understand how their students learn. For example, they can compare the students' approaches to solving the tasks, along with the mistakes they made. Based on this understanding, the instructors can assess students in two ways [15]:

- *formatively* – providing feedback to students to support their learning, for example, correcting the struggling students, and
- *summatively* – grading students to evaluate their level of knowledge, for example, distinguishing advanced students and novices.

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA

This paper follows the multi-national, multi-institutional study framework, which addresses the limitations of many computing education research papers [11]. We employ two different interactive learning environments, exercises, and student/instructor populations from two continents. Using the methods of educational data mining [27] and learning analytics [16], we extract relevant information from data of 46 students, model their progress, and present the selected results as graphical models to instructors (see Section 3).

Our research goals are to examine how the graph models can support assessment and how we could improve them. We performed a study with 22 expert instructors who evaluated the graph models of selected students; Section 4 presents the results. In Section 5, we compare the two modeling approaches and discuss their benefits, limitations, and practical implications for teaching and research. Section 6 concludes the paper and summarizes our contributions.

2 RELATED WORK IN ASSESSMENT MODELS

Although in-depth assessment improves learning [36], only a few studies have explored assessment models for security exercises. Section 2.1 reviews such work and explains how we differ. Section 2.2 discusses other models used in computing education research.

2.1 Hands-on Cybersecurity Education

Visualizations of student data and learning content are valuable in education [9]. Ošlejšek et al. [22] demonstrated this in the context of cybersecurity training. They proposed multiple visualizations to support the instructors' classroom awareness and student assessment. The visualizations display data of student interaction with the training environment, such as the submission of incorrect answers and help requests. The authors claim that visual models "should provide an overview as well as detailed per-trainee data."

Andreolini et al. [1] used directed graphs to model student progress in a security exercise. The vertices of the graphs represent the intermediate states of the exercise. The edges represent the actions that trigger a state transition. The graphs are generated automatically from a reference graph to assess trainee performance. A slight shortcoming is that the states' order is fixed, and paths not leading to the solution are disregarded. Braghin et al. [4] proposed a follow-up: automated scoring metrics based on the reference graphs. However, the proposal is yet to be applied in practice.

Weiss et al. [36, 37] collected students' command histories from exercises in the EDURange platform. Using the data, they manually constructed graph models of student progress. The models revealed student approaches and misconceptions that would have been lost if the students were assessed only by the solution (in)correctness.

Mirkovic et al. [17, 20] developed a system that assesses student progress in hands-on assignments in DETERlab and EDURange platforms. The system collects the input and output of the student's command line and matches the logs with pre-defined milestones (subgoals for the assignment). The system helps the instructors to monitor student learning and identify challenging concepts.

We extend the previous work by evaluating the models with instructors from multiple institutions. We also automate some manual aspects and extend the modeling capabilities to include solutions to partially ordered tasks. These improvements allow us to model a wider variety of exercises in multiple platforms.

2.2 Other Areas of Computing Education

Modeling formalisms applied in computing education include:

- *Petri nets* [24], which were used to model how students progressed through a study curriculum [30],
- *Bayesian networks* [5, 19] to predict student attitudes and goals in a tutoring system [2] or test performance [23], and
- *Markov decision processes* to generate automated hints [3].

While these studies used student data as input for statistical and machine learning methods, we construct visual models for teachers.

Piech et al. [25] captured and clustered temporal traces of student interactions with a compiler to study how students learn to program. They applied a hidden Markov model to the traces and visualized it as a state machine for the cluster. The models then predicted student performance. In our case, the exercise milestones are clearer and easier to define, though this approach could be applied as well.

Hooshyar et al. [13] reviewed methods for modeling the players of educational games. They identified "*data-driven approaches to conceptualizing log data*" as a promising research direction. They see a major challenge in determining actions that "*represent key features of player performance.*" Our research attempts to address this problem in the domain of hands-on cybersecurity education.

3 STUDY AND ASSESSMENT METHODS

In this paper, we use the term *exercise* to denote a set of assignments in which the students practice their cybersecurity skills. We host cybersecurity exercises in two interactive learning environments: *KYPO CRP* [34] and *EDURange* [35]. For each, we now describe the exercise content, participating students, and the process of generating the graph models from students' command-line data. Then, we detail the research methods for the graph models evaluation.

3.1 Exercise Environment and Content

In the *KYPO CRP* environment, the students interact with virtual machines (VMs) in an emulated network to solve the exercise tasks. For this research, we used an exercise *Locust 3302* [14] created within the Seminar on the Simulation of Cyber Attacks [31]. Students assume the role of a cyber investigator who tracks a fictional hacker group. The students have to scan a suspicious server using `nmap` [18], identify a vulnerable service, and exploit it using Metasploit [28] to gain access. Then, they have to copy a private SSH key, crack its passphrase using John the Ripper (`john`) [21], and use it to access another host that stores secret documents.

For exercises in the *EDURange* platform, students use an SSH client to connect to one or more Linux VMs. To achieve variety, we chose an exercise called *File Wrangler*, which is entirely different from *Locust 3302*. Students worked only with one VM to perform tasks such as finding hidden files, identifying file formats, and changing access permissions.

In both exercises, the tasks are also gamified in that students find text strings called "flags" by discovering secret files.

3.2 Teaching Context and Student Data

KYPO CRP hosted the *Locust 3302* exercise for 20 participants, undergraduates and advanced high school students, in a summer school held remotely in July 2020. During the two-hour training session,

we recorded 2,382 commands submitted by the students. The data include full commands with their arguments and metadata, such as arbitrary student ID and timestamp [32]. Since the students had limited time for the exercise, not all of them finished all the tasks.

EDURange deployed *File Wrangler* in a class of 26 students in an intermediate class in networking and network security in February 2020. The students were concentrating in computer science, and they had all taken an introductory course. Most students were familiar with the Linux command line. In total, 3,178 commands were recorded and analyzed from participants in this exercise.

For both exercises, the data were anonymized to protect the students' privacy. We received a waiver/approval from our respective institutions to process the data for this study.

3.3 Model Generation from Student Data

The collected student command logs are used to model progress through the exercise. We proposed two methods for generating graph models to support student assessment, which we describe below. Although the methods work in real-time as well, the scope of this study is post-exercise assessment. Therefore, all models were generated after all students finished their exercises.

3.3.1 Trainee Graph. In the first approach, the exercise author manually and iteratively creates a *reference graph* that serves as a sample solution. Similarly to [1], the vertices of the reference graph represent the exercise subgoals, such as using the right tool. These states are desirable to reach. The directed edges represent the commands the student must execute to progress from one state to the next. The graphs are written in human-readable *DOT* language [7].

Then, each student's commands are automatically mapped to the reference graph using the NetworkX Python module [12] and visualized with Graphviz [8]. This results in our first model, a Trainee graph (see Figure 1). The pattern matching can map the student's command to any in-edge of any state. States can be reachable independently in an arbitrary order to allow modeling parallel tasks and skipping steps, or include prerequisite states to model a sequence of actions. On average, the graphs from our data included 39 states and 66 edges. For details about the graph generation, see [6, Sec. 4].

3.3.2 Milestone Graph. The other model, Milestone graph (see Figure 2), was constructed using a similar process but a different tool. The exercises are broken into tasks by the authors. For each task, specific regular expressions represent a milestone. Python scripts then read student Bash history input and output data [26]. Each line of the Bash history is split and checked against the regular expressions to find milestone attempts. If the line does not match all of the expressions, it is considered to be an unsuccessful attempt. The milestones are ordered by the author, but students do not need to complete them in that order.

The graph contains *template nodes* that describe each milestone (the same for each student) and *attempt nodes* connected to them that match the regular expressions for that milestone. Commands that do not match any milestone are not shown in the graph.

The generated files are processed by Graphviz. Successful attempts are drawn as green nodes, unsuccessful ones are yellow, and unattempted milestones are red. A summary node is appended to the chain based on whether the milestone was ultimately achieved.

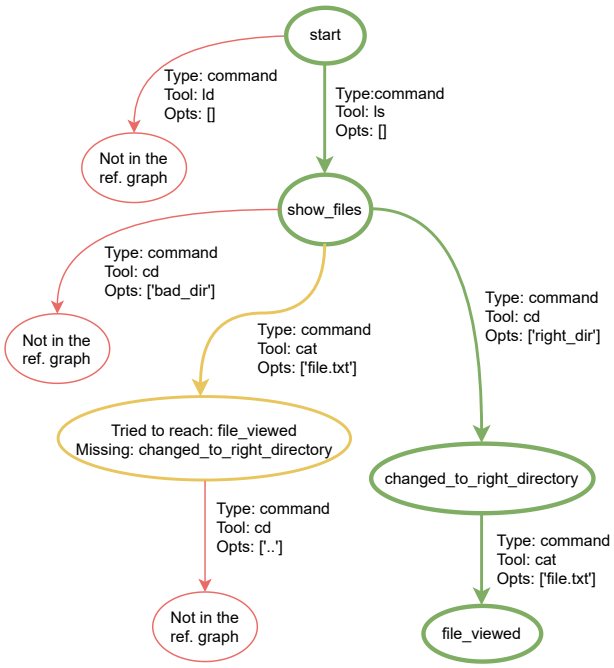

Figure 1: A simplified Trainee graph. The green states and edges represent successful steps mapped to the reference graph. The red states and edges show actions that were likely erroneous or unnecessary. The yellow state and edge show an action with possibly missing prerequisites.

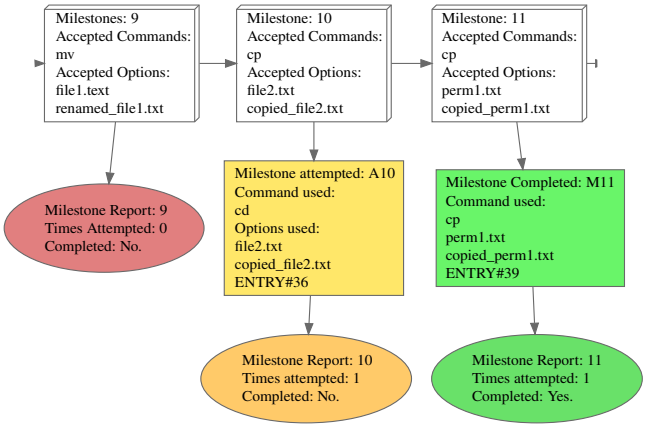

Figure 2: Milestone graph is composed of a chain of template nodes for each task. In this example, the student did not attempt the first of the three tasks, incorrectly attempted the second task (used cd instead of cp) but did not complete it, and completed the third task on their first try.

We also record the number of attempts per milestone (how many relevant commands the student tried). Sometimes matching unsuccessful commands with milestones is ambiguous. The milestones are ordered based on an expected path, which resolves ambiguities by associating an attempt with the earliest similar milestone. The commands' chronological order is encoded in the ENTRY numbers.

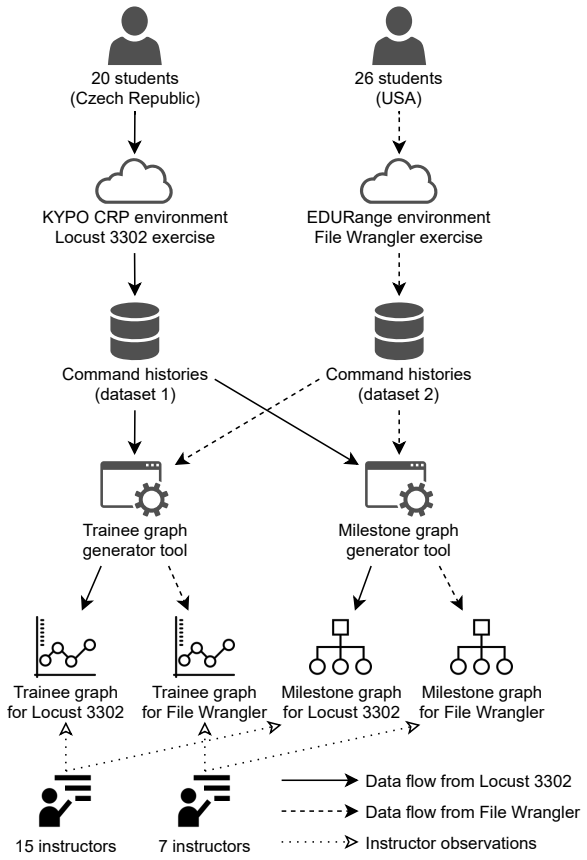

Figure 3: Overview of the factorial design of the study.

Both graph models are generic, and the tools for their creation accept input data from both learning environments. The tools would work with data from other environments too, as long as the structure of the input is preserved. This allowed us to compare the two models and would allow others to adopt or adapt them.

3.4 Model Evaluation by Expert Instructors

Figure 3 summarizes the setup of our study. After generating the graph models from student data, we selected four graphs of two representative students (one from *Locust 3302* and the other from *File Wrangler*) who did well on the exercise but had some problems. We did not choose students who did very well or poorly because that would have been obvious in both models and would have yielded less information.

Then, we asked 40 experienced computing instructors to qualitatively evaluate the models. The instructors were our current or former co-workers from about a dozen different institutions, such as a public college, liberal arts college, and a research university. Most of them had experience teaching cybersecurity but were not familiar with the two exercises. In total, 22 instructors responded (15 for *Locust 3302* and 7 for *File Wrangler*). They all participated voluntarily and were not incentivized.

Each instructor received an e-mail explaining our research goals and the following attachments:

- Briefing that familiarized them with the exercise content (either *Locust 3302* or *File Wrangler*),
- PDF files with a Trainee graph and a Milestone graph generated from the data of one student for each exercise,
- Short instructions on how to interpret the two graphs.

The instructors were asked to examine the graph models and then answer the following questions in an anonymous online survey:

Q1 How do you assess the student's progress based on the two graph models you received?
Q2 In which tasks was the student struggling? Please describe them specifically.
Q3 What feedback would you provide to the student so that his/her learning can improve?
Q4 How do you compare the two types of graph models?
Q5 On a scale from 1 (not at all) to 5 (very), how useful would the graphs be for your classes?

Before the actual study, we performed a pilot test among the paper authors and two other instructors and clarified the instructions and survey wording. After collecting the responses, three authors performed open coding [29] of the replies to qualitatively analyze what information did the graph models convey.

4 RESULTS OF THE SURVEY EVALUATION

We now present the results for each survey question, along with quotes from various participants to illustrate the points they made.

Q1: Assessing Student Progress

Question 1 asked the instructors to assess the student according to the graph models. Out of 22 instructors, 14 said the student progressed pretty well based on at least one of the graphs. They commented that the student made progress and demonstrated a growing understanding over time. One instructor praised the student for experimenting with different approaches.

> *"Overall the student is making good progress, but there [are] a few Linux tasks that need to be reviewed and practiced."*

Only four instructors assessed the student as having struggled with the whole exercise. Next, six other instructors interpreted the graphs as showing disparate amounts of student progress. Specifically, they assessed the student better based on the Milestone graph and worse based on the Trainee graph. The reason is that the Trainee graph shows every student command-line entry, including a lot of trial and error, while the Milestone graph omits student entries that do not match any milestone.

> *"The trainee graph makes the student appear to have fewer skills than the milestone graph – their struggle is much more visually pronounced in that representation."*

The graphs were different enough that, surprisingly, two instructors thought they came from different students (even though the study instructions said both graphs show the same student).

Q2: Identifying Difficulties

Question 2 examined whether the instructors could identify from the graphs which parts of the exercise were problematic for the student, so that they could intervene. Most of the instructors were able to do that. For *Locust 3302*, 10 out of 15 correctly identified at least

one of the areas where the student struggled. For *File Wrangler*, 6 out of 7 identified the problem areas. Therefore, both graphs fulfilled the intended use case, although there is room for improvement.

Q3: Providing Feedback to the Student

Question 3 asked the instructors how they would intervene after they identified where the student struggled. Our goal was to understand how the graphs could be used for in-class feedback.

From the pedagogical point of view, the instructors' feedback to the student differed a lot. Some suggested a direct approach, such as explaining the problem, the correct solution to it, and why the student's attempts were incorrect. They would also provide a tutorial or an example of the tool the student was struggling with.

> *"The student needs feedback on how to better use the [Linux] shell and Metasploit console."*

Some instructors opted for a more indirect approach, such as suggesting to the student to find and learn what the commands do, review and understand their syntax, and read manual pages.

> *"It is not uncommon to do trial and error on the Linux command line. But after the first failed attempt – go to* `man` *[pages]."*

They also emphasized the need to thoroughly read the task assignment and understanding it before starting to type commands.

Other instructors focused on affective feedback, such as encouraging the student to keep trying, praising them for their effort, and inviting them to ask the instructor for help.

All types of feedback were reasonable, given the information the instructors had. Three instructors noted that without the full assignment, it was hard to distinguish conceptual misunderstandings from tool-specific issues, which is a slight limitation of the survey.

> *"As a participant in this study who isn't familiar with the example assignment, it's hard for me to distinguish high-level misconceptions ("I don't understand what* `john` *does, abstractly") with low-level ones ("I know what* `john` *does but don't understand its command line arguments/syntax"). I'd probably focus in on specific learning goals and ask them about* `john` *and Metasploit and see what they do understand."*

Q4: Comparing the Two Graph Models

Question 4 asked the instructors to compare the two graphs, and the vast majority agreed on the strengths and weaknesses of both. The Trainee graph was more detailed, which is a double-edged sword. On the one hand, it gives deeper insight into the student's work, including their used commands, problems, and solution attempts. On the other hand, the graph is difficult to interpret, and working with it is more time-consuming.

> *"Trainee graph has more details, but, as a consequence, it is hard to read. It was much easier for me to understand and work with the Milestone graph. Nevertheless, Trainee graph shows [. . .] the wrong paths and gives the context unavailable in the Milestone graph (e.g., completely wrong directions)."*

Still, instructors found the Trainee graph useful for evaluating and improving their exercise design.

> *"Trainee graph lets me envision the temporal process of the student struggling, see where they got stuck, see where the design of my assignment maybe led them astray. If most of my students have similar graphs, that tells me a lot about which parts of my assignment were tricky, especially if most of them moved on past that challenge point (or didn't), whether I was reasonable in asking them to figure something out."*

The key reported weaknesses of the Trainee graph were that it shows any deviation from the reference solution as a potential error, making it difficult to detect unexpected solutions of students. It also becomes complicated with the growing number of commands and is not 100% colorblind-friendly.

The instructors strongly agreed that Milestone graph is easier to read; only two instructors found it difficult to interpret. Although it omits some details, it is simpler to work with. As a result, it provides a better quick overview (a "*summarized breakdown*") of student actions and is more suitable for batch assessment.

> *"Milestone graph is much easier to assess an individual student's progress quickly, especially when grading many students."*

Since it also captures the attempted and completed task milestones (instead of states), it quickly shows what the student can or cannot do. This translates more directly to skill assessment.

> *"Milestone graph feels more useful as a record of the student's skills and development. [I would use it] for providing learning-goal based formative feedback to my students."*

Q5: Usefulness Rating

Finally, the instructors rated how useful they thought the graphs would be in their classes on a scale of 1 (not at all) to 5 (very). Some responded that their hands-on classes might not fit the structure needed for generating this type of data. However, the average score across 21 responses (one instructor did not answer) was 3.57 out of 5. The median and the mode was 4, which means they considered it beneficial but not perfect. One respondent noted that the payoff of automated solutions like these increases as class size grows.

5 DISCUSSION

This section summarizes the lessons learned from the survey, discusses its limitations, and, based on that, proposes future work.

5.1 Summary of the Results

Developing two graph models proved useful, since their evaluation elicited different perspectives. Most instructors interpreted each graph effectively, and they also identified strengths, weaknesses, and use cases for each graph. They reported that the Trainee graph provided more detail by mapping command history as it happened. This can help review and improve exercise design, as well as discover unexpected solutions by examining the red and yellow elements. In contrast, the Milestone graph showed key stages of student's progress and was easier to read. It is better in providing a quick overview, especially in time-critical situations, and supports assessment based on learning goals. Overall, the instructors found both models useful and novel but also noted their shortcomings, mainly the time required for large classes.

> *"I see the great potential in visualizing commands for better analysis of students' thinking. [The models] might provide better insight into their work. However, in their current form, they*

are far from ideal. I would need to analyze [the models] one by one and formulate suggestions for the students independently."

Another participant commented that the graphs could be shown to students as feedback. This could highlight common misconceptions or missed learning objectives across an entire class.

"I like the idea of graphically summarizing the student experience [...]. I can imagine displaying a bunch of graphs for all students in a class side-by-side and having the common problems jump out visually. This would help the instructor know what to emphasize in the next class session."

5.2 Implications for Teaching Practice

Since both graphs visualize the task subgoals and student attempts, the graphs could be used for the following educational use cases:

- identify high- or low-performing students in class;
- identify successes and struggles of a specific student;
- assess students, both formatively and summatively; and
- give each student their own graph to reflect on their approaches, self-evaluate their learning process, and identify problems in the steps they chose to solve the exercise.

A key feature is that these use cases apply to both in-person and distance education. Supporting remote assessment is crucial when the instructor has limited access to what the students are doing. Moreover, since the graphs can be generated in real-time, they can be used for in-class interventions, not only post-exercise feedback as in this study. This feature becomes especially relevant if the graph generation is incorporated into the learning environment. Last but not least, the graph models are applicable not only in the cybersecurity domain, but generalize to any learning exercise that can be represented by a series of actions.

5.3 Addressing the Limitations

The evaluation also revealed the limitations of the graph models. In complex exercises, the reference graph or milestone definitions that enable model generation can be incomplete. Thus, a correct but unexpected student solution could be marked as erroneous. However, an instructor can gradually update the definitions and generate new graphs.

Another limitation is that the graphs do not scale for sequences of hundreds of commands. This can be resolved by splitting complex exercises into sections, or implementing the graphs as interactive visualizations with collapsible parts and filters.

Some may consider a limitation that the tools are primarily designed for command-line exercises. However, command line interface is important in practice, and the tools would also work with a variety of log files, e.g., webserver or database query logs. Relevant exercises include not only the cybersecurity domain, but also programming, operating systems, and networking. Given a reference graph or regular expressions tailored to these data sources, the graphs can be extended to display many types of student activity.

Regarding the study validity, we received survey responses from 22 instructors out of the 40 asked. Although this number is not high enough to allow generalizing the results, the sample represents instructors with different backgrounds. While selection bias may be present, since we asked mostly our current or former colleagues, the multi-institutional study framework should mitigate the bias. The final limitation is that although we had data from 46 students, we selected only two of them. The reason is that since we did not incentivize the instructors who participated in the survey, we did not want to take too much of their time by asking them to evaluate more graphs. Nevertheless, we selected representative students to illustrate various aspects that appeared in other graphs as well.

5.4 Future Work

Future studies can evaluate the effectiveness of the graph modeling approach with students in real-time. It can be interesting to examine whether students would find the information in the graphs useful. Another follow-up study can be more longitudinal, investigating whether the performance of a single student as displayed by the graphs improves over several training sessions.

Future research can also incorporate machine learning methods. Clustering can group students based on their performance. This would scale to large classes and save the instructors' time because they would not need to examine each student, only the representative of a cluster, and provide feedback applicable to the whole cluster. Alternatively, classification can be used to assess students automatically and even live during the exercise, indicating their skill level or the correctness of their actions. This solution would address the suggestion of one of the study participants:

"I could see having both these graphs being potentially quite useful, especially if it updated live as my students worked, allowing me to catch common areas of concern."

6 CONCLUSIONS

Assessment of learning is crucial to provide instructors with classroom situational awareness, identify students' strengths and shortcomings, and help students learn. This work proposed two methods for assessing student progress in hands-on exercises. The methods visualize and contextualize command history logs, which are very hard to process manually in their raw form. One method provides a quick summary; the other complements it with a detailed view. Together, they improve understanding of students' approaches to learning and represent a faster form of feedback than traditional post-homework assessment. Another strength of this collaborative research is that by giving the instructors two models to compare, they precisely formulated what worked and what did not in each.

We implemented the methods as open-source tools and used them to generate 46 graph models from authentic in-class data: 5,560 commands submitted by students at two universities (approx. 121 lines per student of minimally formatted text logs). The source code and data are available in a public repository [33] to support their adoption by other instructors and researchers.

The resulting graph models were evaluated by 22 instructors from various institutions. Qualitative analysis of their responses revealed strengths, weaknesses, and applications of the two proposed methods in assessment. They can highlight student skills, provide a basis for classroom interventions, and reveal issues in exercise design. Since the methods are generic, they are applicable in other learning environments and exercises. Moreover, they can be applied outside of the cybersecurity domain to enhance assessment in other computing classes that capture student interaction.

ACKNOWLEDGMENTS

The researchers from Masaryk University were supported by ERDF project *CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence* (No. CZ.02.1.01/0.0/0.0/16_019/0000822). Part of this paper is based upon work supported by the National Science Foundation under grant numbers 1723705 and 1723714. Finally, we thank the instructors who participated in this study, as well as the SIGCSE conference reviewers who provided constructive feedback that helped improve the final version of paper.

REFERENCES

[1] Mauro Andreolini, Vincenzo Colacino, Michele Colajanni, and Mirco Marchetti. 2019. A Framework for the Evaluation of Trainee Performance in Cyber Range Exercises. *Mobile Networks and Applications* 25 (2019). https://doi.org/10.1007/s11036-019-01442-0

[2] Ivon Arroyo, David G. Cooper, Winslow Burleson, and Beverly P. Woolf. 2010. Bayesian Networks and Linear Regression Models of Students' Goals, Moods, and Emotions. In *Handbook of educational data mining*, Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan S.J.d. Baker (Eds.). CRC Press, Boca Raton, FL, USA, Chapter 23, 323–338. https://doi.org/10.1201/b10274

[3] Tiffany Barnes, John Stamper, and Marvin Croy. 2010. Using Markov Decision Processes for Automatic Hint Generation. In *Handbook of educational data mining*, Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan S.J.d. Baker (Eds.). CRC Press, Boca Raton, FL, USA, Chapter 33, 467–480. https://doi.org/10.1201/b10274

[4] Chiara Braghin, Stelvio Cimato, Ernesto Damiani, Fulvio Frati, Elvinia Riccobene, and Sadegh Astaneh. 2020. Towards the Monitoring and Evaluation of Trainees' Activities in Cyber Ranges. In *Model-driven Simulation and Training Environments for Cybersecurity*, George Hatzivasilis and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, 79–91. https://doi.org/10.1007/978-3-030-62433-0_5

[5] Eugene Charniak. 1991. Bayesian networks without tears. *AI magazine* 12, 4 (1991), 50–50. https://doi.org/10.1609/aimag.v12i4.918

[6] Radoslav Chudovský. 2020. *Modeling Progress Through Cybersecurity Training Using Command Histories.* Bachelor's Thesis. Masaryk University, Faculty of Informatics. https://is.muni.cz/th/hpykg/?lang=en

[7] John Ellson et al. 2021. *DOT: Graph description language.* GraphViz. Retrieved November 19, 2021 from https://www.graphviz.org/doc/info/lang.html/

[8] John Ellson, Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Gordon Woodhull. 2004. *Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools.* Springer Berlin Heidelberg, Berlin, Heidelberg, 127–148. https://doi.org/10.1007/978-3-642-18638-7_6

[9] Elif E. Firat and Robert S. Laramee. 2018. Towards a Survey of Interactive Visualization for Education. In *Proceedings of the Conference on Computer Graphics & Visual Computing (CGVC '18)*. Eurographics Association, Goslar, DEU, 91–101. https://doi.org/10.2312/cgvc.20181211

[10] CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3467967

[11] Mark Guzdial and Benedict du Boulay. 2019. The History of Computing Education Research. In *The Cambridge Handbook of Computing Education Research*, Sally A Fincher and Anthony V Robins (Eds.). Cambridge University Press, Cambridge, United Kingdom, Chapter 1, 11–39. https://doi.org/10.1017/9781108654555

[12] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). SciPy Conferences, Pasadena, CA, USA, 11–15. https://www.osti.gov/biblio/960616

[13] Danial Hooshyar, Moslem Yousefi, and Heuiseok Lim. 2018. Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review. *ACM Comput. Surv.* 50, 6, Article 90 (Jan. 2018), 19 pages. https://doi.org/10.1145/3145814

[14] Cybersecurity Laboratory. 2021. *Locust 3302.* Masaryk University. Retrieved November 19, 2021 from https://gitlab.ics.muni.cz/muni-kypo-trainings/games/locust-3302

[15] Thomas Lancaster, Anthony V. Robins, and Sally A. Fincher. 2019. Assessment and Plagiarism. In *The Cambridge Handbook of Computing Education Research*, Sally A Fincher and Anthony V Robins (Eds.). Cambridge University Press, Cambridge, United Kingdom, Chapter 14, 414–444. https://doi.org/10.1017/9781108654555

[16] Charles Lang, George Siemens, Alyssa Wise, and Dragan Gašević (Eds.). 2017. *Handbook of Learning Analytics* (1st ed.). Society for Learning Analytics Research (SoLAR). https://doi.org/10.18608/hla17

[17] Paul Lepe, Aashray Aggarwal, Jelena Mirkovic, Jens Mache, Richard Weiss, and David Weinmann. 2019. Measuring Student Learning On Network Testbeds. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, New York, NY, USA, 1–2. https://doi.org/10.1109/ICNP.2019.8888101

[18] Gordon Lyon. 2021. *Nmap Network Scanning.* Nmap. Retrieved November 19, 2021 from https://nmap.org/book/man.html

[19] Eva Millán, Tomasz Loboda, and Jose Luis Pérez-De-La-Cruz. 2010. Bayesian networks for student model engineering. *Computers & Education* 55, 4 (2010), 1663–1683. https://doi.org/10.1016/j.compedu.2010.07.010

[20] Jelena Mirkovic, Aashray Aggarwal, David Weinman, Paul Lepe, Jens Mache, and Richard Weiss. 2020. Using Terminal Histories to Monitor Student Progress on Hands-on Exercises. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. ACM, New York, NY, USA, 866–872. https://doi.org/10.1145/3328778.3366935

[21] Openwall. 2021. *John the Ripper password cracker.* Retrieved November 19, 2021 from https://www.openwall.com/john/

[22] Radek Ošlejšek, Vít Rusňák, Karolína Burská, Valdemar Švábenský, Jan Vykopal, and Jakub Čegan. 2021. Conceptual Model of Visual Analytics for Hands-on Cybersecurity Training. *IEEE Transactions on Visualization and Computer Graphics* 27, 8 (2021), 3425–3437. https://doi.org/10.1109/TVCG.2020.2977336

[23] Zachary A. Pardos, Neil T. Heffernan, Brigham S. Anderson, and Cristina L. Heffernan. 2010. Using Fine-Grained Skill Models to Fit Student Performance with Bayesian Networks. In *Handbook of educational data mining*, Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan S.J.d. Baker (Eds.). CRC Press, Boca Raton, FL, USA, Chapter 29, 417–426. https://doi.org/10.1201/b10274

[24] James L. Peterson. 1977. Petri Nets. *ACM Comput. Surv.* 9, 3 (Sept. 1977), 223–252. https://doi.org/10.1145/356698.356702

[25] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. 2012. Modeling How Students Learn to Program. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 153–160. https://doi.org/10.1145/2157136.2157182

[26] Chet Ramey and Brian Fox. 2020. *The GNU Bash Reference Manual, for Bash, Version 5.0.* Free Software Foundation. https://www.gnu.org/savannah-checkouts/gnu/bash/manual

[27] Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan S.J.d. Baker (Eds.). 2010. *Handbook of educational data mining.* CRC Press, Boca Raton, FL, USA. https://doi.org/10.1201/b10274

[28] Offensive Security. 2021. *Metasploit Unleashed.* OffSec Services Limited. Retrieved November 19, 2021 from https://www.offensive-security.com/metasploit-unleashed/

[29] Josh Tenenberg. 2019. Qualitative Methods for Computing Education. In *The Cambridge Handbook of Computing Education Research*, Sally A Fincher and Anthony V Robins (Eds.). Cambridge University Press, Cambridge, United Kingdom, Chapter 7, 173–207. https://doi.org/10.1017/9781108654555

[30] Nikola Trčka, Mykola Pechenizkiy, and Wil van der Aalst. 2010. Process Mining from Educational Data. In *Handbook of educational data mining*, Cristobal Romero, Sebastian Ventura, Mykola Pechenizkiy, and Ryan S.J.d. Baker (Eds.). CRC Press, Boca Raton, FL, USA, Chapter 9, 123–142. https://doi.org/10.1201/b10274

[31] Valdemar Švábenský, Jan Vykopal, Milan Cermak, and Martin Laštovička. 2018. Enhancing Cybersecurity Skills by Creating Serious Games. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. ACM, New York, NY, USA, 194–199. https://doi.org/10.1145/3197091.3197123

[32] Valdemar Švábenský, Jan Vykopal, Pavel Seda, and Pavel Čeleda. 2021. Dataset of shell commands used by participants of hands-on cybersecurity training. *Data in Brief* 38 (2021), 9. https://doi.org/10.1016/j.dib.2021.107398

[33] Valdemar Švábenský, Richard Weiss, Jack Cook, Jan Vykopal, Pavel Čeleda, Jens Mache, Radoslav Chudovský, and Ankur Chattopadhyay. 2021. *Dataset: Evaluating Two Approaches to Assessing Student Progress in Cybersecurity Exercises.* Zenodo. https://doi.org/10.5281/zenodo.5752288

[34] Jan Vykopal, Pavel Čeleda, Pavel Šeda, Valdemar Švábenský, and Daniel Tovarňák. 2021. Scalable Learning Environments for Teaching Cybersecurity Hands-on. In *Proceedings of the 51st IEEE Frontiers in Education Conference (FIE '21)*. IEEE, New York, NY, USA, 1–9. https://www.muni.cz/en/research/publications/1783808

[35] Richard Weiss, Stefan Boesen, James F. Sullivan, Michael E. Locasto, Jens Mache, and Erik Nilsen. 2015. Teaching Cybersecurity Analysis Skills in the Cloud. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. Association for Computing Machinery, New York, NY, USA, 332–337. https://doi.org/10.1145/2676723.2677290

[36] Richard Weiss, Michael E. Locasto, and Jens Mache. 2016. A Reflective Approach to Assessing Student Performance in Cybersecurity Exercises. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 597–602. https://doi.org/10.1145/2839509.2844646

[37] Richard Weiss, Franklyn Turbak, Jens Mache, and Michael E Locasto. 2017. Cybersecurity Education and Assessment in EDURange. *IEEE Security & Privacy* 15, 3 (2017), 90–95. https://doi.org/10.1109/MSP.2017.54